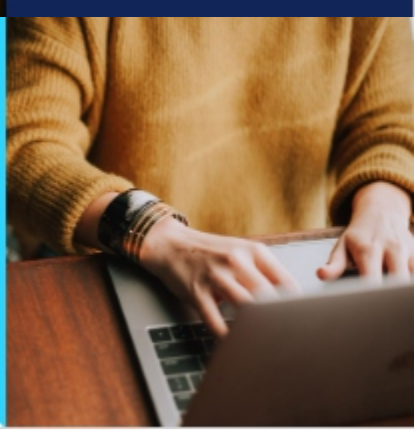
Невлюдов І.Ш., Євсєєв В.В., Андрусевич А.О., Максимова С.С.

МОДЕЛІ ТА МЕТОДИ КІБЕРФІЗИЧНИХ ВИРОБНИЧИХ СИСТЕМ В КОНЦЕПЦІЇ INDUSTRY 4.0



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

І.Ш. Невлюдов, В.В.Євсєєв, А.О. Андрусевич, С.С.Максимова

МОДЕЛІ ТА МЕТОДИ КІБЕРФІЗИЧНИХ ВИРОБНИЧИХ СИСТЕМ В КОНЦЕПЦІЇ INDUSTRY 4.0

МОНОГРАФІЯ

Харків 2023

УДК 62.932:007.52

Рекомендовано науково-технічною радою
Харківського національного університету радіоелектроніки
(протокол № 3/2 від 24.02.2023 року)

Невлюдов І.Ш., Євсєєв В.В., Андрусевич А.О., Максимова С.С. Моделі та методи кіберфізичних виробничих систем в концепції Industry 4.0 Монографія. – Харків: 2023. – 321 с.

ISBN
DOI:

РЕЦЕНЗЕНТИ:

доктор технічних наук, професор, помічник директора Державного підприємства «Південний державний Проєктно-конструкторський та науково-дослідний інститут авіаційної промисловості» **Косенко Віктор Васильович**

доктор технічних наук, професор, директор Державного підприємства «Науково-дослідного технологічного інституту приладобудування» **Замірець Микола Васильович**.

В монографії розглядаються питання забезпечення ефективної стратегії автоматизації керування складними виробничими об'єктами, шляхом реалізації комплексу моделей, методів процесів керування і технології на базі кіберфізичних виробничих систем.

В роботі розроблено архітектурно-логічну модель подання керування процесами в виробничих об'єктах на базі кіберфізичних систем. Вона базується на науково обґрунтованих теоріях мультисистем і моносистем та методах формалізованого подання систем. Це дозволило об'єднати стратегічні, фізичні та кібернетичні складові системи у єдиний інформаційний простір. На базі моделі запропоновано технологію розробки кіберфізичних виробничих систем, що дозволило розробити метод подання структурних системних моделей кіберфізичного керування процесами. Це, в свою чергу, дозволило формалізувати алгоритм функціонування та побудувати структурні і подієві моделі функціонування кіберфізичних виробничих систем. Удосконалено метод синтезу алгоритмів функціонування кіберфізичних виробничих систем. Це призвело до мінімізації кількості операторів і спрощення структури системи функціонування виробничого об'єкту. Розроблено модель формалізації кібернетичної складової виробничого об'єкта у вигляді взаємопов'язаних багаторівневих GUI елементів, що дозволило автоматизувати реалізацію функцій відповідно до вимог HMI кіберфізичних виробничих систем. Також розроблено синтаксичну і семантичну моделі декларативної мови визначення й опису моделювання кібернетичної складової. Проведено ряд досліджень, які показали підвищення продуктивності та ритмічності виробництва.

Монографія може бути корисна науковцям, аспірантам, студентам, які займаються питаннями розробки методологій проєктування кіберфізичних виробничих систем в концепції Industry 4.0.

УДК 62.932:007.52

ISBN
DOI:

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АРМ – автоматизоване робоче місце;

БД – база даних;

ЖЦ – життєвий цикл;

ІС – інформаційна система;

ІТ – інформаційні технології;

ОС – операційна система;

ОТ – операційні технології;

ПМ – програмний модуль;

ПЗ – програмне забезпечення;

ПП – програмний продукт;

ТП – технологічний процес;

ТПВ – технологічна підготовка виробництва;

ТЗ – технічне завдання;

СКБД – системи керування базами даних;

AI (Cloud computing) – штучний інтелект;

APS (Advanced Planning and Scheduling) – система синхронного планування виробництва;

BD (Big Data) – структуровані та неструктуровані дані величезних обсягів і різноманітності, а також методи їх обробки, які дозволяють розподілено аналізувати інформацію;

c-MES (Collaborative Manufacturing Execution System) – спеціалізоване прикладне програмне забезпечення, призначене для вирішення завдань синхронізації, координації, аналізу й оптимізації випуску продукції в рамках будь-якого виробництва;

CASE (Computer-Aided Software Engineering) – набір інструментів і методів програмної інженерії для проєктування програмного забезпечення;

CC (Cloud computing) – технології розподіленої обробки цифрових даних, за допомогою яких комп'ютерні ресурси надаються інтернет-користувачеві як онлайн-сервіс;

CMMS (Computerized Maintenance Management System) – комп'ютеризована система керування технічним обслуговуванням;

CMMI (Capability Maturity Model Integration) – модель зрілості можливостей створення програмного забезпечення;

DCA (Data Collection/Acquisition) – збір і зберігання даних;

DCS (Distributed Control System) – розподілена система керування;

DFD (Data Flow Diagrams) – методологія графічного структурного аналізу;

DIKW (Data, Information, Knowledge, Wisdom) – інформаційна ієрархія, де кожен рівень додає певні властивості до попереднього рівня;

DPU (Dispatching Production Units) – диспетчеризація виробництва;

DT (Digital Twin) – цифровий близнюк;

GUI (Graphical User Interface) – система засобів для взаємодії користувача з комп'ютером, заснована на представленні всіх доступних користувачеві системних об'єктів і функцій у вигляді графічних компонентів екрану;

ERP (Enterprise Resource Planning) – планування ресурсів підприємства;

HMI (Human-Machine Interface) – інженерні рішення, що забезпечують взаємодію людини-оператора з керованими їм машинами;

Industry 4.0 – прогнозована подія, масове впровадження кіберфізичних систем у виробництво і т.д.;

IoT (Internet of Things) – мережа пов'язаних через Інтернет об'єктів, здатних збирати дані й обмінюватися даними, які надходять зі вбудованих сервісів;

i-ERP (Enterprise Resource Planning) – організаційна стратегія інтеграції виробництва й операцій керування трудовими ресурсами, фінансового менеджменту й управління активами, орієнтована на безперервне

балансування й оптимізацію ресурсів підприємства за допомогою спеціалізованого інтегрованого пакета прикладного програмного забезпечення, що забезпечує загальну модель даних і процесів для всіх сфер діяльності;

JAD (Joint Application Development) – методологія управління Проєктами, яка передбачає тісну взаємодію замовників і виконавців з метою домогтися взаєморозуміння в питаннях, що стосуються розроблюваної системи;

LAM (Lifecycle Architecture Milestone) – етап життєвого циклу архітектури;

LOM (Lifecycle Objective Milestone) – етап життєвого циклу мети;

LP (Lean Production) – концепція бережливого виробництва;

LUM (Labor/User Management) – керування людськими ресурсами;

M2M (Machine-to-Machine) – загальна назва технологій, які дозволяють машинам обмінюватися інформацією один з одним, або ж передавати її в односторонньому порядку;

MEMS (Micro-Electro-Mechanical Systems) – пристрої, що поєднують в собі мікроелектронні і мікромеханічні компоненти;

MSI (Module Internal Shared Input) – інтегрована людино-машинна система забезпечення інформацією;

PA (Performance Analysis) – аналіз ефективності;

PIMS (Production Information Management System) – система керування виробничою інформацією;

PLC (Programmable Logic Controller) – спеціальний різновид електронної обчислювальної машини;

PM (Process Management) – керування процесами виробництва;

PoE (Power over Enternet) – технологія, що дозволяє передавати з віддаленого пристрою електричну енергію разом з даними через мережу Ethernet;

PTG (Product Tracking & Genealogy) – відстеження і генеалогія

продукції;

QM (Quality Management) – управління якістю;

RAS (Resource Allocation and Status) — контроль стану і розподіл ресурсів;

RAD (Rapid Application Development) – методологія інкрементного прототипування при Проєктуванні ІС;

RPA (Robotic Process Automation) – автоматизовані робототехнічні процеси для емуляції дій людини;

RUP (Rational Unified Process) – методологія, заснована на інтерактивній моделі проєктування інформаційних систем;

SADT (Structured Analysis and Design Technique) – методологія структурного аналізу і проєктування ІС;

SCADA (Supervisory Control and Data Acquisition) – програмний пакет, призначений для розробки або забезпечення роботи в реальному часі систем збору, обробки, відображення та архівування інформації про об'єкт моніторингу або керування;

UML (Unified Modeling Language) – уніфікована мова моделювання;

WMS (Warehouse Management System) – система управління складом.

ВСТУП

Глобальна конкуренція в галузі виробництва високотехнологічних виробів характеризується короткими життєвими циклами (ЖЦ), ускладненням технічної та технологічної підготовки виробництва (ТПВ) та підвищенням вимог до їх контролю та моніторингу в режимі реального часу. Вимога досягнення високої якості таких виробів викликає потребу постійного удосконалення технологічних процесів (ТП), а також зміни структури їх керування, що є центральними чинниками успіху для виробничих компаній.

З розвитком IoT і Industry 4.0 все більш широке застосування отримує впровадження кіберфізичних виробничих систем (CPPS), в рамках концепції Smart Manufacturing. В свою чергу цифровізація виробничих процесів і процесів керування вимагає обробки та маніпуляції великими обсягами різнорідних промислових даних в масштабі реального часу та протягом всього життєвого циклу виробу. Щоб використовувати промислові дані, для отримання конкурентних переваг, необхідно забезпечити гнучке, адаптивне, бережливе виробництво (LP), яке повинно бути орієнтоване на людину.

Автоматизація процесів управління розробкою нових і модернізацією існуючих CPPS, в рамках концепції Smart Manufacturing, пов'язана зі складною науково-технічною задачею розробки комплексу математичних моделей і методів, які забезпечують синтез фізичних і кібернетичних властивостей ТПВ високотехнологічних виробів, а також необхідністю пошуку нових підходів до створення систем автоматизації процесів керування їх розробкою, які дозволять підвищити ефективність виробництва, за рахунок інтеграції промислових стандартів Industry 4.0 (RAMI 4.0) і підходів LP.

Істотний внесок у розвиток теоретичних і методологічних основ розробки кіберфізичних виробничих систем внесли: Lee J., Bagheri B., Kao H.

– розроблено 5C архітектуру CPPS; Jiang Jehn-Ruey – вдосконалено архітектуру 5C і на базі неї запропоновано архітектуру CPPS 8C; Rasman M., Pipan M., Šimic M., Herakovič N. – запропоновано модель LASFA побудови Smart Manufacturing на базі архітектурної моделі RAMI 4.0; Radanliev P., Roure D. De, Nicolescu R., Huth M. – розроблено емпіричну архітектурну модель інтеграції CPPS-IoE, на базі архітектури CPPS 5C; Zhang H., Zhang G., Yan Q. – запропоновано розглядати CPPS, як PMDT модель, в якій основна увага фокусується на етапі виробництва в інтелектуальному цеху і є об'єднанням 5 моделей: модель визначення продукту (PDM), геометрична модель і модель форми (GSM), модель виробничих атрибутів (MAM), модель поведінки і правил (BRM) і модель об'єднання даних (DFM); Wagner T., Herrmann C., Thiede S. – розроблено матрицю впливу Industry 4.0 (CPPS) на системні принципи Lean Production Systems і їх ефективне досягнення для виробництва високотехнологічних виробів; Elhoone H., Zhang T., Anwar M., Desai S. – проведено дослідження і запропоновано основу для розробки кібер-адитивного дизайну для CPPS, що може динамічно розподіляти цифрові конструкції для спрощення розробки HMI для CPPS.

Науково-технічні розробки в цій галузі ведуться в США, Німеччині, Японії, Франції, Австралії, Китаї в рамках державних програм розвитку цифрового виробництва.

Розвиток методології автоматизації процесів керування на базі CPPS були відображені в роботах таких вчених: Хаустова В. Є., Хаханова В.І., Крамарева Г. В., Зінченка В. А., Бажал Ю. М., Бужімської К. О., Варшавського О. Є., Васечко Д. Ю., Геєць В. М., Глазьєва С. Ю., Грущінської Н. М., Кизим М. О., Кіндзерського Ю. В., Сухарєва О. С., Якубовського М. М, Яструб Н. А., Жолткевича Г. М., Ладанюка А. П., Невлюдова І. Ш.

Незважаючи на численні публікації в галузі автоматизації процесів керування розробкою CPPS, на даний момент часу існує протиріччя між підвищенням ефективності розробки CPPS, з урахуванням концепції Lean

Production (LP), і обмеженістю існуючих методологій, математичних моделей і методів їх реалізації. Це обумовлює актуальність науково-технічної задачі розробки ефективної стратегії автоматизації управління складними організаційно-технічними виробничими об'єктами, шляхом реалізації комплексу моделей, методів процесів управління і технології на базі кіберфізичних систем для «безлюдного виробництва».

Кіберфізичні виробничі системи (CPPS) є цифровими близнюками (Digital Twins) реальних виробничих процесів і вирішують великий спектр завдань, які пов'язані з автоматизацією всіх процесів, що протікають при виробництві високотехнологічних виробів на всіх етапах їх життєвого циклу. Впровадження таких систем дозволить управляти і контролювати виробництво в режимі реального часу, що дасть можливість аналізувати і оптимізувати виробничий процес і життєвий цикл виробу, що є одним з критеріїв для досягнення цілей LP.

Дослідження в цій галузі є досить новим напрямком, який виник з появою концепції Industry 4.0 і технології IIoT. Аналіз останніх публікацій показав, що CPPS є складним об'єктом, який об'єднує в собі фізичні та кібернетичні складові, архітектура та структура яких залежить від вимог, що висуваються до виробу (конструкторські, технологічні параметри, вимоги до обладнання, PLC, датчиків, виконавчих механізмів і т.д.) і вибору методів контролю й управління, способів моніторингу, зберігання великих обсягів даних і їх аналіз, які повинні бути узгоджені в режимі реального часу для уникнення появи браку або простою обладнання, що зменшує економічну ефективність виробництва.

Незважаючи на сучасні дослідження в даній галузі та велику кількість науково-дослідних робіт, залишається невирішеною проблема відсутності ефективної стратегії автоматизації управління складними організаційно-технічними виробничими об'єктами на базі кіберфізичних виробничих систем для «безлюдного виробництва». Тому дана монографія, спрямована на розробку нових методів, моделей, нової технології, алгоритмічного і

програмного забезпечення управління організаційно-технічними об'єктами на базі кіберфізичних виробничих систем є актуальною.

З огляду на це необхідно провести удосконалення існуючих архітектурних моделей RAMI 4.0, ISA-95 (S95), 5C, 8C, CPPS-IoE, а також інтерпретаційної моделі DIKW, які покладені в теоретичні основи управління організаційно-технічними об'єктами на базі кіберфізичних виробничих систем.

Запропоновано методологію та розроблена узагальнена схема теоретичних, експериментальних і практичних досліджень, які передбачають розробку нових і вдосконалення існуючих математичних моделей, методів, технології, алгоритмічного і програмного забезпечення управління організаційно-технічними об'єктами на базі кіберфізичних виробничих систем для розв'язання задачі підвищення продуктивності і ритмічності виробництва від впровадження результатів дослідження.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНИХ МЕТОДОЛОГІЙ, МЕТОДІВ І МОДЕЛЕЙ РОЗРОБКИ ТА УПРАВЛІННЯ КІБЕРФІЗИЧНИМИ ВИРОБНИЧИМИ СИСТЕМАМИ

1.1 Концепція Industry 4.0 в технології IIoT

Тенденції розвитку сучасного світу ведуть до збільшення обсягів інформації, підвищення вимог до її точності і своєчасного подання для аналізу і ухвалення рішень в режимах реального часу, вимагають перегляду підходів до використання високих технологій та їх ролі в різних сферах діяльності людини, що, у свою чергу, потребує змінити підходи до промислових технологій.

Провідні країни в галузі інноваційних технологій в промисловості запропонували нову концепцію стратегії цифрової революції – Industry 4.0 [1-9]. У Німеччині концепція Industry 4.0 підтримується міністерствами Federal Ministry of Education and Research Germany (BMBF) [10] і Federal Ministry for Economic Affairs and Energy Germany (BMWi) [11]. В роботах Хаустова В. Є., Крамарєва Г. В., Зінченко В. А. обґрунтовується актуальність впровадження концепції Industry 4.0 в пріоритетні сфери промисловості України [12].

Дана концепція, за даними IoT Analytics (https://iot-analytics.com/), підтримана трьома провідними міжнародними галузевими організаціями: BitKom (інформаційні технології) [13-15], VDMA (машинобудування) [16-18] і ZVEI (електротехніка) [19-21]. Графік зміни інтересу до пошуку Industry 4.0 і суміжних термінів з 2011 до 2022 наведено на рис.1.1.

IOT ANALYTICS

Your Global IoT Market Research Partner

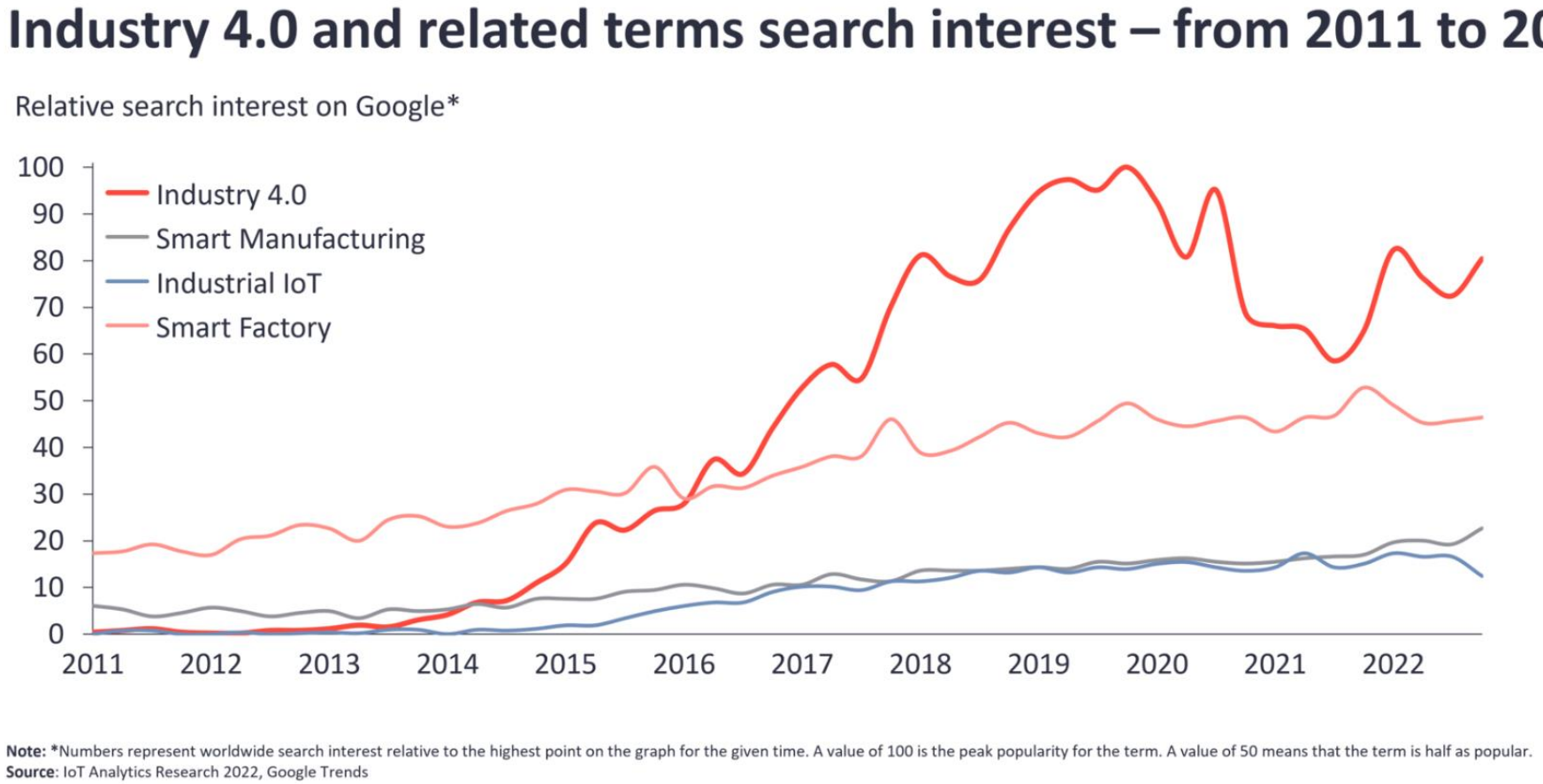


Рисунок 1.1 – Інтерес до пошуку Industry 4.0 і суміжних термінів з 2011 до 2022

Як можна бачити з представленого аналізу, технологія Industry 4.0 застосовується в двох основних блоках: Connected Industry Building і Supporting Technologies, в які входять всі провідні корпорації і фірми з надання послуг у всіх сферах діяльності людства від побутової електроніки до розробки складних космічних об'єктів, систем аналізу і обробки великих обсягів даних.

Основними представниками на ринку Industry 4.0 є ABB (Швейцарія), Mitsubishi (Японія), Yaskawa (Японія), KUKA (Німеччина), FANUC (Японія), General Electric (США), IBM (США), Cisco (США), Microsoft (США), Stratasys (США), Google (США), Intel (США), HP (США), Siemens (Німеччина), Ansys (США), AIBrain (США), SAP (США), Amazon Web Services (США) і General Vision (США) [22].

Аналізуючи вендори можна умовно виділити групу корпорацій і підприємств, сферою діяльності яких є виготовлення виробів різного призначення від мікрочіпів до складних високотехнологічних пристроїв, в які інтегровані елементи технології Industrial Internet of Things (IIoT) [23-25].

Забезпечення виробничого циклу високотехнологічних пристроїв неможливе без впровадження систем автоматизації на всіх рівнях виробництва і його супроводу, що в синтезі дозволяють створити єдиний інформаційний простір Smart Factory або Smart Manufacturing [26-29]. Національний інститут стандартів і технологій США (NIST) визначає поняття Smart Factory або Smart Manufacturing як повністю інтегровані корпоративні виробничі системи, які здатні в реальному масштабі часу реагувати на мінливі умови виробництва, вимоги мереж поставок і задоволення потреб клієнтів [30, 31].

Для досягнення даної мети необхідно використовувати дивергенцію інформаційно-комунікаційних технологій (ICT) [32-35], операційних технологій (OT) [36-40] і кіберфізичних систем (CPPS) [34, 41-49] на всіх етапах виробництва високотехнологічної продукції.

Але якщо розглянути цю дивергенцію більш детально, то можна

зауважити, що вона включає в себе хмарні технології (CC) [4, 50, 51], великі дані (Big Data) [51-54], механізми штучного інтелекту (AI) [51, 55, 56], аналіз даних на межі мереж (туманні і приграничні обчислення) [57, 58], мобільну передачу даних [59, 60], мережеві технології [61, 62], інтерфейси користувачів (HMI) [63-65], SCADA системи [66-68], системи управління (c-MES, i-ERP) [69-73], програмовані логічні контролери (PLC) [67, 74, 75], датчики і виконавчі механізми (MEMS) [76-78], автоматизовані робототехнічні процеси (RPA) [79, 80], автономні роботи (AR) [81, 82]. У роботах [83, 84] запропоноване схематичне подання Smart Manufacturing в компонентах Industry 4.0, яке представлено на рис. 1.2.

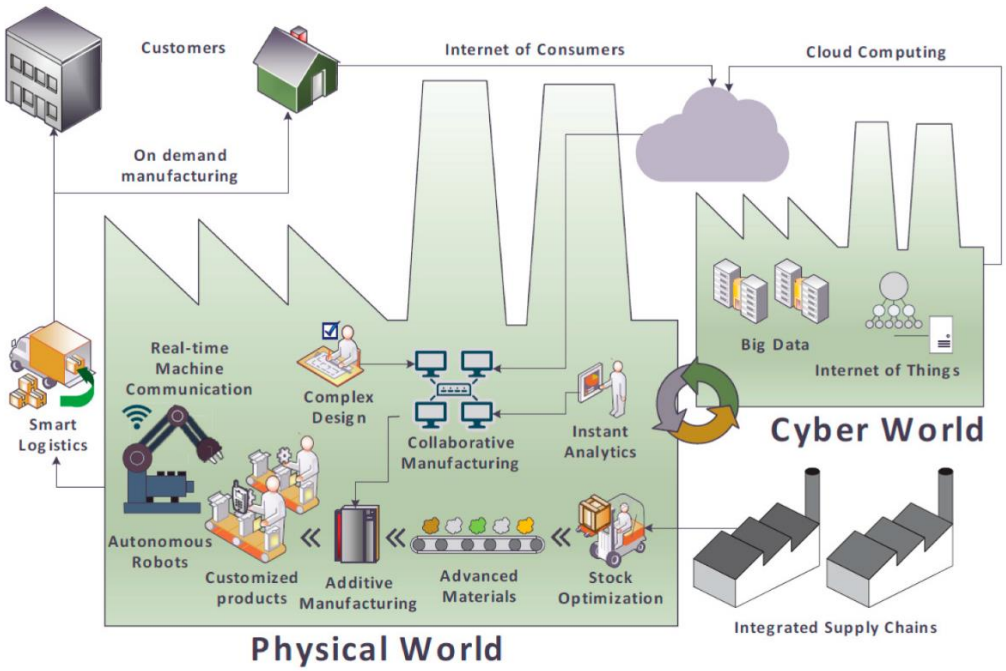


Рисунок 1.2 – Схематичне представлення Smart Manufacturing в компонентах Industry 4.0 [83, 84]

Представлене схематичне подання Smart Manufacturing в компонентах Industry 4.0 повністю відображає основну концепцію німецького стандарту DIN SPEC 91345: 2016-04 (E) Reference Architecture Model Industry 4.0 (RAMI4.0). Його основна концепція полягає у створенні правил опису інформаційних даних і параметрів технічного об'єкта в формі еталонної

моделі архітектури Industry 4.0 (RAMI4.0), яка дає представлення даного технічного об'єкта з усіма відповідними аспектами, від його створення до виробництва і використання аж до його утилізації. Це є компонент Industry 4.0, який фактично представляє її як об'єднання фізичного та кібернетичного світу з впровадженням технології їх комунікації між собою [85,86].

IEC PAS 63088 діє до: 2017 (E) Smart manufacturing-Reference Architecture Model Industry 4.0 (RAMI4.0) описує модель еталонної архітектури у формі моделі кубічного рівня, яка показує технічні об'єкти (активи) в формі шарів і дозволяє їх описувати, відстежувати протягом усього терміну їх служби (або «vita»), що віднесено до технічної та / або організаційної ієрархії. У ньому також описуються структура і функції компонентів Industry 4.0 як найважливіших частин віртуального і фізичного представлення [87].

Аналізуючи DIN SPEC 16593-1: 2018-04 (E) RM-SA – Reference Model for Industry 4.0 Service Architectures – Part 1: Basic Concepts of an Interaction-based Architecture можна виділити основні вимоги до загальних концепцій і процедур для специфікації технологій, орієнтованих на обслуговування еталонних архітектур Industry 4.0 на базі концепції IoT і Internet-of-Services (IoS) [88].

PD IEC / TS 62832-1 діє до: 2016 Industrial-process measurement, control and automation. Digital factory framework. General principles описує основні концепції структури Smart Manufacturing [89].

IEC 62264-1 діє до: 2013 Enterprise-control system integration – Part 1: Models and terminology – описує область управління виробничими операціями (рівень 3) і її дії, а також вміст інтерфейсу і пов'язані транзакції на рівні 3 і між рівнем 3 і рівнем 4. Цей опис забезпечує інтеграцію між виробничими операціями і областю управління (рівні 3, 2, 1) і домен підприємства (рівень 4). Його метою є підвищення одноманітності і узгодженості термінології інтерфейсу і зниження ризику, вартості і помилок,

пов'язаних з реалізацією цих інтерфейсів [90].

IEC 62541-100 діє до: 2015 OPC Unified Architecture – Part 100: Device Interface є розширенням всієї серії стандартів OPC Unified Architecture і визначає інформаційну модель, пов'язану з пристроями. Запропоновані три моделі, які базуються одна на одній:

– (базова) модель пристрою, призначена для забезпечення єдиного подання пристроїв;

– модель зв'язку пристрою, яка додає інформаційні елементи мережі і з'єднання для створення топології зв'язку;

– нарешті, модель хоста інтеграції пристроїв, яка додає додаткові елементи і правила, необхідні хост-системам для управління інтеграцією всієї системи.

Це дає змогу побачити топологію системи автоматизації з пристроями, а також з мережами зв'язку [91]. На рис. 1.3 наведені 28 міжнародних стандартів, які за допомогою семантичних технологій підтримують сумісність концепції Industry 4.0. В даному дослідженні будуть використовуватися не всі семантичні зв'язки, а тільки ті, які необхідні і достатні для досягнення мети, на базі якої будуть сформульовані і формалізовані основні принципи автоматизації процесів управління і модифікації CPPS.

Дане рішення дозволить обґрунтувати і довести правильність положень у відповідності до концепцій міжнародних стандартів, а також інтегрувати усі запропоновані науково-теоретичні та практичні рішення в рамках загального підходу до автоматизації процесів управління розробкою CPPS, що є невід'ємною частиною в структурі Smart Manufacturing.

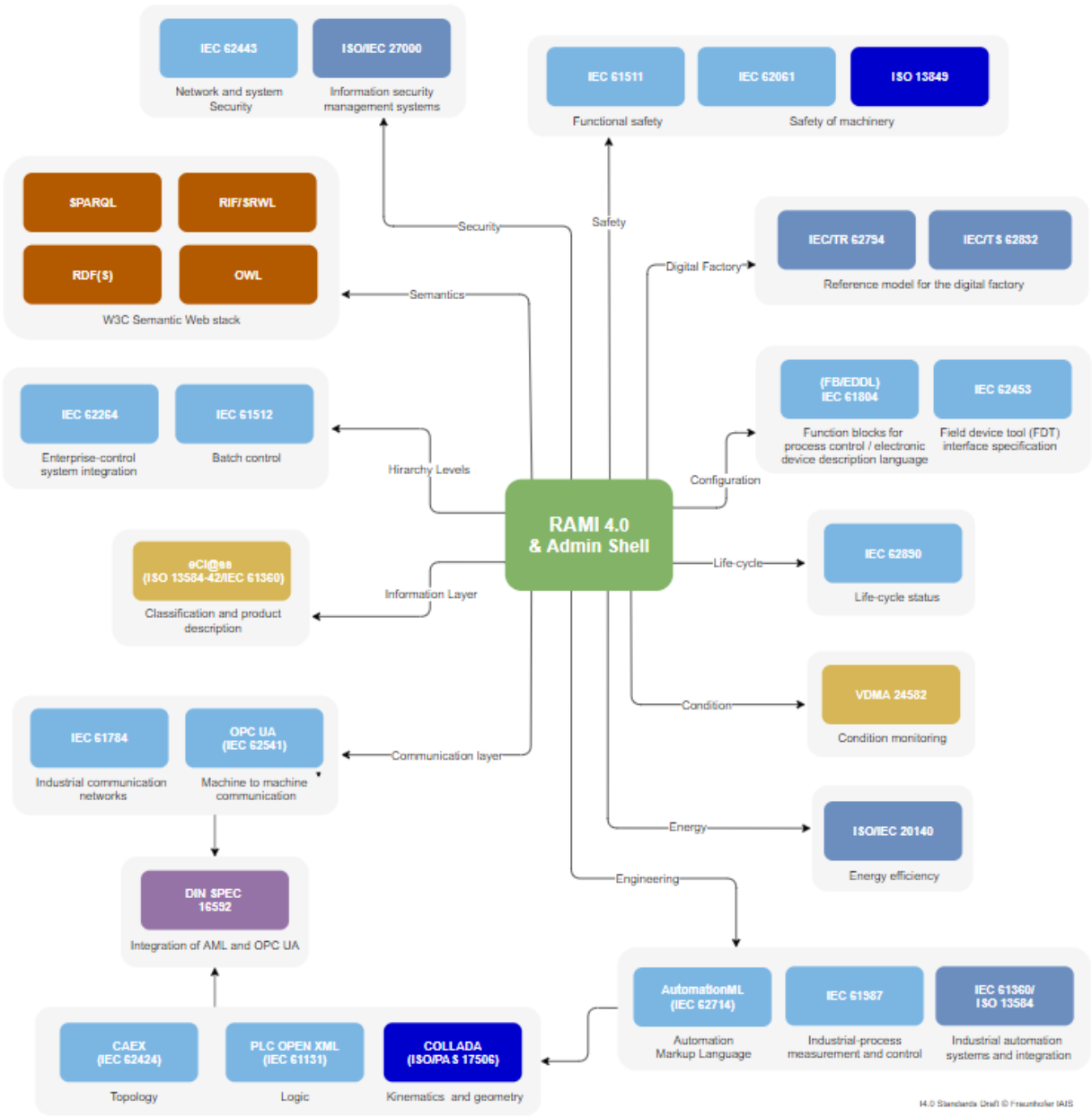


Рисунок 1.3 – Схема сумісність стандартів Industry 4.0 за допомогою семантичних технологій [92]

1.2 Кіберфізичні системи і кіберфізичні виробничі системи в Industry 4.0

Кіберфізичні системи (CPPS) – це складні системи з глибокою інтеграцією і взаємодією обчислювальних, комунікаційних керуючих технологій. З урахуванням теорії і технології існуючих кіберфізичних систем, розробка та управління CPPS є складною науково-технічною і

дослідницькою задачею [93]. Майбутні промислові системи можуть бути реалізовані з використанням CPPS, які об'єднують кібернетичні і фізичні складові в єдиному інформаційному просторі за допомогою мережевої структури для спільного виконання заданих функцій, незалежно від області їх застосування [94, 95].

У [96] визначено, що основними факторами розвитку і впровадження CPPS є скорочення витрат і часу процесу виготовлення виробів. Це також стосується аналізу типів систем і пов'язаних з ними процесів переходу від мехатроніки до CPPS і IoT систем. Далі автори [96] розглядають вимоги, які роблять методології для CPPS-проєктування частиною міждисциплінарного процесу розробки та управління, в рамках якого розробники повинні зосередитися не тільки на окремих фізичних і обчислювальних компонентах, але також на їх інтеграції та взаємодії.

Однією з найбільших проблем в розробці кіберфізичних систем є їх внутрішня складність, неоднорідність і міждисциплінарний характер. Нові розподілені CPPS об'єднують широкий спектр різнорідних аспектів, таких як фізична динаміка, управління, машинне навчання і обробка помилок. Крім того, системні компоненти часто розподілені по декількох фізичних місцях розташування, апаратних платформ і мереж зв'язку [97, 98].

В [99] розглядається проблема кіберфізичних систем з точки зору безперервного генерування великих обсягів даних, який вимагає обробки і візуалізації, що дає можливість підвищити масштабованість, безпеку і ефективність CPPS з метою досягнення повної автономії в рамках технології Industry 4.0 [99].

По суті, новий термін «кіберфізичні системи» представляє собою архітектурну парадигму, в якій технології всеосяжного зондування є фундаментальною частиною. Започаткований в області комп'ютерних наук термін «кіберфізичні системи» був адаптований до дуже різних галузей, таких як теорія управління або електронна інженерія. Навіть деякі автори [100] розуміють CPPS як особливий сценарій Інтернету речей (IoT),

заснований на всепроникному зондуванні та пропонується визначення CPPS, включаючи всі функції, описані в різних областях.

Запропоноване визначення CPPS, в рамках Industry 4.0, відповідає опису запропонованими міжнародними грантами:

– Trade and Invest «Industry 4.0» віднесено до технологічної еволюції від вбудованих систем до кіберфізичних систем [101];

– McKinsey «Industry 4.0» – це наступний етап оцифрування виробничого сектора [102];

– Boston Consulting Group «Industry 4.0» відноситься до конвергенції і застосування дев'яти цифрових технологій: передової робототехніки, адитивного виробництва, доповненої реальності, моделювання, горизонтальної / вертикальної інтеграції, IIoT, хмарних даних, кібербезпеки, Big Data і аналітики [103].

Виходячи з цього можна з упевненістю стверджувати, що дані системи є унікальними і їх математичне, інформаційне, функціональне і алгоритмічне забезпечення напряму залежать від специфіки виробництва, обладнання та вимог, які пред'являються Smart Manufacturing.

Багато авторів такі як Pericles Loucopoulos, Evangelia Kavakli, Natalia Chechina [104]; O. Cardin [105], S. Vijayakumar, N. Dhasarathan, P. Devabalan, C. Jehan [106]; HermannMeissner, Jan C. Auricha [107]; Fazel Ansari, Robert Glawar, Tanja Nemeth [108]; Luis Alberto Cruz Salazar, Daria Ryashentseva, Arndt Lüder, Birgit Vogel-Heuser [109]; Pericles Loucpoulos, Evangelia Kavakli, Natalia Chechina [110]; Dawn M. Tilbury [111]; Sven Hoffmann, Aparecido Fabiano Pinatti de Carvalho, Darwin Abele, Marcus Schweitzer, Peter Tolmie, Volker Wulf [112] розкривають питання і завдання пов'язані з використанням CPPS в рамках Smart Manufacturing і визначають термін кіберфізичних виробничих систем.

Зростаюча складність виробничих систем вимагає відповідних архітектур управління, які забезпечують гнучку адаптацію під час їх роботи.

CPPS надають засоби для подолання складності і гнучкості, але

інтеграція даних складових з існуючими системами управління все ще залишається проблемою. Термін CPPS визначає мехатронні системи (фізичний світ) в поєднанні з програмними об'єктами і цифровою інформацією (кібер частина) та дозволяє реалізувати концепцію Smart Manufacturing для парадигми Industry 4.0 (I4.0). Але все ж таки дослідження проведені у [104-112], в основному, носять концептуальний характер або знаходяться на ранній стадії, де даються пропозиції щодо еталонної архітектури.

Внаслідок чого, для досягнення практичної реалізації CPPS потрібна систематична методологія процесу управління, збору, обробки та застосування даних для CPPS, що розробляється. Це пов'язано з тим, що CPPS може бути успішно реалізована тільки тоді, коли розроблені всі критерії обробки і методи застосування для різноманітних даних, визначена послідовність процесу розробки з урахуванням їх існування в реальному часі через характер виробничих процесів. Різні технології та системи були розроблені для збору необроблених даних цеху, але вони в основному спрямовані на автоматизацію виробництва.

Таким чином, роблячи висновок з аналізу публікацій, існує складна науково-дослідна задача систематизації процесів управління кіберфізичними виробничими системами, яка буде служити ядром для створення систем автоматизації процесів управління складними CPPS для різних сфер виробничої діяльності людини [113-117].

В рамках даних досліджень пропонується прийняти наступне визначення поняття CPPS – це інформаційно-технологічна концепція, що передбачає інтеграцію обчислювальних ресурсів і фізичних сутностей будь-якого виду.

У кіберфізичних системах обчислювальна компонента розподілена по всій фізичній системі, яка є її носієм, і синергетично ув'язана з її складовими елементами. Ґрунтуючись на даному визначенні можна зробити висновок що CPPS – це складна багаторівнева жорстко ієрархічна система, яка об'єднує в

собі всі необхідні і достатні потоки інформації від апаратної складової (фізичної, мехартонної системи) до верхнього рівня візуалізації, аналізу і прийняття рішення (кіберсистеми).

Один з підходів використання теорії розподіленого управління є підхід, який базується на багатоагентних системах (MAS) [118], які використовується для вирішення завдання управління CPPS.

Підходи MAS класифіковані для полегшення переходу від традиційних систем автоматизації до CPPS. Тому автори [118-119] використовують шаблон, який складається з переліку критеріїв класифікації, затверджених експертами німецького співтовариства FA 5.15.

Оскільки всі підходи були створені для використання в різних областях і на різних рівнях піраміди автоматизації, їх значна частина сконцентрована для забезпечення гнучкості або мінливості (Flexibility / Changeability (FC)) системи. Інші концентруються на інших функціях, таких як: надійність (Reliability (RL)), адаптивність або спритність (Adaptability / Agility (AA)), Реконфігурація (Reconfigurability (RC)) і надійність (Dependability (DP)). Опис характеристик MAS для CPPS представлені в таблиці 1.1.

Таблиця 1.1 – Основні характеристики MAS для CPPS

Характеристика	Опис
1	2
Flexibility/changeability (FC) [120,121]	Ступінь, з якою система може використовуватися з ефективністю, дієвістю, свободою від ризику і задоволенням в ситуаціях, які виходять за рамки початково зазначених вимог
Reliability (RL) [121]	Набір атрибутів, які впливають на здатність кібер складової CPPS підтримувати свій рівень продуктивності в зазначених умовах протягом зазначеного періоду часу (чотири атрибути: зрілість, відмовостійкість, можливість відновлення, відповідність надійності)

Продовження табл. 1.1

1	2
Reconfigurability (RC) [122]	Атрибут швидкої зміни структури, компонентів фізичної та кібернетичної складових, для швидкого налаштування виробничих потужностей і функціональності в залежності від раптових змін вимог
Adaptability/agility (AA) [120]	Ступінь здатності «виживати» в конкурентному середовищі з безперервними і непередбачуваними змінами і ефективно реагувати на мінливі вимоги розроблені для клієнтів
Dependability (DP) [122]	Набір незалежних атрибутів виробничих подій (ES), який повністю визначає доступні процеси у виробничій системі

За результатами порівняння існуючих системних архітектур і їх основних напрямків, стосовно конкретних вимог до CPPS і RAMI 4.0 (табл. 1.2), а також порівняння структури підходів CPPS щодо класів за типами механізмів управління і прийняття рішень, можна зробити висновки, що майже всі архітектури концентруються на забезпеченні гнучкості систем автоматизації.

Для даного напрямку CPPS, згідно [123-129], існує п'ять ключових вимог:

- незалежність додатків (вимога 1.1), що означає, що MAS, його протоколи і повідомлення повинні бути незалежними від конкретного додатка;

- незалежність від рівня (вимога 1.2), яка вказує, що всі рівні автоматизації для ISA-95.00.01-2000 [130] доступні в залежності від сценаріїв, в яких буде застосовуватися CPPS;

- незалежність від платформи реалізація (вимога 1.3), що має на увазі, що модулі легко інтегруються з незалежної реалізацією (відкриті технології);

Таблиця 1.2 – Порівняння існуючих системних архітектур і їх основних напрямків до CPPS і RAMI 4.0

Автор(и)	Характерні переваги					Класифікація		Вимоги CPPS					Вимоги RAMI 4.0				
	FC	RL	RC	AA	DP	Сфера	Тип	1.1	1.2	1.3	1.4	1.5	2.1	2.2	2.3	2.4	2.5
S. Cruz [123]	+	–	+	+	–	CPPS архітектура	II	+	+	+	+	+	–	–	+	–	–
J. Fischer [174]	–	–	–	–	+	MFS на основі агентів	III	+	+	+	+	–	–	–	+	–	–
A. Lüder [125]	+	+	–	–	–	MAS для промисловості	III	+	+	+	–	–	–	–	+	–	–
S. Rehberger [126]	+	–	–	–	+	MAS для промисловості	III	+	+	+	+	–	–	–	+	–	–
L. Ribeiro [127]	+	+	+	+	–	CPPS архітектура	III	+	+	+	+	+	+	+	+	+	–

Примітка:

"+" – може застосовуватись;

"–" – не застосовують або описаний дуже слабо;

Тип II – напівгетерархічна система управління (наприклад, архітектура ADACOR);

Тип III – повністю гетерархічна система управління (наприклад, архітектура D-MAS).

- стійкість до помилок (вимога 1.4) означає, що MAS повинні реагувати на несправності та динамічні умови відповідним чином, тобто вони повинні бути стійким до непередбачених обставин;

- децентралізація (вимога 1.5) означає, що MAS повинен мати справу з тимчасовою втратою мережевого з'єднання, і критичні дані повинні бути розподілені між кількома вузлами.

Що стосується моделі RAMI 4.0, існують інші п'ять найважливіших вимог до концепції компонентів Industry 4.0 [87, 88, 131].

Підмоделі (вимога 2.1) повинні підтримувати різні інженерні дисципліни.

Границя системи (вимога 2.2) має на увазі, що підмодель описує відносини між рівнями RAMI 4.0.

Принцип вкладеності (вимога 2.3) для конкретної підмоделі повинін існувати свої власні принципи організації для відповідних ресурсів (активи в ієрархічних вимірах).

Віртуальне подання (вимога 2.4), адміністративна оболонка, може позначати цифровий актив з його частинами.

Нарешті, функціональні властивості (вимога 2.5) вимагають, щоб маніфест мав доступний ззовні набір метамоделей, що описують його функціональні і нефункціональні властивості.

Аналізуючи таблицю 1.2 можна виділити [127], в якій запропонована адаптована структурна модель CPPS на базі структурної моделі [132], яка представлена на рисунку 1.4.

Варто зауважити, що запропонована структурна модель CPPS є повністю гетерархічною, всі елементи знаходяться в різноманітних, але рівноцінних зв'язках, тому не існує переважаючого способу їх структурування. Будь-яка структура гетерархії сприймається розробником як неповне, супроводжується існуванням суперечливості, отже, неприйнятним для вирішення завдання автоматизації процесів управління складними CPPS.

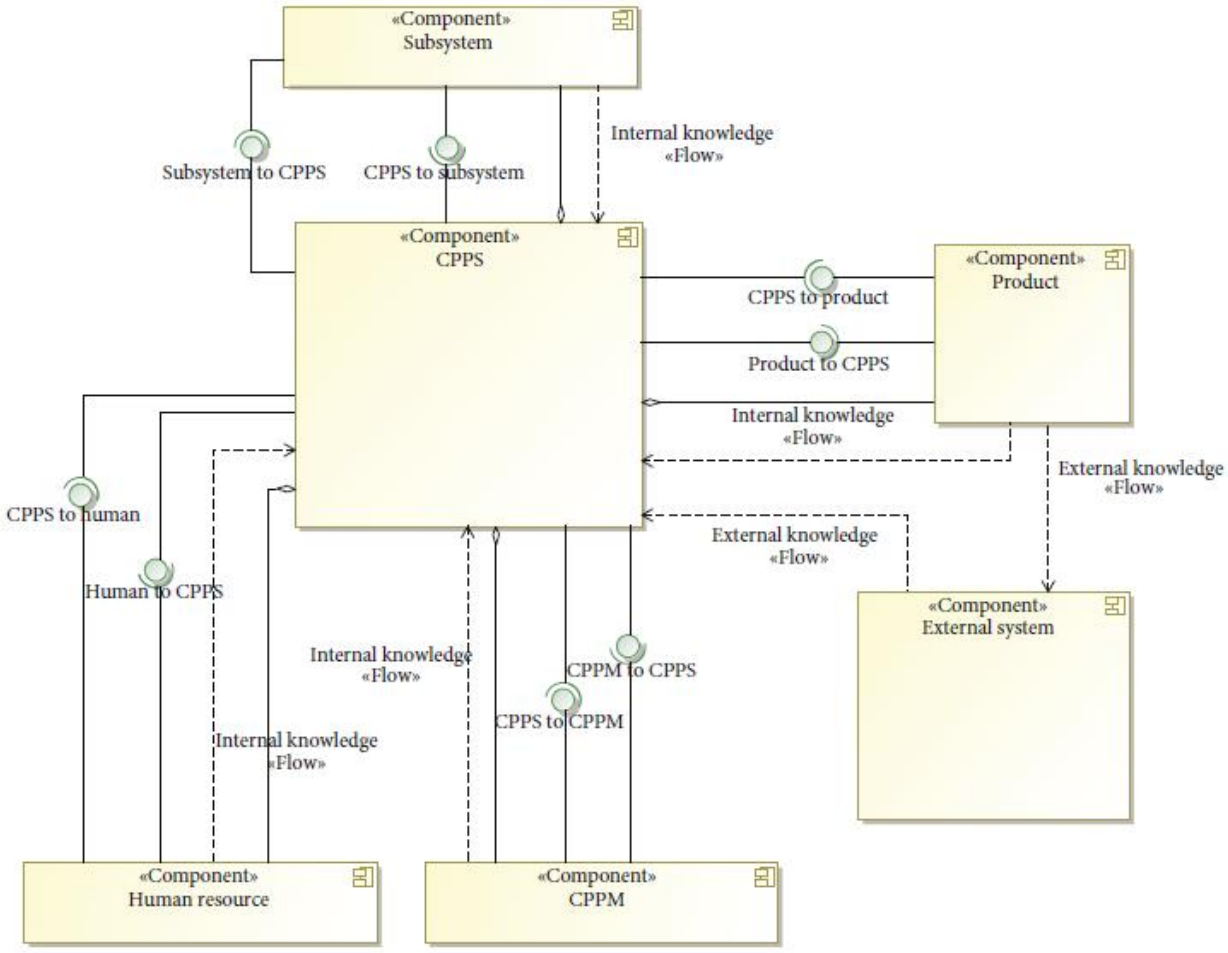


Рисунок 1.4 – Структурна модель CPPS [127]

А як видно з таблиці 1.2, не існує системних архітектур, які б мали доступний набір метамоделей, що описують його функціональні і нефункціональні властивості, що показує, що існуючі рішення мають суто декларативний опис і рекомендаційний характер.

Ґрунтуючись на результатах аналізу можна зробити висновок, що запропонована структурна модель демонструє зв'язки, але не показує послідовність, правила процесу управління складними CPPS. Виходячи з цього, в даному дослідженні прийнято рішення використовувати ієрархічний підхід до процесу управління CPPS в трьох концепціях «згори вниз», «знизу вгору» і «зліва направо», що дозволить на кожному етапі процесу управління виділити базові елементи і оцінити прогресію управління складною CPPPS.

1.3 Industry 4.0 і кіберфізичні виробничі системи як засіб досягнення мети Lean Production

Основними ключовими технологіями Industry 4.0 в рамках Smart Manufacturing є застосування CPPS, які являють собою результат замкнутого циклу збору даних про фізичні процеси на основі датчиків в поєднанні з програмною (кібер) обробкою даних і автономним виконавчим механізмом управління технологічними процесами, їх візуалізації для прийняття рішення. На основі елементів CPPS реалізуються такі зв'язки: збору даних, обробки даних, міжмашинні зв'язки і взаємодія людини з машиною, що дає можливість реалізації децентралізованого автономного управління виробництвом.

Щоб впровадити технології Industry 4.0 в структуру виробництва, засновану на елементах CPPS, пропонується згрупувати їх у 3 кластери:

– збір і обробка даних;

– M2M;

– HMI.

До кластеру збору і обробки даних віднесено об'єднання апаратних датчиків і виконавчих механізмів для взаємодії з фізичним світом. Це у поєднанні з підходами технології хмарних обчислень дає можливість застосовувати інтелектуальні об'єкти CPPS, оснащені мікроелектронікою, датчиками, модулями зв'язку і обробки. В результаті продукти, ресурси, машини і обладнання набувають форму базового інтелекту. IIoT створює середовище для з'єднання таких інтелектуальних об'єктів в єдиний інформаційний простір. Всі отримані дані технологічного процесу на основі інтелектуальних об'єктів будуть зберігатися на великих платформах даних в якості бази даних для аналітичних додатків. Аналітичні додатки, що залежать безпосередньо від датчиків і виконавчих механізмів, генерують дані інтелектуальних пристроїв і інтелектуальних машин.

Ці дані дозволяють аналізувати величезну кількість статистичних

даних технологічних процесів, щоб визначити нестабільні параметри і уникнути зниження якості в межах встановленого діапазону [133-137]. Традиційні дерева key performance indicator (KPI) використовуються для управління, контролю і вимірювання ефективності виробничого процесу на різних рівнях CPPS [138-140]. Збір і розрахунок значень у традиційний спосіб займають багато часу, що повністю суперечить принципам Smart Manufacturing, в якому ключовим параметром є можливість обробки і прийняття рішень в режимі реального часу. Дане протиріччя вирішується за допомогою використання різних засобів автоматизації (PLC, SCADA) в залежності від рівня, на якому вони застосовуються у вертикальній структурі CPPS.

Кластер M2M дає можливість забезпечити автоадаптивне управління взаємопов'язаними машинами та обладнанням без участі людини [141, 142]. Дана концепція поєднує в собі підхід вертикальної і горизонтальної інтеграції. Вертикальна інтеграція з'єднує машини і дані на різних рівнях, що дозволяє створити нерозривний зв'язок даних машинних процесів на фізичному рівні з MES і ERP. Дані з ERP-системи містять інформацію про параметри виробничого процесу кожного окремого продукту. Вертикальна інтеграція без зазорів дозволяє здійснювати індивідуальний потік без ручного перемикання. Підхід горизонтальної інтеграції визначає глобальний зв'язок між обладнанням на одному рівні. На підставі цієї інформації виробничий процес може бути змінений автономно відповідно до автоадаптивного плану виробництва [143].

Кластер HMI реалізує підхід промислової взаємодії людини з машиною, що базується на обміні інформацією і спільну роботу виробничого устаткування і співробітників за допомогою графічних інтерфейсів. Всі HMI засновані на даних, отриманих в реальному часі на рівні виробничих процесів фізичної складової CPPS. Ця концепція демонструє рішення, що дозволяє аналізувати велику кількість даних з датчиків в поєднанні з можливістю екстреного втручання людини в протікаючі технологічні

процеси з метою його корекції для досягнення необхідних виробничих параметрів [144].

В [145] проводитися аналіз House of Lean Production, запропонованого Toyota Production System (TPS), який є символом принципів Lean Production (LP). Ґрунтуючись на структурі House of Lean Production і його параметрах, автори провели аналіз впливу концепції Industry 4.0 і CPPS для підвищення LP. У таблиці 1.3 наведена матриця оцінки впливу Industry 4.0 і кластерів CPPS, описаних вище, на параметри LP. Код оцінки «+» означає, що дана технологія Industry 4.0 і CPPS може надати незначний позитивний вплив на цей принцип LP. Два показники «++» показують високу оцінку впливу, а три показники «+++» означають максимально можливий вплив технології на відповідний принцип LP.

Матриця оцінки впливу Industry 4.0 і кластерів CPPS на параметри LP показує, що розробка і впровадження CPPS на підприємствах підвищить економічність і дозволить домогтися максимально бережливого виробництва. Запропоновану матрицю оцінки необхідно використовувати на початковому етапі процесу управління CPPS в залежності від головної мети створення або модифікації існуючої CPPS, вимог до неї і обраних параметрів LP, які необхідно досягти, замовник надає в ТЗ.

Таблиця 1.3 – Матриця оцінки впливу Industry 4.0 і кластерів CPPS на параметри LP [145]

Параметри Lean Production [151]	Кластери							
	Збір і обробка даних				M2M		HMI	
	Sensors and Actuators [150]	Cloud Computing [148,149]	Big Data	Analytics	Vertical Integration [147]	Horizontal Integration [147]	Virtual reality	Augmented Reality [146]
5S	+	+	+	+	+	+	++	+++
Kaizen	+	++	+++	+++	+++	+++	+++	+++
Just-in-Time	++	++	+++	+++	+++	++	+	++
Jiboka	+	+++	+++	+++	++	++	+	+
Heijunka	++	++	+++	+++	+++	++	++	+
Standardisation	+	+++	+++	+++	++	++	+++	+++
Takt time	+	+	+++	+++	+++	+++	+	+
Pull flow	++	+	+	+	+++	+++	+	+
Man-machine separation	+	+	+	+	+	+	+++	+++
People and teamwork	+	+	+	+	+	+	+++	+++
Waste reduction	+	+	++	+++	+++	+++	+	+

1.4 Еталонна архітектура розробки кіберфізичних виробничих систем в моделях RAMI 4.0 (DIN SPEC 91345) і ISO-95

В рамках форуму ЄС «Оцифрування європейської промисловості» [87] запропонована еталонна архітектурна модель розробки CPPS (RAMI 4.0). Вона є тривимірною моделлю, яка описує простір Industry 4.0. Дана архітектурна модель представлена на рисунку 1.5.

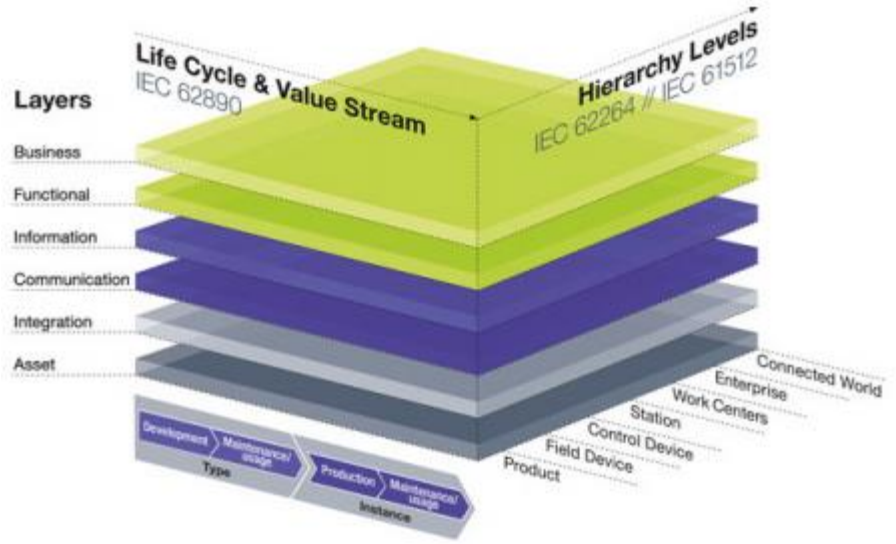


Рисунок 1.5 –Архітектурна модель Industry 4.0 (RAMI 4.0) [152,153]

Модель RAMI 4.0 складається з шести шарів по вертикальній осі, і двох на горизонтальній. Перший шар на вертикальній осі має назву «Asset layer», він показує фізичні об'єкти, такі як деталі, документи, архіви, діаграми, люди і т. д.

На один рівень вище «Integration Layer», де представляється перетворення і з'єднання фізичних об'єктів в кібер об'єкти. Компоненти «Asset layer» пов'язані з кібер об'єктами на шарі «Integration Layer» на базі обробки інформації, виконує роль сполучної ланки між фізичним і цифровим світом. Цей рівень включає комп'ютерний контроль процесу, системні драйвери, пристрої HMI, комутатори, концентратори тощо.

Наступний рівень – це «Communication layer», який забезпечує стандартизований зв'язок між «Integration Layer» і «Information layer».

Стандартизація досягається за допомогою єдиного формату даних, який використовується на «Information layer» та забезпечує управління «Integration layer». «Information layer» зберігає дані в упорядкованому ієрархічному порядку і основною метою цього шару є надання інформації про обладнання та компоненти, які використовуються для створення виробу. «Information layer» базується на програмному забезпечення (форма додатка, даних, рисунків або файлів). У цьому шарі відбувається перетворення події у вигляді набору даних для більш високих рівнів.

Наступний шар на вертикальній осі «Functional layer», який відповідає за виробничі правила, дії, обробку, систему, контроль та моніторинг. Крім того, він включає в себе різні інші види діяльності, такі як координація компонентів виробничого процесу, вмикання / вимикання елементів системи, елементи управління. «Functional layer» реалізує концепцію віддаленого доступу і горизонтальну інтеграцію.

«Business layer» складається з бізнес-стратегії, бізнес-середовища, бізнес-цілей і управління взаємовідносинами, бюджетом і моделлю ціноутворення [154, 155].

На рис. 1.5 праворуч показана друга горизонтальна вісь, яка представляє собою шар ієрархії, рівень якої заснований на міжнародних стандартах IEC 62264-1: 2013 Enterprise-control system integration - Part 1: Models and terminology [157] і IEC 61512-3: 2008 Batch control Part 3: General and site recipe models and representation [158].

Крім чотирьох шарів: «Enterprise», «Work Centers», «Station» і «Control Device», існують два шари «Field Device» і «Product», які не включені у вище перелічені міжнародні стандарти, але є основними для розробки методології процесу розробки CPPS .

Шар «Field Device» дозволяє врахувати управління обладнанням або групою обладнання (при горизонтальній архітектурі) за допомогою інтелектуальних датчиків і виконавчих механізмів.

Шар «Product» забезпечує взаємозв'язок виробу і виробничих

потужностей підприємства.

Верхній шар «Connected World» забезпечує можливість зв'язку із зовнішніми партнерами через сервісні мережі на базі технології IoT [159-161].

Горизонтальна вісь (на лівій стороні рис. 1.5) показує життєвий цикл і значення потоків в процесі промислового виробництва (рис. 1.6). Вона складається з двох етапів: «Type» і «Instance».

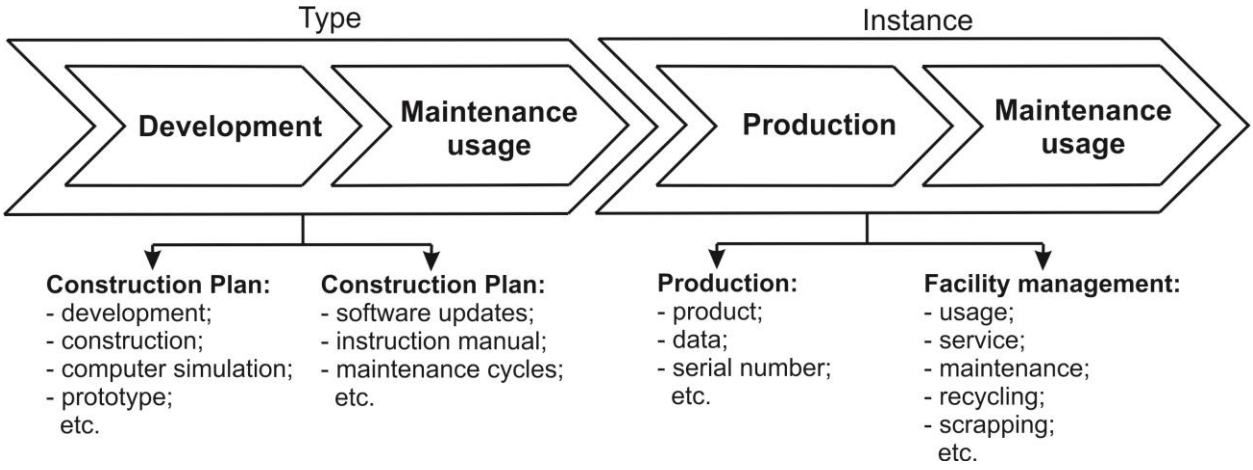


Рисунок 1.6 – Життєвий цикл і значення потоків в процесі промислового виробництва [156]

Коли виріб знаходиться на стадії розробки, то знаходиться в фазі «Type», а коли продукт знаходиться у виробництві, він знаходиться в фазі «Instance». Кожен раз, коли один і той же виріб переміщується в стадію розробки то він переміщується в фазу «Type», цей цикл може повторитися кілька разів [154, 156].

На підставі проведеного детального аналізу еталонної моделі RAMI 4.0, яка включає в себе всі елементи вертикальної та горизонтальної архітектури, в рамках концепції Industry 4.0, можна виділити наступні недоліки:

- не існує точного визначення щодо того, яким чином окремі елементи всередині кожного шару з'єднані один з одним і з елементами, які знаходяться на шарах вище і нижче даного;

- в еталонній архітектурній моделі RAMI 4.0 не описані шари «Field Device» і «Product», не визначені елементи і типи зв'язків між ними, їх взаємодія з елементами на шарі «Control Device».

Внаслідок чого, процес управління CPPS, в рамках Smart Manufacturing, на базі архітектурної моделі RAMI 4.0 показує декларативний концепт, а не конкретні рішення і дії у вигляді методологій, моделей і методів, послідовності процесу управління CPPS в залежності від мети, вимог замовника, парку обладнання ділянок, цехів підприємства і т.д.

В [162-166] проводитися аналіз існуючих архітектур Smart Manufacturing в рамках концепції Industry 4.0. Автори досліджують базову архітектуру ISA-95 (S95), запропоновану Американським національним інститутом стандартів (ANSI), для спрощення процесу системної інтеграції підприємства в рамках Smart Manufacturing. Міжнародний стандарт IEC 62264-1: 2013 Enterprise-control system integration визначає базову архітектуру ISA-95 як послідовності рівнів від 0 до 4.

Рівень 0 – це фізичний виробничий процес;

Рівень 1 – визначення і керування виробничим процесом за допомогою датчиків і виконавчих механізмів;

Рівень 2 – призначений для моніторингу, диспетчерського контролю та автоматичного управління виробничим процесом. Можливі об'єкти на цьому рівні: SCADA системи, розподілені системи управління (DCS) і PLC;

Рівень 3 – пов'язаний з робочим процесом і діяльністю з виготовлення виробу. Можливі об'єкти на цьому рівні є: MES, система управління виробничою інформацією (PIMS), система управління складом (WMS) і комп'ютеризована система управління технічним обслуговуванням (CMMS).

Рівень 4 – пов'язаний з управлінням підприємством в цілому. Можливі об'єкти на цьому рівні: планування ресурсів підприємства (ERP), управління життєвим циклом продукту (PLM), управління людськими ресурсами (HRM), управління взаємовідносинами з клієнтами (CRM) і відносини з постачальниками системи управління (SCM). У загальному вигляді

архітектура ISA-95 (S95) представлена на рис. 1.7.

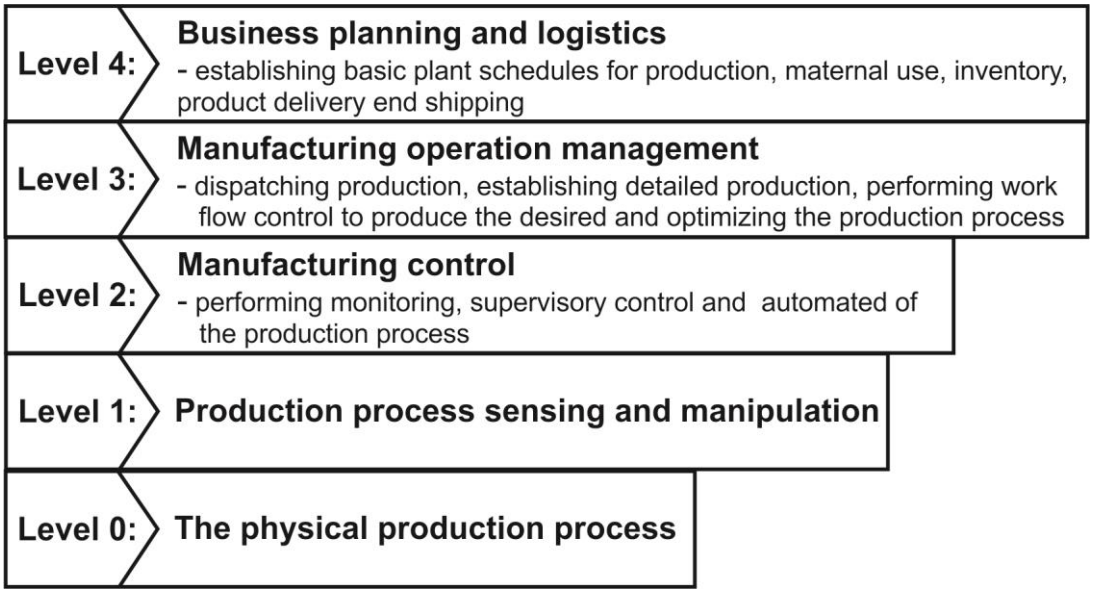


Рисунок 1.7 – Архітектура ISA-95 (S95) [167]

З появою поняття CPPS автори [168-170] запропонували п’ятирівневу архітектуру CPPS 5C, яка складається з рівнів: «Smart Connection», «Data-to-Information Conversion», «Cyber», «Cognition» і «Configuration». Загальний вигляд архітектури CPPS 5C представлений на рис. 1.8.

На першому рівні «Smart Connection» в [167] пропонується використовувати машини і їх компоненти для отримання точних і надійних даних на першому кроці управління CPPS для Smart Manufacturing. Датчики використовуються для збору різних параметрів протікання технологічного процесу виробництва виробів в режимі реального часу. Дані можуть також надходити від PLC, PAC або виробничих систем ERP, MES, SCM і CMM. Технології IIoT, використовуються для здійснення передачі даних і управління за обраним протоколами.

«Data-to-Information Conversion» призначений для перетворення отриманих даних в інформацію. В [167] розглядається цей рівень з позиції, коли деякі пристрої можуть реалізовувати функції прогнозування і моніторингу зносу обладнання , це вносить поняття якостей «інтелекту».

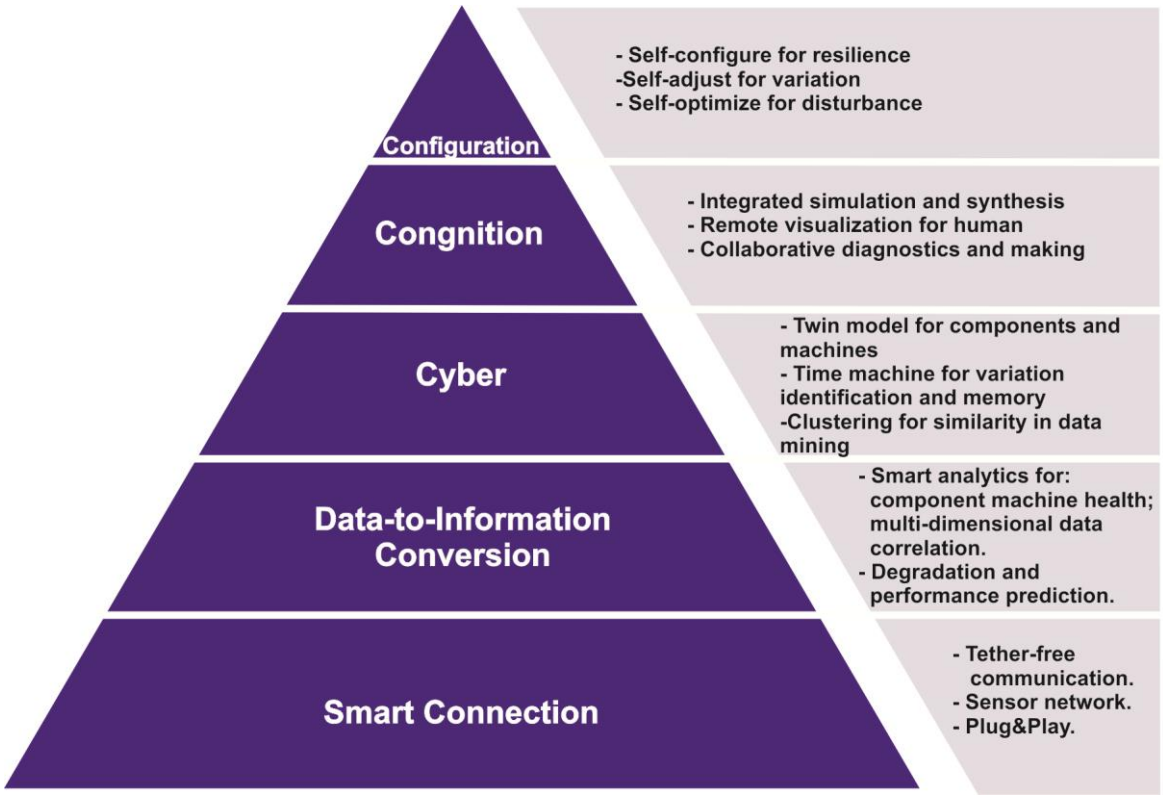


Рисунок 1.8 – Загальний вигляд архітектури CPPS 5C [167]

«Cyber» рівень представляє роль головного інформаційного центру, який збирає масив інформації з обладнання в промислову мережу. На цьому рівні додаткова аналітична інформація витягується з зібраної інформації. За отриманими даними продуктивність одного обладнання порівнюється і ранжується серед всіх однотипних типів обладнання, які находяться в промисловій мережі. Більш того, аналітична інформація про роботу обладнання може бути застосована для прогнозування і планування виробництва для досягнення необхідних заданих параметрів LP.

Рівень «Cognition» призначений для подання аналітичної інформації безпосередньо операторам для прийняття виробничих рішень. Цей рівень уможливлює дистанційну і спільну діагностику і прийняття рішень в умовах виробництва. Пріоритет завдань для процесу обслуговування може бути легко визначеним завдяки наявності порівняльної інформації та індивідуального стану групи або одиничного обладнання.

Рівень «Configuration» повертає зворотний зв'язок від «Cyber» рівня до

«Smart Connection», тобто виконує диспетчерське управління яке дозволяє зробити обладнання самоконфігурованим, самоналаштовуємим і самооптимізуємим. Рівень діє як система контролю стійкості (RCS), щоб застосувати засоби управління для відповідних рішень на рівні контрольованого обладнання.

В роботі [167] розглядається архітектура CPPS 8C, яка утворюється шляхом додавання в архітектуру 5C граней 3C (рис. 1.9).

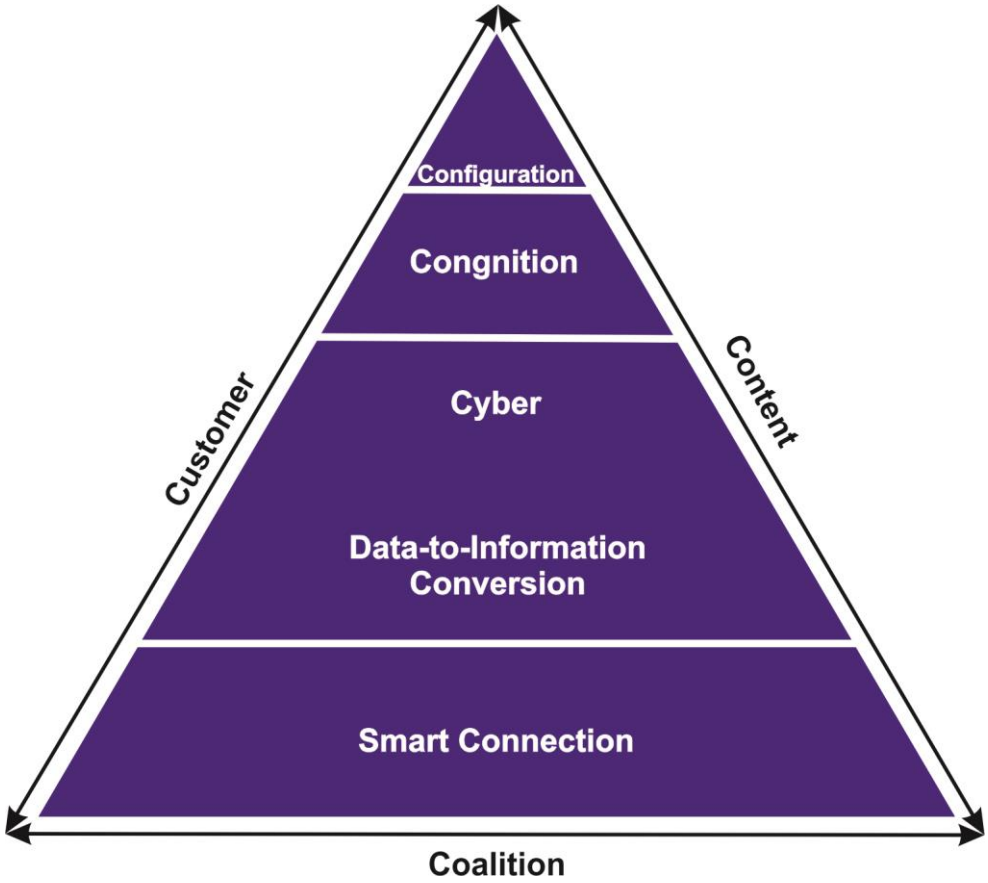


Рисунок 1.9 – Архітектура CPPS 8C запропонована на базі архітектури 5C [167]

Аспекти запровадження граней 3C обґрунтовуються в [167] як інтеграція клієнту і змісту. Автор підкреслює горизонтальну інтеграцію CPPS, такі як інтеграція різних сторін, і пов'язаною з ними інформаційну складову (контент). Також грані дозволяють виділити найважливішу сторону, а саме участь клієнта в процесі виробництва. Проведемо аналіз кожної

введеної грані 3С:

«Coalition» – грань фокусується на інтеграції ланцюжка створення вартості і інтеграції виробничого ланцюжка між різними сторонами, залученими у виробничий процес. Сторони можуть спільно побудувати ланцюжок поставок і планувати виробничі лінії для виробничих потоків. Якщо є будь-які коригування у виробничому процесі, сторони можуть спільно реконструювати їх для досягнення максимального економічного ефекту.

«Customer» – грань, що орієнтована на роль, яку клієнти грають в процесі виробництва і післяпродажне обслуговування виробу. Smart Manufacturing можуть приймати різні замовлення в невеликих кількостях від різних клієнтів і виконувати замовлення вчасно. Клієнти або індивідуальний покупець, можуть брати участь при розробці і навіть змінити технічні характеристики виробу в процесі виробництва.

Це досягається за допомогою концепцією виготовлення виробу. Тобто фабрика може автоматично підготувати матеріал, провести гнучке планування виробничого процесу, динамічно переналаштувати виробничі лінії і організацію зберігання і доставки виробу. Замовники можуть бути повідомлені про виробничі процеси, отримуючи звіти на електронну пошту.

Міркування в аспекті клієнта можуть відповісти на зрушення від звичайної «mass production» парадигми в парадигму «mass customization». Колишня парадигма призначалася для виробництва великої кількості продуктів однієї специфікації, в той час як нова призначена для виробництва різноманітних товарів різних специфікацій. Крім того, нова парадигма зачіпає доставку виробів замовнику, при цьому він може продовжувати отримувати післяпродажне обслуговування, інформацію про експлуатацію та використання виробу.

«Content» – грань спрямована на отримання, зберігання інформації про товари. Вся виробнича інформація, така як постачальники сировини, виробничі процеси, фізичні параметри навколишнього середовища

(наприклад, температура, вологість, вібрація і т.д.), виробничі параметри зберігаються в БД. Всі вироби контролюються на етапах післяпродажного обслуговування, такі як технічне обслуговування, заміна деталей, усунення несправностей, утилізація. Скарги, пропозиції та коментарі користувачів вважаються важливими даними і зберігаються в БД. Положення в [167] для даної грані можуть допомогти досягти повного сервісного обслуговування виробу, аналізуючи всі отримані дані, а не тільки виробничий процес, дозволяє замовнику і виробникові відстежувати тенденції ринку, проводити аналіз і прогнозувати його попит для кожного виробу.

З трьома додатковими гранями, архітектура 8C запропонована в [167] орієнтована на вертикальну і горизонтальну інтеграцію, але це рішення приділяє велику увагу ЖЦ виробу на етапах сервісного обслуговування, що не спрощує розуміння принципів рішення з точки зору поставлених завдань в даній роботі.

Розглядаючи модель RAMI 4.0, в рамках першого і другого вимірів, розробнику необхідно чітке розуміння послідовності ієрархічних рівнів процесу управління CPPS, а дана модель, в загальному вигляді, містить тільки аспекти та рекомендації. Моделі архітектури 5C, запропоновані [166], і 8C, розроблені [167], проаналізовані вище, вони представляють загальну концепцію, парадигму та рекомендації до розробки, не пропонуючи послідовність процесу управління розробкою CPPS. При цьому не вказується і не обґрунтовується початкова точка (стартовий рівень) для створення процесу управління CPPS.

Ґрунтуючись на проведеному аналізі пропонується використовувати модель «Data, Information, Knowledge, Wisdom» (DIKW), яка дає можливість уявити вертикальну інтеграцію процесу управління розробкою CPPS. У роботах [171-176] дана модель застосовується і широко використовується в різних сферах виробничої діяльності людини. На рис. 1.10 представлена інтерпретація моделі DIKW на базі моделей архітектур: RAMI 4.0, 5C, ISA-95 (S95) – для реалізації вертикальної інтеграції ієрархічних рівнів CPPS.

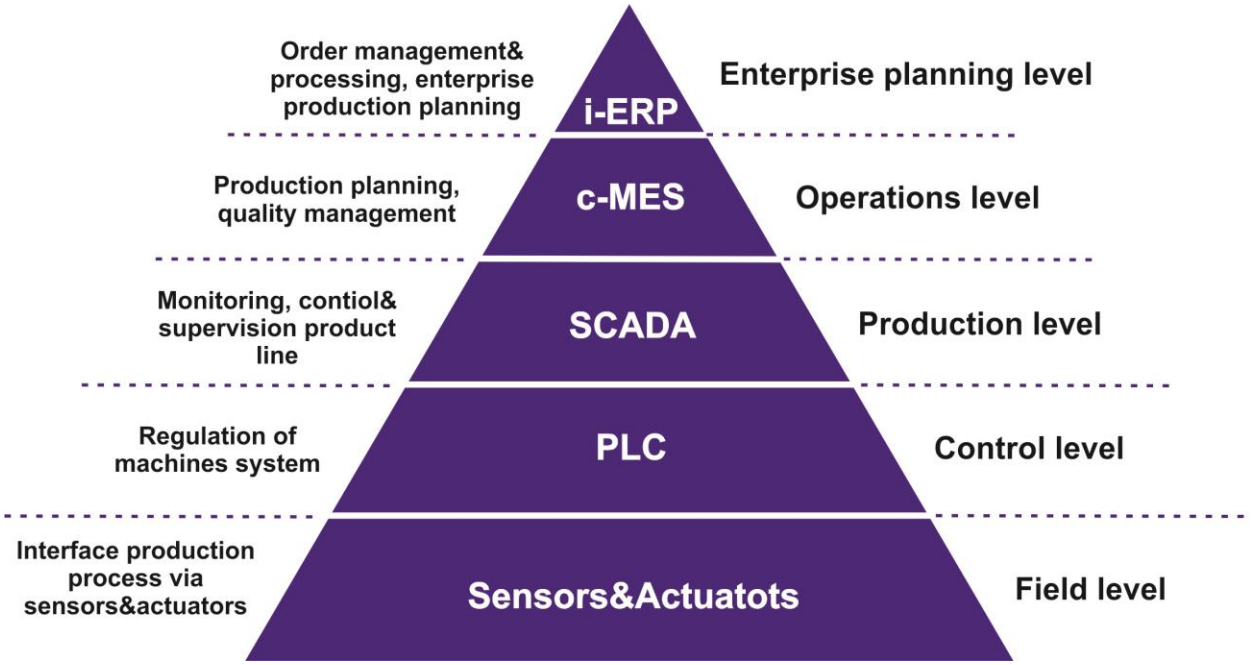


Рисунок 1.10 – Інтерпретація моделі DIKW для вертикальної інтеграції ієрархічних рівнів CPPS

Ґрунтуючись на моделі DIKW інтеграція ієрархічних рівнів CPPS формується за принципом «знизу вгору». На першому польовому рівні (field level) представляються взаємодія з виробничим процесом за допомогою датчиків і виконавчих механізмів; на рівні управління (control level) здійснюється регулювання і управління виконавчими механізмами з використанням програмованих логічних контролерів (PLC); рівень технологічних ліній (production level) (фактично рівень виробничого процесу) виконує функцію контролю і моніторингу технологічного процесу виробництва; рівень операцій (operations level) призначений для планування виробництва, управління якістю, і т.д.; рівень планування підприємства (enterprise level) забезпечує управління замовленнями, їх обробкою, загальне планування виробництва і т.д.

Інтерпретаційна модель DIKW дає можливість провести декомпозицію процесу управління CPPS на ієрархічних рівнях за ключовими точками. Це може використовуватися як початкова базова конфігурація для розробки

методологій і методів управління CPPS на фізичному і кібернетичному рівні.

Ґрунтуючись на запропонованій [166] 5C архітектурі CPPS група вчених у складі: Petar Radanliev, David De Roure, Razvan Nicolescu, Michael Huth запропонували емпіричну архітектурну модель бачення інтеграції CPPS-IoE в архітектури 5C для Industry 4.0, фрагмент якої представлений на рис. 1.11. Автори стверджують, що запропонована модель дозволяє спростити процес щодо інтеграції IIoT в I4.0, а в ширшій перспективі, дозволити спростити процес розробки CPPS (CPPS) в рамках Industry 4.0. Дана модель призначена для підтримки розробки нових національних стратегій Industry 4.0, що дасть можливість поліпшення і переформулювання існуючих концепцій і практичних ініціатив. Архітектурна модель також буде корисна для розробників в області IIoT, які прагнуть поліпшити і розвинути свою виробничу діяльність в концепції Industry 4.0.

При цьому автори звертають увагу, що архітектурна модель інтеграції CPPS-IoE-5C в Industry 4.0, вимагає подальшої перевірки та розмежування, шляхом застосування даної концепції при розробці реальних CPPS [177].

Проводячи аналіз запропонованої архітектурної моделі інтеграції CPPS-IoE-5C в Industry 4.0 можна виділити такі недоліки:

– дана модель архітектури містить синтез усіх стратегій і теорій концепції Industry 4.0, які засновані на процесі каскадування і систематизації академічної літератури, при цьому запропонована модель не має чітко визначеного опису та подання рівнів і етапів процесу управління CPPS, або не визначені ключові точки початку розробки CPPS і набір необхідних вхідних даних;

– модель містить рекомендаційний характер, в якому не прописані інформаційні зв’язки між її елементами, їх зміст і параметри, що обмежує її використання при розробці CPPS.

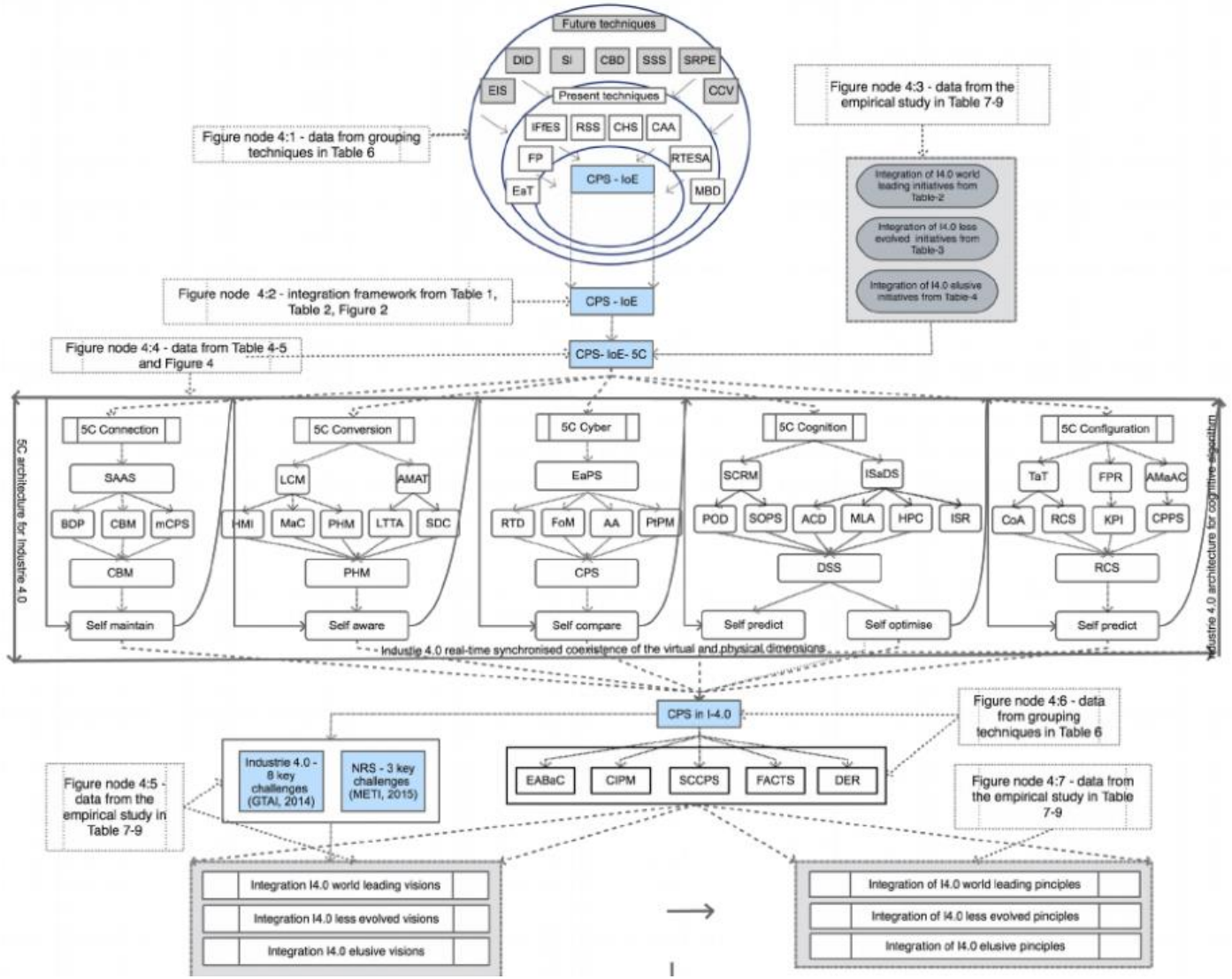


Рисунок 1.11 – Фрагмент емпіричної архітектурної моделі інтеграції CPPS-IoE в архітектурі 5C для Industry 4.0 [177]

1.5 Модель CMMI в розробці кіберфізичних виробничих систем

Один з підходів до розробки систем управління CPPS, запропонована в [178], заснований на моделі Capability Maturity Model Integration (CMMI). Класична методологія враховує в основному окремі вироби або вже існуючі сімейства виробів, де аналізується їх структура на рівні компонентів (фізичному рівні), щоб забезпечити підходи Industry 4.0 до розробки систем управління CPPS. Умовно їх можна розділити на дві групи:

– цілісні підходи (holistic approaches), які прагнуть оцінити і використовувати максимально всі існуючі аспекти Industry 4.0 при розробці та впровадженні CPPS. В роботах [179-186] пропонується багатовимірна концептуальна модель CMMI у вигляді «Industry 4.0 toolbox» як центральний елемент для визначення відповідних областей застосування CPPS;

– специфічні підходи (specific approaches), де зосереджується увага на обмежену кількість аспектів, які стосуються Industry 4.0, і розглядається їх доцільність застосування у вирішенні задач розробки CPPS. Аналізуючи роботи [187-190] можна зробити висновок, що даний підхід показує очевидні обмеження, ізольованість фокуса і функціональних можливостей в межах одного підприємства, нехтуючи взаємозалежностями. В результаті це призводить до зменшення потенціалу від розробки і впровадження CPPS, а також підвищує ризики некоректної розробки архітектури та управління CPPS.

В [178] запропоновано нову модель зрілості, яка базується на наступних основних вимогах:

– операціоналізація абстрактних концепцій Industry 4.0 (кіберфізична система, вертикальна / горизонтальна інтеграція) як практичні проблеми застосування і оцінки цих концепцій, в рамках підприємства, для якого розроблюється CPPS;

– перетворення результатів за термінами, яке включає інтерпретацію розробників / результати оцінки впровадження, внаслідок необхідності

задоволення вимог висунутими в ТЗ на кожному рівні і етапі розробки CPPS.

Ґрунтуючись на вищенаведених вимогах, пропонуються наступні фази реалізації розробки і впровадження CPPS, на базі концептуальної моделі CMMI, представленої на рис. 1.12.

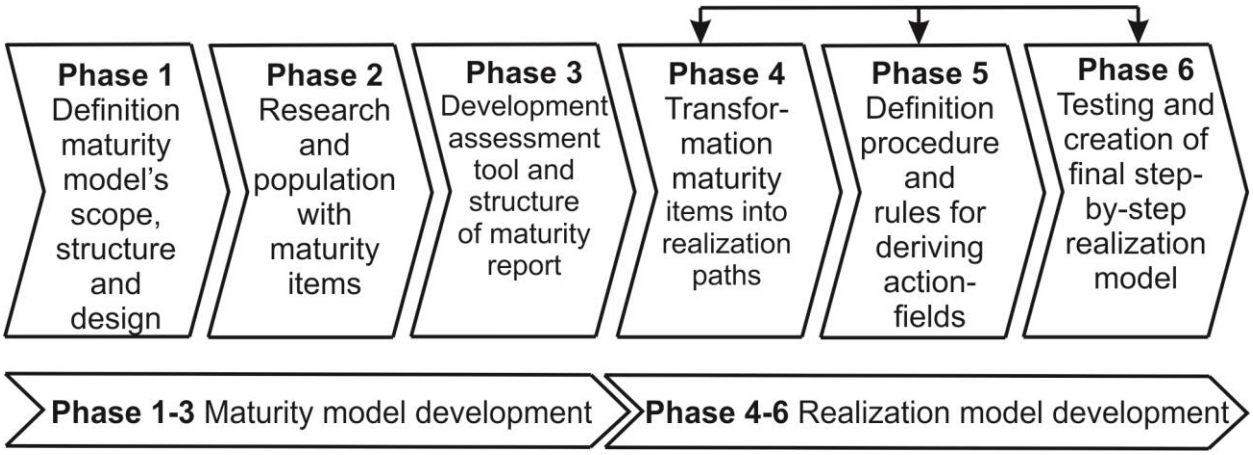


Рисунок 1.12 – Фази реалізації розробки і впровадження CPPS на базі концептуальної моделі CMMI [178]

Як видно з рис.1.12 автор [178] визначає наступні фази:

– фази 1-3 фокусуються на розробці моделі зрілості CPPS (визначення сфери, дослідження структури і дизайну, оцінка обладнання та його структури, вимоги до виробу і його вихідні характеристики);

– фази 4-6 реалізують розробку моделі остаточної реалізації CPPS (визначення процесу управління розробкою, знаходження шляхів, процедур і правил реалізації, розробка, тестування і моделювання отриманої моделі CPPS і її реалізація).

На жаль, модель запропонована в [178] має узагальнений характер концептуальної моделі CMMI з точки зору застосування її для розробки CPPS. У свою чергу автори [191] запропонували розширення моделі CMMI, за допомогою розроблених рекомендацій щодо її поліпшення, відповідно до контекстуальних факторів:

– стратегічні цілі;

– основні процеси;

– ключові показники ефективності.

В роботі [192] представлено дослідження реалізації CPPS реального виробництва, в ході якого розроблена всеосяжна описова модель для подання відповідних систем, їх інтерфейсів, взаємозалежностей, параметрів. Автор приділив увагу необхідності визначення системи цілей (які включають технічні вимоги) на кожному рівні при розробці CPPS. З точки зору даних досліджень пропонується інтерпретувати модель CMMI у вигляді послідовності декомпозиції головної мети розробки CPPS як набір підцілей і завдань для кожного ієрархічного рівня архітектури процесу управління розробкою CPPS, які в сумі складових повинні задовольняти меті, зазначеній в ТЗ на розроблювальну CPPS.

1.6 Дослідження застосування сучасних життєвих циклів розробки програмного забезпечення для реалізації складних кіберфізичних виробничих систем

В даному підрозділі проведено дослідження можливості застосування існуючих моделей ЖЦ розробки програмного забезпечення (ПЗ), з точки зору їх застосування для автоматизації процесів управління розробкою кібер складової CPPS. Структура життєвого циклу ПЗ визначаються і регламентовані в міжнародному стандарті ISO / IEC 12207: 1995 року "Information Technology – Software LifeCycle Processes". Відповідно до цього стандарту всі процеси моделі ЖЦ розробки ПЗ можна представити у вигляді етапів, показаних на рис. 1.13.

Наведемо коротку характеристику кожного етапу:

– системний аналіз – етап, в рамках якого йде визначення ролі кожного елемента, їх взаємодія при виконанні процесу планування проєкту, в ході якого визначається обсяг роботи, ризик, необхідна трудомісткість і вартість;

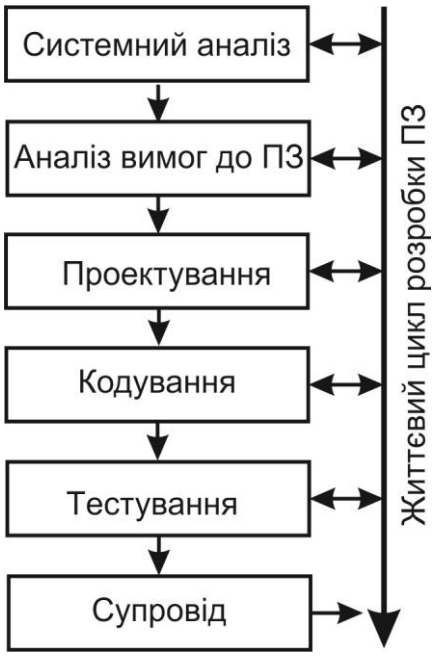


Рисунок 1.13 – Життєвий цикл розробки ПЗ

– аналіз вимог до ПЗ – етап, на якому уточнюються і деталізуються функції, характеристики та створюється прототип інтерфейсу. Необхідно зазначити, що запропоноване ТЗ на початку розробки ПЗ, затверджене замовником і розробником, під час проєктування ПЗ проходить стадію редагування і уточнення, що відповідно збільшує трудомісткість, вартість і складність проєкту. Це підтверджують аналітичні дослідження компанії Standish Group, яка проаналізувала роботу 364 американських корпорацій, а також підсумки виконання 23 тис. проєктів, пов'язаних з розробкою ПЗ, і отримала наступні результати:

– тільки 16,2% проєктів завершені в строк і не перевищили розраховану трудомісткість і вартість, а всі функції, зазначені замовником, були реалізовані;

– 52,7% проєктів завершилися з запізненням, перевищивши при цьому запропоновану в ТЗ трудомісткість, з неповною реалізацією всіх функцій;

– 31,1% - проєкти, які були закриті до завершення, так як вони не відповідали вимогам ТЗ і перевищили ліміт трудомісткості і вартість реалізації ПЗ. В ході аналізу отриманих даних можна зробити висновок, що для проєктів, які завершилися з запізненням або були закриті до завершення,

трудомісткість в середньому була перевищена на 89%, а термін виконання на 122%.

А основною причиною закриття проєктів або перевищення їх терміну проєктування є нечітке і неповне формулювання вимог до програмного забезпечення, допущених на стадії проєктування ТЗ:

– проєктування складається зі створення уявлень про розроблювану архітектуру, модульну структуру, алгоритмічну і функціональну структури ПЗ, структуру даних і розробку інтерфейсу. Однак, в реальних умовах, розробка інтерфейсу користувача і реалізація основних візуальних компонентів програмного забезпечення зазнає ряд змін, відповідно до подання замовника по інформативності та взаємозв'язку робочих областей (робочих вікон, функціональних кнопок і т. д.) ПЗ, а також його специфіки та призначення;

– кодування полягає в перекладі результатів проєктування на мови високого рівня програмування;

– тестування – використання ПЗ для виявлення дефектів у функціях, логіці і в формі реалізації ПЗ;

– супровід – це підтримка у використанні ПЗ, а також зміни в ході виявлення помилок, вдосконалення за вимогами замовника або адаптація ПЗ в залежності від зміни операційних систем (ОС) і т. д.

Досліджуючи можливість застосування моделі ЖЦ розробки ПЗ для рішень завдань управління процесом розробки кібер складової складних CPPS дозволяє зробити наступні висновки:

– етапи системного аналізу та аналізу вимог ЖЦ розробки ПЗ не має сенсу і їх неможливо використати внаслідок того, що системний аналіз і аналіз вимог до майбутньої кібер складової CPPS, що розробляється, визначаються на етапі розробки фізичної складової і залежить від технічних характеристик обладнання, датчиків, виконавчих механізмів, а також вимог до технологічного процесу, що контролюється;

– етап проєктування вже інтегрований в кібер складову CPPS, яка має

«жорстко» виражену ієрархічну архітектуру, прив'язана до процесу управління розробкою CPPS, тобто послідовністю рівнів і етапів, які об'єднані логічною послідовністю, зв'язками і описами алгоритмів функціонування, які обираються в процесі управління розробкою CPPS.

Найбільшого поширення набули такі моделі ЖЦ розробки ПЗ, які запропоновані в стандарті ISO / IEC 12207:

– waterfall model (каскадна модель);

– v-shaped model (V-образна модель);

– prototype model (модель прототипування);

– rapid application development model або RAD-model (модель швидкої розробки додатків);

– incremental model (багатопрохідна модель);

– spiral model (спіральна модель).

Проведемо аналіз існуючих моделей ЖЦ розробки ПЗ для визначення можливості їх застосування в розробці кібер складової складних CPPS.

Waterfall model (каскадна модель) – застосовувалась при розробці однорідних інформаційних систем в 1970-1980 роках, коли ПЗ представляло собою єдине ціле. Основні етапи каскадної моделі представлені на рис. 1.14.

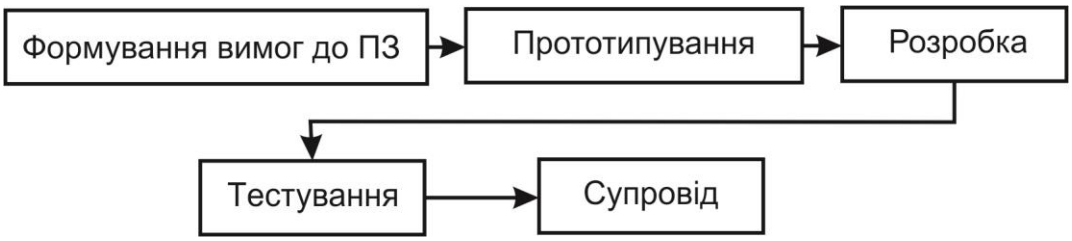


Рисунок 1.14 – Основні етапи каскадної моделі

Принципова особливість каскадної моделі полягає в тому, що перехід між етапами здійснюється тільки після повного завершення робіт на попередньому етапі моделі. Це пов'язано з тим, що результати, отримані при виконанні етапу, є вихідними даними для наступного етапу. При цьому вимоги, що пред'являються в ТЗ на ПЗ, строго документуються і фіксуються

на весь час виконання розробки.

Дана модель добре зарекомендувала себе при розробці ПЗ, для якого досить точно або повністю сформульовано ТЗ, а, отже, можна точно розрахувати трудомісткість і вартість розробки. Однак застосування даної моделі при розробці кібер складової складних CPPS неможливе тому, що реальний процес розробки ніколи не вкладається в жорстку схему каскадної моделі, результати чергового етапу часто викликають зміни в прийнятих рішеннях, вироблених на попередніх етапах. Це змушує постійно повертатися до попередніх етапів і вносити зміни і уточнення в раніше прийняті рішення.

V-shaped model (V-образна модель) – заснована на систематичному підході до розробки ПЗ і визначає чотири базових етапи проєктування: аналіз, проєктування, розробка і огляд. Дана модель є різновидом каскадної, в якій приділяється велика увага верифікації та атестації ПЗ. Структура V-образної моделі представлена на рис. 1.15.

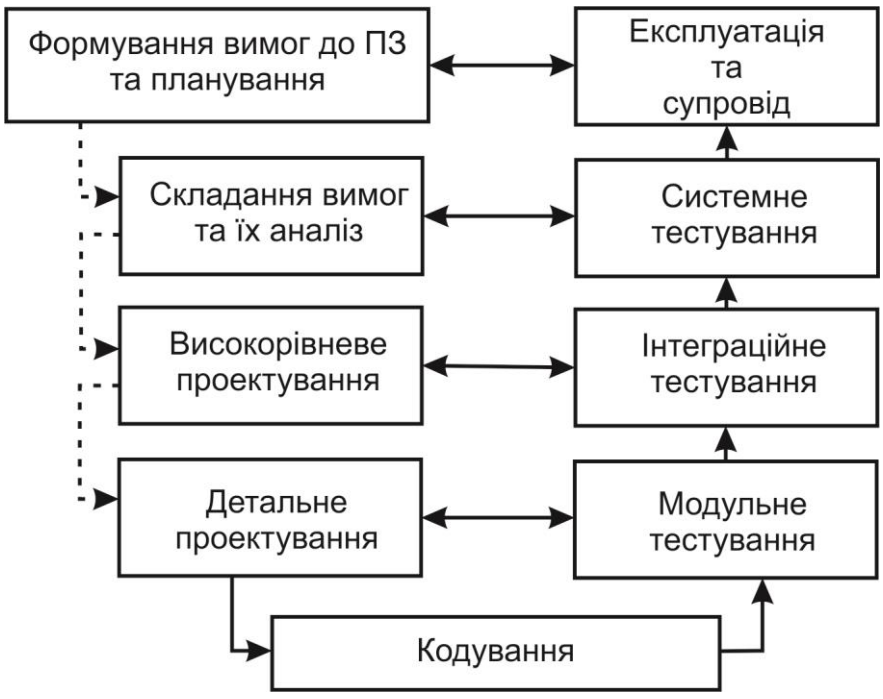


Рисунок 1.15 – Структура V-образної моделі

Штрихпунктирні стрілки (рис. 1.15) вказують на виконання даних

етапів паралельно основним, отже, системні вимоги до ПЗ і планування. Вони об'єднані з такими етапами: формування вимог до продукту і їх аналіз (складання повного технічного завдання); високорівневе проєктування (визначається структура програмного забезпечення і внутрішні зв'язки); детальне проєктування (визначається алгоритм роботи кожного компонента). При такому підході приділяється велика роль верифікації та атестації програмного забезпечення, починаючи з ранніх етапів проєктування, що дозволяє легко відстежувати хід роботи, а також виконання кожного етапу, так як завершення кожного етапу є контрольною точкою.

Однак основними недоліками даної моделі, з точки зору застосування її при розробці кібер складової складних CPPS, є такі: до уваги береться інтеграція між етапами; немає можливості внесення змін на різних етапах ЖЦ; тестування вимог відбувається занадто пізно, а отже, і внесення змін впливає на графік виконання робіт.

Prototype model (модель прототипування) – дозволяє створити прототип ПЗ до етапу або протягом складання ТЗ. Потенційний замовник працює з прототипом, визначає його сильні і слабкі сторони, а результати повідомляє розробникові. В результаті забезпечується взаємний зв'язок між розробником і замовником, який використовується для змін і коригування прототипу. Модель прототипування представлена на рис. 1.16.

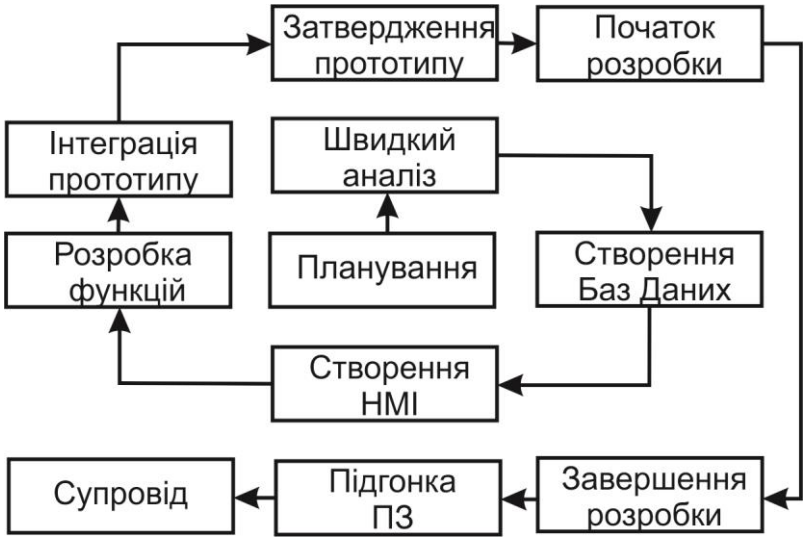


Рисунок 1.16 – Модель прототипування

Цей процес триває до тих пір, поки замовник не буде задоволений ступенем відповідності прототипу вимогам, що пред'являються до ПЗ в ТЗ. Однак, крім зазначених переваг, при розробці кібер складової складних CPPS, наприклад, етапи рішення складних технічних, технологічних і специфічних завдань, відсуваються на останній етап, що неприйнятно при розробці CPPS внаслідок того, що вони є первинними функціями, які необхідні замовнику (замовник може віддати перевагу отримання прототипу, а не закінченій повній версії ПЗ). Розробка прототипу може невиправдано затягнутися у зв'язку з невідомістю кількості виконаних ітерацій під час розробки, що збільшує трудомісткість і вартість розробки CPPS.

Rapid Application Development model (RAD-model) – на відміну від попередніх моделей, в RAD-model замовник грає вирішальну роль, в тісній взаємодії з розробниками ПЗ він бере участь у формуванні технічного завдання та апробації прототипу. Модель має ряд переваг: використання сучасних інструментальних засобів дозволяє скоротити трудомісткість розробки; залучення замовника на етапах складання вимог і розробки інтерфейсу користувача, що зводить до мінімуму ризик отримання програмного продукту, яким залишиться незадоволений замовник; повторне використання компонентів вже існуючих програм. Основні етапи RAD-model показані на рис. 1.17.

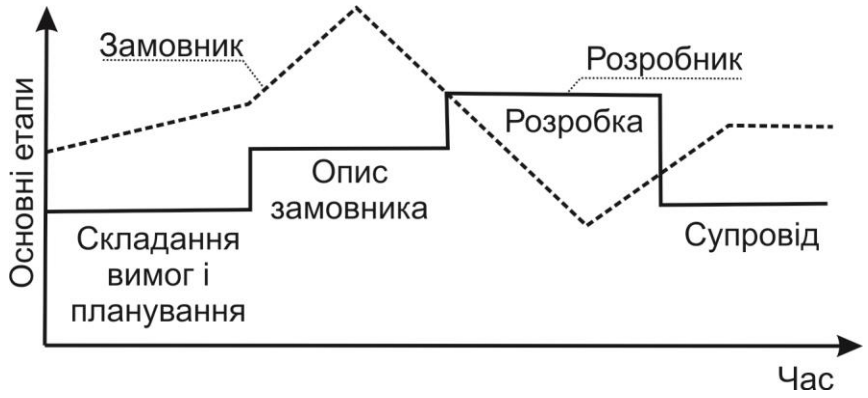


Рисунок 1.17 – Основні етапи RAD-model

Однак, дана модель має ряд недоліків, пов'язаних з розробкою кібер

складової складних CPPS: в залежності від специфіки ПЗ, що розробляється, використання компонентів існуючих програм неможливо або зведено до мінімуму; існує великий ризик, що розробка CPPS ніколи не буде закінчена, в зв'язку з зацикленням робіт з її розробки.

Incremental model (багатопрохідна модель) – складається з об'єднання процесу побудови ПЗ, з додаванням на кожній інтеграції нових функціональних можливостей або підвищенням ефективності ПЗ. Дана модель передбачає, що на початковому етапі ЖЦ розробки виконується конструювання ПЗ в цілому і визначається число інкрементів та належних до них функцій. Далі кожен інкремент проходить через етапи ЖЦ, що залишилися (кодування і тестування). Багатопрохідна модель представлена на рис. 1.18.

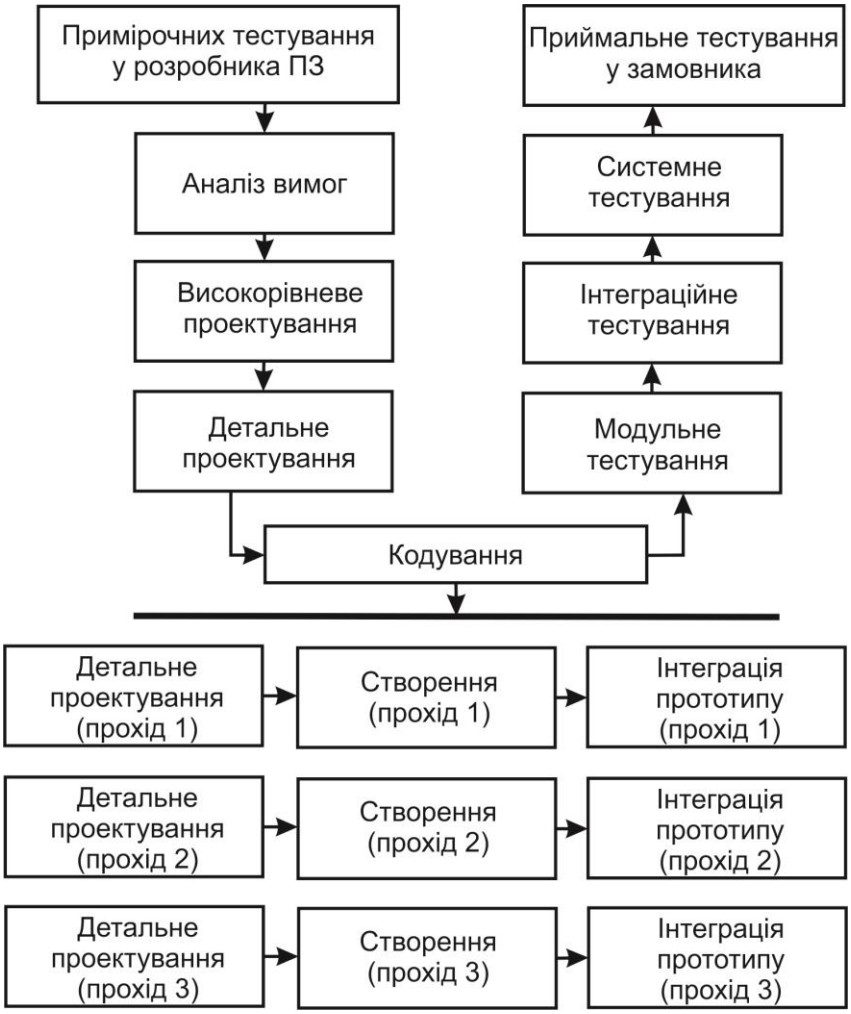


Рисунок 1.19 – Багатопрохідна модель

Однак, дана модель має такі недоліки: непередбачені етапи інтеграції всередині кожного інкремента; повна функціональність ПЗ повинна бути вказана на початку ЖЦ в технічному завданні; є можливість тенденції відтягування рішень складних завдань на завершальному етапі; загальні витрати трудомісткості і вартості ПЗ не знижуються в порівнянні з іншими моделями; обов'язковою умовою даної моделі є наявність гарного планування і проєктування програмного забезпечення. В результаті наведених вище недоліків дану модель рекомендують використовувати при проєктуванні ПЗ за умови заздалегідь сформульованих вимог і виділення великого періоду часу.

Spiral model (спіральна модель) – особливість даної моделі полягає в тому, що прикладне програмне забезпечення створюється не відразу, а частинами (модулями) з використанням методу прототипування. У даній моделі прототип – чинне ПЗ, що реалізує окремі функції і зовнішній інтерфейс користувача. Створення прототипу здійснюється за декілька ітерацій (витків спіралі), кожна ітерація відповідає створенню фрагмента або версії програмного забезпечення, на якій уточнюються мета і характеристики, оцінюється якість отриманих результатів, а також проводиться ретельний аналіз ризику перевищення трудомісткості і вартості ПЗ. Розробка ПЗ ітераціями відображає об'єктивно існуючий спіральний цикл, дозволяючи переходити на наступну стадію, не чекаючи повного завершення робіт на поточній стадії, оскільки при ітеративному способі розробки відсутні роботи можна зробити на наступній ітерації. Основні етапи спіральної моделі представлені на рис. 1.19.

До недоліків даної моделі можна віднести наступні: план роботи складається на основі статистичних даних, отриманих в попередніх проєктах і з особистого досвіду розробника. Спіральна модель може тривати нескінченно, в результаті пропозицій замовника, які можуть породити нову спіраль, а, отже, збільшити трудомісткість, вартість і час розробки ПЗ, у зв’язку з чим дана модель не підходить для розробки кібер складової

складних CPPS.

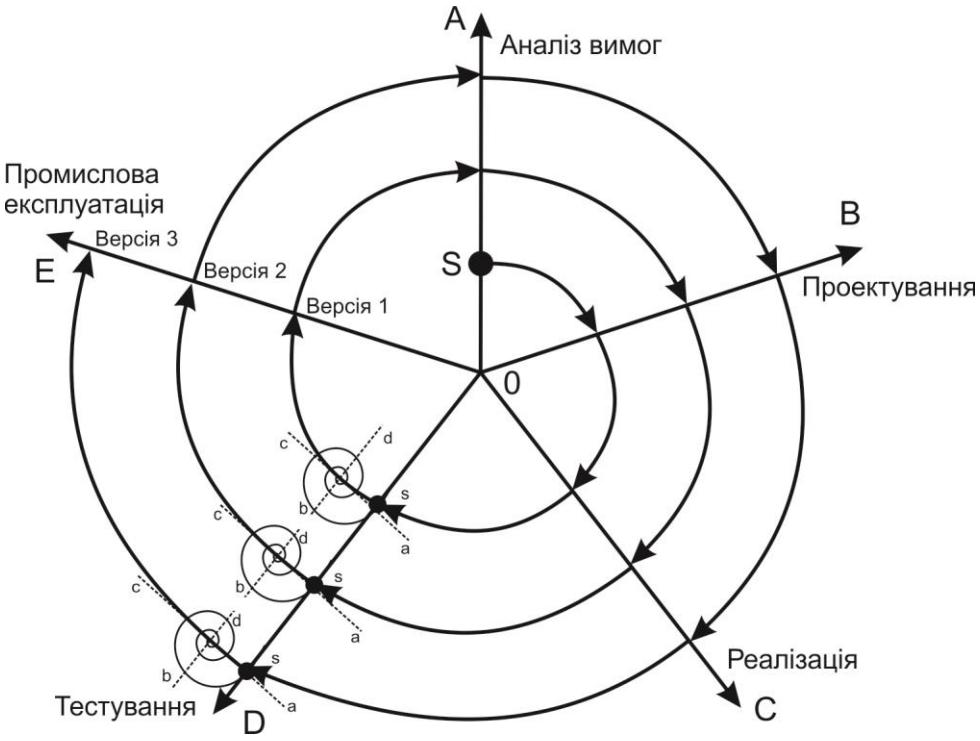


Рисунок 1.19 – Спіральна модель

1.7 Аналіз застосування методологій проєктування інформаційних систем для розробки кіберфізичні виробничих системи

Розглядаючи CPPS як складну взаємопов'язану багаторівневу архітектуру, яка об'єднує в собі кібернетичні і фізичні властивості, та враховуючи специфіку процесу управління розробкою складних CPPS, можна визначити, що домінантною для розробки кібернетичної складової є інформаційна база, яка отримується з фізичного рівня. Кібернетична складова являє собою великий набір елементів, який складається з: програмних скриптів PLC, які обробляють дані на PLC з датчиків; програмних модулів управління на кожному етапі; елементів системи SCADA для візуалізації технологічної інформації, а також програмних рішень для рівнів MES і ERP, які пов'язані в єдине цифрове середовище на базі сучасних інформаційних технологій. Внаслідок цього, в рамках даного

дослідження, необхідно провести аналіз існуючих методологій проєктування інформаційних систем, з точки зору їх можливого часткового або повного застосування, при рішенні задач автоматизації процесів управління розробкою складних CPPS.

Для проведення аналізу визначимо наступні поширені методології проєктування інформаційних систем:

- методологія функціонального моделювання SADT;
- методологія RED;
- методологія RUP.

Проведемо дослідження кожної методології і можливість їх застосування для вирішення завдань автоматизації процесів управління розробкою складних CPPS.

Методологія SADT (Structured Analysis and Design Technique) – методологія структурного аналізу і проєктування запропонована Дугласом Т. Россом, яка базується на структурному аналізі систем і графічному поданні організації у вигляді системи функцій, які розділені на три класи структурних моделей:

- функціональна модель;
- інформаційна модель;
- динамічна модель.

Підхід до процесу моделювання заснований на наступних етапах:

- збір інформації та аналіз предметної області;
- документування отриманої інформації;
- моделювання (IDEF0);
- коригування моделі в процесі інтерактивного рецензування.

Методологія заснована на формалізації процесу моделювання і базується на стадіях:

- аналіз;
- проєктування;
- реалізація;

- об'єднання;

- тестування;

- установки і функціонування.

Проєктування на основі IDEF0 (Function Modeling) зводиться до декомпозиції основних функцій організації. Результатом є розробка ієрархічної моделі проєктованої інформаційної системи при багатократній рівневій декомпозиції для отримання чіткого і детального опису всіх процесів [244, 245];

IDEF1 (Information Modeling) – методологія моделювання інформаційних потоків усередині системи, що дозволяє візуалізувати і аналізувати їх структуру і взаємозв'язки;

IDEF1X (Information Modeling Extended) – методологія розроблена на реляційних структурах і застосовується, в основному, для проєктування БД і відносяться до типу ER (Entity – Relationship) «сутність – взаємозв'язок» [245];

IDEF2 (Simulation Model Design) – методологія динамічного моделювання систем. На сучасному етапі представлені алгоритми і їх комп'ютерні реалізації у вигляді бібліотек, які дають можливість конвертувати статичні діаграми IDEF0 в динамічні за допомогою CPN (Color Petri Notes);

IDEF3 (Process Description Capture) – методологія документування процесів всередині системи, на базі сценаріїв і послідовності операцій для кожного процесу. Дана методологія прямо пов'язана з методологією IDEF0, у вигляді функціональних блоків кожного процесу засобами IDEF3 [242];

IDEF4 (Object-Oriented Design) – методологія побудови об'єктно-орієнтованих систем дає можливість відображати структуру об'єктів і закладені в них принципи взаємодії, що дозволяє аналізувати і оптимізувати складні об'єктно-орієнтовані системи;

IDEF5 (Ontology Description Capture) – методологія онтологічного дослідження систем за допомогою певного словника термінів і правил, на

основі яких формуються достовірні твердження про стани системи в деякий момент часу. На базі даних тверджень формуються висновки про подальший розвиток системи та проводиться її оптимізація;

IDEF6 (Design Rationale Capture) – методологія проєктних рішень, основним завданням якої є спрощення отримання «знань про здатність моделювання», їх подання і використання при розробці системи управління для Smart Manufacturing;

IDEF8 (User Interface Modeling) – метод розробки інтерфейсів користувача. Визначає увагу розробників на інтерфейс та поведінку користувача за:

- виконуваною операцією;

- сценарієм взаємодії, що визначається специфічною роллю користувача;

- деталями інтерфейсу, які визначають елементи управління для виконання операції та управління потоками даних.

IDEF9 (Business Constraint Discovery method) – метод дослідження і аналізу бізнес обмежень, в яких працює Smart Manufacturing, їх характеристик і вплив на процес моделювання.

IDEF10 (Implementation Architecture Modeling), IDEF11 (Information Artifact Modeling), IDEF12 (Organization Modeling), IDEF13 (Three Schema Mapping Design) – ці методи були визначені як затребувані, але не були розроблені;

IDEF14 (Network Design) – метод проєктування комп'ютерних мереж, який базується на аналізі вимог мережевих компонентів і забезпечує підтримку рішень, пов'язаних з раціональним управлінням матеріальними ресурсами.

Методологія SADT у більшості випадків підходить для опису моделей верхнього рівня, а наявність жорстких вимог забезпечує розробку моделей систем стандартного виду. При даних перевагах методологія SADT має ряд недоліків: складність сприйняття (велика кількість зв'язків (дуг) на діаграмі);

велика кількість рівнів декомпозиції, що призводить до труднощів зв’язку декількох процесів, які представлені в різних моделях в рамках одного Smart Manufacturing. На базі методології SADT (IDEF0 + DFD) розроблена уніфікована мова моделювання UML, яка дає можливість реалізації специфікації, візуалізації, конструювання та документації складних інформаційно насичених об'єктних систем. При всіх заявлених можливостях мови UML, вона має ряд недоліків, які не дозволяють її застосувати як базову методологію проєктування CPPS:

- надмірність мови, обумовлена тим, що вона включає багато практично не використовуваних діаграм і конструкцій;

- нечітка семантика, обумовлена тим, що UML визначена комбінацією за допомогою абстрактного синтаксису, мовою опису обмежень (OCL) і Англійською мовою (детальна семантика). У деяких випадках абстрактний синтаксис комбінації UML, OCL і Англійська мова суперечать один одному, в інших випадках вони неповні. Неточність опису UML однаково відображається на користувачах і розробниках інструментів, приводячи до несумісності розроблюваних інструментів під мову UML через унікальне трактування специфікації;

- UML розроблялась як мова моделювання загального призначення, в спробі досягти сумісність з усіма можливими мовами розробки. В контексті даних досліджень для досягнення мети автоматизації процесів управління розробкою CPPS, повинні бути обрані застосовні можливості UML, що накладає обмеження на її застосування, у зв'язку з чим на даний момент не існують повністю сформульовані формалізації опису CPPS.

Внаслідок чого застосування методології SADT, а саме мова моделювання UML, для автоматизації процесів управління розробкою складних CPPS неможлива без існуючих математичних моделей і методів, а також структури, яка формалізує всі рівні, етапи і послідовність розробки складних CPPS. Отже,на даний момент часу він носить описовий характер з нечіткою семантикою, що не припустимо в жорстко ієрархічних моделях

побудови CPPS.

Методологія RAD (Rapid Application Development) заснована на принципах інкрементного прототипування, що на ранній стадії проєктування інформаційної системи дає можливість продемонструвати замовникові діючу інтерактивну модель прототипу, уточнити прийняті проєктні рішення та оцінити експлуатаційні характеристики. У даній методології швидкість розробки досягається за рахунок використання компонентно-орієнтованого конструювання і має сенс при:

- обмеженому бюджеті розробки;

- нечітко визначених вимогах до розроблюваної системи;

- мінімальному терміні розробки;

- декомпозиції проєкту на складові елементи за функціональним призначенням.

В процесі розробки за методологією RAD система проходить чотири фази:

- **фаза аналізу і планування**, в ході якого визначаються вимоги, функції додатку і їх пріоритетні, а також інформаційні потреби, які вказуються замовником при безпосередній участі розробника. Визначаються рамки масштабу проєкту, тимчасові терміни і платформи для запуску системи;

- **фаза проєктування** проходить за участю користувачів розроблюваної системи під керівництвом розробників. Групи і підгрупи RAD на даному етапі використовують комбінацію технік спільної розробки додатків JAD і CASE – інструментів для реалізації вимог користувачів в розроблюваних моделях. Це дає можливість майбутнім користувачам зрозуміти, модифікувати і визначити робочу модель системи, яка повинна відповідати вимогам ТЗ, і при необхідності створити частковий прототип системи, що розробляється.

В результаті даної фази розробник проєктує:

- загальну інформаційну модель системи;

- функціональні моделі системи і підсистем;

- робочі прототипи інтерфейсу користувача, звітів та діалогових вікон. Особливістю даної фази в RAD моделі є те, що кожен прототип стає частиною розроблюваної системи.

- **фаза побудови** спрямована на швидку розробку системи на основі результатів, отриманих в попередній фазі. При цьому замовник продовжує брати участь в розвитку системи і в залежності від необхідності пропонує внесення змін і поліпшень в систему, що розробляється. Особливістю даного етапу є те, що тестування проходить синхронно під час розробки.

- **фаза впровадження** дозволяє синхронізувати три ключові моменти: навчання користувачів розробленої системи; тестування функціоналу на відповідність вимогам ТЗ; заміна старої системи на вдосконалену нову.

Розглядаючи методологію RAD, в контексті даних досліджень, і можливість її застосування при розробці CPPS можна виділити наступні недоліки:

- відсутність масштабованості (аналіз показав, що методологія RAD переважно використовується для швидкої розробки невеликих і малих систем невеликими і середніми проєктними командами);

- фокусування методології RAD на прототипах в більшості випадків призводить до проблеми постійного внесення дрібних змін в окремі елементи системи і при цьому ігноруються проблеми системної архітектури, що є неприпустимим для складних жорстко ієрархічних CPPS;

- в реальних проєктах неможливо дотримати баланс між гнучкістю і контролем процесу розробки, а в більшості випадків обирається один з них. В останньому випадку методологія RAD не буде життєздатною;

- методологія RAD не підійде для систем, в яких основними вимогами є якість і контроль всіх етапів розробки; для створення великомасштабних проєктів, час реалізації яких більше 90 днів; для реалізації систем, в яких критично важливим є високий рівень планування і жорстка архітектура; систем, для яких притаманні суворе дотримання розроблених протоколів і

інтефейсу.

Ґрунтуючись на вище перелічених недоліках методології RAD можна зробити висновок, що її застосування не є доцільним при вирішенні задач автоматизації процесів управління розробкою складних CPPS.

Методологія RUP (Rational Unified Process) – заснована на ітеративній моделі розробки систем на базі ЖЦ «Спіральної моделі». В кінці кожної ітерації розробники повинні досягти заплановані цілі, створити або допрацювати «проєктні артефакти» і отримати проміжну, але функціональну версію кінцевої системи. Основними перевагами методології RUP є швидкість реагування на мінливі вимоги до розроблюваної системі, виявлення та усунення ризиків на ранніх стадіях проєкту, а також ефективність контролю якості створюваного продукту.

ЖЦ розробки системи складається з чотирьох основних фаз:

- Inception (початкова стадія) в ході якої формується початкове бачення і межі проєкту; розроблюється економічне обґрунтування; визначаються основні вимоги, обмеження і ключові функції продукту; формується базова версія моделі прецедентів і проводитися оцінка ризиків. Як результат цієї фази розробник оцінює досягнення LOM і узгодженості між замовником і розробником.

- Elaboration (уточнення) на цій фазі проводиться дослідження та аналіз предметної області та побудови архітектури системи, яка включає детальний опис прецедентів, на базі яких проводитимуться тестування розробленої архітектури й уточнюються терміни та вартість розробки системи. Успішне виконання фази Elaboration означає досягнення LAM.

- Construction (побудова), на даній фазі ЖЦ розробки здійснюється реалізація частини функціоналу системи і завершується розробка першої бета-версії системи.

- Transition (впровадження) в ході виконання даної фази, створюється фінальна версія системи і передається від розробника до замовника для повного тестування та оцінки якості. Якщо якість не відповідає вимогам і

критеріям ТЗ, встановленим на фазі Inception, то дана фаза повторюється знову, поки всі умови, зазначені в ТЗ замовником, не будуть задоволені.

Ґрунтуючись на вищевказаному можна виділити основні риси методології RUP:

- інтерактивний процес розробки (controlled interactive);

- наскрізне застосування апарату Use Case Driven;

- приділення особливої уваги розробці Architecture Centric;

- управління вимогами та змінами (Requirements Configuration and Change Management);

- використання концепції Component Based Development;

- базування на принципах Visual Modeling Techniques.

Аналізуючи публікації B. Boehm можна виділити наступні загальні ризики при використанні методології RUP:

- планування нереалістичних термінів, за якими неможливо виконати повний обсяг роботи, а отже,профіцит бюджету;

- повністю не враховується функціональність, що неприпустимо і критично при розробці складних CPPS, так як функціональність є головним пріоритетом для замовника;

- розробка некоректного візуального інтерфейсу користувача, що не дозволить реалізувати візуалізацію складних інформаційних потоків вертикальної інтеграції ієрархічних рівнів CPPS на рівнях SCADA і c-MES, а отже,зменшує ефективність від розробки CPPS;

- відведення великої кількості часу на оптимізацію;

- базування методології на ЖЦ «Спіральної моделі», яка вносить на кожній ітерації новий потік змін, що в рамках розробки «жорстко» ієрархічних CPPS неприпустимо і призводить до порушення архітектури, алгоритмів функціонування, а отже,ускладнює процес управління розробкою CPPS;

- брак інформації про зовнішні компоненти, які визначають оточення системи або використовуваних для інтеграції, що призводить до зменшення

продуктивності, а це суперечить основним принципам роботи CPPS в умовах «реального часу».

З огляду на вище перераховані ризики використання методології RUP можна зробити однозначний висновок про недоцільність його застосування при розробці CPPS. Так само, в рамках даної роботи, були досліджені і проаналізовані наступні методології: ASD (Adaptive Software Development), DevOps (Development і Operations), DAD (Disciplined Agile Delivery), DSDM (Dynamic Systems Development Method), FDD (Feature Driven Development), LeSS (Large Scale Scrum), MDD (Model-Driven Development), MSF (Microsoft Solutions Framework), PSP (Personal Software Process), OpenUP (Open Unified Process), SAFe (Scaled Agile Framework), Scrum (SCRibing Unified Methodology), TSP (Team Software Process), UP (Unified Process), XP (Extreme Programming), а також наступні основні парадигми і моделі: Agile, Cleanroom, моделей ЖЦ «Інтерактивної», «Спіральної», «Каскадної», «V-Model», «Dual Vee Model». В результаті їх аналізу зроблені висновки про можливість застосування до вирішення завдань при розробці кібер складової CPPS:

- методологія DevOp базується на автоматизації розробки, яка заснована на стандартизації і схожа з моделлю Agile. Метою методології DevOp є використання архітектурних стилів мікросервісів. Дослідження компанії «F5 Labs» показало, що тільки 17% розробників із 2200 опитаних позначили методологію DevOp як базову в своїх розробках. З огляду на специфіку розробки CPPS, складність і багатогранність архітектури, а також унікальність розробки під кожну конкретну задачу і мету, можна зробити висновок про неможливість використання стандартизації мікросервісів для розробки CPPS;

- методологія DAD доповнює методологію Scrum за допомогою гібридних доповнень: XP, UP, Канбана, ощадної розробки ПЗ, які були запропоновані компанією IBM. В основі даної методології лежить підхід заснований на меті розробки, який представляє можливість вибору і

модифікації фреймворку в залежності від вимог до кожної розробки та обраної стратегії. Розглядаючи методологію DAD, з точки зору вирішення завдань розробки CPPS, можна зробити висновки, що дана методологія допускає масштабування, але не дозволяє його реалізувати в повній мірі, обмеження фреймворку накладає рамки його використання, що вимагає доопрацювання його кожен раз під новий тип завдань і вимог, які накладаються на розроблювальний CPPS і доступний тільки на верхніх рівнях його ієрархії;

- методологія DSDM заснована на концепції методології RAD, тобто інтерактивному підході до розробки, недоліки якої, з точки зору даних досліджень, повторюють недоліки методології RAD які описані вище;

- методологія FDD заснована на моделі гнучкої розробки (Agile) і використовує модульний підхід до розробки ПЗ. Основна парадигма даної методології заснована на обмеженні тимчасових рамок розробки функції (2 тижні), при неможливості їх досягнення в заданий термін, функції розбиваються на підфункції і т.д. Основним недоліком методології FDD є залучення невеликих команд розробників певних функції, що ускладнює видимість основних цілей розробки, повної структури процесів управління розробкою CPPS і зв'язків між рівнями та етапами, регулярна збірка розроблюваних функцій кожного разу, коли виявляються помилки при їх об'єднанні. Дані недоліки, у відповідності до вимог з розробки CPPS, можуть спричинити зміну архітектури, функціональних алгоритмів і призвести до відхилення або недосягнення головної мети розробки CPPS;

- методологія MDD заснована на абстрактному описі ПЗ у вигляді моделей. Моделі представлені за допомогою ПОМ (предметно-орієнтованої мови). Модель визначається нотацією (сукупністю графічних елементів) і метамоделлю (інформація у вигляді метаданих), які в сумі використовуються у вигляді каркасів, які описують ПЗ, що розробляється. На базі даної методології розроблено Eclipse Modeling Framework, але має він має ті ж недоліки як і методологія DAD з точки зору його застосування для розробки

CPPS;

- методологія MSF використовується в гібридному зв'язку з методом Agile, і представляє собою загальну базу методів, які відображають загальні методологічні підходи до інтерактивного проєктування, що тягне за собою наступні недоліки їх застосування при розробці CPPS: немає певних принципів управління проєктами та пояснень різноманітних методів роботи, з самого початку не проводиться декомпозиція CPPS за рівнями, що тягне «проблеми» розробки на рівнях Field level, Control level (рис. 1.4), а дані рівні є основними при рішенні задач розробки CPPS, у зв’язку з чим можна стверджувати про неможливість повного використання методології MSF для реалізації розробки кібер складової CPPS;

- методологія PSP за визначенням не підходить, з точки зору його застосування для розробки CPPS, так як основним його аспектом є застосування принципів моделі зрілості процесів до практики одного розробника;

- методологія OpenUP заснована на ітеративно-інкрементальній моделі розробки і базується на легкому і гнучкому варіанті методології RUP, у зв'язку з чим він успадкував всі недоліки методології RUP, з точки зору його застосування для розробки CPPS;

- методологія SAFe являє собою гнучкий фреймворк на базі Agile і розглядає процес розробки з точки зору керівників для аналізу та управління поточних проєктів, а отже,дана методологія не підходить до вирішення завдань розробки складних CPPS.

Як можна бачити, що всі проаналізовані методології, методи і моделі, в загальному випадку відносяться до розробки інформаційних систем, що не враховує фізичну складову CPPS, яка є основоположною для розробки на рівнях Field level, Control level і впливають на структуру і реалізацію інформаційних потоків на рівнях Production, Operations level і механізмів математичного, функціонального і алгоритмічного опису інформації на атомарних рівнях (датчики, PLC), які впливають на процес управління,

архітектуру і функціональні алгоритми розробки CPPS, як єдиної інформаційної екосистеми.

В результаті чого можна зробити висновки, що досліджені методології мають узагальнені підходи до розробки інформаційних систем і програмних продуктів і частково можуть використовуватися для розробки програмних модулів і налаштувань на Operations і Enterprise planning level у вигляді окремих програмних рішень для певних типів завдань.

CPPS являє собою складний «жорстко» ієрархічний об'єкт, який поєднує в собі природу як фізичних так і кібернетичних процесів, які в колаборації дозволяють створити цифрового близнюка реального виробничого процесу високотехнологічних виробів з можливістю автоматизації керування їх в режимі реального часу. Розробка нових і модернізація існуючих CPPS керування складними організаційно-технічними виробничими об'єктами, є складною науково-технічною задачею, яка носить індивідуальний характер, який залежить від мети її розробки, застосування, обладнання і датчиків, які використовуються в даному операційному процесі.

При цьому використання аналогічних CPPS керування складними організаційно-технічними виробничими об'єктами неможливі через використання різних параметрів сенсорів, актюаторів, PLC навіть в схожих операційних процесах, що веде до зміни кібернетичних потоків контролю фізичних величин і алгоритмів їх виконання, що в подальшому буде впливати на забезпечення ефективності процесами керування.

Виходячи з цього одним з ефективних варіантів розробки нових або модернізації існуючих CPPS є застосування нових методів і моделей автоматизації процесів керування складними організаційно-технічними виробничими об'єктами, як синтезу фізичних і кібернетичних складових в єдиний цифровий інформаційний простір, який дозволить об’єднати всі інформаційні потоки в цілісну систему.

Основною науковою гіпотезою наукової роботи є підвищення продуктивності і ритмічності виробництва за рахунок розробки нових

методів і моделей автоматизації процесів керування складними організаційно-технічними виробничими об'єктами на базі кіберфізичних виробничих систем. Для обґрунтування наукової гіпотези на етапі експериментальних досліджень в роботі поставлена наступна мета – підвищення ефективності виробничого процесу шляхом розробки методів, моделей і нової технології, алгоритмічного і програмного забезпечення керування організаційно-технічними об'єктами на базі кіберфізичних виробничих систем.

Для досягнення поставленої мети в монографії розроблена узагальнена схема теоретичних, експериментальних і практичних досліджень, яка представлена на рис. 1.20.

На першому *теоретико-методологічному етапі* проводитися аналіз об'єкта дослідження – процесу кіберфізичного керування складними організаційно-технічними виробничими об'єктами, як синтез фізичних і кібернетичних параметрів в єдиному інформаційному середовищі. Досліджуються і аналізуються основні етапи зрілості CPPS на базі моделі CMM, аналізуються і інтерпретуються вимірювання моделі еталонної архітектури розробки CPPS (RAMI 4.0) з точки зору мети і завдань даних досліджень. Проводиться аналіз вертикальної інтеграції ієрархічних рівнів CPPS на базі моделі DIKW. Об'єднання результатів аналізу архітектури та функціонування складних CPPS, а також ряду вимог, які висуваються до Smart Manufacturing, з точки зору автоматизації процесу їх керування, та поряд з дослідженням існуючих методологій, парадигм і моделей розробки програмних систем (ПС) і програмного забезпечення (ПЗ) для можливості їх застосування при реалізації кібернетичної складової CPPS, показує тільки часткову можливість їх використання лише на верхньому рівні моделі вертикальної інтеграції ієрархічних рівнів керування організаційно-технічними об'єктами на базі кіберфізичних виробничих систем.

Таки чином виявляється і обґрунтовується науково-прикладна проблема розробки ефективної стратегії автоматизації управління складними

організаційно-технічними виробничими об'єктами, шляхом реалізації комплексу моделей, методів процесів управління і технології на базі кіберфізичних систем для «безлюдного виробництва».

Для досягнення поставленої мети в даному дослідженні необхідно вирішити наступні завдання:

- провести аналіз сучасних методологій, методів і моделей автоматизації процесів керування складними організаційно-технічними об'єктами на базі кіберфізичного виробничих систем, з метою виявлення недоліків та проблем;

- розробити архітектурно-логічну модель і методи декомпозиції кіберфізичного керування процесами в складних організаційно-технічних об'єктах;

- розробити групу методів прийняття рішень на кожному етапі архітектурно-логічної моделі кіберфізичного керування;

- розробити метод синтезу алгоритмів функціонування кіберфізичної виробничої системи;

- розробити технологію безперервного процесу керування організаційно-технічним об'єктом на базі кіберфізичної виробничої системи;

- розробити моделі формалізації кібернетичної складової організаційно-технічного об'єкта;

- розробити синтаксичну та семантичну модель декларативної мови визначення і маніпулювання даними;

- програмно реалізувати розроблені методики, моделі, методи і провести експериментальні дослідження ефективності та практичну апробацію отриманих теоретичних результатів.

<table>
<tr><td rowspan="2">І етап</td><td colspan="2">Теоретико-методологічні дослідження</td></tr>
<tr><td colspan="2">Аналіз об'єкта дослідження - процес кіберфізичного керування складними організаційно-технічними виробничими об'єктами</td></tr>
<tr><td></td><td>Проблема - відсутність ефективної стратегії автоматизації управління складними організаційно-технічними виробничими об'єктами на базі кіберфізичних виробничих систем для «безлюдного виробництва»</td><td>Проблема як потенційна можливість розробки нових методів, моделей і технологій процесів керування складними організаційно-технічними виробничими об'єктами на базі кіберфізичних виробничих систем</td></tr>
<tr><td rowspan="4">ІІ етап</td><td colspan="2">Експериментальні дослідження</td></tr>
<tr><td colspan="2">Визначення рівнів і етапів процесу управління складними організаційно-технічними виробничими об'єктами вигляді синтезу кібернетичних і фізичних параметрів на базі теорії метасистем і методів декомпозиції</td></tr>
<tr><td>Мета досліджень-підвищення ефективності виробничого процесу шляхом розробки методів, моделей і нової технології, алгоритмічного і програмного забезпечення управління організаційно-технічними об'єктами на базі кіберфізичних виробничих систем.</td><td rowspan="2">- розробити архітектурно-логічну модель і методи декомпозиції кіберфізичного керування процесами в складних організаційно-технічних об'єктах;
- розробити методи процесу керування організаційно-технічним об'єктом, як логічно узгоджені послідовності прийнять рішень;
- розробити технологію безперервного процесу керування організаційно-технічним об'єктом;
- розробити метод синтезу алгоритмів функціонування кіберфізичного керування;
- розробити модель формалізації кібернетичної складової організаційно-технічного об'єкта;
- розробити синтаксичну та семантичну модель декларативної мови визначення і маніпулювання даними;</td></tr>
<tr><td>Завдання дослідження:
- провести аналіз сучасних методологій, методів і моделей автоматизації процесів керування складними організаційно-технічними об'єктами з метою виявлення недоліків та проблем;
- розробити архітектурно-логічну модель і методи декомпозиції кіберфізичного керування процесами в складних організаційно-технічних об'єктах;
- розробити технологію безперервного процесу керування організаційно-технічним об'єктом на базі кіберфізичної виробничої системи;
- розробити моделі формалізації кібернетичної складової організаційно-технічного об'єкта;</td></tr>
<tr><td colspan="2">«Система розробки кібернетичної складової для автоматизації процесів керування організаційно-технічним виробничим об'єктом»</td></tr>
<tr><td rowspan="2">ІІІ етап</td><td colspan="2">Практична реалізація результатів досліджень і наукових розробок</td></tr>
<tr><td>Оцінка техніко-економічної ефекту (продуктивності і ритмічності виробництва) від впровадження результатів дослідження</td><td>Висновки за результатами досліджень. Отримання свідоцтва про реєстрацію авторського права. Промислове впровадження отриманих результатів наукової роботи.</td></tr>
</table>

Рисунок 1.20 – Узагальнена схема теоретичних, експериментальних і практичних досліджень

На останньому етапі практичної реалізації результатів дослідження і наукових розробок проводиться оцінка техніко-економічної ефекту (продуктивності і ритмічності виробництва) від впровадження результатів дослідження, отримання свідоцтв про реєстрацію авторських прав України на розроблену «Систему розробки кібернетичної складової для автоматизації процесів керування організаційно-технічним виробничим об'єктом».

Для вирішення поставлених завдань в рамках даної роботи запропонована наступна методологія дослідження автоматизації процесів керування складними організаційно-технічними об'єктами на базі кіберфізичного виробничих систем, яка представлена на рис. 1.21.

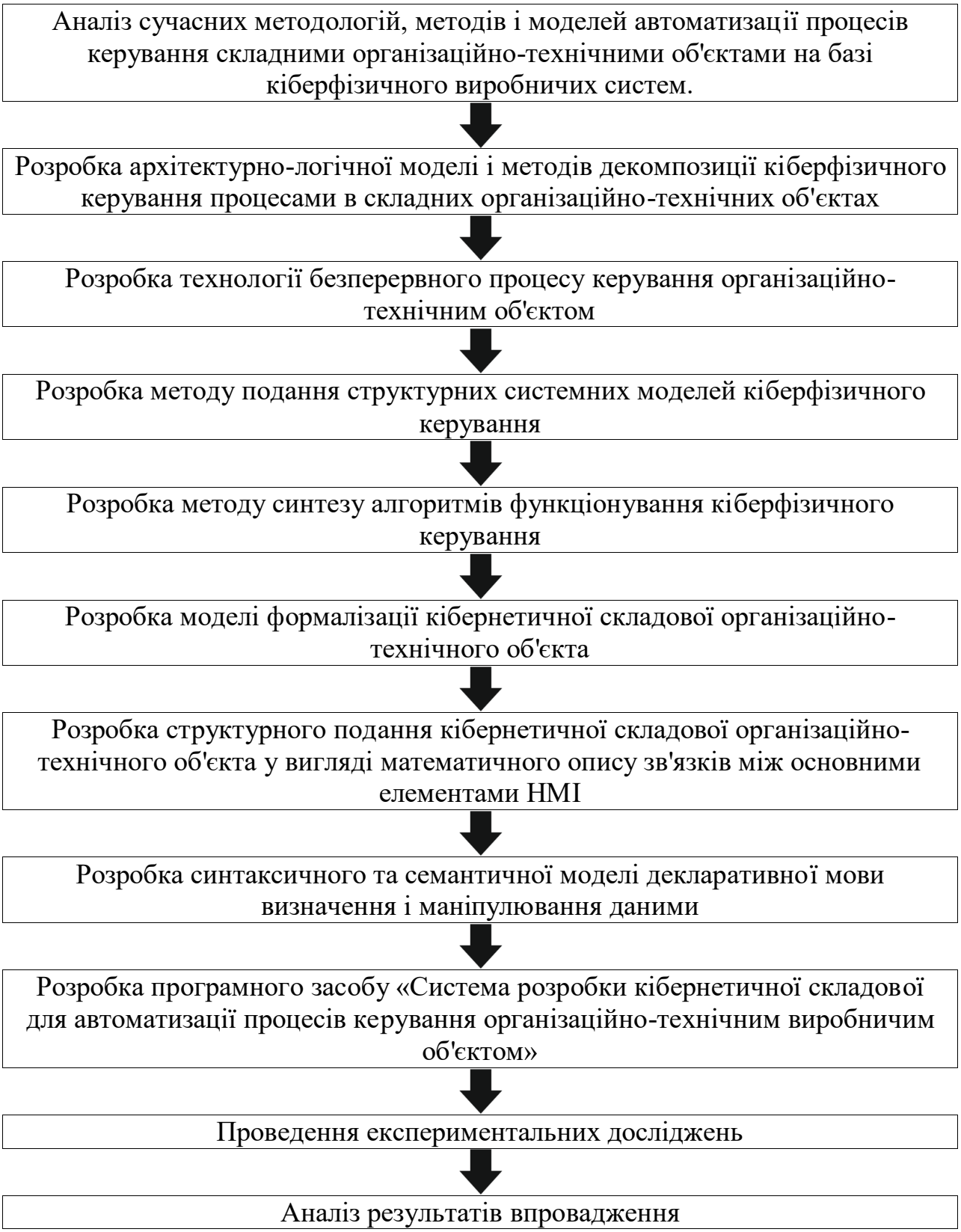


Рисунок 1.21 – Методологія дослідження автоматизації процесів керування складними організаційно-технічними об'єктами на базі кіберфізичного виробничих систем.

РОЗДІЛ 2
МЕТОДОЛОГІЯ І МЕТОДИ ПРОЦЕСУ УПРАВЛІННЯ РОЗРОБКОЮ СКЛАДНИХ КІБЕРФІЗИЧНИХ ВИРОБНИЧИХ СИСТЕМ

2.1 Архітектурно-логічна модель процесу управління розробкою кіберфізичних виробничих систем

Сучасні кіберфізичних виробничі систем в рамках Industry 4.0 можна розглядати як складну багаторівневу комп'ютерно-інтегровану систему на всіх рівнях управління. Прикладом таких систем можуть служити системи, які застосовуються в таких сферах діяльності людини: Hostig (Amazon, IBM, Oracle, Bosch, TAT, Rockspace, і т.д.), Industrial IoT Platforms (Samsung, OLEX, PTC , Siemens, Exosite, Cisco, Atlizon, і. т.д.), Analytics (SAR, Mnubo, Seeq, Smart Cloud, ARM Treasure data, MAPR, Relayr, і т.д.), Microchips (ARM, Micron, ST , NVidia, Intel, і т.д.), Sensor (ABB, JUMO, FESTO, SICK, KISTLER, TE і т.д.), Connectivity Hardware (ABB, DELL, MOXA, LOGIC, KEB, iEi, ADELINK), Cybersecurity (ARGUS, ARM, Arilou, Bastille, CyberBit, Indegy, Senrio, MOCANA і т.д.), Systems Integration (Accenture, Atos, CGI, Cognizant, HCL, InfoSys, Kalypso, T-Systems і т.д.) , Additive Manufacturing (3D System, AddUp, AutoDesk, ArcamEBM, Arkema, DesktopMetal, HP, DMG MORI, ExOne, і т.д.), Augmented and Virtual Reality (Sony, HTC, Atheer, Brother, UBiMAX, і т.д.), Collaborative robots (ASS, KUKA, YASKAWA, ROBOTIQ, MAGAZINO, і. т.д.), Connected Machine Vision (BASLER, COG NEX, FESTO, FLIR, OMRON, SICK), Dron / UAVs (Kespry, DJI, Skycatch і т.д.), Self-Driving (AETHON, AmazonRobotics, BALYO, STILL, GreyOrange, Omron і т.д.).

Особливістю управління процесами розробки таких систем є їх складний багаторівневий характер, що визначає великі труднощі, пов'язані з нескінченними варіантами їх побудови як на кібернетичному так і на

фізичному рівнях з урахуванням інтеграції всіх можливих зв'язків між ними. Іншою особливістю розробки складних CPPS є їх індивідуальність і одиничний характер, тобто не існує прототипів і аналогів, з яких можна частково або повністю перенести структуру кіберфізичної системи. В результаті чого необхідно знаходити спільні закономірності процесів управління при розробці таких систем і проводити узагальнення не на типових рішеннях, а на рівні складних багаторівневих комп'ютерно-інтегрованих систем.

Третьою особливістю є слабка методологічна база розробки складних багаторівневих CPPS, що пояснюється відсутністю адекватних засобів формального опису таких складних об'єктів і засобів алгебраїчного перетворення цих описів, що стримує розвиток методів структурно-параметричного синтезу CPPS і їх аналізу у вигляді системних складних моделей.

Четвертою особливістю є синхронне проєктування, як самого об'єкта, так і його системи управління, яке спочатку носить паралельно-послідовний характер «згори-вниз» і суворо відповідає логічній послідовності.

Методологія створення CPPS включає в себе два основних підходи до кожного рішення:

- підхід до вибору конкретного рішення (синтез) цільових, функціональних, структурних, інформаційних і алгоритмічних рішень;

- аналіз кожного прийнятого рішення за якістю, змістом і методами моделювання.

Методологія об'єднує в собі складний комплекс логічно пов'язаних методів і методик прийняття рішень. Спочатку вводяться і визначаються такі поняття: «древо», «рівень декомпозиції», «базис». Далі можна сказати, що CPPS – це структура розподілених за «рівнем декомпозиції» і «базисами» методів рішень, а «древо» представляє собою деревоподібну ієрархічну структуру з «рівнів декомпозиції».

Взявши до уваги, що CPPS в рамках технології Industry 4.0 є складним *n*-рівневим об'єктом, розробку даної методології пропонується почати з системного аналізу методом декомпозиції і розбиття CPPS на складові його компоненти за класифікаційними властивостями, або ознаками, що дозволять визначити рівень його ієрархії, а отже, покажуть його системний вид. Одним з ключових моментів багаторівневої декомпозиції CPPS є виявлення і виділення його властивостей, які визначаються метою і завданнями створення CPPS. Дане рішення впливає на всі етапи процесу управління розробкою CPPS, бо визначає не тільки структуру, а й «древо».

Ґрунтуючись на вищевказаному можна зробити висновок, що на ранніх етапах розробки CPPS необхідно використовувати цільову декомпозицію. Тобто на першому кроці процесу управління розробкою CPPS неможливо обійтися без системного аналізу мети і завдання, які дозволять окреслити і виділити пріоритетні ознаки і властивості, за якими з'являється можливість провести декомпозицію на підцілі нижнього рівня.

Дані ознаки і властивості можна отримати за допомогою аналізу близьких за метою та завданнями існуючих CPPS, а також цілей і задач нового CPPS, що розробляється. Варто звернути увагу, що рівнева декомпозиція дає можливість визначити функціонал, структуру і алгоритм функціонування CPPS на базі мети і завдання процесу управління розробкою. Отже,цілі та завдання розробки є домінантними, так як централізують в собі всі інші властивості CPPS.

Ґрунтуючись на базі теорії великих систем, введемо в дане дослідження наступні поняття:

- атомарний елемент системи ($AEofS_i$) – це неподільна найелементарніша частина системи, яка не складається з будь-яких частин, де $i=0…n$. Тобто атомарні елементи системи, які за своїми властивостями, що характеризують їх фізичну або інформаційну природу, можуть бути як однаковими так і різними. Звідси можна записати умову, яка на самому нижньому елементарному рівні складатиметься з множини $AEofS_i$. У

результаті ліквідним є запис $CPPS \subset AEofS_i$. Даний нижній рівень, в даній методології, визначимо як «рівень декомпозиції першого рівня» CPPS. Тоді подання CPPS через елементи першого рівня можна описати у вигляді $CPPS\_AEofS_i$ при $i=1...n$;

- група елементів ($G\{AEofS\}$) *та* $AEofS_i$, які об'єднані за принципом володіння однаковими властивостями на атомарному рівні. Введення ($G\{AEofS\}$) дозволить представити CPPS через ($G\{AEofS\}$), а Отже,дозволить її описати вигляді $CPPS \subset G\{AEofS\}_j$, де $j=1...m$, що відповідає другому рівню складності подання CPPS і в даній методології визначається як декомпозиція другого рівня;

- підсистема (Sub_S) – це подання CPPS на базі ($G\{AEofS\}$), які об'єднані в групу і володіють властивими їм природними властивостями. Тоді подання CPPS, на базі підсистеми, дасть можливість визначити третій рівень складності об'єкта $CPPS \subset Sub\_S_k$, де $k=0...d$;

- мультисистема *(MS″)*, яка є об'єднанням (Sub_S) складних об'єктів, які мають свої властивості і дає можливість представити об'єкт як четвертий рівень декомпозиції CPPS, а Отже,її можна описати виразом $MS'' \subset Sub\_S_k$;

- метасистема *(MS')* це об'єднання MS'' за властивостями відповідно до вище описаної логіки.

Використовуючи запропоновану методику подання CPPS через рівні і глибину розробки з’являється можливість провести декомпозицію моносистеми до четвертого порядку, четвертого – як мультисистеми, з п'ятого як метасистеми з нескінченним числом рангів. Дане рішення дозволить провести дослідження процесів управління розробкою складних CPPS, визначити рівень декомпозиції, а також рівень розробки структури на кожному етапі, використовуючи метод «згори-вниз». Запропоновані в даній роботі рішення дають можливість будувати вертикальний каркас декомпозиції всіх рівнів процесу управління розробкою CPPS у відповідності до поставленої мети.

Проведений аналіз ISO 22745-1-2016, ISO 22745-11-2017, ISO 22745-13-2017, ISO 22745-2-2017 , ISO 22745-20-2018, ISO/TS 22745-30-2018 показав, що розробці методів процесів управління в даних стандартах не приділено достатньо уваги, а наведено загальні відомості і основні принципи та послідовність завдань на різних етапах, що неможливо застосувати до сучасних кіберфізичних систем в рамках Industry 4.0, і тому не може бути прийнятий за основу розроблюваної методології.

Для подальших досліджень процесу управління розробкою CPPS необхідно провести класифікацію прийняття рішень за певними властивостями, які дозволять досягти мету розробки CPPS.

В ході аналізу існуючих методів, які дозволяють провести класифікацію, був обраний метод стратифікації. Даний метод включає в себе певний набір рішень за різними властивостями об'єкта. Виходячи з цього, для визначення властивостей об'єкта, необхідно використовувати методи системного аналізу.

Дослідження процесу управління сучасними кіберфізичними системами, в рамках Industry 4.0, можна в загальному випадку згрупувати за такими рішеннями:

Загальні (цільові):

- мета управління;
- функції управління;

Кібернетична складова:

- інфологічна структура;
- алгоритми управління;
- інформаційна структура (PLM, MES, ERP);
- програмне забезпечення (CAD / CAM / CAE);

Фізична складова:

- функції управління;
- організаційно-технічна структура;

- комплекс технічних засобів (ПЛК, датчики, технологічне обладнання, мережі і т.д.).

При цьому необхідно врахувати, що процес управління розробкою сучасних CPPS може протікати як в синхронному, так і в асинхронному часовому режимі, що вимагає встановлювати жорстку послідовність виконання рішень, так як їх невиконання призведе до зміни термінів і підвищення вартості проєктування і впровадження CPPS на підприємстві.

Як обґрунтування існування суворо логічної послідовності процесу управління розробкою складних CPPS можна навести такі протиріччя:

- не маючи проєктних рішень з алгоритмічного забезпечення і комплексу технічних засобів неможливо розробити PLM, MES, ERP і адаптувати CAD / CAM / CAE;

- необхідне алгоритмічне забезпечення функціонування і проєктування відбувається на базі математичного забезпечення;

- математичне забезпечення розробляється на базі завдань управління, які повинні обиратися в залежності від мети і завдання розробки кожної CPPS.

Тому процес управління розробкою сучасних кіберфізичних систем будується на базі цільових алгоритмів управління та інформаційних потоків взаємодії структурних елементів об'єкту управління, в залежності від його природи і фізичних властивостей. Простий приклад вищенаведених протиріч показує, що кожне рішення строго залежить від попереднього і існує «жорстка» логічна послідовність всіх рішень при розробці тієї чи іншої CPPS.

Виходячи з цього, для методології, що розробляється, необхідно визначити послідовність процесу управління прийняття рішень, для чого пропонується наступна ієрархія:

- цільова;

- функціональна;

- організаційно-технічна;

- інфологічна;

- інформаційна;

- алгоритм функціонування;

- фізичне і імітаційне моделювання елементарних завдань;

- математичний опис елементарних завдань.

Запропонована послідовність процесу управління прийняття рішень не є повною, бо вона не охоплює завдання управління CPPS, що розробляється.

Для забезпечення повного кортежу всіх необхідних цілей і параметрів, для досягнення завдань розробки сучасних CPPS пропонується розширити його за допомогою таких установок керування:

- цільова система управління;

- функціональна система управління;

- організаційна система управління;

- інформаційна система управління;

- фізична система управління;

- математична;

- алгоритми управління.

Ґрунтуючись на запропонованих параметрах управління і ухвалення рішень в даній послідовності, можна зробити висновки, що етапи носять складний багаторівневий характер, їх об'єднання дозволить окреслити древо розробки складних CPPS з розбиттям їх на рівні декомпозиції. Системне подання процесу управління розробкою складних CPPS представлене на рис. 2.1

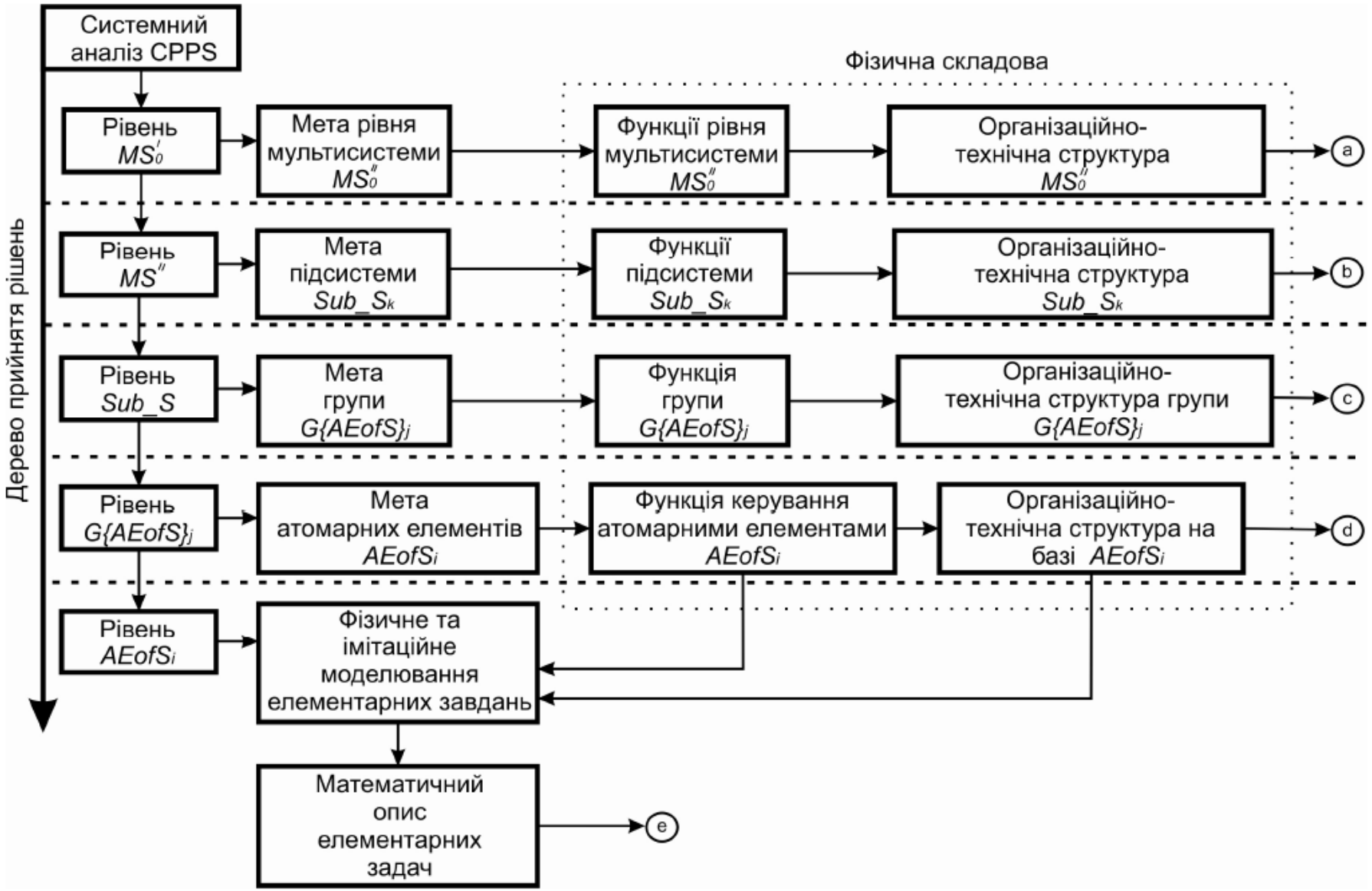


Рисунок 2.1 – Архітектурно-логічна модель автоматизації процесу управління розробкою складних CPPS

Рисунок 2.1, аркуш 2

2.2 Метод декомпозиції архітектурно-логічної моделі процесу управління розробкою кіберфізичних виробничих систем на початковому етапі

Кіберфізична виробнича система являє собою інтеграцію тісно взаємопов'язаних властивостей груп об'єктів: цільові, функціональні, структурні, інфологічні і алгоритму їх функціонування на кожному рівні декомпозиції. В даному дослідженні пропонується за вище перерахованими ознаками проводити декомпозицію метасистем в системи, між якими можуть існувати зв'язки з більш або менш вираженим ефектом.

Проводячи аналіз реального комп'ютерно-інтегрованого виробництва (КІВ) можна провести декомпозицію гнучкого автоматизованого заводу за функціонально-технічними ознаками на автоматизовані цехи з верстатами з ЧПУ, які тісно пов'язаними властивостями: технологія виробництва, технологічне обладнання і комп'ютерно-інтегроване управління. Однак, варто зауважити, що зв'язки між цехами, у порівнянні з внутрішньоцеховими, мають не настільки яскраво виражений ефект.

У свою чергу, декомпозицію автоматизованого цеху можна провести за такими ознаками: дільниці, які мають свої функціональні цілі; технологічні процеси і функціонування. Використовуючи метод діалектичного перенесення, з наведеного прикладу ознак декомпозиції метасистем, можна виділити ознаки, які дозволять декомпозувати метасистеми в системи:

- цільові;
- функціональні;
- організаційні;
- інфологічні;
- функціонування.

Основне завдання розробника CPPS – провести повний системний аналіз сучасних аналогів та виділити базові об'єкти, виходячи з головної мети розробки CPPS, визначити древо прийняття рішень.

В даному дослідженні фіксуються такі рівні декомпозиції розробки CPPS по MS'_0, MS''_0, Sub_S_k, $G\{AEofS\}_j$, $AEofS_i$, що дасть можливість провести повний аналіз і отримати древо побудови будь-якої CPPS, з урахуванням як кібернетичних, так і фізичних складових.

На початковому етапі розробки CPPS, в залежності від головної мети, необхідно побудувати древо прийняття рішень. Для цього розробник повинен провести декомпозицію головної мети розробки CPPS на підцілі. З цією метою проводиться повнофакторна науково-дослідна робота з аналізу головної мети, яка в більшості випадків містить якісні та кількісні характеристики у вигляді масиву вимог до технічних і інформаційних складових.

У відповідності до запропонованої архітектурно-логічної моделі автоматизації процесу управління розробкою складних CPPS, наведеної на рис. 2.1, можна визначити, що на 5 рівні розробки необхідно провести декомпозицію головної мети метасистеми (MS'_0) на підцілі, які формують цілі для кожної мультисистеми (MS''_0), за умов $MS'_0 \subset MS''_0$, які повністю відповідають головній меті розробки. Методика декомпозиції MS'_0 наступна:

- проводиться аналіз властивостей MS''_0;
- вивчаються і аналізуються зв'язки між групами MS''_0;
- досліджується можливість досягнення головної мети MS'_0;
- визначаються закономірності відносин між MS'_0 і MS''_0;
- аналітичним методом розраховуються якісні і кількісні показники вимог для кожної підцілі $(QandQIR\_MS''_0)$, при цьому вони можуть бути як поодинокі, так і об’єднані в групи;
- розглядається можливість існування співвідношень в групах MS''_0, які мають свої цілі. Дані співвідношення аналізуються і розраховується $QandQIR\_MS''_0$;
- будується графова модель за таким принципом: вузли – це значення $QandQIR\_MS''_0$ графа мети MS'_0, а співвідношення – ребра графа;

- проводиться перевірка правильності побудови $QandQIR\_MS''_0$ для MS''_0 і $QandQIR\_\ MS'_0$ для MS'_0 за допомогою розв'язання оберненої задачі суперпозиції параметрів;

- якщо в ході перевірки система параметрів підцілей MS''_0 складе кількісне значення параметрів головної мети MS'_0, то розробник може вважати, що проєктне рішення з декомпозиції головної мети MS'_0 розробки CPPS на цілі MS''_0, є правильними.

Узагальнений алгоритм запропонованого методу представлений на рис. 2.2. Для зручності розуміння позначимо через системне моделювання CPPS – процес моделювання за допомогою графа цілей, а побудова графа цілей системи – системної цільової моделі CPPS. Логічно, що запропоновані рішення повинні проводитися на кожному рівні процесу управління розробкою CPPS: системному (MS''), підсистемному (Sub_S), груповому ($G\{AEofS\}$) і за елементарними цілями на рівнях атомарних елементів системи ($AEofS$). Варто зауважити що подання MS'_0, через декомпозицію кожного рівня, підвищить розмірність графа цілей, що, в свою чергу, ускладнить зв'язки між усіма елементами.

Звідси виникає завдання адекватної обробки і аналізу цільової декомпозиції, особливо це буде видно на атомарних нижніх рівнях, для чого необхідним буде застосовування методів системного аналізу та комп'ютерні системні моделі, а графові системні цільові моделі будуть необхідні тільки для наочного подання процесу моделювання.

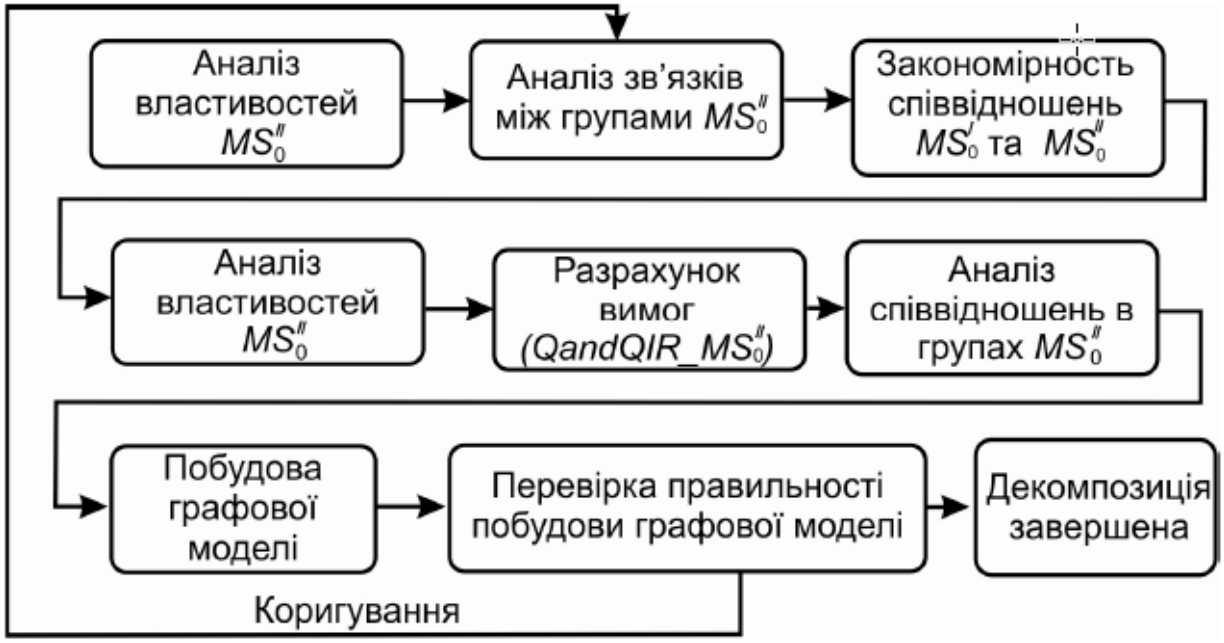


Рисунок 2.2 – Узагальнений алгоритм методу декомпозиції кіберфізичних систем як метасистеми (MS'_0)

2.3 Розробка методів прийняття рішень на рівні фізичної складової CPPS

2.3.1 Розробка методу прийняття рішень на функціональному етапі

У відповідності до запропонованої архітектурно-логічної моделі автоматизації процесу управління розробкою складних CPPS (рис. 2.1) першим йде етап подання та опису процесів управління розробкою фізичної складової CPPS. На цьому етапі розробляється комплекс завдань, за допомогою яких досягається головна мета розробки CPPS, який формується на базі апаратних засобів. Логічно зауважити, що всі рішення, які ухвалюватимуться на функціональному рівні, будуть виконуватися на всіх рівнях декомпозиції і будуть дзеркальним відображенням головної мети CPPS. В рамках даних досліджень пропонується наступний метод рішень на функціональному рівні:

- фіксується перший рівень декомпозиції на атомарному рівні (*AEofS*) (рис. 2.1);

- проводяться дослідження галузі знань (засоби досягнення мети функціонування фізичної складової CPPS) на цьому рівні;

- визначаються завдання або група завдань для досягнення поставленої мети. Тобто для кожної мети розробляються завдання, які дозволяють досягти мети, з урахуванням їх ефективності та запобігання надлишковості даних. Виходячи з цього залежність між цілями (Aim_i) і завданнями ($Task_j$) можна уявити у вигляді бінарних співвідношень:

$$Aim_i \cong Task_j. \tag{2.1}$$

У випадку існування декількох (n) Aim_i, або $Task_j$ бінарні співвідношення можна представити у вигляді математичного запису:

$$Aim_i \cong \underset{j=1}{\overset{n}{\Omega}} Task_j, \text{ де } j = 1 \ldots n \tag{2.2}$$

де Aim_i – цілі;

$Task_j$ – задачі.

- для кожного кількісного параметра мети *QandQIR* обираються атомарні елементи $AEofS_i$, для яких розраховується тактико-технічні характеристики завдання (*PCofT*), які повинні повністю відповідати заданим цілям та враховувати, що розрахунок буде необхідно обов'язково проводити за правилами опису явищ предметної області, а отже, функціонування об'єкта даного рівня розробки;

- проводиться побудова системної графової функціональної моделі CPPS за наступними принципами: вузли графа розташовуються у відповідності послідовності розроблених завдань ($Task_j$), а зв'язком між $Task_j$ виступають Aim_i у вигляді ребер графа. У результаті чого розробник отримує орієнтований функціональний граф для досягнення мети у вигляді завдань ($FCofTask_j$). Запропоноване рішення необхідне для контролю правильності

ухвалення рішень на функціональному етапі заданого рівня процесу управління розробкою CPPS;

- проводиться перевірка на функціональну повноту $Task_j$ для вирішення всіх поставлених Aim_i. Для цього порівнюються графи заданого рівня розробки і завдань цього рівня. Отже,можна стверджувати, що граф мети для MS''_0 (Aim_MS'') є «батьківським» для графа завдань MS''_0 ($Task\_MS''_0$), що дозволяє стверджувати про гомоморфізм графів $Aim\_MS''_0$ і $Task\_MS''_0$;

- розмічається системний функціональний граф параметрами $PCofI\_MS''_0$, що дозволить зробити системне моделювання декомпозиції $PCofI\_MS''_0$ на відповідність тактико-технічним вимогам мети $QandQIR\_MS''_0$ за критеріями ефективності та специфічних параметрів даної MS'_0. Варто зауважити, що особливістю моделювання зазначених критеріїв, є те, що його можливо провести на функціональному графі системи. Якщо результати моделювання $PCofI\_MS''_0$ повністю задовольняють заданим $QandQIR\_MS''_0$ і головній меті MS'_0, тоді можна з упевненістю вважати рішення на функціональному етапі системного рівня процесу управління розробкою CPPS правильним. Узагальнений алгоритм запропонованого методу представлений на рис. 2.3.

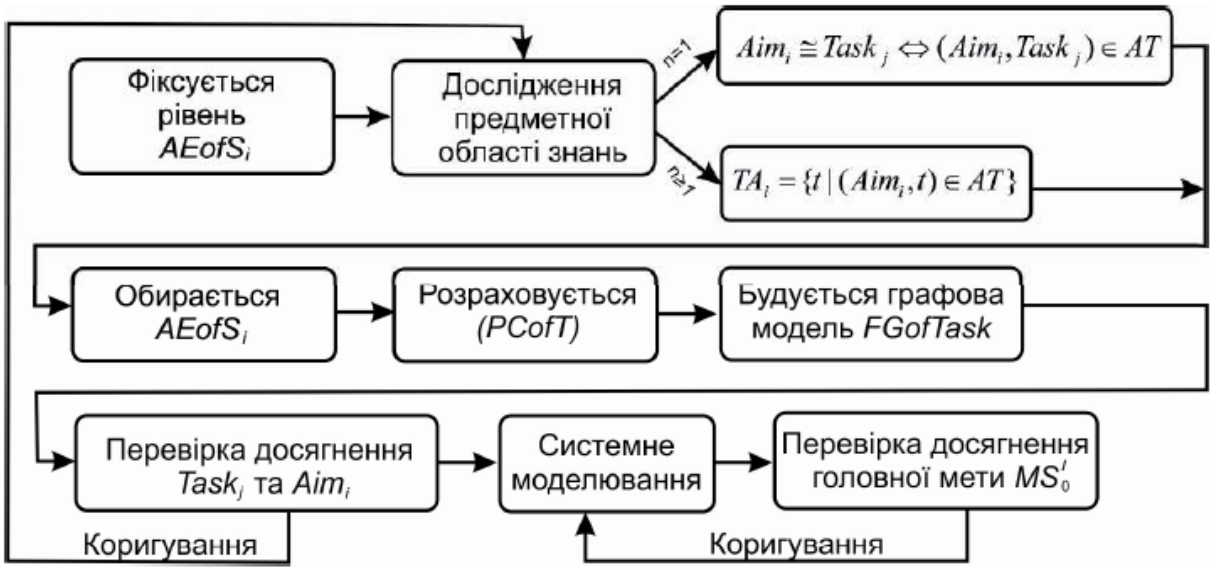


Рисунок 2.3 – Узагальнений алгоритм методу прийняття рішень на функціональному етапі

2.3.2 Розробка методу прийняття рішень на організаційно-технічному етапі

Розглянемо наступний етап розробки фізичної складової CPPS. Як можна бачити з рис. 2.1, даний етап визначає засоби (апаратні), на базі яких вирішується завдання, або комплекс завдань для досягнення головної мети розробки CPPS. Позначимо їх як організаційно-технічну структуру CPPS, на певному рівні декомпозиції (*Pattern_MS″0*), для рівня *MS″0*, що складається з елементів (*1…m*) даного рівня декомпозиції об'єкта. Уявімо запропонований структурний елемент *Pattern* у вигляді абстрактного процесу, який призначений для функціонального рішення *Task*, або групи $Task_j$ функціонування CPPS на даному рівні декомпозиції.

Як можна бачити з цього твердження розробник створює прообраз комплексу технічних засобів CPPS, на рівні фізичної складової, які функціонально здатні вирішувати необхідні завдання (Aim_i). Для досягнення проєктних рішень на організаційно-технічному етапі пропонується наступний метод:

- проводиться дослідження $Task_j\_MS''_0$ системного рівня розробки на предмет можливості бути реалізованим і досягти $Aim_i\_MS''_0$, а також аналізуються зв'язки у *FCofTask_MS″0* на предмет «сильних» і «слабких» зв'язків;

- на базі ознак «реалізованості» і «слабкості» зв'язків $Task_j\_MS''_0$ групуються з метою реалізації на своєму структурному елементі (*StrE_MS″0*) даного рівня розробки об'єкта. Це дасть можливість знайти відповідності між $Task_j\_MS''_0$ і *StrE_MS″0*, на яких вона може бути реалізована. Далі проводиться структурно-функціональний синтез об'єкту для досягнення необхідного рівня за специфічними особливостями (технічними даними), що властиві $AEofS_i$ або $G\{AEofS\}_j$. За результатами будується бінарне співвідношення:

$$Task_j\_MS''_0 \cong Pattern\_MS''_0 \qquad (2.3)$$

- на даному етапі методу, що розробляється, проводиться з'єднання $Pattern\_MS''_0$ в систему за допомогою ототожнення зв'язків між $Task_j\_MS''_0$, як всередині $AEofS_i$, так і між ними. Існуючі зв'язки між $Task_j\_MS''_0$ різних структурних елементів ($AEofS_i$) визначають зв'язки $Pattern\_MS''_0$, у результаті чого розробник отримує системну модель організаційно-технічної структури CPPS, яка відповідає заданому рівню процесу управління розробкою;

- аналізуються $PCofI\_MS''_0$ (тактико-технічні характеристики) на рівні MS''_0, які закріплені за кожним $StrE\_MS''_0$. За результатами даного аналізу розробник розраховує всі параметри тактико-технічних характеристик $Pattern\_MS''_0$ структурних елементів, що реалізують *Aim*. Модель розрахунків параметрів для $PCofI\_MS''_0$ і для будь-якого рівня визначаються закономірностям предметної області знань, до якої належить використаний розрахунковий процес, аналогічно розраховуються параметри продуктивності, надійності, точності для кожного *Pattern*;

- на останньому етапі проводиться моделювання $PCofI\_MS''_0$ на відповідність $TrandTR\_Aim\_MS''_0$ та критеріям ефективності декомпозиції структурних елементів. Отриманий результат моделювання повинен показати відповідність продуктивності, точності, надійності елементів, що піддані декомпозиції: $StrE\_MS''_0$ і $TrandTR\_Aim\_MS''_0$ і, як наслідок, можна зробити висновок, що прийняті рішення структурно-функціонального синтезу даного рівня декомпозиції відповідають ТЗ на CPPS.

Узагальнений алгоритм запропонованого методу представлений на рис. 2.4

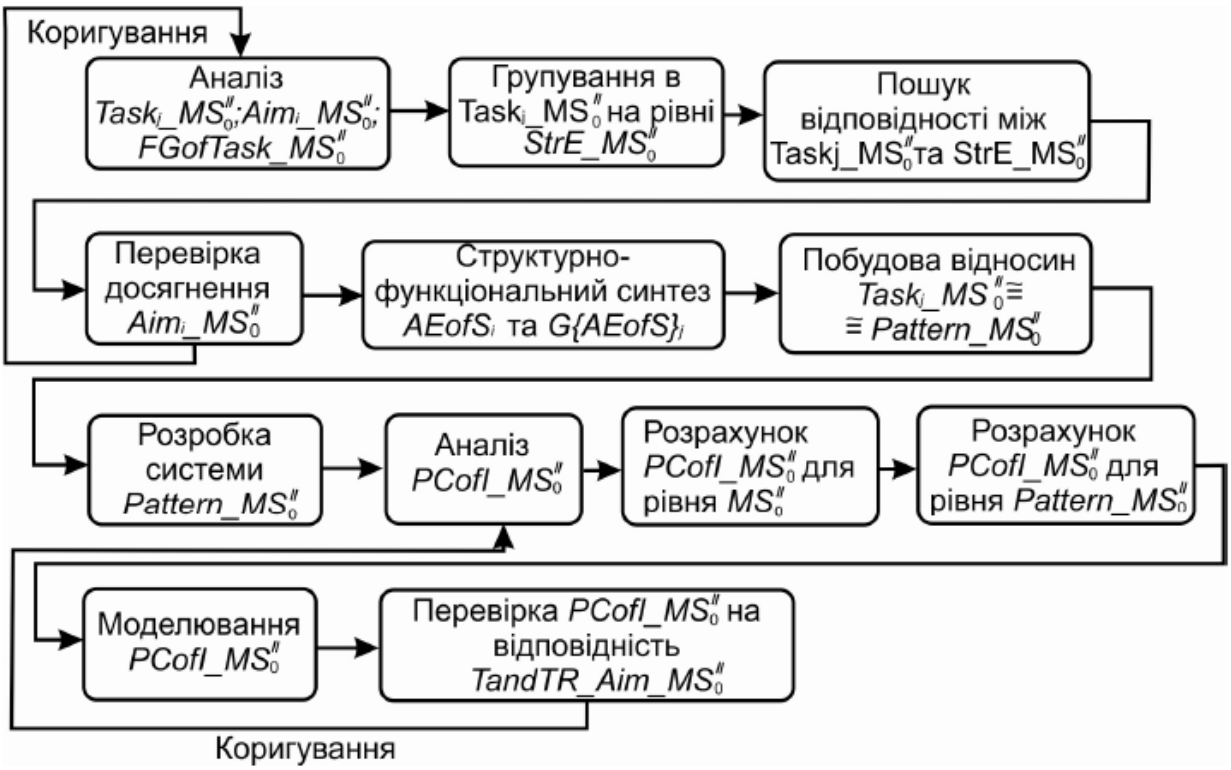


Рисунок 2.4 – Узагальнений алгоритм методу прийняття рішень на організаційно-технічному етапі

2.3.3 Розробка методу прийняття рішень на атомарному етапі

Сучасні CPPS – це складна метасистема, яка має на нижньому рівні декомпозиції (рівень $AEofS_i$ на рис. 2.1) множину технічних приладів (ПЛК, датчики, верстати з ЧПУ, і т.д.), здатних вирішити елементарні завдання, сукупність яких задовольняє вимогам ТЗ на CPPS. Наочним прикладом можуть виступити: верстат з ЧПУ, транспортний робот або робот-маніпулятор, Отже,можна відзначити, що елементарна задача – це виконання технологічної операції (основної, допоміжної, транспортної і т.д.), або технологічного переходу. Тому дослідження з вивчення атомарних елементів (*AEofS*), які можуть вирішувати елементарні завдання в даній методології, позначимо через побудову і аналіз фізичних моделей. У зв'язку з цим, для виконання прийнятих рішень на атомарному рівні декомпозиції «фізичного і імітаційного моделювання елементарних завдань» необхідно проводити відразу після прийняття рішень за *Aim* елементів об'єкта. Аналізується *Aim*

для *AEofS* і її *TrandTR*, в ході яких визначатимуться галузі фізики, до яких відносяться процеси в технічній системі по кожній елементарній *Aim*.

Будується фізична модель процесу по кожному атомарному елементу, з визначенням граничних умов, які представлені в специфікації та тактико-технічних вимогах, для досягнення елементарної мети при рішенні елементарних завдань. Вивчається мета атомарного елемента (*Aim_AEofS*), його тактико-технічні вимоги (*TrandTR_AEofS*) на даному рівні процесу управління та проводиться ідентифікація фізичної моделі процесу, який протікає всередині *AEofS*. Проводиться ряд експериментів за допомогою імітаційного, а потім і фізичного моделювання, в результаті якого уточнюються параметри *AEofS*. В результаті отримується масив даних, який містить тактико-технічні характеристики фізико-інформаційної моделі атомарного елемента (*PCofI_PIM_AEofS*) CPPS. Ґрунтуючись на запропонованому методі розробнику необхідно розробити фізико-інформаційну модель (*PIM*) для всіх *AEofS* і елементарних *Aim*. Узагальнений алгоритм методу прийняття рішень на атомарному етапі процесу управління розробкою CPPS представлено на рис. 2.5.

Отже,при підвищенні рівня розробки можна об'єднати елементарні *Aim* в систему групових цілей (*G{Aim}*), що дозволить об'єднати фізико-інформаційні моделі в групову систему ($PIM\_G\{AEofS\}_j$). Для зручності візуалізації позначимо вузлами графа атомарні фізико-інформаційні моделі (*PM_AEofS*), а ребрами графа виступатимуть зв'язки (фізичні та / або інформаційні) між *AEofS*.

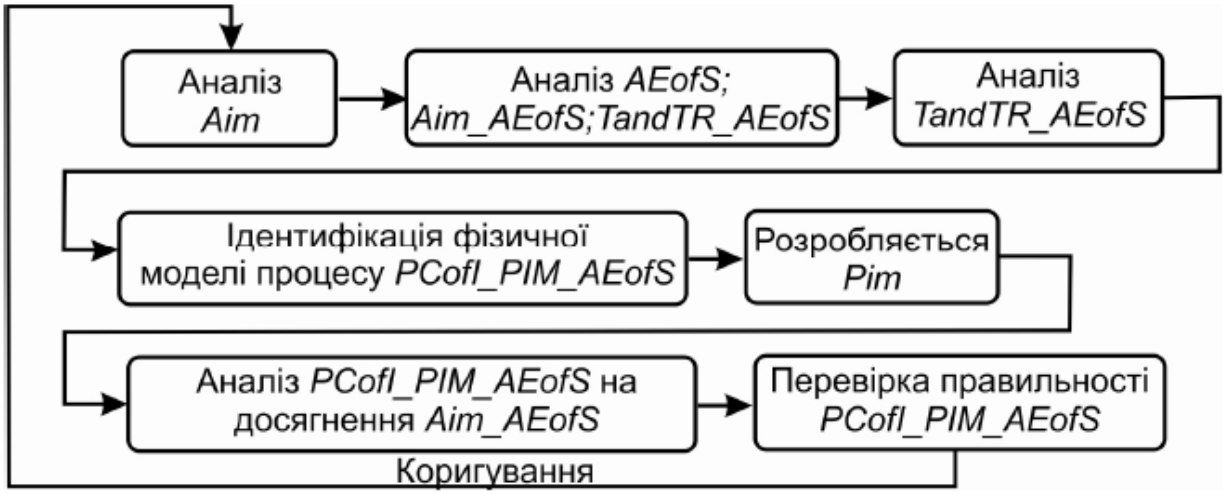


Рисунок 2.5 – Узагальнений алгоритм методу прийняття рішень на атомарному етапі розробки CPPS

Для представлення зв'язків в даному дослідженні вводяться такі типи: фізичний зв'язок (*Pcom*), інформаційний зв'язок (*Icom*), фізико-інформаційний зв'язок (*PIcom*), які описуються закономірностями, властивими предметній області знань.

Системна $PIM\_G\{AEofS\}_j$ з її тактико-технічними характеристиками дає можливість контролю правильності розробки, за допомогою тактико-технічних характеристик *AEofS*, підданих декомпозиції і їх відповідників для досягнення графової мети на даному рівні розробки CPPS.

Фізичні моделі елементарних цілей є основою для розробки математичних моделей елементарних цілей (MM_Aim_i), елементарних завдань (E_Task_j) і інформаційного алгоритму роботи (*LAW*).

2.4 Розробка методів ухвалення рішень на рівні кібернетичної складової CPPS

2.4.1 Розробка методу прийняття рішень на інфологічному етапі

Проводячи дослідження існуючих CPPS, які спеціалізуються на повній автоматизації складних технологічних процесів виробництва, велика увага приділяється її кібернетичній (комп'ютерно-інтегрованій) складовій, яка

об'єднує всі етапи технології і управління в єдине інформаційне середовище. В рамках даних досліджень пропонується процес функціонування CPPS розглянути як завдання, які вирішуються на структурних елементах, для досягнення головної мети CPPS, зазначених в ТЗ. Зробимо припущення, що ці завдання мають вихідні ($InputIP_i$) і вихідні ($OutIP_j$) інформаційні параметри.

Розглядаючи завдання верхніх ієрархічних рівнів можна сказати, що це будуть тільки інформаційні потоки, а на нижніх рівнях це будуть параметри матеріальних потоків атомарних елементів ($AEofS_i$) або групи атомарних елементів ($G\{AEofS\}_j$). Виходячи з особливостей розробки і функціонування CPPS пропонується наступні рішення на цьому етапі:

- проводиться аналіз і дослідження $InputIP_i$ і $OutIP_j$ для кожної $Task\_MS''_0$ системного рівня. Залежно від рівня MS''_0 параметри будуть носити «збільшений» характер. Розробник визначає об'ємні і тимчасові характеристики $InputIP_i$ і $OutIP_j$, а також об'ємні і часові характеристики $Pattern\_MS''_0$. Відстежується перетворення параметра $InputIP_i$ в $OutIP_j$ на базі рішення кожної $Task_j\_MS''_0$. Обираються математичні моделі і методи перетворення інформаційних параметрів, властивих предметній області $Task_j$, які адекватно описують даний процес;

- кожен $AEofS_i$ представляється у вигляді інформаційного перетворювача (*IC*). Вибирається і розраховується швидкодія, обсяги оброблюваної інформації, функціональна точність і необхідність перетворення інформації;

- проводиться об'єднання $InputIP_i$ всіх завдань, що вирішуються на одному $AEofS_i$ у вигляді вхідного інформаційного каналу (*InputCanal*), а $OutIP_j$ у вигляді вихідного інформаційного каналу (*OutCanal*) з $AEofS_i$;

- об'єднуючи *InputCanal* і *OutCanal* різних $AEofS_i$ за ознаками системного зв'язку отримуємо, що вихідні параметри $OutCanal_i\_Task_j$ будуть служити $InputCanal_{i+1}Task_j$, що дозволить розробнику сформувати системну інфологічну модель CPPS на системному рівні розробки;

- проводиться аналіз параметричної сумісності всіх вихідних і вихідних параметрів $Task_j\_G\{AEofS_i\}$, розв'язуваних на IC, для перевірки правильності побудови системної інфологічної моделі;

- на останньому етапі рекомендується провести моделювання інфологічної моделі CPPS системного рівня за наступними мінімальними критеріями: продуктивність, точність, надійність, а також обов'язково на відповідність $TrandTR\_Aim_i\_G\{AEofS_i\}$. Якщо отримані результати повністю задовольняють головній меті і ТЗ на розробку CPPS, то всі прийняті рішення на інфологічному етапі є правильними. Узагальнений алгоритм методу прийняття рішень на інфологічному етапі представлений на рис. 2.6.

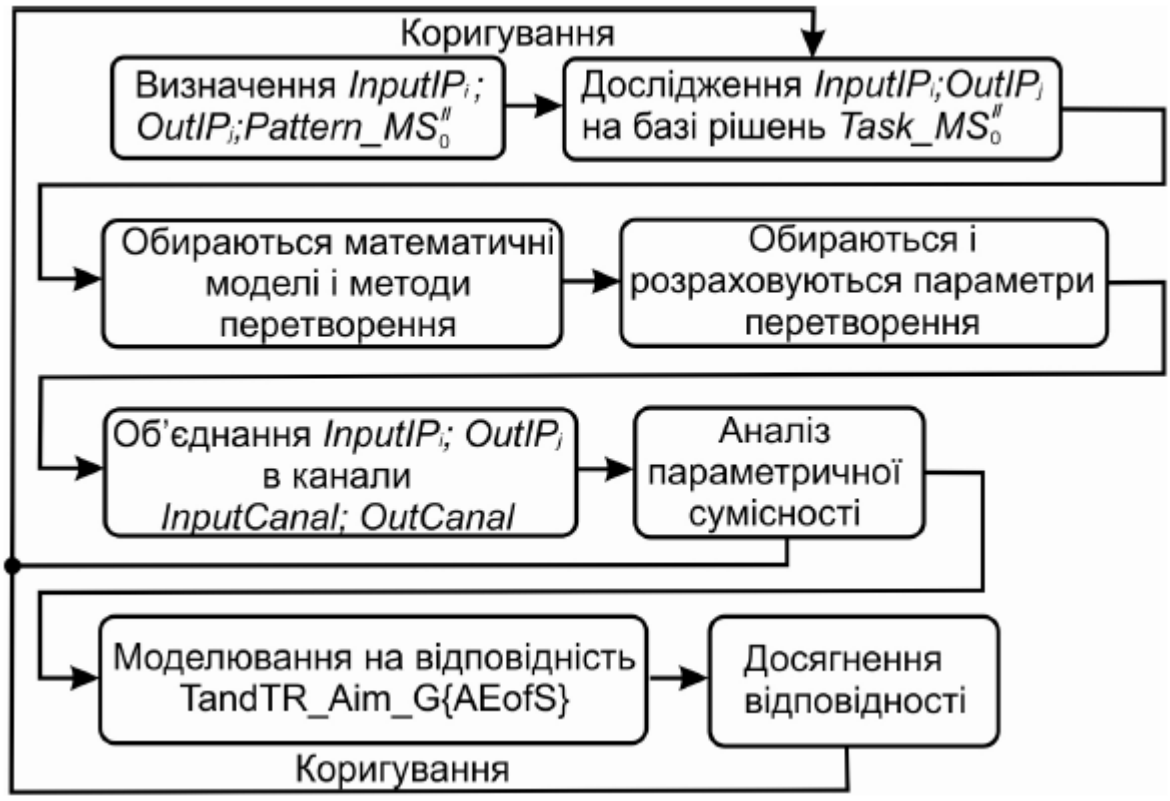


Рисунок 2.6 – Узагальнений алгоритм методу прийняття рішень на інфологічному етапі

2.4.2 Розробка методу прийняття рішень на інформаційному етапі

Розробка прийняття рішень на інформаційному етапі є одним з важливих етапів процесу управління розробкою CPPS, який забезпечує функціонування

та інформаційну взаємодію між інфологічим і алгоритмічним рівнями. На базі даного етапу розробником приймаються рішення, які прямо впливають на адекватність і якість розробки всієї кібернетичної складової CPPS. Даний рівень забезпечує функціонування і розвиток інформаційного простору CPPS, а також засобів інформаційної взаємодії не тільки на кожному рівні розробки, але і несе в собі рішення щодо візуалізації і відображення (HMI, GUI) всієї необхідної інформації на рівнях функції c-MES.

На даному етапі розробнику CPPS необхідно реалізувати всі інформаційні потоки на кожному рівні щодо забезпечення досягнення головної мети розробки CPPS відповідно до вимог зазначених в ТЗ. Як результат, в рамках даного дослідження, пропонується на даному етапі зробити декомпозицію методів «знизу-вгору», що обумовлено наступними завданнями:

- розробник повинен володіти повними тактико-технічними характеристиками кожного атомарного елемента системи (*PCofI_AEofS*$_i$), з точки зору його інформаційної складової, а також кількості *OutIP*$_j$, *InputIP*$_i$, *PCofI_OutIP*$_j$, *PCofI_InputIP*$_i$;

- робочий алгоритми вирішення елементарних завдань базується на всіх *AEofS*$_i$, які досягають *Aim*$_i$*_AEofS*$_i$ для кожного атомарного елементу;

- інформацію вхідних *InputCanal_AEofS*$_i$ і вихідних *OutCanal_AEofS*$_i$ каналів забезпечують зв'язки всередині групової *G{AEofS*$_i$*}*, що дозволяє досягти *Aim*$_i$*_G{AEofS*$_i$*}* і т.д.

Виходячи з вище перерахованих завдань, в даному дослідженні, пропонується наступний метод ухвалення рішень на даному етапі:

- проводиться аналіз *OutIP*$_j$, *InputIP*$_i$, *PCofI_OutIP*$_j$, *PCofI_InputIP*$_i$, для кожного *AEofS*$_i$;

- перевіряють і моделюють інформацію вхідних *InputCanal_AEofS*$_i$ і вихідних *OutCanal_AEofS*$_i$ каналів *AEofS*$_i$;

- *AEofS*$_i$ об'єднують у необхідну групу *G{AEofS*$_i$*}* атомарних елементів *AEofS*$_i$, які відповідають *Task*$_j$*_G{AEofS*$_i$*}* і дозволяють досягти

$Aim_i\_G\{AEofS_i\}$, а також перевіряють інформаційну сумісність $PCofI\_G\{AEofS_i\}$ всередині кожної $G\{AEofS_i\}$;

- проводиться моделювання досягнення вирішення задач, або завдань для кожної $Aim_i\_G\{AEofS_i\}$ і перевіряється досягнення правильності рішення, точності і швидкодії необхідних (Sub_S_k) на вході наступного рівня;

- виходячи з отриманих результатів моделювання розробник приймає рішення про успішність правильності побудови інформаційної структури рівня Sub_S_k;

- використовуючи отримані результати розробник синтезує послідовність інформаційних зв'язків елементів рівня Sub_S_k для досягнення вимог до вхідної інформації на рівень MS''_0, яка є необхідною для досягнення $Aim_i\_MS''_0$, $Task_j\_MS''_0$ на даному рівні розробки;

- на останньому рівні розробнику необхідно розробити всі інформаційні зв'язки між елементами MS''_0, врахувати всі канали зв'язків і їх тактико-технічні характеристики, що в сумі надасть можливість досягти головної мети розробки CPPS, з урахуванням вимог ТЗ.

2.4.3 Розробка методу прийняття рішень на алгоритмічному етапі

Алгоритмічний етап є одним з важливих складових при розробці CPPS, в ході якого на базі отриманих результатів розробляється комплекс програмного забезпечення від програм управління ПЛК, SCADA систем до ERP і MES, які в сукупності повинні забезпечити повний рівень автоматизації на кожному етапі виробництва. Розробник повинен повністю розробити алгоритми роботи на кожному рівні і забезпечити інформаційну сумісність кожного елемента від атомарного до елементів візуалізації і прийняття рішень на найвищому рівні ієрархії, при цьому не допустити втрати інформаційних каналів або їх несумісності і досягти головної мети розробки CPPS, заданої в ТЗ. Для опису прийняття рішень на алгоритмічному етапі, в даному дослідженні, пропонується словесно-формульний опис послідовності дій розробника CPPS.

Позначимо під алгоритмом функціонування CPPS наступну послідовність виконання: $Task_j\_MS''_0$ на $StrE\_MS''_0$, з урахуванням $InputCanal\_MS''_0$ і $OutCanal\_MS''_0$, для виконання $Aim_i\_MS''_0$ функціонування CPPS на даному рівні розробки. Уявімо алгоритм у вигляді наступного виразу:

$$AF\_MS''_0 = f(Task_j\_MS''_0, StrE\_MS''_0, InputCanal\_MS''_0, OutCanal\_MS''_0, t) \quad (2.4)$$

де $AF\_MS''_0$ – алгоритм функціонування CPPS на рівні MS''_0;

$Task_j\_MS''_0$ – завдання на рівні MS''_0;

$StrE\_MS''_0$ – структурні елементи на рівні MS''_0;

$InputCanal\_MS''_0$ і $OutCanal\_MS''_0$ – вхідні і вихідні системні канали рівня MS''_0;

t – час.

На початковому етапі розробки $AF\_MS''_0$ необхідні рішення з декомпозиції CPPS за:

- $Aim_i\_MS''_0$ (цілі на рівні MS''_0);
- $PCofI\_Aim_i\_MS''_0$ (тактико-технічні характеристики цілі на рівні MS''_0);
- $Task_j\_MS''_0$ (завдання на рівні MS''_0);
- $PCofI\_Task_j\_MS''_0$ (тактико-технічні характеристики завдань на рівні MS''_0);
- $StrE\_MS''_0$ (структурні елементи рівня MS''_0);
- $PCofI\_StrE\_MS''_0$ (тактико-технічні характеристики елементів рівня MS''_0);
- $InputCanal\_MS''_0$ і $OutCanal\_MS''_0$ (вхідні та вихідні системні канали рівня MS''_0 і зв'язки між ними);
- $PCofI\_InputCanal\_MS''_0$ і $PCofI\_OutCanal\_MS''_0$ (тактико-технічні характеристики вихідних і вихідних каналів рівня MS''_0).

Основною і єдиною метою розробки є синтез алгоритму функціонування (*AF*) для досягнення $Aim_i\_MS''_0$ на рівні MS''_0. В даному дослідженні пропонується наступна послідовність рішень:

- аналізується початковий стан: MS''_0, $StrE\_MS''_0$, $InputCanal\_MS''_0$, $OutCanal\_MS''_0$ і по $PCofI\_StrE\_MS''_0$, $PCofI\_InputCanal\_MS''_0$, $OutCanal\_MS''_0$ і визначається готовність до функціонування;

- аналізується $Aim_i\_MS''_0$, $PCofI\_Aim_i\_MS''_0$ та їх взаємодія для досягнення генеральної мети розробки CPPS, яка вказана в ТЗ. На базі аналізу розробник будує граф цільової моделі;

- проводитися аналіз $Task_j\_MS''_0$ і визначається послідовність для досягнення $Aim_i\_MS''_0$;

- перевіряється сумісність $PCofI\_Aim_i\_MS''_0$ і $PCofI\_StrE\_MS''_0$ при послідовному їх проходженні по структурних елементах;

- аналізується і перевіряється сумісність $PCofI\_InputCanal\_MS''_0$ і $PCofI\_OutCanal\_MS''_0$ з $PCofI\_StrE\_MS''_0$;

- розробляється послідовність виконання $Task_j\_MS''_0$ на базі $StrE\_MS''_0$ з урахуванням $InputCanal\_MS''_0$ і $OutCanal\_MS''_0$ та просторових характеристик $PCofI\_Task_j\_MS''_0$, $PCofI\_StrE\_MS''_0$, $PCofI\_InputCanal\_MS''_0$ і $PCofI\_OutCanal\_MS''_0$;

- будується граф алгоритму функціонування рівня MS''_0, де вузлами виступають $Task_j\_MS''_0$. Зв'язками між $Task_j\_MS''_0$ виступають $InputCanal\_MS''_0$ та $OutCanal\_MS''_0$, які будуть ребрами даного графа. Таким чином розробник отримує графову модель алгоритму функціонування CPPS рівня.

- розробник проводить аналіз на графі алгоритму функціонування за допомогою статичного моделювання за критеріями ефективності розроблювального CPPS рівня MS''_0 і перевіряє досягнення $Aim_i\_MS''_0$;

- використовуючи методи імітаційного моделювання та програмне забезпечення розробник перевіряє розроблену графову модель алгоритму функціонування CPPS рівня MS''_0 під навантаженням, якщо результати

моделювання задовольняють умовам головної мети і вимогам ТЗ, то можна вважати прийняті рішення по розробці графової моделі правильними.

Розроблений метод дає можливість комплексувати рішення за цільовою, функціональною, інформаційною, алгоритмічною сумісністю з метасистемою, яка представляє собою складні сучасні CPPS.

Розглядаючи сучасні CPPS як складну багаторівневу метасистему пропонується використовувати вище представлені словесно-формульні описи для розробки рішень на алгоритмічному етапі для всіх рівнів декомпозиції і розглядати їх як певну кількість підсистем (Sub_S_k), при умові, що $MS''_0 \subset Sub\_S_k$. Отже, з цього можна визначити, що прийняті рішення розробником здійснюються так само як за цільовою, функціональною організаційно-технічною структурами на інфологічному і алгоритмічному етапах. Ґрунтуючись на (2.4) можна провести декомпозицію $Aim_i\_MS''_0$ на підцілі $Aim_i\_Sub\_S_k$ з побудовою графа алгоритму $Aim_i\_MS''_0$ на алгоритмічному етапі. Для цього розробник може уявити $Aim_i\_MS''_0$ як:

$$AF\_Aim_i\_MS''_0 = f(Aim_i\_Sub\_S_k, t) \tag{2.5}$$

де $AF\_Aim_i\_MS''_0$ – алгоритм досягнення мети на рівні MS''_0;

$Aim_i\_Sub\_S_k$ – цілі на рівні підсистем Sub_S_k;

t – час.

Для цього розробнику необхідно знати всі $PCofI\_Aim_i\_Sub\_S_k$, розрахунок критеріїв ефективності кожної складової, надійність і результати системного моделювання всіх прийнятих рішень на кожному підсистемному рівні. Так само можна уявити алгоритмічний опис і на функціональному етапі кожної підсистеми:

$$AF\_Aim_i\_Sub\_S_k = f(Task_j\_Sub\_S_k, t) \tag{2.6}$$

де $AF\_Aim_i\_Sub\_S_k$ – алгоритм досягнення мети підсистеми Sub_S_k;

$Task_j\_Sub\_S_k$ – завдання підсистеми Sub_S_k;

t – час.

Розробник повинен провести розрахунок основних критеріїв ефективності по $PCofI\_Aim_i\_Sub\_S_k$ за розробленим алгоритмічним описом і на функціональному етапі за допомогою моделювання довести правильність прийнятих рішень, які відповідають головній меті CPPS.

На організаційно-технічному етапі розробник визначає структурні елементи підсистеми ($Pattern\_Sub\_S_k$), аналізує і розраховує $PCofI\_Pattern\_Sub\_S_k$, на базі якої будує структурну модель системи підсистем:

$$AF\_Patten\_Sub\_S_k = f(Patten\_Sub\_S_k, t) \qquad (2.7)$$

де $AF\_Pattern\_Sub\_S_k$ – алгоритм моделі підсистеми рівня Sub_S_k;

$Pattern\_Sub\_S_k$ – організаційно-технічна структура рівня Sub_S_k;

t – час.

Розробник на базі (2.7) проводить системне моделювання правильності прийнятих рішень, які показують досягнення необхідних цілей MS''_0 за допомогою сукупності Sub_S_k на базі їх $PCofI\_Pattern\_Sub\_S_k$ і з урахуванням $PCofI\_InputCanal\_Sub\_S_k$, $PCofI\_OutCanal\_Sub\_S_k$.

Для забезпечення процесу управління розробкою кібернетичної складової CPPS в даному дослідженні пропонується наступний алгоритмічний опис прийняття рішень на інфологічному етапі:

- розробником аналізуються і визначаються підсистемні канали $InputCanal\_Sub\_S_k$ та $OutCanal\_Sub\_S_k$;

- виділяються і аналізуються необхідні $PCofI\_InputCanal\_Sub\_S_k$, $PCofI\_OutCanal\_Sub\_S_k$;

- перевіряються $PCofI\_InputCanal\_G\{AEofS\}_j$ і $PCofI\_OutCanal\_G\{AEofS\}_j$ кожної групи;

- будується інфологічна модель системи на базі:

$$AF\_InputCanal\_Sub\_S_k = f(InputCanal\_G\{AEofS\}_j, PCofI\_InputCanal\_G\{AEofS\}_j, t) \quad (2.8)$$

$$AF\_OutCanal\_Sub\_S_k = f(OutCanal\_G\{AEofS\}_j, PCofI\_OutCanal\_G\{AEofS\}_j, t) \quad (2.9)$$

де *AF_InputCanal_Sub_S*$_k$ – алгоритм функціонування *InputCanal* підсистеми рівня *Sub_S*$_k$;

InputCanal_G{AEofS}$_j$ – вхідний канал *G{AEofS}*$_j$ з притаманними йому каналами рівня атомарних елементів *G{AEofS}*$_j$;

PCofI_InputCanal_G{AEofS}$_j$ – тактико-технічні характеристики вхідного каналу групи атомарних елементів *G{AEofS}*$_j$;

AF_OutCanal_Sub_S$_k$ – алгоритм функціонування *OutCanal* підсистеми рівня *Sub_S*$_k$;

OutCanal_G{AEofS}$_j$ – вихідний канал *G{AEofS}*$_j$ з притаманними йому каналами рівня атомарних елементів *G{AEofS}*$_j$;

PCofI_OutCanal_G{AEofS}$_j$ – тактико-технічні характеристики вихідного каналу групи атомарних елементів *G{AEofS}*$_j$;

t – час.

Після побудови інфологічної моделі розробником розраховується критерій ефективності підсистемних каналів і проводитися моделювання правильності побудови інфологічної структури рівня *Sub_S*$_k$.

- на останньому етапі розробником CPPS будується алгоритм функціонування на підсистемному рівні MS''_0.

В рамках даних досліджень пропонуються наступні рішення:

$$AF\_MS''_0 = f(Aim_i\_Sub\_S_k, StrE\_Sub\_S_k, OutCanal\_Sub\_S_k, InputCanal\_Sub\_S_k, t) \quad (2.10)$$

де $AF\_MS''_0$ – алгоритм функціонування підсистеми рівня MS''_0;

$Aim_i\_Sub\_S_k$ – досягнення мети підсистеми рівня Sub_S_k;

$StrE\_Sub\_S_k$ – вхідні канали рівня Sub_S_k;

$OutCanal\_Sub\_S_k$ – вихідні канали рівня Sub_S_k;

$InputCanal\_Sub\_S_k$ – вхідні канали рівня Sub_S_k.

t – час.

Після побудови $AF\_MS''_0$ обов'язково проводиться його дослідження з метою виявлення досягнення $Aim_i\_MS''_0$, шляхом моделювання за просторово-тимчасовими характеристиками і критеріям ефективності при навантаженні.

Виходячи із запропонованих рішень (2.1-2.10) розробник CPPS на підсистемному рівні декомпозиції проводить комплексування в єдину систему по всіх рівнях з урахуванням тактико-технічних характеристик їх підсистемних рівнів, які повинні відповідати тактико-технічними умовам мети та завданню даного рівня. Варто звернути увагу, що запропоновані методи прийняття рішень можна поділити на два типи:

- моделювання декомпозованих елементів нижнього рівня на відповідні елементи вищого рівня з урахуванням їх тактико-технічних характеристик;

- побудова моделі MS'_0 як сукупності елементів підсистемних рівнів від першого до четвертого, для перевірки досягнення головної мети розробки CPPS та відповідності ТЗ.

Ґрунтуючись на вище запропонованих методах прийняття рішень розробнику необхідно провести декомпозицію від п'ятого до першого атомарного рівня процесу управління розробкою CPPS з отриманням на кожному етапі відповідних тактико-технічних характеристик і досягнення задач і мети, які задовольняють вимогам даного рівня і прагнути до досягнення головної мети і параметрів, зазначених в ТЗ.

2.4.4 Розробка методу прийняття рішень на рівні математичного опису елементарних завдань

Виходячи із запропонованої архітектурно-логічної моделі автоматизації процесу управління розробкою складних CPPS (рис. 2.1) можна бачити, що блок «Математичного опису елементарних завдань» впливає на кібернетичну складову CPPS, що розробляється. Це пов'язано з тим, що математичний опис елементарних завдань, розв'язуваних на атомарних елементів (), в контексті даних досліджень можуть виступати апаратні складові вигляді ПЛК, датчиків, одноплатних комп'ютерів і т.д.

Визначимо, що під математичною моделлю атомарного елемента $AEofS_i$ розуміється математичне подання фізичного процесу, що протікає в ньому, в термінах математичних рівнянь і їх послідовності різних балансів фізичних моделей. Результати математичних моделей можуть бути використані розробником при проєктуванні технологічного процесу, який напряму може бути пов'язаний з інформаційною складовою у вигляді псевдокоди для програм з ЧПУ, які забезпечують вимоги даної задачі.

Позначимо елементарну задачу через найпростіший одиничний перехід (основний, допоміжний, транспортний і т.д.), який є невід'ємною частиною технологічної операції. З цього можна з упевненістю заявити, що математичні моделі, які використовують фізичну природу свого процесу, можуть бути представлені вигляді математичного подання, яке можна перевірити за допомогою математичного моделювання з використанням пакетів MatCAD, MatLab і т.д.

Результати моделювання показують можливість досягнення мети поставлені перед елементарними завданнями, а отримані математичні описи виступатимуть основою для розробки алгоритму і програм управління, моніторингу перебігу технологічного процесу в часі. Ґрунтуючись на цьому пропонується наступний метод прийняття рішень на даному блоці:

- на першому етапі розробник повинен провести аналіз і дослідити результати фізичного та імітаційного моделювання елементарних завдань (Elm_Task_j) на наявність досягнення елементарної мети (Elm_Aim_i);

- за отриманими результатами розробник виявляє деякі фізичні закономірності, при яких об'єкт досягає заданих вимог Elm_Task_j;

- на базі виявлених закономірностей, аналізуються і підбираються математичні апарати, від найпростіших (системи лінійних рівнянь) до найскладніших (інтегро-диференційних рівнянь). Розробнику необхідно підібрати ефективний математичний апарат, який відповідає вимогам точності, надійності, часу виконання і збіжності результатів;

- проводиться ідентифікація математичної моделі на параметри фізичної моделі об'єкта, проводиться моделювання при різних припустимих значеннях і вимог тактико-технічних характеристик, що дає можливість визначити фізичний діапазон вхідних і вихідних параметрів;

- розробник ідентифікує всі математичні моделі і виявляє фізичний діапазон їх вхідних і вихідних параметрів;

- ґрунтуючись на ідентифікованих математичних моделях, розробник формує список апаратних атомарних елементів ($AEofS_i$), які за тактико-технічними характеристиками дозволяють вирішувати Elm_Task_j, при цьому на один $AEofS_i$ може вирішуватися кілька Elm_Task_j, кожна з яких досягає Elm_Aim_i;

- для кожного обраного $AEofS_i$ розробляється алгоритм вирішення елементарних завдань (рис. 2.1, продовження), які в майбутньому дозволяють розробити інформаційну структуру і алгоритм роботи ПЗ, в залежності від типу і вимог $AEofS_i$.

2.5 Технологія процесу управління розробкою кіберфізичних виробничих систем

Запропоновані в підрозділах 2.2-2.4 методи декомпозиції основних етапів процесу управління розробкою CPPS, що дозволяють прийняти рішення на кожному кроці, але при цьому не показують загальну технологію підходу до вирішення мети даного дослідження, а лише вирішують завдання розробки на етапах.

Отже,наступному кроком в цій роботі є розробка технології управління процесом розробки CPPS. Цей крок є логічно пов'язаною послідовністю ухвалення рішень розробником, на основі методів декомпозиції, у відповідності з древом і етапами розробки (рис. 2.7.).

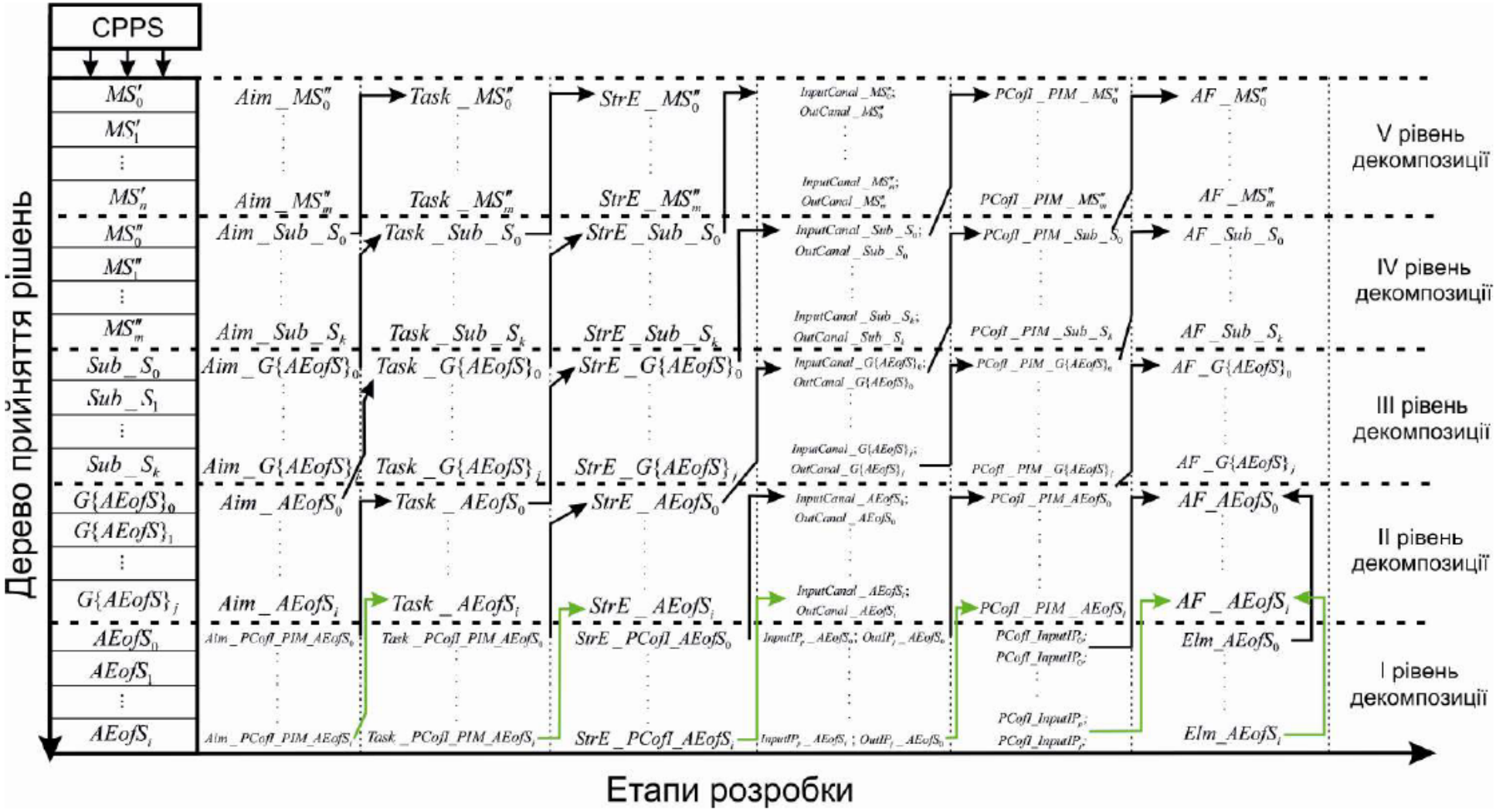


Рисунок 2.7 – Технологія процесу управління розробкою CPPS

В основі запропонованої технології лежить принцип «згори-вниз» і «зліва-направо». Такий вибір обумовлено складною ієрархічною структурою будь-якої сучасної промислової CPPS. Отже, на початковому етапі необхідно вирішити завдання системного підходу до технологій процесу управління розробкою CPPS. Ґрунтуючись на даному припущенні розробнику необхідно визначити п'ятий рівень ієрархії CPPS, який ґрунтується на критеріях головної мети і вимог ТЗ. Вибір критеріїв визначення п'ятого рівня декомпозиції, в ході проведення системного аналізу, проводиться на базі методу, запропонованого в підрозділі 2.2.

Розробник виділяє необхідний рівень ієрархії процесу управління розробки і проводить декомпозицію метасистеми MS'_0 на мультисистеми MS''_0 і фіксує цей рівень.

Це дозволяє приймати рішення на даному рівні в суворій послідовності, використовуючи принцип «зліва-направо» у відповідності з запропонованими етапами розробки (рис. 2.1) та врахуванням послідовності рішень від m до $m+n$. Отже, структуру технології процесу управління розробки CPPS можна візуалізувати у вигляді систематизованого дерева рішень на етапі декомпозиції MS'_0, який представлений на рисунку 2.8.

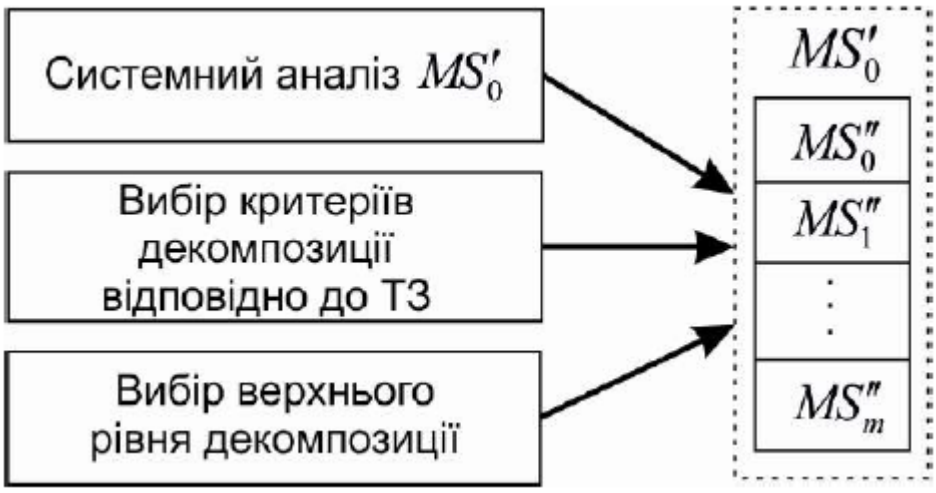


Рисунок 2.8 – Древа рішень на етапі декомпозиції MS'_0

Після виділення рівня декомпозиції розробки MS'_0, розробник проводить декомпозицію MS'_0 на $Aim_i\_MS''_0$, на базі головної мети ТЗ. Наступним кроком є виявлення і обґрунтування всіх $PCofI\_MS''_0$, на базі яких здійснюється побудова цільової моделі MS'_0 і проводитися перевірка правильності, а також оцінка ефективності прийняття рішень за допомогою статичного або динамічного моделювання, яке повинно показати досягнення головної мети розробки MS'_0.

У відповідності до запропонованої технології процесу управління розробки CPPS (рис.2.7) розробник може перейти до наступного етапу – розробка функціональної моделі. На цьому етапі знаходиться відповідність для кожної підцілі ($Aim_i\_MS''_0$) свого завдання ($Task_j\_MS''_0$), або групи завдань. Для кожного завдання $Task_j\_MS''_0$ розробляються $PCofI\_Task_j\_MS''_0$, на базі яких будується функціональна модель досягнення $Aim_i\_MS''_0$. Дерево рішень декомпозиції на функціональному етапі представлене на рис. 2.9.

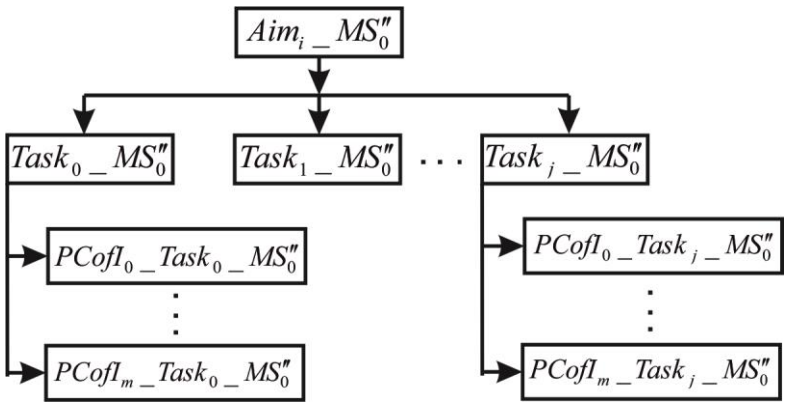


Рисунок 2.9 – Древо рішень декомпозиції на функціональному етапі

Методом системного моделювання на $Task_j\_MS''_0$ розробник проводить статистичне моделювання $PCofI\_Task_j\_MS''_0$ функціональних елементів MS''_0, які піддавались декомпозиції. Якщо результати моделювання відповідають головній меті MS'_0, то можна вважати, що рішення на функціональному етапі декомпозиції завдань і тактико-технічних характеристик, розроблені адекватно і розробник може переходити до етапу організаційно-технічної розробки.

На цьому етапі розробник розробляє структурні елементи системного рівня $StrE\_MS''_0$, проводить дослідження і вибір їх $PCofI\_StrE\_MS''_0$ в залежності від вимог $PCofI\_Task_j\_MS''_0$. Будується системна структурна модель CPPS рівня MS''_0 і проводитися статичне моделювання $PCofI\_StrE\_MS''_0$ для досягнення $Aim_i\_MS''_0$.

Дерево рішень декомпозиції на організаційно-технічному етапі представлений на рис. 2.10.

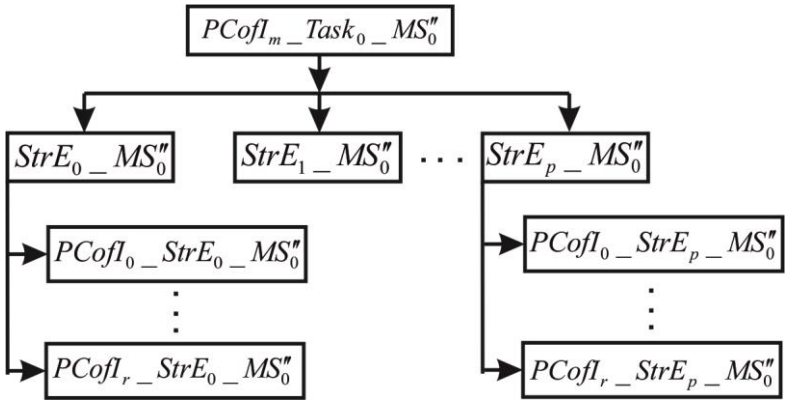


Рисунок 2.10 – Дерево рішень декомпозиції на організаційно-технічному етапі

Якщо результати моделювання задовольняють $PCofI\_StrE\_MS''_0$ і $Aim_i\_MS''_0$ – можна вважати, що рішення на даному рівні відповідають головній меті розробки CPPS і вимогам ТЗ. Отже, розробник може приступити до проєктування інфологічного етапу.

Ґрунтуючись на методі розробки, представленого в підрозділі 2.4.1, розробник проводить аналіз $StrE\_MS''_0$ і визначає $InputIP_i$ і $OutIP_j$ для кожного структурного елементу, що вимагає дослідження інфологічних перетворювачів параметрів $IC\_MS''_0$ і системних канали зв'язків між ними $InputCanal\_MS''_0$ і $OutCanal\_MS''_0$, в тому числі $PCofI\_IC\_MS''_0$, $PCofI\_InputCanal\_MS''_0$, $PCofI\_OutCanal\_MS''_0$.

На базі проведеного аналізу будується інфологічна модель CPPS системного рівня MS''_0 і проводиться статичне моделювання на відповідність $PCofI\_IC\_MS''_0$, $PCofI\_InputCanal\_MS''_0$, $PCofI\_OutCanal\_MS''_0$ і $Aim_i\_MS''_0$.

При відповідності $PCofI\_IC\_MS''_0$, $PCofI\_InputCanal\_MS''_0$ та $PCofI\_OutCanal\_MS''_0$ головній меті розробки MS'_0, розробник може вважати, що прийняті рішення на даному етапі задовольняють ТЗ на CPPS. Древо рішень декомпозиції на інфологічному етапі представлений на рисунку 2.11.

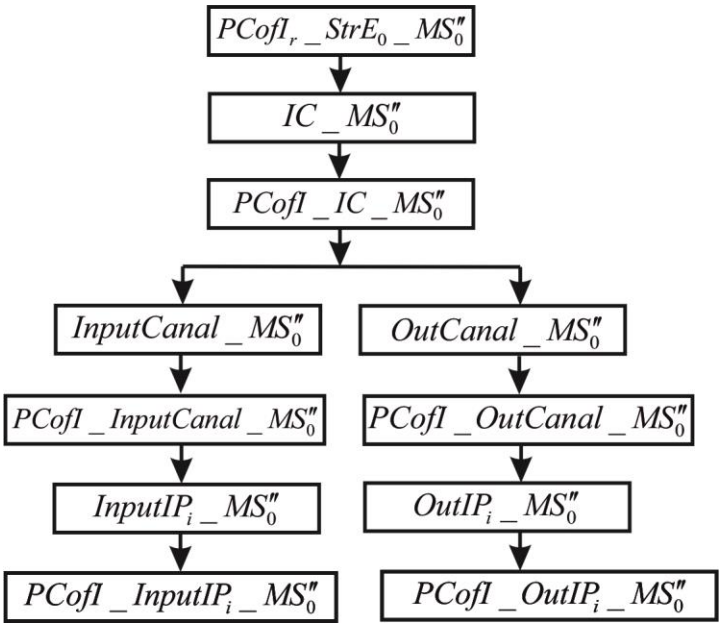


Рисунок 2.11 – Дерево рішень декомпозиції на інфологічному етапі

Ґрунтуючись на результатах попереднього етапу розробник може приступити до розробки інформаційної структури обраного рівня. Для цього на базі результатів статичного моделювання $PCofI\_InputIP_i\_MS''_0$, $PCofI\_OutIP_i\_MS''_0$, $InputCanal\_MS''_0$, $OutCanal\_MS''_0$, проводиться розробка тактико-технічних характеристик фізико-інформаційної моделі даного рівня декомпозиції ($PCofI\_PIM\_MS''_0$). На його базі синтезуються послідовності інформаційних потоків, параметрів і зв'язків. Проводиться імітаційне моделювання під навантаженням для перевірки досягнення головної мети розробки MS'_0 і вимог ТЗ. Дерево рішень декомпозиції на інфологічному етапі представлений на рис. 2.12.

Базуючись на отриманих результатах, розробник може приступити до розробки алгоритмічного етапу даного рівня декомпозиції. Завданням даного

етапу є розробка алгоритму функціонування, на базі якого в подальшому проводитися розробка кібернетичної складової CPPS.

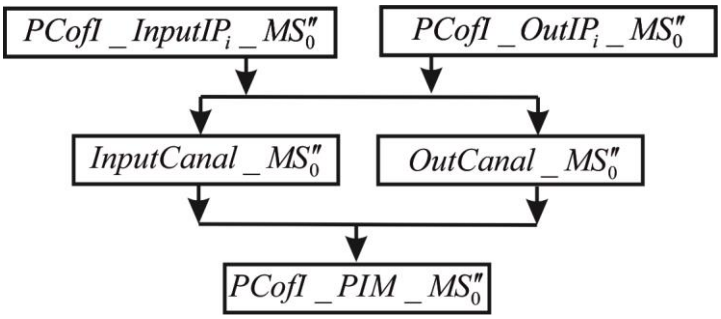


Рисунок 2.12 – Дерево рішень декомпозиції на інформаційному етапі

На базі методу, запропонованого в підрозділі 2.4.3, розробник проводить аналіз $PCofI\_PIM\_MS''_0$ і за (2.4) проводить розробку алгоритму рівня $AF\_MS''_0$. Для цього розробляються оператори Op_h алгоритму функціонування CPPS на даному рівні декомпозиції, проводитися розрахунок їх тактико-технічних і інформаційних характеристик $PCofI\_PIM\_Op_h\_MS''_0$ системного рівня $AF\_MS''_0$ для кожного $Op_h\_MS''_0$.

Використовуючи отримані результати розробник будує алгоритм $AF\_MS''_0$ і проводить спочатку статичне, а потім імітаційне моделювання під навантаженням, для перевірки правильності прийнятих рішень. Дерево рішень декомпозиції на алгоритмічному етапі представлене на рис. 2.13.

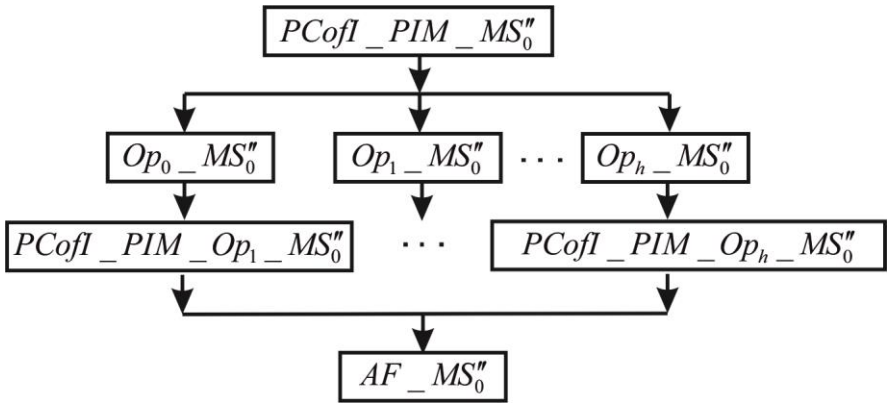


Рисунок 2.13 – Дерево рішень декомпозиції на алгоритмічному етапі

Якщо отриманий результат повністю задовольняє вимогам $Task_j\_MS''_0$ і завданням $Aim_i\_MS''_0$ головної мети CPPS, то можна вважати, що прийняті рішення декомпозиції на інформаційному етапі правильні, отже, всі рішення на попередніх етапах процесу управління розробкою CPPS даного рівня декомпозиції «життєздатні».

Для зручності розуміння всі запропоновані рішення декомпозиції рівня MS''_0 за етапами: цілі, функціональному, організаційно-технічному, інфологічному, інформаційному і алгоритмічному, можна уявити у вигляді узагальненого дерева рішень декомпозиції рівня MS''_0, який представлений на рис. 2.14.

Ґрунтуючись на представленому дереві декомпозиції рівня MS''_0, можна припустити, що використовуючи методи, запропоновані в розділах 2.3-2.4, можна провести декомпозицію всіх рівнів розробки від Sub_S_k до $AEofS_i$ і забезпечити досягнення головної мети розробки CPPS.

Варто зауважити, що запропоновані методи декомпозиції дозволяють розробнику не тільки розробити структуру CPPS, але і забезпечити вибір необхідних атомарних елементів $AEofS_i$ (фізичної складової), на базі яких можливі рішення елементарних завдань і отримання математичного опису кожного завдання, або групи, для розробки їх алгоритму роботи (кібернетичної складової).

Отримані рішення з розробки CPPS дають можливість провести статичне і імітаційне моделювання під навантаженням, а також оцінити тимчасові, якісні та функціональні параметри розроблюваної CPPS.

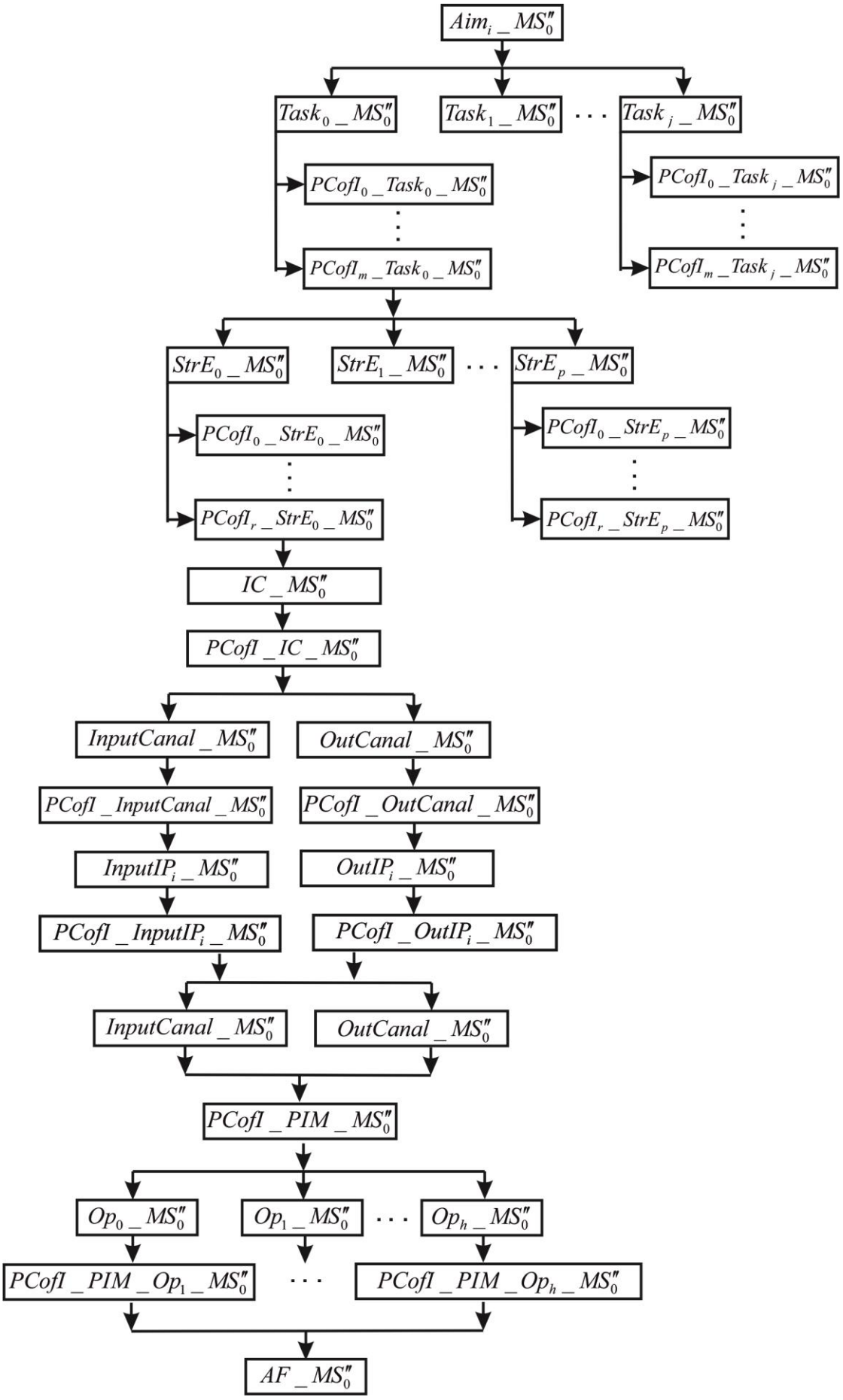


Рисунок 2.14 – Дерево рішень декомпозиції рівня MS''_0

РОЗДІЛ 3
РОЗРОБКА СИСТЕМНИХ МОДЕЛЕЙ КІБЕРФІЗИЧНИХ ВИРОБНИЧИХ СИСТЕМ

3.1 Групування методів розробки CPPS

Ґрунтуючись на твердженнях, що розробка кіберфізичних систем базується на застосуванні кількох приватних системних методів прийняття рішень, етапи яких запропоновано у 2 розділі даної роботи, можна відмітити, що при аналізі цих рішень існує деяка закономірність використання одного методу на різних рівнях розробки CPPS. Для спрощення розробки системних моделей CPPS, пропонується провести таке групування даних методів:

- згрупувати методи декомпозиції Aim_i, $Task_j$ структур, алгоритмів функціонування верхнього рівня на елементи нижнього рівня;

- згрупувати методи розробки Op_h_AF, по $Task_j$ і структурах об'єкта *InputCanal*, *OutCanal* по структурах $AEofS_i$, по $Task_j$, $Task_j$ по Aim_i, застосувавши метод трансформації можна бачити таку закономірність: цілі трансформуються в завдання, завдання в структуру, канали зв'язків, алгоритм функціонування об'єкта;

- згрупувати методи розрахунку *PCofI*, Aim_i, $Task_j$, структур *InputCanal*, *OutCanal*, Op_h;

- згрупувати методи розробки системної, цільової, функціональної, інфологічної, інформаційної та алгоритмічної моделей;

- згрупувати аналітичні методи ухвалення рішень строго за етапами розробки.

Використання даних методів розробником CPPS для різних етапів розробки – обов'язковий, закономірний характер. Але всі запропоновані методи мають відмінності в застосування, у залежності від етапу розробки.

Проведемо аналіз першої групи методів і визначимо головні положення:

- дослідження і аналіз аналогів, прототипів і визначення основних властивостей, критеріїв, ознак – системний аналіз CPPS;

- виявлення складових частин CPPS за певними ознаками (цільовою, функціональною, інфологічною і т.д.);

- визначення зв'язків між складовими частинами CPPS і всередині їх;

- декомпозиція CPPS на складові частини на базі принципу слабких зв'язків між його частинами.

Логічно, що для кожного CPPS і його складових частин зв'язки і властивості між ними різні і будуть верифіковані закономірностями тієї предметної області, до яких відносяться процеси, що відбуваються всередині. Отже, приватні методики визначення складових частин CPPS (вид, зв'язки, властивості, і т.д.) будуть визначені даною предметною областю знань, але при цьому метод декомпозиції для всіх буде один.

Проведемо дослідження групи трансформації, за результатами якого можна бачити можливість ізоморфізму, отже ліквідним буде такий запис:

$$Aim_i \cong Task_j \cong StrE_q \cong IC_v \cong Op_h$$

Це дає можливість виділення елементів, груп, підсистем і систем, віднести їх до об'єктів системного дослідження, що є безпосереднім завданням розробника.

Групування методів розрахунку *PCofI* обумовлено тісним зв'язком з групою трансформації і закономірне за предметною областю знань, внаслідок чого визначають математичні закони, методи і способи розрахунку параметрів *PCofI*. Дана група вивчається фахівцями даної галузі, і на базі їх рішень розробляється методика розрахунків в кожному випадку при розробці CPPS.

Ґрунтуючись на вищевказаних основних вимогах, для успішної розробки CPPS розробник повинен маніпулювати системними моделями які:

- описуються в графічній формі;

- мають можливість алгебраїчних перетворень для зручності їх моделювання.

Отже, проведений аналіз методів формалізації, побудови і перетворення структур і алгоритмів показав, що для опису структури та зв'язків, що підходять до вище перерахованих вимог, можна віднести використання теорії графів, а їх аналіз проводити на базі регулярних схем і мов.

3.2 Метод подання структурних системних моделей CPPS

Визначення 1. Системна модель – це графічне подання, в якому вузли – об'єкти, відповідні етапам і рівням розробки CPPS, а ребра – канали зв'язків між ними. Базуючись на розроблених методах декомпозиції (2.1)–(2.4) і дереві рішень (рис. 2.1, 2.7), в даному дослідженні пропонуються наступні подання системних моделей. Логічно, що для досягнення головної мети розробки CPPS необхідно досягти всі підцілі на всіх рівнях декомпозиції:

$$Aim_i \_ MS_0' = \Omega Aim_i \_ MS_0'' \tag{3.1}$$

де $Aim_i\_MS'_0$ – головна мета розробки CPPS у відповідності з ТЗ;

$Aim_i\_MS''_0$ – цілі на рівні декомпозиції MS''_0.

Ґрунтуючись на (3.1) можна провести декомпозицію мети $Aim_i\_MS''_0$ на підцілі рівня $Sub\_S_k\_MS''_0$ у вигляді (3.2).

$$Aim_i \_ MS_0'' = \Omega Aim_i \_ Sub \_ S_k \_ MS_0'' \tag{3.2}$$

де $Aim_i\_Sub\_S_k\_MS''_0$ – цілі на рівні декомпозиції $Sub\_S_k\_MS''_0$.

Проведемо декомпозиції мети розробки по всьому дереву:

$$Aim_i\_Sub\_S_k\_MS''_0 = \Omega Aim_i\_G\{AEofS\}_j\_MS''_0 \quad (3.3)$$

де $Aim_i\_G\{AEofS\}_j\_MS''_0$- мета для групи атомарних елементів $G\{AEofS\}_j\_MS''_0$.

$$Aim_i\_G\{AEofS\}_j\_MS''_0 = \Omega Aim_i\_AEofS_i\_MS''_0 \quad (3.4)$$

де $Aim_i\_AEofS_i\_MS''_0$ - мета атомарного елемента $AEofS_i$;

Проводячи аналіз (3.1)–(3.4) можна помітити, що головна мета розробки CPPS є наслідком досягнення цілей на кожному рівні декомпозиції на кожному етапі розробки, отже, можна з упевненістю стверджувати існування «спадковості» (→) цілей за принципом декомпозиції «згори-вниз» (3.5).

$$\begin{aligned} &Aim_i\_MS'_0 \rightarrow Aim_i\_MS''_0 \rightarrow Aim_i\_Sub\_S_k\_MS''_0 \rightarrow \\ &\rightarrow Aim_i\_G\{AEofS\}_j\_MS'' \rightarrow Aim_i\_AEofS_i\_MS''_0 \end{aligned} \quad (3.5)$$

Ґрунтуючись на (3.5) і визначенні 1, можна уявити граф досягнення головної мети розробки CPPS, який представлено на рис. 3.1, з урахуванням його багаторівневості, що дозволяє враховувати вплив досягнення цілей одна на одну всередині одного рівня декомпозиції.

Виходячи з рис. 3.1 можна сказати, що розробник CPPS отримає набір системних моделей декомпозиції головної мети $Aim_i\_MS'_0$ на всіх рівнях розробки. Розмістивши кожен вузол Aim_i за відповідними $TandTR\_Aim_i$, розробник має можливість провести статистичне і динамічне моделювання під навантаженням на досягнення головної мети розробки CPPS.

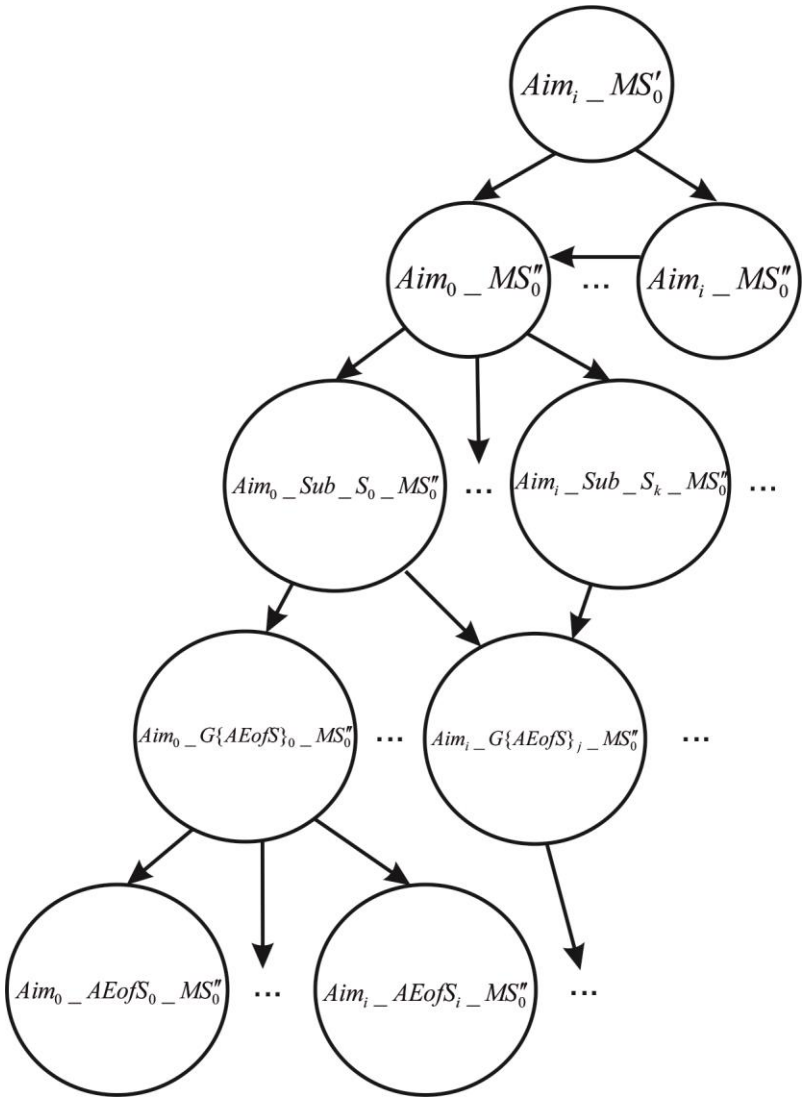


Рисунок 3.1 – Граф досягнення головної мети розробки CPPS

Аналогічно розробник може побудувати граф функціонального рівня. Для цього розробник розробляє за (3.6) комплекс завдань ($Task_j$) для рівня MS_0'.

$$Aim_i\_MS_0' = \Omega Task_j\_MS_0'' \tag{3.6}$$

де $Aim_i\_MS_0'$ – головна мета розробки CPPS;

$Task_j\_MS_0''$ – завдання рівня MS_0'' для досягнення $Aim_i\_MS_0'$.

Визначимо вузлами графа *Task* на кожному рівні декомпозиції, а ребрами графа – зв'язок (→) між завданнями для досягнення головної $Task_j$

на цьому рівні, тоді необхідно провести фіксацію зв'язків між завданнями за допомогою лічильника задач:

$$\begin{aligned} &Task_j \to Task_{j+1}; Task_j \to Task_{j+2}; \\ &Task_{j+1} \to Task_{j+3}; Task_{j+n-1} \to Task_{j+n} \end{aligned} \tag{3.7}$$

Запропонована «спадковість» дозволяє отримати підграфову модель подання $Task_j$, приклад якої наведено на рис. 3.2.

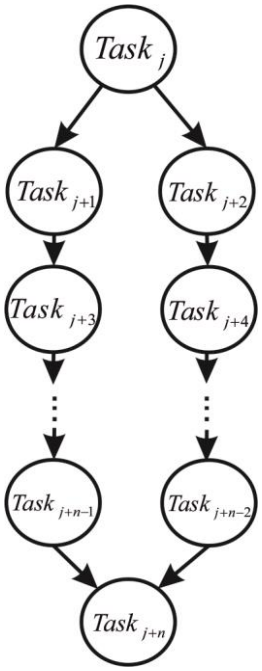


Рисунок 3.2 – Підграфова модель подання $Task_j$

Базуючись на підграфовій моделі представлення $Task_j$, отримуємо можливість провести декомпозицію наступного підрівня $Sub\_S_k\_MS_0''$. Логічно, що для даного підрівня тотожний запис:

$$Task_j\_MS_0'' = \Omega Task_j\_Sub\_S_k\_MS_0'' \tag{3.8}$$

де $Task_j\_Sub\_S_k\_MS_0''$ – завдання на функціональному етапі рівня Sub_S_k.

Підставивши підграф (3.2) в математичне подання (3.8) можна отримати граф декомпозиції (рис. 3.3.) головного завдання рівня MS_0' до рівня Sub_S_k.

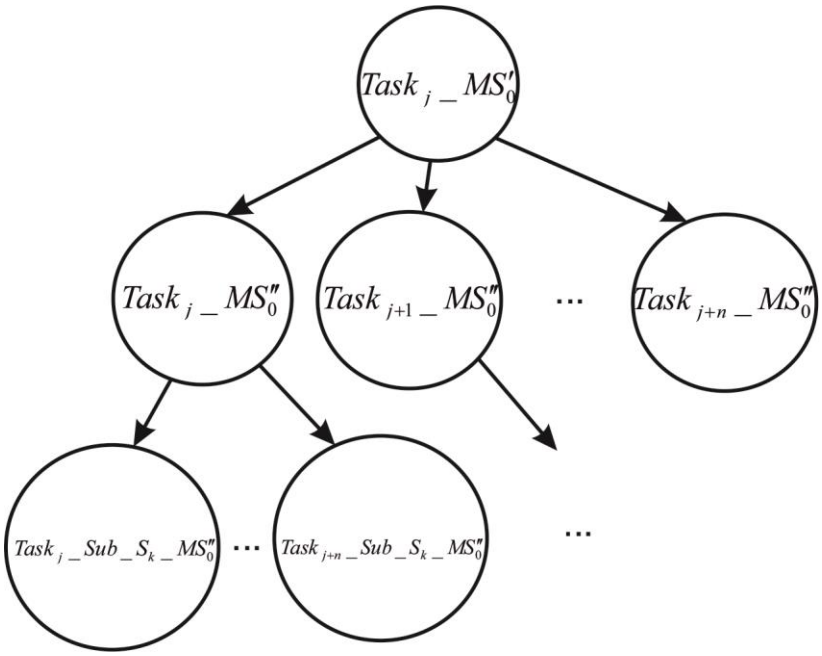


Рисунок 3.3 –Граф декомпозиції функціонального етапу до Sub_S_k рівня

Аналогічно (3.6) та (3.8) можна вивести вирази для всіх рівнів декомпозиції на функціональному етапі:

- для $Task_j\_Sub\_S_k\_MS_0''$

$$Task_j\_Sub\_S_k\_MS_0'' = \Omega Task_j\_G\{AEofS\}_j\_MS_0'' \qquad (3.9)$$

- для $Task_j\_G\{AEofS\}_j\_MS_0''$:

$$Task_j\_G\{AEofS\}_j\_MS_0'' = \Omega Task_j\_AEofS_i\_MS_0'' \qquad (3.10)$$

Використовуючи даний підхід отримуємо набір системних функціональних моделей на всіх рівнях декомпозиції, а ребрами будуть зв'язки між ними.

Метод побудови системної організаційно-технічної моделі буде аналогічним функціональній моделі і будуватися у вигляді графа, в якому вузлом виступатиме $StrE_q$, а ребрами – «спадковість» зв'язків у вигляді:

$$\begin{aligned} &StrE_q \to StrE_{q+1}; StrE_q \to StrE_{q+2}; \\ &StrE_{q+1} \to StrE_{q+3}; StrE_{q+n-1} \to SrtE_{q+n} \end{aligned} \tag{3.11}$$

Ґрунтуючись на цьому можна представити системну модель організаційно-технічного етапу у вигляді графа представленого на рис. 3.4.

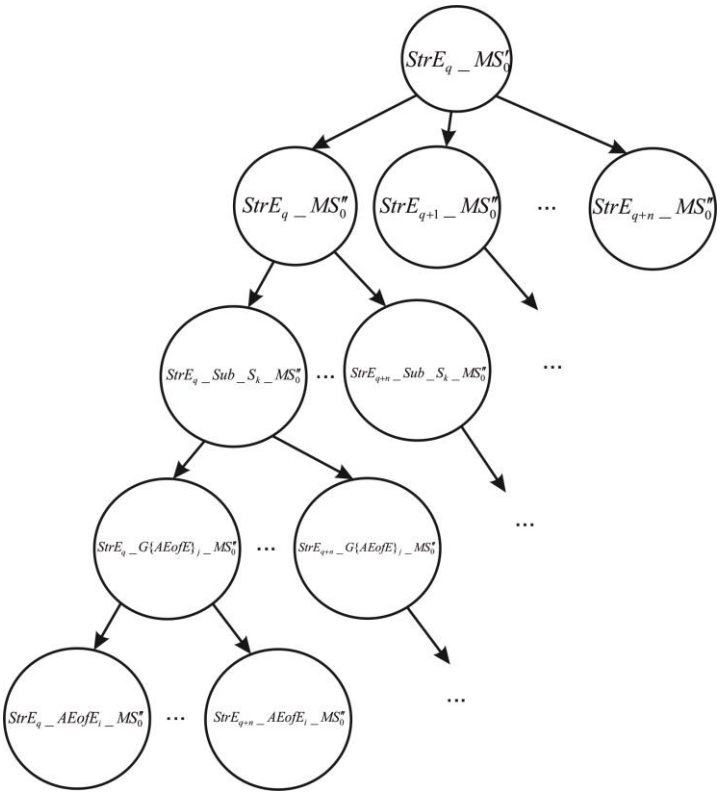


Рисунок 3.4 – Граф системної моделі організаційно-технічного етапу

Отже, на базі запропонованого методу можна побудувати комплекс системних моделей організаційно-технічного етапу для аналізу досліджень прийнятих рішень на будь-якому рівні декомпозиції.

Для опису системної інфологічної моделі приймемо у вигляді вузла графа IC_v (підрозділ 2.5), а в якості ребер графа виступатимуть *InputCanal* і

OutCanal зв'язків. Ґрунтуючись на розробленому методі побудови системної моделі *Aim*$_i$, опишемо системну інфологічну модель рівня MS_0'':

$$\begin{aligned} &I^{MS}\_MS_0' = \Omega IC_v\_MS_0'', \\ &InputCanal\_IC_v\_MS_0'', OutCanal\_IC_v\_MS_0'' \end{aligned} \tag{3.12}$$

де $I^{MS}\_MS_0'$ – інфологічна модель системи рівня MS_0'';

$IC_v\_MS_0''$ – інформаційний перетворювач на рівні MS_0'';

$InputCanal\_IC_v\_MS_0''$ – вхідні канали на рівні MS_0'';

$OutCanal\_IC_v\_MS_0''$ – вихідні канали на рівні MS_0''.

Графова системна модель інфологічного етапу на рівні декомпозиції MS_0' представлена на рисунку 3.5.

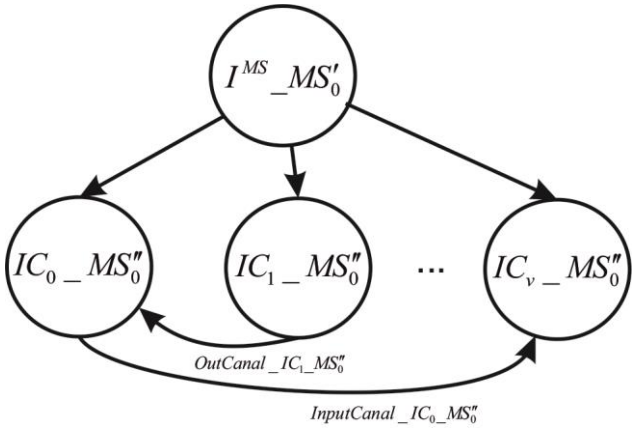


Рисунок 3.5 – Графова системна модель інфологічного етапу на рівні декомпозиції MS_0'

Аналогічно (3.12) можна подати системну модель інфологічного етапу за наступними підсистемними рівнями:

- системна модель на рівні $I^{MS}\_MS_0''$:

$$\begin{aligned} &I^{MS}\_MS_0'' = \Omega IC_v\_Sub\_S_k, \\ &InputCanal\_IC_v\_Sub\_S_k, OutCanal\_IC_v\_Sub\_S_k \end{aligned} \tag{3.13}$$

- системна модель на рівні $I^{MS}\_Sub\_S_k$:

$$I^{MS}\_Sub\_S_k = \Omega IC_v\_G\{AEofS\}_j, \\ InputCanal\_IC_v\_G\{AEofS\}_j, OutCanal\_IC_v\_G\{AEofS\}_j \quad (3.14)$$

- системна модель на рівні $I^{MS}\_G\{AEofS\}_j$:

$$I^{MS}\_G\{AEofS\}_j = \Omega IC_v\_AEofS_i, \\ InputCanal\_IC_v\_AEofS_i, OutCanal\_IC_v\_AEofS_i \quad (3.15)$$

- системна модель на рівні $I^{MS}\_AEofS_i$

$$I^{MS}\_AEofS_i = \\ = \Omega AEofS_i(InputIP_p\_AEofS_i, OutIP_j\_AEofS_i), \quad (3.16)$$

Таким чином, на базі розроблених системних моделей (3.12)–(3.16) можна отримати набір інфологічних системних моделей на будь-якому рівні представлення «верхнього» рівня через «нижній» для аналізу правильності вибору інформаційного перетворювача в залежності від каналів зв'язків.

За результатами, отриманими на інфологічному етапі, можна приступити до реалізації системних моделей інформаційного етапу розробки CPPS. На даному етапі основним завданням є розробка системних тактико-технічних характеристик фізико-інформаційної моделі ($PCofI\_PIM$) на всіх рівнях декомпозиції CPPS. Базуючись на запропонованих методах в підрозділі 2.5 можна представити системну модель інформаційного етапу за наступними підсистемними рівнями:

- системна модель рівня $PCofI\_PIM\_MS_0''$:

$$PCofI\_PIM\_MS_0'' = \\ = \Omega I^{MS}\_Sub\_S_k\_MS_0'', PCofI\_PIM\_Sub\_S_k \quad (3.17)$$

- системна модель рівня $PCofI\_PIM\_Sub\_S_k$:

$$PCofI\_PIM\_Sub\_S_k = \Omega I^{MS}\_G\{AEofE\}_j\_MS_0'', \\ PCofI\_PIM\_G\{AEofE\}_j \quad (3.18)$$

- системна модель рівня $PCofI\_PIM\_G\{AEofE\}_j$:

$$PCofI\_PIM\_G\{AEofE\}_j = \\ = \Omega I^{MS}\_AEofS_i\_MS_0'', PCofI\_PIM\_AEofS_i \quad (3.19)$$

- системна модель рівня $PCofI\_PIM\_AEofS_i$:

$$PCofI\_PIM\_AEofS_i = \\ = \Omega PCofI\_InputIP_p\_AEofS_i, PCofI\_OutIP_p\_AEofS_i \quad (3.20)$$

Для опису системної інформаційної моделі приймемо $PCofI\_PIM$ у вигляді вузла даного рівня, а ребрами виступатимуть тактико-технічні характеристики інформаційних зв'язків ($PCofI\_InputIP_p$, $PCofI\_OutIP_p$). Приклад графової інформаційної системної моделі для (3.19) представлений на рис. 3.6. При цьому потрібно дотримуватися умови існування бінарних співвідношень $PCofI\_OutIP_p\_AEofS_o$ і $PCofI\_InputIP_p\_AEofS_i$:

$$PCofI\_PIM\_AEofS_0 \cong PCofI\_PIM\_AEofS_i \quad (3.21)$$

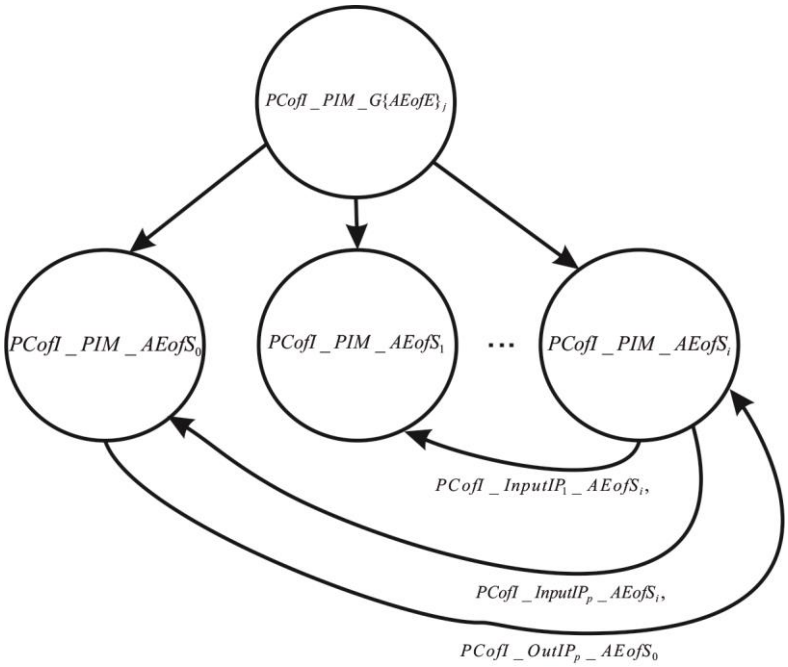


Рисунок 3.6 – Графова системна модель інформаційно етапу на рівні декомпозиції $PCofI\_PIM\_G\{AEofE\}_j$

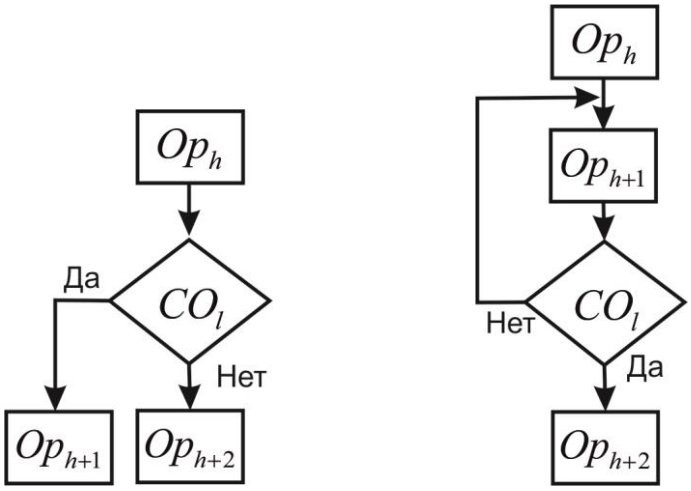
Останнім етапом розробки CPPS, у відповідності до запропонованої архітектурно-логічної моделі автоматизації процесу управління розробкою складних CPPS (рис 2.1) є розробка алгоритму функціонування (AF) на кожному рівні декомпозиції CPPS. Для його побудови пропонується використовувати принцип граф-схем через зручність їх подання. Отже, пропонується наступний метод їх побудови:

- проводиться аналіз Aim_i, $Task_j$, $InputCanal$, $OutCanal$ системного рівня розробки;

- у відповідності кожній $Task_j$ обираємо Op_h граф-схему алгоритму, на базі запропонованого методу (підрозділ 2.5). Позначимо, що першим завданням рівня $Task_0\_MS_0'$, буде Op_0, отже, для наступного рівня декомпозиції $Task_0\_MS_0'$ є $Task_1\_MS_0''$, а йому буде відповідати оператор Op_1 і за аналогією до $Task_j\_MS_o''$ оператором буде Op_h;

- проводиться аналіз послідовності виконання $Task_j\_MS_o''$ для даного рівня, де досягається $Aim_i\_MS_0''$, або є необхідним для досягнення головної мети розробки CPPS;

- проводиться розстановка Op_h за логічною послідовністю для виконання $Task_j\_MS_o''$. Вводяться поняття умовного і безумовного переходу від $Task_j\_MS_o''$ до $Task_{j+1}\_MS_o''$, якщо перехід без умови то Op_h з'єднується $\rightarrow$, якщо Op_h має зв'язок з іншими операторами Op_{h+1}, то необхідно ввести поняття умовного оператора (CO_l), який працює за аналогією з блоком «умови» з базової теорії побудови алгоритму. На базі даного припущення AF може враховувати не тільки «лінійний вид», а й реалізувати «диз'юнктивний перехід», «цикли» і «умови переходу» для досягнення $Task_j$. Приклад варіантів побудови з CO_l представлений на рис. 3.7.



а) б)

а) представлення «диз'юнктивного переходу»;

б) представлення «циклу»

Рисунок 3.7 – Приклад варіантів побудови з CO_l

- розробити на AF кожному рівні системного подання і об'єднати їх в AF_MS_0', який досягає головну мету розробки CPPS. На базі розробленого AF_MS_0' проводитися перевірка за допомогою імітаційного моделювання

під навантаженням, що покаже правильність прийнятих рішень як по всіх етапах, так і за рівнями процесів розробки CPPS.

3.3. Формалізація системних моделей

Для формалізації системних моделей, розроблених в підрозділі 3.2 даного дослідження, було запропоновано використовувати математичний апарат регулярних схем алгоритму і алгоритмічних алгебр. Обґрунтуванням цього вибору є забезпечення доступності формалізації подання системних моделей, що дає можливість реалізації результатів формалізації за допомогою мов високого рівня програмування для розробки автоматизованих систем управління процесами розробки CPPS. Ґрунтуючись на теорії апарату регулярних схем і алгоритмічних алгебр, введемо такі позначення:

OA_q – алгебра операторів, елементами алгебри операторів є Op_h, а також для зручності подання ведемо додаткові операції, що не є операторами перетворення інформації: H – тотожний оператор, $\varnothing$ – порожній оператор;

AP_r – алгебра умов включає в себе всі логічні умови CO_l, які у відповідності до підрозділу 3.2, можуть набувати наступних значень $true, false$ або $CO_l^x \quad x = [true, false, b, c, y, \ldots, r]$. Отже, можна уявити OA_q у вигляді набору алгебр:

$$OA_q = (Op_h, CO_l, H, \varnothing, true, false, x) \tag{3.22}$$

Аналізуючи Op_h можна помітити, що реалізація кібер складової CPPS заснована на візуальних подієвих моделях, побудованих на візуальних об'єктах інтерфейсів користувача (розділ 4), які об'єднують в собі цілі або різні частини алгоритмів та мають в собі розгалуження, цикли, лінійні ділянки, які доцільно замінити функціоналами (представити алгоритм більш

укрупненої форми), а окремі частини алгоритму уявити у вигляді підалгоритму.

Для зручності маніпулювання OA_q (в рамках алгоритмічних алгебр) необхідно визначити основні типи операцій:

Визначення 1. Множення операторів – строго послідовне виконання операторів в порядку їх черги.

$$OA = OA_i \cdot \underset{CO_l}{(} OA_k \vee OA_n) \cdot OA_m \tag{3.23}$$

де $OA_i = Op_i, \ldots, Op_j$;

$OA_k = Op_k \underset{CO_l}{(} Op_l \vee Op_m)$, де CO_l – логічні умови;

$OA_n = Op_n \{Op_p\} Op_t$;

$OA_m = Op_m, \ldots, Op_s$.

Визначення 2. Додавання операторів – це умовне розгалуження простих, або вкладених одна в одну операцій ($\tilde{O}p_h$).

$$OA_q = \tilde{O}p_1 \cdot \tilde{O}p_2 \cdot, \ldots, \cdot \tilde{O}p_h \tag{3.34}$$

$$OA_{q-1} = \underset{CO_1}{(} \tilde{O}p_1 \vee \underset{CO_2}{(} \tilde{O}p_2 \vee \underset{CO_3}{(} \tilde{O}p_3 \vee \ldots \vee \underset{CO_l}{(} \tilde{O}p_h \vee e))) \tag{3.35}$$

На прикладі (3.35) пропонуються наступні розуміння: $\underset{CO_l}{(} \tilde{O}p_h \vee e)$ результати складання простих операцій повинні задовольняти умови CO_l, після цього виконується складання до умови CO_3, і т.д. Але варто врахувати, що Op_h може бути складеним компонентом алгоритму, тому пропонується вести поняття «процес складання».

Визначення 3. Процес складання в системній моделі – це дотримання умовного розгалуження і з'єднання шляхів алгоритму, в залежності від CO_l. Синтаксис запису представлений у вигляді (3.36):

$$OA_q = (\underset{CO_l}{} Op_{h-1} \vee Op_h)^{CO_l} \tag{3.36}$$

Приклад представлення процесу складання в системні моделі можна представити таким виразом:

$$OA_q = (\underset{CO_1}{} Op_1 \vee Op_2 \cdot (\underset{CO_3}{} Op_3 \vee Op_4)^{CO_1} \vee Op_5)^{CO_3} \tag{3.37}$$

За допомогою (3.37) в системних моделях можна описати ітерацію «вперед» після перевірки умов CO_l, при цьому необхідно характеризувати поширену ітерацію «назад», в якій повернення здійснюється до Op_h, а не до умови CO_l (3.40), що вимагає ввести нове визначення.

Визначення 4. Ітераційний процес складання – це правила запису і читання послідовних і концентрично вкладених циклів. Ґрунтуючись на даному визначенні представимо існуючі типи циклів:

- проста послідовність циклів:

$$CY = CY_1 \cdot CY_2 \cdot \ldots CY_n \tag{3.38}$$

де $\mathrm{CY_n} = \{Op_h\}_{CO_l}$;

- концентричне вкладення циклів:

$$CY_n = \{\{\{\ldots\{Op_1\}_{CO_1} Op_2\}_{CO_2} Op_3\}_{CO_3} \ldots Op_h\}_{CO_l} \tag{3.39}$$

- окремий випадок циклу CY_n з поверненням до оператора Op_h при виконанні умови CO_l:

$$CY_n = \{Op_{h-2} \ldots Op_{h-1} \}_{CO_l} Op_h \tag{3.40}$$

Ґрунтуючись на визначеннях 1–4 можна подати такі описи ітеративних шляхів проєктованого алгоритму:

- виконання інтерактивного шляху: за умовою $CO_{l-1} = false$, до закриття фігурної дужки і повторення, потім повернення до перевірки $CO_l = true$, тоді дія алгоритму переносяться за дужку, що закривається до наступного за ним оператору.

$$CY_{n-3} = \{_{CO_{l-1}} Op_1 \ldots Op_{h-1} \}^{CO_l} Op_h \tag{3.41}$$

- виконання інтерактивного шляху за умовою $CO_{l-1} = false$, повернення до дужки, що відкривається, і повторення. За умовою $CO_l = true$ дія переноситься за дужку, що закривається.

$$CY_{n-m} = {}^{CO_{l-1}}\{ Op_1 \ldots Op_{h-1} \}_{CO_l} Op_h \tag{3.42}$$

Умова CO_l, представлена знизу дужок, визначає перевірку її в залежності від його параметра ($false, true$) і в залежності від заданого параметра відбувається умовний перехід. Умова CO_{l-1}, представлена зверху дужок, показує те місце, куди треба повернутися при параметрі $false$ і продовжити дію алгоритму. Приклад даного виду опису алгоритму представлений в (3.43).

$$OA_q = \overset{CO_{l-1}}{\{} Op_{h-5} \underset{CO_{l-3}}{(} Op_{h-4} \vee Op_{h-3} \overset{CO_{l-3}\,CO_l}{)} \{ \underset{CO_{l-2}}{(} Op_{h-2} \underset{CO_{l-1}}{\}} \vee Op_{h-1} \overset{CO_{l-2}}{)} \underset{CO_l}{\}} Op_h \quad (3.43)$$

Запропонований опис алгоритмів дозволяє уявити клас послідовних алгоритмів будь-якої складності перетинання циклів. Але дослідження сучасних алгоритмів CPPS показує, що необхідно розробити математичну формалізацію паралельних шляхів його виконання.

Визначення 5: Кон'юнкція алгоритму – це безумовне розгалуження з виконанням декількох паралельних шляхів алгоритму.

$$OA_q = [Op_{h-2} \wedge Op_{H-1} \wedge Op_h] \quad (3.44)$$

Введемо припущення: якщо дії операторів алгоритму, які укладені в квадратні дужки, починаються паралельно, то їх можна виносити за дужки. Після виконання алгоритму за допомогою одного з шляхів здійснюється перехід до виконання оператора, що стоїть за дужками. Для коректності цього запису необхідно дотримати рівність всіх шляхів (3.45), тому визначимо через P – довжину шляху (кількість операторів) на даній гілці алгоритму.

$$POp_{h-2} = POp_{H-1} = POp_h \quad (3.45)$$

Для виконання умови (3.45) при виникненні $POp_{h-2} \neq POp_{H-1}$ необхідно до меншої довжині алгоритму додати число n операторів.

Для визначення розгалуження алгоритму пропонується наступний метод запису: CO_l^x вказується внизу відкритої квадратної дужки, що означає перевірку умови і початок паралельної роботи алгоритму при x, а зверху над дужкою, що закривається, вказується умови виходу CO_{l-1}^x з алгоритму.

Варто зауважити, що при виконанні паралельних алгоритмів необхідно врахувати параметр x при CO_l, який повинен визначатися розмірністю умов і завжди повинен бути визначений для кожної конкретної умови $x=[true,\ false, b, c, y, \ldots, r]$, виконання умов b, c r, які задовольняють вимогам умов виходу CO_{l-1}, тоді можна вважати що паралельний алгоритм виконав свою функцію. Приклад запису представлений в (3.46).

$$OA_q = [\underset{CO_l^x}{} Op_{h-4} \overset{b}{\wedge} \ldots \overset{c}{\wedge} Op_{H-1} \overset{r}{\wedge} Op_h \overset{CO_{l-1}^x}{}] \tag{3.46}$$

Аналогічно можливо існування CO_l^x при $x=[true,\ false, b, c, y, \ldots, r]$ для операції ітераційного процесу складання алгоритмів. Для зручності запису системної моделі пропонується наступні правила синтаксису:

- умова перевірки CO_l записується знизу закритої фігурної дужки;

- визначимо наступний запис у вигляді визначення верхніх індексів для фігурних дужок $CO_l^{true,\ false, b, c, y, ..., r}$ і розставимо в тих місцях, куди повертається дія алгоритму, в залежності від умови значень $x=[true,\ false, b, c, y, \ldots, r]$

$$OA_q = \overset{CO_{l-1}^b}{\{} Op_{h-4} \overset{CO_{l-2}^{true}}{\{} Op_{h-3} \{Op_{h-2} \overset{CO_{l-3}^y}{\cdot} Op_{h-1}\}_{CO_i} \tag{3.47}$$

На базі розробленої мови формалізації системних моделей, проводиться алгебраїчний опис всіх операторів і умов їх взаємодії у вигляді алгоритму функціонування (AF) на всіх етапах і рівнях декомпозиції CPPS.

3.4 Метод синтезу алгоритмів функціонування CPPS

Формалізація системних моделей дає можливість розробити алгоритм функціонування певного етапу на обраному рівні розробки CPPS, але при цьому не вирішує загальної задачі взаємодії їх як загального цілого для вирішення $Aim_i \_ MS_0'$. Отже, алгоритми функціонування необхідно порівняти, комплексувати з часткових рішень в загальний, декомпозувати із загального в часткові, провести мінімізацію і здійснити алгебраїчні дії. Ідеальним рішенням є абстрактний синтез алгоритмів функціонування для отримання ефекту «одного пристрою або рішення». Тому виникає задача синтезу приватних алгоритмів функціонування в загальний AF.

Ґрунтуючись на вищесказаному пропонуються наступні методи синтезу:

Визначення 6. Об'єднання алгоритмів функціонування – якщо набір Op_h для виконання часткових алгоритмів є загальними і CO_l теж загальний, отже, їх можна об'єднати за CO_l.

$$AF = \underset{OC_l}{(} AF_r \vee AF_m) \tag{3.48}$$

за умови, що $CO_l = (true, false)$

$$AF_r + AF_m = \underset{OC_l}{(} AF_r \vee AF_m) = \begin{cases} AF_r \text{ при } CO_l = true \\ AF_m \text{ при } CO_l = false \end{cases} \tag{3.49}$$

Ґрунтуючись на (3.48)–(3.49) можна застосувати систему аксіом тотожних перетворень алгоритмів, що дасть можливість спростити логічну структуру реалізованого алгоритму.

Визначення 7. Декомпозиція алгоритмів функціонування – це розчленування алгоритму на прості алгоритми без втрати тотожності умов

його роботи. Приклад декомпозиції алгоритму функціонування (3.50), представлений в (3.51)–(3.52).

$$AF_i = \underset{OC_{l-2}}{(}\ AF_j \vee \underset{OC_{l-1}}{(}\ AF_m \vee AF_n \overset{OC_{l-1}\ OC_{l-2}}{)}\ \overset{)}{} \cdot \underset{OC_l}{(}\ AF_p \vee AF_g \overset{OC_l}{)} \qquad (3.50)$$

- укрупнений приклад декомпозиції:

$$AF_i = \begin{cases} AF_1 = AF_j \underset{OC_l}{(}\ AF_p \vee AF_g\ ) \text{ при } OC_{l\text{-}2} = true \\ AF_2 = \underset{OC_{l-1}}{(}\ AF_m \vee AF_n\ ) \underset{OC_l}{(}\ AF_p \vee AF_g\ ) \text{ при } OC_{l\text{-}2} = false \end{cases} \qquad (3.51)$$

- деталізований приклад декомпозиції:

$$AF_i = \begin{cases} AF_1 = \begin{cases} AF_1' = AF_j \cdot AF_p \text{ при } (OC_{l\text{-}1} \wedge OC_l) = true \\ AF_2'' = \underset{OC_{l-1}}{(}\ AF_m \vee AF_n) AF_g \text{ при } (OC_{l\text{-}1} \wedge OC_l) = false \end{cases} \\ AF_2 = \begin{cases} AF_j \underset{OC_l}{(}\ AF_p \vee AF_g\ ) \text{ при } (OC_{l\text{-}2} \wedge OC_{l\text{-}1}) = true \\ AF_n \underset{OC_l}{(}\ AF_p \vee AF_g\ ) \text{ при } (OC_{l\text{-}2} \wedge OC_{l\text{-}1}) = false \end{cases} \end{cases} \qquad (3.52)$$

Приклад декомпозиції (3.52) доводить можливість декомпозиції складного алгоритму на прості, а на базі простих можна проводити перетворення з метою його мінімізації.

В ході аналізу існуючих математичних апаратів і алгебр маніпуляції з алгоритмами, для вирішення питань об'єднання (склеювання) різних алгоритмів, пропонується модифікувати апарат регулярних схем системних моделей. Для цього введемо такі поняття:

- шлях – це фрагмент AF_i, який містить певну послідовність Op_h, шляхом алгоритму буде $Op_{h-1} \to Op_h$. Звідси випливає, що шлях алгоритму є

простим, якщо не має розгалужень і складним, коли містить ітераційний процес (визначення 4) та кон'юнкцію (визначення 5) і т.д.

- довжина шляху – часова характеристика алгоритму, яка служить для доказу тотожності алгоритму при структурній мінімізації за однаковими шляхами. Позначимо $\vec{A}F_i$ частковий алгоритм або шлях алгоритму.

$$\vec{A}F_i = \vec{A}F_1 \cdot \vec{A}F_2 \cdot \vec{A}F_3 \cdot \ldots \cdot \vec{A}F_{i-1} \tag{3.53}$$

Для об'єднання вихідних алгоритмів, необхідно провести його декомпозицію у відповідності до визначення 7 і визначити прості шляхи. Проводиться аналіз Op_h і OC_l по кожному $\rightarrow$ вихідних AF_i, з метою визначення однакових. Виходячи з цього опису приймемо такі припущення щодо еквівалентності алгоритмів:

Аксіома 1. Припустимо, що два будь-яких алгоритми $\vec{A}F_i$ і $\vec{A}F_{i-1}$ будуть тотожно еквівалентні, якщо вони складаються з однакових $\rightarrow$, довжин шляху, Op_h і OC_l в $\rightarrow$ будуть рівнозначні, OC_l – ідентичні. Приклад реалізації аксіоми 1, нехай дано три будь-яких алгоритми (3.54)–(3.56):

$$\vec{A}F_1 = Op_1 \cdot Op_2 \underset{OC_1}{(} Op_3 \vee Op_4 \overset{OC_1}{)} \underset{OC_2}{\{} Op_5 \cdot Op_6 \overset{OC_2}{\}} \vec{A}F_{i-1} \tag{3.54}$$

$$\vec{A}F_2 = \underset{OC_1}{(} Op_3 \vee Op_4 \overset{OC_1}{)} Op_1 \cdot Op_2 \underset{OC_2}{\{} Op_5 \cdot Op_6 \overset{OC_2}{\}} \vec{A}F_{i-1} \tag{3.55}$$

$$\vec{A}F_3 = \vec{A}F_{i-1} \cdot Op_1 \cdot Op_2 \underset{OC_1}{(} Op_3 \vee Op_4 \overset{OC_1}{)} \underset{OC_2}{\{} Op_5 \cdot Op_6 \overset{OC_2}{\}} \tag{3.56}$$

Як можна бачити, алгоритми $\vec{A}F_1, \vec{A}F_2$, і $\vec{A}F_3$ мають однакові оператори $(Op_1, Op_2, Op_3, Op_4, Op_5, Op_6)$ і умови (OC_1, OC_2), отже однакові $\rightarrow$. Для

отримання об'єднаного алгоритму і спрощення маніпуляцій з даними введемо такі скорочення:

$$Op_1 \cdot Op_2 = W \tag{3.57}$$

$$\underset{OC_1}{(} Op_3 \vee Op_4 \overset{OC_1}{)} = M \tag{3.58}$$

$$\overset{OC_2}{\{} Op_5 \cdot Op_6 \underset{OC_2}{\}} = K \tag{3.59}$$

$$\vec{A}F_{i-1} = Z \tag{3.60}$$

Підставимо прийняті скорочення з (3.57)–(3.60) в (3.54)–(3.56), отримаємо наступний вид запису алгоритмів (3.61)–(3.63):

$$\vec{A}F_1 = WMKZ \tag{3.61}$$

$$\vec{A}F_2 = MWKZ \tag{3.62}$$

$$\vec{A}F_3 = ZWMK \tag{3.63}$$

Визначимо додаткові умови існування алгоритмів $\vec{A}F_1, \vec{A}F_2$, і $\vec{A}F_3$ через системи:

$$\hat{S}_1 = \begin{cases} \vec{A}F_1 \text{ при } s_1 = true \\ \vec{A}F_1 \ \vec{A}F_2 \text{ при } s_1 = false \end{cases} \tag{3.64}$$

$$\hat{S}_2 = \begin{cases} \vec{A}F_2 \text{ при } s_2 = true \\ \vec{A}F_3 \text{ при } s_2 = false \end{cases} \tag{3.65}$$

На першому кроці синтезу алгоритмів $\vec{A}F_1, \vec{A}F_2$, і $\vec{A}F_3$ необхідно визначити однакові шляхи за критерієм найменшої повторюваності і зробити запис відповідно до умов існування $\vec{A}F_1, \vec{A}F_2$, і $\vec{A}F_3$. В результаті маніпуляцій можна отримати багато варіантів синтезу алгоритмів в один, і, в залежності від вимог, які необхідно досягти, обирається найбільш оптимальний. Запис (3.66) наводить приклад синтезу алгоритму за критерієм мінімізації кількості операторів, використовуючи наведені скорочення з (3.61)–(3.63) з урахуванням додаткових умов (3.64)–(3.65).

$$\vec{A}F_{1-3} = (\underset{s_2}{} M \vee Z) \cdot W \cdot (\underset{s_1 \vee \bar{s}_2}{} M \vee e) \cdot K \cdot (\underset{s_1 \vee s_2}{} Z \vee e) \tag{3.66}$$

Проведемо підстановку в (3.66) прийняті в (3.57)–(3.60) скорочення, отримаємо синтезований алгоритм $\vec{A}F_{1-3}$, у вигляді набору операторів і умов.

$$\begin{aligned} \vec{A}F_{1-3} = (\underset{s_2}{} (\underset{OC_1}{} Op_3 \vee Op_4) \vee \vec{A}F_{i-1})^{s_2} \cdot Op_1 \cdot Op_2 \, (\underset{s_1 \vee \bar{s}_2}{} (\underset{OC_1}{} Op_3 \vee Op_4) \vee e)^{s_1 \vee \bar{s}_2} \\ \{Op_5 \cdot Op_6\}_{OC_2} \, (\underset{s_1 \vee s_2}{} \vec{A}F_{l-1} \vee e) \end{aligned} \tag{3.67}$$

Аналізуючи результат синтезу об'єднаного алгоритму (3.67) і вихідних алгоритмів (3.54)–(3.56), можна побачити, що в алгоритмах $\vec{A}F_1, \vec{A}F_2$ і $\vec{A}F_3$ є 21 оператор і 6 умов, при заданому критерії мінімізації операторів було отримано 10 операторів при збереженні кількості умов. Тому можна з упевненістю стверджувати, що кількість операторів при цьому синтезі, скоротилася в 2 рази, при цьому алгоритм виконує всі умови, поставлені спочатку. Використовуючи запропоновані в підрозділі 3.4 методи можна маніпулювати з операторами алгоритмів і проводити синтез для мінімізації їх

структури і отримання еквівалентних алгоритмів на всіх етапах і рівнях розробки CPPS, в залежності від вимог ТЗ.

РОЗДІЛ 4
МЕТОДОЛОГІЯ ПРОЦЕСУ РОЗРОБКИ КІБЕРНЕТИЧНОЇ СКЛАДОВОЇ CPPS

4.1 Аналіз вимог, що пред'являються до Smart Factory

Розробка кіберфізичних систем (CPPS), в рамках технології Industry 4.0, є складним завданням синтезу проєктних, фізичних і програмних рішень, які в сукупності повинні забезпечити безперервний процес функціонування, моніторингу та аналізу даних в контексті хмарного і віддаленого виробництва з використанням глобальних інформаційних мереж (iFactory).

Запропоновані в другому розділі методи процесу управління розробкою CPPS, дозволяють уявити CPPS як метасистему, яка декомпозується за етапами і рівнями розробки, що дає можливість прийняти комплексу рішень і реалізувати їх вигляді формалізованого підходу для оцінки правильності прийнятих рішень на кожному етапі і рівні розробки. Розроблені в третьому розділі системні моделі та їх формалізація дозволяють спроєктувати алгоритми функціонування CPPS, як окремо за рівнями і етапам розробки, так і в єдиному алгоритмі функціонування CPPS. Але варто зауважити, що найскладнішою задачею є розробка кібернетичної складової CPPS.

Досліджуючи рішення компаній промислової автоматизації Advantech, можна помітити, що принцип Industry 4.0 – це використання технології M2M (машина до машини), що вимагає постійного забезпечення інформацією про стани системи управління виробничими процесами для аналізу і швидкості прийняття виробничих і управлінських рішень. Таким чином досягається інтеграція в єдину корпоративну мережу iFactory, яка забезпечує синтез функції систем виробничого процесу (c-MES). Основні функції c-MES:

– PM (Process Management) – управління процесами виробництва;

– DPU (Dispatching Production Units) – диспетчеризація виробництва;

– QM (Quality Management) – управління якістю;

– DCA (Data Collection / Acquisition) – збір і зберігання даних;

– PA (Performance Analysis) – аналіз ефективності;

– RAS (Resource Allocation and Status) – контроль стану і розподіл ресурсів;

– LUM (Labor / User Management) – управління людськими ресурсами;

– PTG (Product Tracking & Genealogy) – відстеження і генеалогія продукції.

Як можна бачити з функцій, c-MES – дуже складна колаборація взаємопов'язаних програмних рішень в єдиний цілий інформаційний простір підприємства.

Варто врахувати ще один важливий факт того, що розробка CPPS є одиничним рішенням, яке залежить від спрямованості підприємства та його технологічного забезпечення, що ставить перед розробником складне завдання спроєктувати і розробити єдине інформаційне середовище. Звідси виникає дилема: з одного боку існують методики розробки програмних продуктів та комерційні аналоги для вирішення деяких функцій c-MES, з іншого – розробка інформаційного простору в розроблюваній CPPS повинна охоплювати і об'єднувати всі етапи виробництва в єдине ціле, в рамках специфіки (обладнання, ПЛК, датчики, і т.д.) і завдань (типи і види технологічних процесів, види виробництв, методи аналізу та управління, і т.д.) роботи iFactory. Виходячи з вищевказаного можна сказати, що не існує певних рекомендацій і методологій розробки кібернетичної складової CPPS виробничого призначення в рамках технології Industry 4.0.

Проаналізовано такі міжнародні стандарти: ISO/IEC 90003:2004, ISO 9001:2000, ISO/IEC 25062, ISO/IEC 25030:2007, ISO/IEC 25000:2005, ISO/IEC 24774:2007, ISO/IEC TR 20000:2005, ISO/TR 18529:2000, ISO PAS 18152:2003, ISO/IEC 18019:2004, SO/IEC 16085:2006, ISO/IEC 15939:2007, ISO/IEC 15504, ISO/IEC 15289:2006, ISO/IEC 15288:2007, ISO/IEC

15288:2002, ISO/IEC TR 15271:1998, ISO/IEC 12207, ISO/IEC 14764:2006, ISO 13407:1999, ISO/IEC TR 9294:2005, ISO 9241-11:1998, ISO/IEC TR 9126-4:2004, ISO/IEC TR 9126-3:2003, ISO/IEC TR 9126-2:2003, ISO/IEC 9126-1:2001, ISO 13407:1999, ISO 10007:2003, ISO 9004:2000, ISO 9001:2000, ISO 9000:2005, IEEE/EIA 12207.0-1996, ISO/IEC 12207:1995, IEEE Std 1517-1999, IEEE P90003 – 2007, ISO 9001:2000.

А також міжнародні стандарти процесу менеджменту моделей життєвого циклу: IEEE 1074 – стандарт описує підхід до визначення процесів життєвого циклу програмних засобів; IEEE Std 1175 – описує інтеграцію CASE-інструментарію в продуктивне середовище програмної інженерії; IEEE Std 1462 – керівні вказівки з оцінки та вибору CASE-інструментарію; IEEE Std 16085 – викладає процес менеджменту ризиків програмних засобів; IEEE Std 1061 – описує методологію, що охоплює життєвий цикл для встановлення вимог до якості та ідентифікації, реалізації та валідації відповідних показників; IEEE Std +1362 – представляє собою керівництво за форматом і змістом концепції операційного документа, описуючи характеристики запропонованої системи з точки зору користувача.

Процес аналізу системних вимог IEEE Std 1233 – викладає керівництво з розробки специфікації системних вимог, характеристик і якості вимог.

Процес проєктування архітектури системи IEEE Std 1471 – рекомендує концептуальну структуру і зміст для опису архітектури систем, що інтенсивно використовують програмні засоби.

Процес аналізу вимог до програмних засобів IEEE Std 830 – стандарт рекомендує зміст і характеристики специфікацій вимог до програмних засобів.

Процес проєктування архітектури програмних засобів IEEE Std 1471 – рекомендує концептуальну структуру і зміст для опису архітектури систем, що інтенсивно використовують програмні засоби.

Процес детального проєктування програмних засобів IEEE Std 1016 – стандарт рекомендує зміст і організацію детального проєктування програмних засобів.

Процес менеджменту документації IEEE Std 1063 – містить вимоги до структури, змісту та формату користувальницької документації, IEEE Std 12207.1 – містить вказівки з реєстрації даних в результаті виконання процесів життєвого циклу.

Процес менеджменту конфігурації програмних засобів IEEE Std 828 – стандарт конкретизує зміст плану маркетингу конфігурації програмних засобів разом з вимогами до специфічної діяльності з планування.

Ґрунтуючись на проведеному аналізі можна зробити висновок, що існуючі аспекти та рекомендації, представлені в сучасних міжнародних стандартах, не дозволяють використовувати їх в рамках комплексного підходу для вирішення завдання розробки кібернетичної складової CPPS і в основному мають часткові пропозиції і рекомендації щодо систематизації розробки прикладних програмних продуктів загального призначення. Виходячи з цього, для реалізації єдиного підходу до розробки CPPS, в даному дослідженні, пропонується нова модель життєвого циклу (ЖЦ) процесу управління розробкою CPPS «Jump», яка представлена на рис. 4.1.

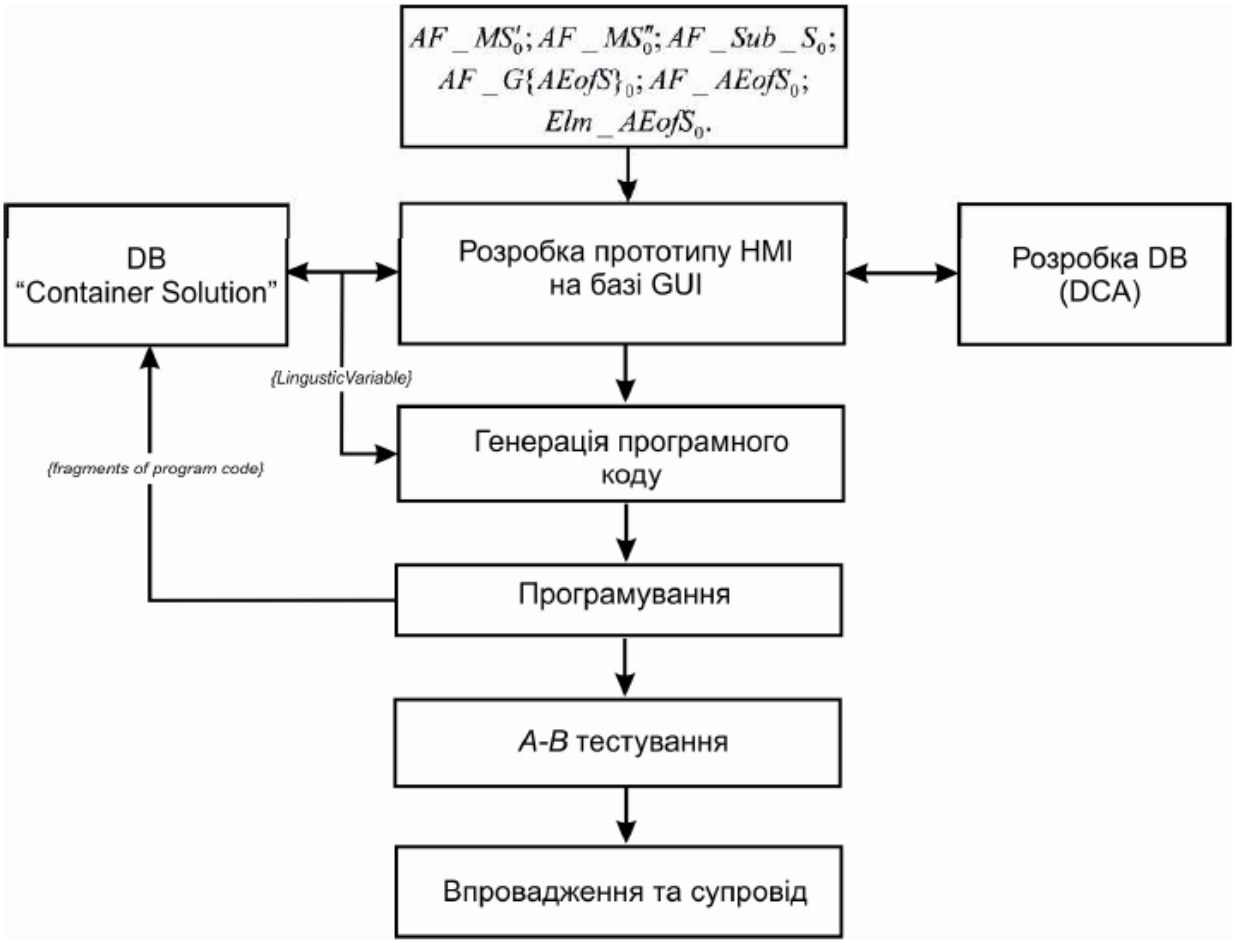


Рисунок 4.1 –Модель ЖЦ автоматизації процесу управління розробкою CPPS «Jump»

Переваги даної моделі ЖЦ полягають в тому, що вона складається з трьох основних етапів:

– аналізу алгоритму функціонування CPPS (загальний алгоритм функціонування CPPS, за етапами і рівнями декомпозиції);

– розробка прототипу інтерфейсу користувача для візуалізації функції DPU, PM, QM і т.д. в залежності від вимог ТЗ на CPPS;

– генерація програмного коду прототипу інтерфейсу, який враховує оператори і події при виконанні заданих умов, визначених алгоритмом функціонування CPPS на кожному рівні і етапі декомпозиції;

– тестування і впровадження в єдину корпоративну мережу iFactory.

Запропонована модель ЖЦ розробки CPPS включає в себе синтез існуючих концепцій і підходів до розробки прикладних програмних

продуктів, модифікований під запропоновані моделі і методи процесу управління розробкою CPPS, які представлені в другому і третьому розділі.

Розглянемо детально кожен етап ЖЦ розробки CPPS «Jump», як сукупність підсистем, що знаходяться у відносинах і зв'язках між собою і утворюють цілісну послідовність для вирішення завдання автоматизації процесу управління розробки кібернетичної складової CPPS.

Запропонована модель ЖЦ заснована на моделях і методах розробки CPPS і виконаються наступні принципи: єдності та ієрархії, відкритий ступінь взаємодії з зовнішнім середовищем, властивості самоорганізації при зміні специфіки завдань і вимог до мов високого рівня програмування і середовища розробки в залежності від вимог, що пред'являється до неї.

Для опису всіх зв'язків в підсистемах і всередині системи запропоновано використовувати методи структуризації на базі теоретико-множинних представлень і інформаційного підходу формалізації процесів, зв'язків, що протікають всередині моделі і кожного етапу. Даний підхід дозволить вирішити такі важливі завдання:

- забезпечить представлення моделі ЖЦ процесу управління розробкою CPPS «Jump» на такому рівні абстракції, на якому вона стає виразною і інтуїтивно зрозумілою аналітикам, розробникам CPPS, Software Product Manager, програмістам і кінцевим користувачам;

- забезпечити специфікацію CPPS, її структуру і поведінкову організацію, в залежності від заданого в ТЗ середовища розробки та специфіки виробництва.

Результати аналізу і вимог, що пред'являються ТЗ і алгоритмом функціонування CPPS на всіх етапах і рівнях декомпозиції для досягнення головної мети розробки CPPS, обумовлюють необхідність розробки нових або вдосконалення існуючих моделей і методів. Для досягнення мети досліджень в даній роботі необхідно досягти максимально можливу автоматизацію процесу управління розробкою кібернетичної складової CPPS, за рахунок наступних вимог, що висуваються до моделі ЖЦ:

- модель ЖЦ процесу управління розробкою CPPS «Jump» повинна володіти необхідною і достатньою спільністю, щоб її засоби надавали можливість забезпечити прозоре, адекватне, комплексне подання даних і зв'язків для всіх учасників проєкту з виконанням умов єдиної структури і цілісності;

- розбіжність між запропонованою моделлю процесу управління розробкою CPPS «Jump» і об'єктно-орієнтованими підходами, на основі яких реалізуються необхідні функції c-MES, ERP в єдиному інформаційному середовищі Smart Factory, повинні бути адаптивні до динамічних змін предметної області CPPS в залежності від ТЗ, а алгоритм функціонування має бути мінімальним, але достатнім.

Отже, склад і структура набору формальних об'єктів, створюваних в моделі ЖЦ розробки CPPS «Jump» на базі принципу об'єктно-орієнтованої побудови, повинні бути максимально близькими, щоб при відображенні концептуальної моделі CPPS можна було скористатися відомими правилами перетворення структур, побудови, ієрархії без втрат цілісності ;

- можливість комплексного використання розроблених алгоритмів функціонування на кожному етапу та рівні декомпозиції як загального функціонального алгоритму розробки CPPS;

- реалізація «часткової автоматизації» за рахунок генерації програмного коду події, інтерфейсу користувача на базі функціональних алгоритмів CPPS і головної мети розробки у відповідності до ТЗ;

- можливість використання природних мов опису подій і об'єктів, заданих в функціональному графі CPPS, для розробки інтерфейсу користувача (HMI) на базі графічних елементів (GUI) об'єктно-орієнтованого підходу до програмування.

Для забезпечення сформульованих вище вимог виділимо чотири основні етапи, відповідно до загального підходу, викладеного в К. Дейтом:

- виявлення деякої множини латентно-семантичних понять (концепцій) (LSA), які будуть використовуватися для опису моделі;

- визначення набору формальних об'єктів, які можуть використовуватися для подання обраних понять;

- визначення формальних правил підтримки цілісності даних в моделі ЖЦ розробки CPPS «Jump»;

- визначення формальних операторів взаємодії моделі ЖЦ розробки CPPS «Jump» з «реальним світом» між етапами як підсистемами даної моделі, а також взаємодії всередині кожної підсистеми до атомарного рівня декомпозиції графу функціонування CPPS.

Інформація про розглянуту предметну область («реальний світ») представляється через сприйняття розробника або аналітика, і складається з множини взаємозалежних і взаємопов'язаних факторів, що орієнтує на розгляд моделі ЖЦ розробки CPPS «Jump» з позиції закономірності системного цілого і взаємодії його складових сутностей (етапів).

Ґрунтуючись на концепції багаторівневої ієрархії суті моделі ЖЦ розробки CPPS «Jump», процеси і явища доречно розглянути як множину дрібніших ознак (підмножин) опису. При цьому етапи логічно розглядати як елементи більш високих класів узагальнень.

Виходячи з запропонованої моделі, ЖЦ розробки CPPS «Jump» повинний забезпечувати адекватне правильне сприйняття. Отже, доцільно використовувати принцип, за яким таке уявлення спочатку вибудовує для себе аналітик або розробник природною мовою.

Ґрунтуючись на визначенні, що всі процеси «реального часу» протікають в просторі і в часі, то модель ЖЦ розробки CPPS «Jump» являє собою динамічну систему, яка складається з певних послідовностей станів, де в кожен момент часу визначається її множина кінцевих станів.

Кожна множина кінцевих станів описується елементарними факторами природною мовою в термінах об'єкта, зв'язків і приналежності об'єкту, значеннях параметрів і його властивостей під час виконання тієї чи іншої події, що задаються за допомогою розроблених моделей і методів представлення алгоритму функціонування, які описані в 2-3 розділах.

Опис моделі ЖЦ розробки CPPS «Jump» можна уявити з точки зору доцільності у термінах предметної області, розширивши і синтезувавши їх набір. В контексті розробленої моделі ЖЦ розробки CPPS «Jump» запропоновані наступні поняття:

- об'єкт, що пов'язується з різними частинами модельованого програмного продукту або модулями для CPPS;

- властивість (опис характеристик) об'єкта, який дозволяє співвіднести множину об'єктів (підкласів) і їх властивостей в один загальний клас;

- подія – реакція, яка відбувається на об'єкті (певного підкласу), кожна подія описується його властивостями і визначається моментом часу його настання або виклику;

- значення характеристик об'єкта і події – необхідні ознаки (параметри, що відносяться до певних класів і їх підкласів) подання об'єкта (опис його візуалізації), або реакції на реалізацію події, у вигляді певних необхідних дій, яке представлену як «контейнер рішень»;

- елемент – частина об'єкта (підклас), який формується як залежна частина об'єкта і визначає необхідність логічної взаємодії всередині об'єкта або групи об'єктів в контексті певного і необхідного рішення.

Ці поняття є неформальними концепціями (метапоняттями) моделі як невід'ємного елементу згідно підходу К.Дейта. При цьому варто зауважити, що основним в розробленій моделі ЖЦ розробки CPPS «Jump», є обов'язкова умова існування «об'єкта» (форми, GUI елемента) і події, які можуть належати як об'єкту або класу, так і підкласам які входять в них.

Виникнення об'єкта обумовлено інтересом суб'єкта і зникає, коли він втрачається. У монографії Д. Цікрітзіса визначена єдина властивість існування об'єкта – час його виникнення, зникнення і зміни, пов'язані з подіями, що відбуваються з ним.

У запропонованій моделі ЖЦ розробки CPPS «Jump» знайшли застосування певні підходи, рішення і методології, які вже апробовані: методологія RUP (Rational Unified Process) – розроблена компанією Rational

Software; методологія Microsoft Solution Framework – запропонована компанією Microsoft, заснована на практичному досвіді; методологія Scrum – управління проєктами, яка заснована на принципах Time-менеджменту; гнучка методологія Agile-розробки (швидке проєктування і розробка ПП і ПМ, де головне завдання – працюючий продукт, а не його документація; методологія візуального об'єктно-орієнтованого програмування (абстрагування, поліморфізм, інкапсуляція), семантичної, мережевої, онтологічної, інфологічної логіки предикатів з розширеною підтримкою в часі.

Відштовхуючись від того, що парадигма методології візуального об'єктно-орієнтованого програмування, концептуальної та логічної моделі близькі, можна спростити перехід від концептуальної моделі до логічної. Також на реалізацію моделі ЖЦ розробки CPPS вплинули: семантичні моделі, що дають можливість опису і представлення складних ситуацій; концепція ієрархії типів, що дозволяє обґрунтувати і систематизувати подання класів, метакласів; концепція ролі понять і часу, що дозволяє реалізувати висунуті вище вимоги, які пред'являються до моделі ЖЦ процесу управління розробки CPPS «Jump».

4.2 Формалізація моделі ЖЦ процесу управління розробкою CPPS «Jump»

У відповідно до певних вимог і понять, модель ЖЦ процесу управління розробки CPPS «Jump» ($\aleph$) можна представити і формалізувати як наступний кортеж на базі четвертого виду представлення системи і на основі класичної теорії системного аналізу:

$$\aleph = \langle P(\Im, \Re, H), K, L \rangle, \tag{4.1}$$

де $P(\Im, \Re, H)$ – сукупність правил і структурування даних про об'єкт

моделювання, яка визначається AF_MS';

P – об'єкт моделювання CPPS або окремих функції c-MES;

$\Im$ – множина базових понять (концепції подання) об'єкта моделювання;

$\Re$ – множина відносин між базовими поняттями моделі;

H – множина функцій описів і трактування базових понять і відносин;

K – множина вимог обмеження цілісності;

L – мова подання моделювання даних.

Для подальшого опису сукупності правил і структурування даних P необхідно розробити словник опису і трактування базових понять H, який складений для множини базових понять $\Im$. Уявімо множину $\Im$ як набір базових понять в такому вигляді:

$$\begin{aligned}\Im = \{&Form, ParameterForm, ValueForm, EventForm,\\ &LinguisticVariable, ElementForm, ParameterElement,\\ &ValueElement, EventElement, ContainerSolutions\}\end{aligned} \quad (4.2)$$

Ці поняття є основними елементами подання деякого формального об'єкта моделювання P. При розробці опису розроблюваного CPPS кожне поняття з (4.2) буде містити конкретні імена (назви), які будуть запозичуватись зі словників і визначень того чи іншого середовища розробки, або неформально, шляхом узгодження базових понять між розробником і користувачем автоматизованої системи процесів управління розробкою CPPS.

Звідки $\Re$ – безпосередня множина відносин між однойменними множинами, співвідносними з даними поняттями і їх елементами як формальними об'єктами, що дає можливість представляти властивість модельованих елементів CPPS, їх взаємодію і зв'язки, за умови, що модель має кінцеву множину таких зв'язків і взаємодій та визначаються алгоритмом функціонування і цілям (Aim_i) для кожного етапу так і в цілому для CPPS.

Наведемо опис основних базових понять з (4.2), зумовивши неформальне визначення та призначення, в рамках формалізації об'єкта моделювання, на базі запропонованої моделі ЖЦ розробки CPPS «Jump»:

Form (Windows Form) – деяка виділена і унікально ідентифікована частина предметної області. Її призначення – опис і подання візуальної структури CPPS у вигляді основних блоків;

ParameterForm – сукупність типів і способів опису властивостей предметної області, виділених і згрупованих за деякими ознаками, а також ідентифікованих за ім'ям. Призначення – опис параметрів, необхідних і достатніх для відображення і моделювання візуального представлення *Form*;

ValueForm – значення, яке присвоюються типу і способу опису властивостей предметної області. Призначення – привласнення конкретного значення (цілочисельного, лінгвістичного, булевого) типу або способу опису параметрів в залежності від функціональних ознак і допустимих рішень.

EventForm – подія або група подій (дія), які можуть відбуватися (вже відбулися або відбудуться) з предметною областю в певний момент або інтервал часу. Ідентифікується часом (необхідністю) і об'єктом, до якого належить подія. З одним об'єктом в один момент часу може відбуватися тільки одна подія, яка попередньо ініціалізується користувачем. Необхідні для опису призначення допустимого набору подій (умов CO_l) (підрозділ 2.3), які задаються в алгоритмі функціонування для кожного етапу і рівня в залежності від вимог до CPPS.

LingusticVariable («Лінгвістична змінна») – іменований (природною мовою системи) логічний опис дій при виникненні подій. Такі описи можуть бути згруповані за рядом ознак. Призначення – привласнення класу події або одиничній події лінгвістичної інтуїтивно-зрозумілої користувачеві моделі змінної для опису реакцій при виникненні тієї чи іншої події.

ElementForm – елемент або група елементів GUI (підклас об'єкта) для візуального представлення взаємодії користувача і модельованого функціоналу CPPS. Містять необхідні ознаки для реалізації інтерфейсу

користувача або управління і взаємодії з інформаційними потоками і даними, в залежності від вимог до атомарних елементів ($AEofS_i$) і алгоритмів їх функціонування при рішенні елементарних завдань до рівня MS'_0. Призначення – графічне відображення елементів або групи елементів, які можуть мати деревоподібну структуру візуального представлення і використовуватися як для реалізації інтуїтивно зрозумілого інтерфейсу, так і безпосередньо для взаємодії з користувачем CPPS.

ParameterElement – типи і способи опису властивості елементів, поодинокі або згруповані за деякими ознаками і ідентифіковані ім'ям. Призначення – опис параметрів, необхідних і достатніх для подання та моделювання візуального представлення елемента в рамках єдиного інформаційного об'єкта.

ValueElement – значення, що привласнюється типу і способу опису властивостей GUI елемента. Призначення – привласнення конкретного значення (цілочисельного, лінгвістичного, булевого) типу або способу опису параметрів в залежності від функціональних ознак і реалізації візуального інтуїтивно-зрозумілого інтерфейсу кожної частини предметної області у відповідності з алгоритмом функціонування. Для деяких *ParameterForm* і *ParameterElement* при їх функціональному призначенні і поіменній ідентифікації значення можуть бути однаковими в залежності від вимог CPPS.

EventElement – подія, група подій або умова (CO_l), які можуть відбуватися (вже відбулися або відбудуться) з елементом GUI, який виконує певну функцію в певний момент або інтервал часу. Ідентифікується часом (необхідністю) та елементом до якого належить подія. З одним об'єктом в один момент часу може відбуватися тільки одна подія, що ініціалізується користувачем. Призначення – одна з основних властивостей елемента, що обмежується: функціональними можливостями цього GUI елемента, областю застосування (з точки зору необхідності використання в даній моделі CPPS), необхідністю і роллю в загальній концепції застосування в інтерфейсі

користувача для відпрацювання інформаційних потоків.

ContainerSolutions («Контейнер рішень») – іменований опис реакцій при виникненні події або групи подій в певний момент часу на елемент (групу елементів) або предметну область. Є жорстко структурованим, в залежності від мови високого рівня програмування і середовища розробки, який необхідний для досягнення мети розробки. Застосування – часткове або повне рішення виконання необхідних дій з даними (інформаційними потоками), що необхідні для досягнення мети розробки за умови досягнення головної мети розробки CPPS або на певних рівнях декомпозиції.

$\Re$ – множина відносин між базовими поняттями моделі, що дозволяють формально представляти класифікацію відносин, які визначають тип взаємодії між об'єктами предметної області та їх елементами. Визначимо в даній системі такі види відносин: «Об'єкт–об'єкт», «елемент–об'єкт», «елемент–елемент», «об'єкт–елемент», за допомогою яких можна зробити класифікацію елементів предметної області. При цьому утворюються класи подій, встановлюються правила відносин між усіма учасниками процесу розробки CPPS з використанням *LingusticVariable* і *ContainerSolutions* як невід'ємної частини множини $\Re$ і існуючих в будь-якому вигляді відносин визначених вище.

Виходячи з методів візуального об'єктно-орієнтованого програмування, для розроблюваної методології необхідно розглянути поняття багаторівневої абстракції існування двох видів підмножин на базі ЖЦ процесу управління розробкою CPPS «Jump». Припустимо як аксіому, що будь-який P володіє одним (обов'язковим) або набором $Form$. Для зручності побудови моделі розділимо цю множину на дві підмножини: «*master*» і «*slave*». «*Master*» будемо називати $Form_n^{master}$, який в розробленому нижче методі нумерується через $n=1$ та є головною в цій ієрархії, а «підлеглий» $Form_n^{slave}$ (відповідно $Form_{n+1}^{slave}$, де $n=\overline{1,i}$). Виходячи з цього, за умови, що $n \neq 0$ можна представити у вигляді такої записи:

$$Form = Form_1^{master} \cup Form_n^{slave} \quad (4.3)$$

при $n = 0$ запис (4.3) прийматиме такий вигляд:

$$Form = Form_1^{master} \quad (4.4)$$

Тобто, кібернетична складова CPPS, що розроблюється (моделюється), буде містити єдиний об'єкт взаємодії з користувачем, і в даному випадку це буде $Form_1^{master}$.

Виходячи з (4.3) представимо CPPS як множину об'єктів візуальних $Form_n^{slave}$ різного рівня ієрархії, що підпорядковуються $Form_1^{master}$. На базі цього тотожним є такий запис:

$$Form_1^{master} = \bigcup_{n=2} Form_n^{slavr} \quad (4.5)$$

Відповідно запису (4.5) для $Form_1^{master}$ існує множина $Form_n^{slave}$ n-го рівня ієрархії, що мають унікальні імена:

$$Form_1^{master} = \{Form_1^{slave}, \ldots, Form_{n-1}^{slave}, Form_n^{slave}\} \quad (4.6)$$

Вони взаємодіють між собою і з $Form_1^{master}$ через події, що належать множинам *EventForm* або *EventElement* з використанням *ContainerSolutions*.

Введемо наступні ознаки відносини типу взаємодії між базовими поняттями: «включає», «є елемент», «є параметр», «є подія», «є значення», «є ім'я», «містить рішення». Нижче наведені приклади формалізації математичного опису відносин для кожної ознаки:

- «включає»

$$[(Form_1^{slave}, Form_2^{salve}, \ldots, Form_n^{slave}) \in Form_1^{master}] \in P \quad (4.7)$$

або за умови (4.4) запис матиме вигляд:

$$(Form_1^{master}) \in P \tag{4.8}$$

де P – візуальна модель, яку змодельовано в ЖЦ розробки CPPS "Jump". Розуміється, що змодельований CPPS «включає» в себе множину $Form$, які мають ієрархію взаємозв'язків відповідно до (4.7)

- «є елемент»

$$\exists(ElementForm_1^1, ElementForm_2^1, \ldots, ElementForm_t^1) \in Form_1^{master} \tag{4.9}$$

можливий наступний запис, коли $(ElementForm_t^1) \in Form_1^{master}$ при $t = \overline{1,i}$, містить у собі ще й елементи графічного інтерфейсу (GUI):

$$\begin{aligned}&[(ElementForm_1'^{1(t)}, ElementForm_2'^{1(t)}, \ldots, ElementForm_q'^{1(t)}) \in \\ &\in ElementForm_t^1] \in Form_1^{master}\end{aligned} \tag{4.10}$$

Слід розуміти як $ElementForm_1'^{1(t)}$, $ElementForm_2'^{1(t)}, \ldots, ElementForm_q'^{1(t)}$ при $(t) = \overline{1,i}$ і $q = \overline{1,i}$, «є елементи» підкласу, класу $ElementForm_t^1$, що входять в нього (для зручності реалізації групування за деякою умовою: логічною, інформаційною, змістовною).

- «є параметр» для $Form_1^{master}$ і $Form_n^{slave}$:

$$\begin{aligned}&((parameter_1, \ldots, parametr_p) \in ParameterForm_z^1) \in Form_1^{master} \equiv \\ &\equiv ((parameter_1, \ldots, parameter_p) \in ParameterForm_z^1) \in Form_2^{slave}\end{aligned} \tag{4.11}$$

За умови, що використовується одне середовище розробки візуалізації кібернетичної складової CPPS:

$$((parameter_1,\ldots, parametr_p) \in ParameterElement_h^1) \in ElementForm_t^1 \quad (4.12)$$

Множина *ParameterForm* і *ParameterElement* «є параметр» ($parameter_p$ при $p=\overline{1,i}$), який описує його візуальні характеристики (розміри, відображення, колір) в залежності від його призначення. Для *Form* можна уявити у вигляді запису (4.11), для опису $ElementForm_t^1$ вираз (4.12).

- «є подія» для $Form_n^{master}$ і $Form_n^{slave}$:

$$\begin{aligned}&((event_1,\ldots event_e) \in EventForm_c^1) \in Form_1^{master} \equiv \\ &\equiv ((event_1,\ldots event_e) \in EventForm_c^2) \in Form_2^{slave}\end{aligned} \quad (4.13)$$

З (4.13) видно, що $event_e \in EventForm_c^1 \equiv event_e \in EventForm_c^2$ при $e=\overline{1,i}$ і $c=\overline{1,i}$, мають однаковий набір певних подій, який притаманний усім *Form* в рамках одного CPPS.

$$((event_1,\ldots event_e) \in EventForm_c^1) \in Form_1^{master} \quad (4.14)$$

Визначає належність того чи іншого $event_e$ як невід'ємної частини множини $EventForm_c^1$ і $ElementForm_t$ як «є подія».

- "є значення"

$$\exists(ValueForm_1, ValueForm_2,\ldots ValueForm_u) \in parameter_p,\ u=\overline{1,i} \quad (4.15)$$

За умови, що $parameter_p \in ParameterForm_z$ з (4.11).

$$\exists(ValueElements_1, ValueElements_2,\ldots ValueElements_y) \in parameter_p \quad (4.16)$$

За умови, що $parameter_p \in ParameterElement$ і $y=\overline{1,i}$ з (4.12).

- «є значення» (цілочисельне, лінгвістичне, булеве), яке задається обмеженням $parameter_p$ в залежності від середовища розробки і призначення того чи іншого параметра та входить до множини $ParameterForm_z$ і $ParameterElement_h$.

- «є ім'я»

$$\exists! LingusticVariable_w \in LingusticVariable \forall event_e \in EventForm_c \quad (4.17)$$

$$\exists! LingusticVariable_w \in LingusticVariable \forall event_e \in EventElement_r \quad (4.18)$$

при $w = \overline{1,i}$ і $r = \overline{1,i}$

За умови, що хоч одна «Логічна змінна», яка описує вимоги, «Має ім'я» необхідне для виконання події, яке належить множині $EventForm_c$ або $EventElement_r$.

- «є рішення»

$$\exists! ContainerSolutions_d \in ContainerSloution = LingusticVariable_w \in LingusticVariable;\ d = \overline{1,i} \quad (4.19)$$

Розуміється так: існує хоч один «Контейнер рішень», який «є рішення» (процедуру, операцію з програмним кодом) для даної «Логічної змінної». Ґрунтуючись на даному припущенні, представимо для наочності множину існуючих в моделі ЖЦ розробки CPPS «Jump» основних базових понять і їх взаємодію у вигляді орієнтованого графа $L=(D,O)$. Вершину графа D варто розуміти як умовне позначення базових понять, а O – тип відносин (взаємодії) між ними. Подання основних базових понять для моделі ЖЦ розробки CPPS «Jump» виходячи з (4.2) представлено на рисунку 4.2.

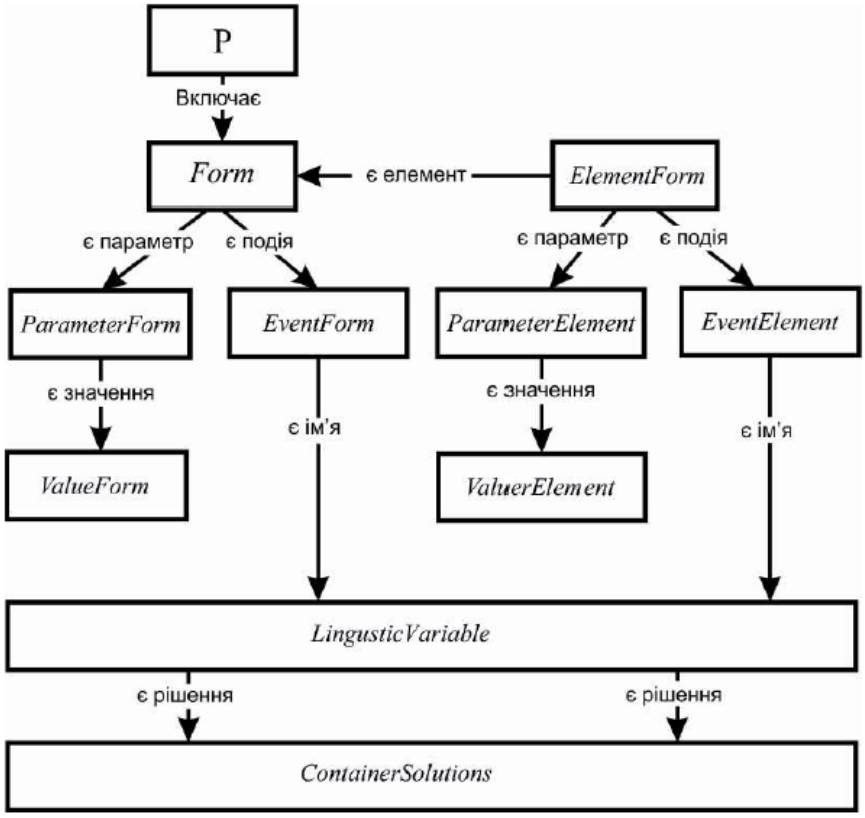


Рисунок 4.2 – Подання основних базових понять і відносин між ними

H – множина функцій описів і трактування базових понять і відносин $\Re$, що визначає правила подання і опису структури даних предметної області моделювання. Внаслідок цього, набір формальних об'єктів (математичну структуру представлення об'єкту і опис його властивостей і подій) можна представити у вигляді математичної структури, яка дозволяє реалізувати в рамках об'єктно-орієнтованої моделі, достатньої для моделювання даних в рамках певної предметної області. Розглянемо функції інтерпретації, як вид відносин між множиною понять. У запропонованій моделі ЖЦ розробки CPPS «Jump» визначимо такі поняття:

- «є значення»:

$$f : Form_n^{master} \times ParameterForm_z \rightarrow ValueForm_u \Leftrightarrow$$
$$\Leftrightarrow Form_n^{master} \times ParameterForm_z \xrightarrow{f} ValueForm_u \qquad (4.20)$$

де для будь-якої впорядкованої пари $(Form_n^{master}, parameter_p)$ із $Form_n^{master} \times ParameterForm_z$ існує не більше одного елемента $value_v$ з $ValueForm_u$, який є $(Form_n^{master}, parameter_p, value_v) \in f$. На базі (4.20) наступний запис можна представити в такому вигляді:

$$value_v = f(Form_n^{master}, parameter_p) \quad (4.21)$$

де $Form_n^{master} \in Form$; $parameter_p \in ParameterForm_z^{master}$; $value_v \in ValueForm_u^{master}$.

Аналогічно можна описати «є значення» для $ElementForm_t^{master}$:

$$value_v = f(ElementForm_t^{master}, parameter_p) \quad (4.22)$$

де $ElementForm_t^{master} \in Form_n^{master}$; $parameter_p \in ParameterElement_h$; $value_v \in ValueElement_y$.

- «є подія» ґрунтується на структурі запису (4.20).

Представимо дане поняття у вигляді (4.23) для $Form_n^{master}$ і (4.24) відповідно для $ElementForm_t^{master}$:

$$name_m = f(Form_n^{master}, event_e) \quad (4.23)$$

де $Form_n^{master} \in Form$; $event_e \in EventForm_t$; $name_m \in ValueElement_y$.

$$name_m = f(ElementForm_t, event_e) \quad (4.24)$$

де $ElementForm_{t\text{-}1} \in ElementForm_t$; $event_e \in EventElement_r$; $name_m \in ValueElement_y$.

- «має рішення» для $ElementForm_t$ виразу (4.25), а для $EventForm_c$

відповідно (4.26).

$$cod_l = f(event_e, name_m) \quad (4.25)$$

де $event_e \in EventElement_r$; $name_m \in LingusticVarable_w$; $cod_l \in ContainerSolutions_d$

$$cod_l = f(event_e, name_m) \quad (4.26)$$

де $event_e \in EventForm_c$; $name_m \in LingusticVarable_w$; $cod_l \in ContainerSolutions_d$

K – множина вимог щодо обмеження цілісності. В даному контексті – це деяка логічна умова, що обмежує семантику обраного об'єкта або елемента для уникнення втрати відносин всередині CPPS.. В рамках даного дослідження виділимо два типи обмежень цілісності: явні (накладаються семантикою) і неявні (накладаються для підтримки і досягнення головної мети моделювання CPPS самою моделлю ЖЦ розробки CPPS «Jump»). Систематизуємо і подаємо опис обмежень за такими типами.

Явні обмеження накладаються щодо списку параметрів $ParameterForm_z$ об'єкта ($Form$), $ParameterElement_h$ – елемента, як підкласу об'єкта ($ElementForm_t$) і подій для $EventForm_c$, $EventelEment_r$ (за визначенням множини припустимих значень для відповідних подань):

$$\begin{aligned}
&ParmeterForm_z^1 = \{parameter_1^z, parameter_2^z, ..., parameter_p^z\};\\
&ParmeterElement_h^1 = \{parameter_1^h, parameter_2^h, ..., parameter_p^h\};\\
&EventForm_c^1 = \{event_1^c, event_2^c, ..., event_e^c\};\\
&EventElement_r^1 = \{event_1^r, even_2^r, ..., event_e^r\};\\
&ParameterForm_z = dom(parameter_{p_спис}^z); \quad (4.27)\\
&ParameterElement_h = dom(parameter_{p_спис}^h);;
\end{aligned}$$

$$EventForm_c^1 = dom(event_{e\_спис}^c);$$

$$EventElement_r^1 = dom(event_{e\_спис}^r);$$

де $dom(parameter_{p\_спис}^z)$, $dom(parameter_{p\_спис}^h)$, $dom(event_{e\_спис}^c)$, $dom(event_{e\_спис}^r)$ – домени відповідних характеристик (значень), що належать до перераховуваних (облікових) типів, які вибираються з заздалегідь сформованого списку.

Прикладом можуть виступати деякі параметри $ParameterForm_z$, наприклад, параметр відображення $Form_n^{master}$, який може приймати значення $true$ або $false$, так і параметр $Align$ і т.д., який притаманний $dom(parameter_{p\_спис}^z)$, $dom(parameter_{p\_спис}^h)$ і може приймати значення фіксовані середовищем розробки ($Top, Down, Center$ і т.д.). Ґрунтуючись на (4.27) конкретні значення зазначених характеристик можна вибрати з представлених для відповідних доменів:

$$\begin{aligned}
&parameter_p^1 \in dom(parameter_{p_спис}^z);\\
&parameter_p^1 \in ParameterForm_z;\\
&ParameterForm_z \in Form_1^{master};\\
&parameter_p^1 \in dom(parameter_{p_спис}^h);\\
&parameter_p^1 \in ParameterElement_h;
\end{aligned} \qquad (4.28)$$

Аналогічно вибору певного параметра опису (4.28) можна описати вибір події з $dom(event_{e\_спис}^c)$ для $Form^{master}$ і $Form^{slaver}$, а також подій візуальних елементів $dom(event_{e\_спис}^r)$:

$$\begin{aligned}
&event_e^1 \in dom(event_{e_спис}^c);\\
&event_e^1 \in EventForm_t;\\
&EventForm_t \in Form_1^{master};\\
&Form_1^{master} \in P;
\end{aligned} \qquad (4.29)$$

$$
\begin{aligned}
&event_e^1 \in dom(event_{e\_спис}^r);\\
&event_e^1 \in EventElement_1^r;\\
&EventElement_1^r \in CD_x^1;\\
&CD_1^x \in CF_f^1;\\
&CF_1^f \in Form_1^{master};\\
&Form_1^{master} \in P;
\end{aligned}
\tag{4.30}
$$

Ґрунтуючись на (4.28)–(4.30) визначимо параметри опису $Form_1^{master}$ і подій, які належать даній формі, а параметри візуальних елементів і їх властивості можна уявити у вигляді dom-списку, який обмежується середовищем розробки і є фіксованим.

Неявне обмеження, що накладаються на модель формалізації опису P в даному дослідженні буде введено на використання значень фізичних величин ($value_v$), які описують параметри $ParameterForm_z$ і $ParameterElement_h$. Модель опису CPPS, що розроблюється, дозволяє представити обмеження за існуванням, яке полягає в тому, що для існування в даній моделі параметра (4.28) необхідно і достатньо, що б він був і мав деяке значення $value_v$.

Аналогічно для (4.29) і (4.30) для існування $event_e^1$ необхідно, щоб вони мали $LibgusticVarable_w$, який посилається на певний і єдиний $ContainerSolution_d$ за (4.25)–(4.26). Наприклад, у запропонованій моделі без конкретного заданого значення ($value_v$) певний параметр $parameter_p^1 \in ParameterForm$ і $parameter_p^1 \in ParameterElement_h$ не буде використовуватися або буде заданим за замовчуванням.

В якості обмежень за припустимими операціями в моделі, що розроблюється, пропонуються:

- операція створення елементів множин, однойменних базових понять моделі $\Im$ за (4.1) і їх зв'язків між собою в відповідностей з $\Re$ і H;

- операція зміни зазначених елементів множин, однойменних базовим

поняттям;

- операція видалення зазначених елементів множини, однойменних базовим поняттям, а отже, залежних від них відповідно до $\Re$ і H елементів інших множин;

- операції вибірки у відповідності із зазначеними умовами існування елементів як частини множини, однойменних базових понять запропонованої моделі.

Обмеження за унікальною ідентифікацією елементів множин, співвіднесених з однойменними базовими поняттями запропонованої моделі, досягається унікальністю іменування:

- для елементів множин *Form*, *ParameterForm*, *EventForm*, *ElementForm*, *ParameterElement*, *EventElement* – унікальністю відповідних в ієрархії імен у рамах даної предметної області;

- для значень параметрів множин *ValueForm*, *ValueElement* – унікальністю імен в рамках розглянутої предметної області;

- для елементів множин *LingusticVariable* – унікальністю імен конкретних рішень, які посилаються на *ContainerSolutions*, що містить однозначне рішення для множин *EventForm* і *EventElement*.

Посилальна цілісність передбачає обов'язкову наявність об'єкта, на який посилається інший об'єкт і забезпечується завдяки запропонованим типам взаємодії між елементами множини.

L – мова подання даних, яка може бути представлена теоретико-множинною моделлю, що не дозволяє непідготовленому фахівцю правильно і повно описати всі необхідні дані для адекватного представлення моделі і вхідної інформації для неї.

Таким чином, була розроблена спеціальна мова, що поєднує в собі простоту синтаксису і опису даних. Вона побудована на базі близької до природної мови, яка дає зрозумілість і інтуїтивність використання CPPS, а також подання даних модельованої предметної області для всіх учасників. Більш детально розроблена мова моделювання і опис її синтаксису

представлені в розділі 5.

На даному етапі позначимо такий умовний поділ метаопису мови моделювання даних в розроблюваній моделі як: дані та метадані. Під метаданими або інтенсіоналом предметної області розуміється і сукупність конкретних множин.

4.3 Метод синтезу візуальних компонентів і елементів структури в моделі ЖЦ процесу управління розробки CPPS «Jump»

Для реалізації автоматизації процесу управління розробкою CPPS необхідно розробити метод синтезу візуальних компонентів на базі алгоритму функціонування, який буде забезпечувати повноту опису вхідних даних, логічний взаємозв'язок між ними і властивостями основних параметрів, притаманних для елементів об'єктно-орієнтованих мов високого рівня програмування і інтерфейсу користувача.

Визначимо P як впорядковану множину призначених для користувача форм ($Form_n$), які виконують умову:

$$\exists Form_n \in P \quad \text{при } n = \overline{1, i} \tag{4.31}$$

де $i \le 40$ – максимальна кількість візуальних призначених для користувача форм інтерфейсу, з якого складається проєктований програмний продукт, який взаємодіє з головною (основною) формою ($Form_1^{master}$).

Ґрунтуючись на аналізі конструкторів інтерфейсу для об'єктно-орієнтованих мов програмування, які реалізовані в середовищах розробки, а також на ISO 9241-12-1988, $Form_n$ можна представити у вигляді підмножин $Form_n\ Propertied\ and\ evens$ і $Components\ on\ Form_n\ structure$, що містять необхідну і достатню кількість даних для даного дослідження.

Виходячи з цього, визначимо об'єкт $Form_n$ де $n = \overline{1, i}$, як множину:

$$Form_n \in \{Form_n Propertird\ and\ evens,\ Components\ on\ Form_n\ structure\} \quad (4.32)$$

за умови що $Form_n Propertird\ and\ evens$ і $Components\ on\ Form_n\ structure \neq \varnothing$.

Теорема 1. Про існування $Form_n$ тільки за умов $Form_n Propertird\ and\ evens$ і $Components\ on\ Form_n\ structure \neq \varnothing$.

Доказ: Припустимо $Components\ on\ Form_n\ structure = \varnothing$, отже,на $Form_n$, як робочому вікні інтерфейсу, не будуть присутні елементи візуалізації роботи з даними (GUI елементи: Button, Grid, і т.д.), отже, інтерфейс розроблюваного програмного продукту не має функціонального призначення, і його розробка не доцільна, і це логічно. Виходячи з цього $Components\ on\ Form_n\ structure \neq \varnothing$. Цей вислів ліквідний так само для $Form_n Propertird\ and\ evens$ і дозволить уникнути парадоксу. Теорема доведена.

Визначимо $Form_n Propertird\ and\ evens$ як множину параметрів $Main\ propertied$ і значень $Properties\ parameters$, притаманних кожній $Form_n$, як невід'ємної частини $Form_n Propertird\ and\ evens$. Уточнимо, що кожному параметру множини $Main\ propertied$ належить хоч одна підмножина значень $Properties\ parameters$, але при цьому деякі значення підмножини можуть бути $Properties\ parameters = 0$. У результаті можна записати:

$$\exists Properties\ parameters_s \in Properties\ parameters: \\ : Properties\ parameters_s \neq 0 \quad (4.33)$$

Наведемо підмножини $Main\ propertied$ і $Properties\ parameters$ як впорядковану жорстко фіксовану послідовність параметрів $mp_x \in Main\ propertied$, де mp_x – параметри, які є невід'ємною частиною

підмножини $Main\ propertied \in Form_n Propertied\ and\ evens$. Відповідно, множина значень $pp_a \in Properties\ parameters_s$, яка, в свою чергу, задовольняє умові $Properties\ parameters_s \in Form_n Propertied\ and\ evens$. Введемо множину Z з елементами $(z_1, z_2, \ldots, z_k)$, як множину можливих варіацій значень (розмір $Form_n$ в піх, розташування «Top», «Left», зарезервовані слова «alNone – немає вирівнювання» і логічних «True» «False» і т.д.), що залежать від синтаксису опису $Form_n$ для певної мови програмування і середовища розробки.

Виходячи з цього, визначимо $\exists z_k \in Z : pp_q \neq 0$. В окремих випадках можливо $z_k = 0$, за умови, що $z_k \in Z \subseteq pp_q \in Properties\ parameters \neq \varnothing$. Також невід'ємною частиною множини є підмножина $Main\ events$, яка представляє набір параметрів $Form_n Propertird\ and\ evens$ таких як: «Create», «Close», «OnClick» і т.д. Для їх опису введемо me_h, $(me_1, me_2, \cdots, me_h) \in Main\ events$, як всілякі допустимі дії над множиною $Form_n$. Множина параметрів обмежується правилами, заданими в тому чи іншому середовищі розробки. Обґрунтуємо існування me_h як елементу множини $Main\ events$, наступним правилом існування:

$$Main\ events = \{me_h \in Main\ events : Form_n Propertied\ and\ event \neq \varnothing\}. \quad (4.34)$$

Кожному елементу me_h властива множина $Events\ on\ action\ whith\ Form_n$, яка складається з нескінченної кількості елементів ea_z – можливий набір імен $LinguisticVariable$, які посилаються на $ContainerSolution$ з певним конкретним «шаблоном» у вигляді програмного коду ($code_l$). Цей код містить в собі всі допустимі процедури/функції, які можуть виконуватися елементом me_h для кожної мови об'єктно-орієнтованого програмування. Наведемо це як:

$$Events\ on\ action\ whith\ Form_n \in \{ea_1, ea_2, \ldots, ea_z\}. \quad (4.35)$$

На базі висунутих припущень представимо множину $Form_n Propertird\ and\ evens$ як:

$$\{(\exists pp_a \cap mp_x) \in Main\ properties: Main\ properties \neq \varnothing) \& \\ \&(\exists ea_z \cap me_h \in Main\ properties \neq \varnothing)\} \subseteq Form_n Propertied\ and\ evens. \quad (4.36)$$

Вираз (4.36) не є достатнім і повним для вирішення завдання, поставленого в даному дослідженні, так як не враховує візуальні компоненти і компоненти роботи з відображення даних. Внаслідок цього, введемо множину $Components\ on\ Form_n\ structure \in Form_n$ за умови:

$$(Components\ on\ Form_n\ structure \backslash Form_n\ Propertied\ and\ evens),$$

як множину елементів опису візуальних компонентів інтерфейсу, необхідних і достатніх для реалізації управління з даними. $Components\ description_f \in Components\ on\ Form_n\ structure$, де $f = \overline{1,i}$ – кількість візуальних компонентів, які присутні на $Form_n$ і виконують певні функції. У свою чергу, для опису візуальних компонентів введемо підмножини $Preporties\ components_f$ і $Component\ events\ on\ action_f$ як невід'ємні частини множини $Components\ description_f$ – кількість різноманітних візуальних форм, за допомогою яких можна працювати з інформацією.

Тоді будь-який візуальний елемент можна описати в наступному вигляді:

$$\{PPreportis\ components_f, Component\ events\ on\ action_f\} \in \\ \in Components\ description_f. \quad (4.37)$$

Для опису властивостей множин $Preporties\ components_f$ і $Component\ events\ on\ action_f$ введемо параметр $pc_m \in Preporties\ components_f$, де $m = \overline{1,i}$ кількість параметрів, які описують властивості об'єкта. Значення pc_m може приймати значення з множини Z, описаної вище. На відміну від $Main\ properties_t$, який єдиний для $Form_n$, кількість $Components\ description_f \in Components\ on\ Form_n\ structure$ повинна бути $f \geq 1$, де $f = \overline{1,i}$. В іншому випадку, при невиконанні цієї умови спрацьовує теорема 1.

Нескінченна різноманітність візуальних компонентів, які можуть використовуватися при формуванні множини $Components\ on\ Form_n\ structure$ вимагає накладення обмежень на $Preporties\ components_f$: за умови, що множина значень параметрів $pc_1 \in Preporties\ components_1 \neq pc_m \in Preporties\ components_f$, при цьому $pc_1 = pc_m$ так само ґрунтується на тому, що множина $Components\ description_f$ може містити однакові назви параметрів pc_m в множині $Preporties\ components_f$, але при цьому різні значення з множини Z або навпаки.

Наведемо множину $Component\ events\ on\ action_f$ як впорядкований набір подій ce_w, які повинні виконувати обов'язкову умову:

$$ce_w \in Componenet\ events\ on\ action_f : Components\ description_f \neq 0. \quad (4.38)$$

Визначимо ce_w як множину значень (набір «контейнерів» з програмним кодом, які можна застосувати для параметру pc_m), тоді логічним буде вираз:

$$\exists ce_w \in \forall pc_m : Components\ description_f \neq 0 \quad (4.39)$$

в іншому випадку це доводить неналежність $Components\ description_i \notin Components\ on\ Form_n\ structure$, тобто воно відсутнє як візуальний елемент в $Form_n$.

Для подальших досліджень і обґрунтувань обраних рішень в даному методі введемо скорочення для зручності математичних записів:

$Form_n Propertid\ and\ evens$ – будемо надалі розуміти як $Form_n PE$;

$Main\ propertied$ – MP_n, де n – номер $Form_n$, котрій він належить, а параметри відповідно mp_x^n, де n – номер форми, якій належить параметр, x – номер параметра в множині MP_n.

$Properties\ parameters$ – PP_n – множина значень, які може приймати mp_x^n у вигляді певних pp_a^n.

Варто врахувати, що кожному $mp_x^n \to pp_a^n$, при чому множина варіацій, які може приймати pp_a^n зберігається в множині Z, як описано вище. Припустимі дії, які може виконувати $Form_n$ у вигляді множини $Main\ events_n$ визначимо як ME_n де n – номер форми. Множина ME_n може володіти унікальним набором параметрів, які позначимо як me_h^n, де n – ідентифікатор приналежності до ME_n, а h – номер параметра. Кожному ME_n відповідає множина $Events\ on\ action\ whith\ Form$ (EA_n), яка в свою чергу містить множину значень ea_z^n, де n – ідентифікатор приналежності до EA_n, а z – номер параметра.

При цьому варто врахувати, що $me_h^n \to ea_z^n$ за умови, що $n = n$, значення ea_z^n для кожного параметра me_n^n можуть обиратися з набору «контейнерів». $Components\ on\ Form_i\ stucture$ позначимо як CF^n, де n – номер $Form_n$, якій належить множина візуальних компонентів, які описані у вигляді $Components\ description_f$ (CD_x^n), де n показує приналежність до тієї чи іншої $Form_n$, а x – номер компонента $Form_n$ в структурі множини CF^n.

Уявімо структуру множини CD_x^n як упорядковану множину параметрів опису візуальних компонентів для роботи з даними $Preporties\ components_f$, яку в подальшому позначимо через PC_y^x, де x показує приналежність до CD_x^n, а n – номер компонента в даній структурі. Уявімо PC_y^x як впорядкований набір параметрів pc_m^y, де $y = y$ номер приналежності до PC_y^x, а m – як порядковий номер параметрів масиву PC_y^x. $Component\ events\ on\ action_i$ позначимо як CE_z^x, де x – номер приналежності до певного CD_x^n, а z – порядковий номер. Наведемо CE_z^x як множину подій, які може обробляти візуальний компонент ce_w^z, де $z = z$ з нумерації CE_z^x.

Множині подій ce_w^z може належати «контейнер рішень». Звернемо увагу, що множина CD_x^n може використовуватися кілька разів, при цьому візуальні форми можуть виконувати різні функції і описуватися різними параметрами, але можуть мати одне і те ж програмне рішення з «контейнера рішень».

Подання кібернетичної складової для CPPS, що розробляється (P), у вигляді математичного опису структур $Form_n$ і зв'язків між ними представлено на рисунку 4.2.

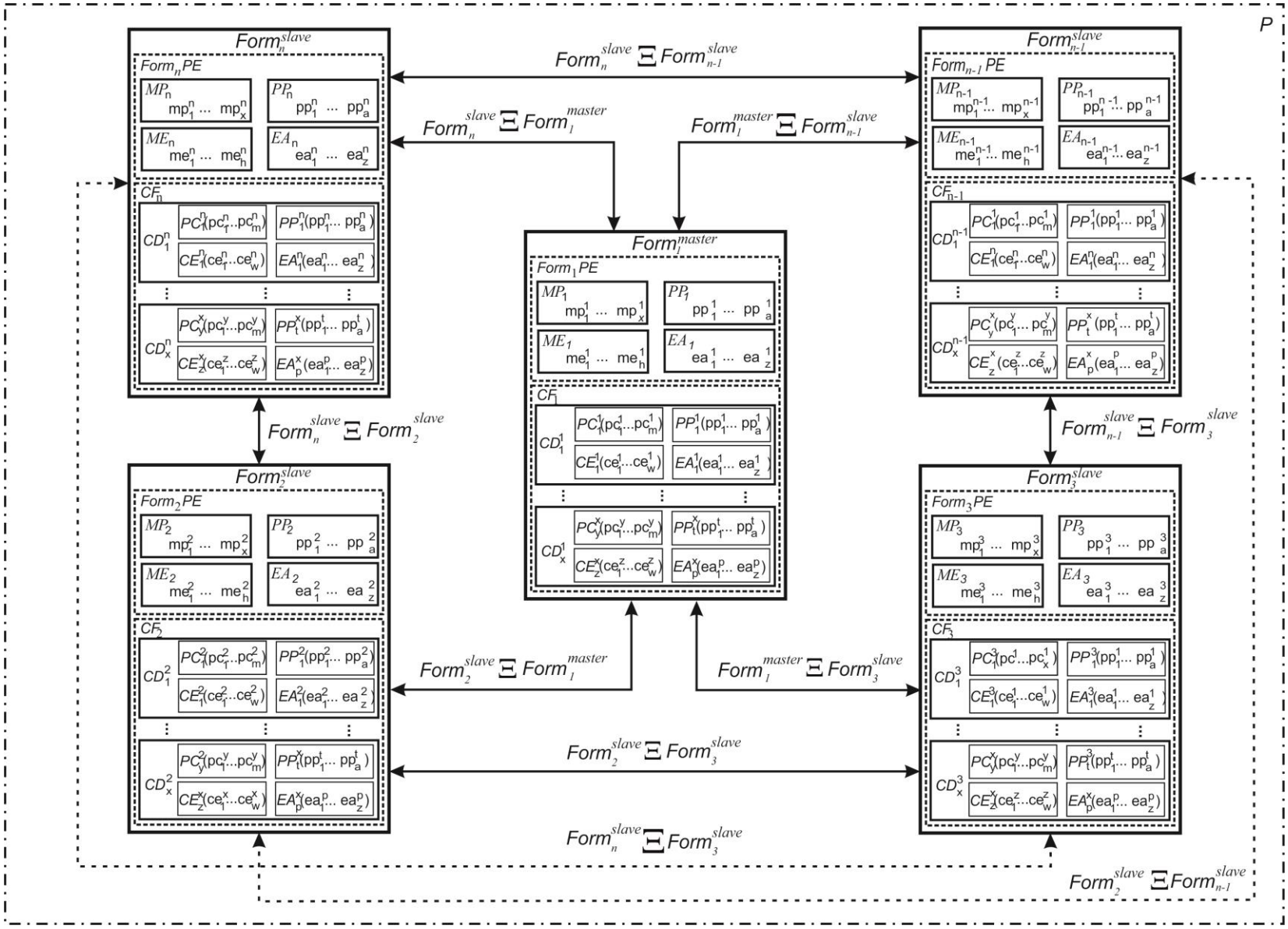


Рисунок 4.2 – Структурне подання $Form_n$ та зв'язків між ними в рамках P

Для визначення зв'язків введемо Ξ – як умову взаємодії між множинами $Form_n$. Надалі будемо розглядати такий запис $Form_1^{master} \xrightarrow{\Xi} Form_n^{slave}$ як взаємодію (передача даних, виклик і т.д.) $Form_1^{master}$ через подію me_h^n, або ce_w^n подію, яка належать будь-якому GUI елементу, який належить $Form_1^{master}$ на $Form_n^{slave}$.

Для математичного опису зв'язку $Form_1^{master}$ (в (4.40)–(4.41) – $Form_1$) через подію ea_a^n з $Form_n^{slave}$ (в (4.40)–(4.41) – $Form_i$) використовується ce_w^z і запис приймає такий вид:

$$Form_1^{master} \xrightarrow{Form_1 PE_1(ea_2^1 \in EA_1 : me_4^1 \in ME_1)\Xi\{(ce_w^z \in CE_z^x) \in CD_x^n\} \in CF_n} Form_n^{slave} \qquad (4.40)$$

Наведена взаємодія одностороння, тобто подія me_4^1 належить $Form_1^{master}$ через «лінгвістичну змінну» ea_2^1, яка виконує якусь дію з події ce_w^z графічного елементу CD_x^n, який належить CF_n на $Form_n^{slave}$, при цьому не повертає $Form_1^{master}$ ніяких результатів. Надалі для роботи з взаємодією між $Form_1^{master}$ і $Form_n^{slave}$, визначимо такий запис:

$$Form_1^{master} \xleftarrow{Form_1(парметри)} \Xi \xleftarrow[Form_n(параметри)]{} Form_n^{slave} \qquad (4.41)$$

де під «параметрами» будемо розглядати які елементи $Form_1 PE$ і CD_x^1 взаємодіють з елементами $Form_n PE$ або CD_x^n в обох напрямках. Кількість взаємозв'язків між $Form_1^{master}$ і $Form_n^{slave}$ теоретично не мають обмежень, але в рамках даних досліджень визначимо, що кількість $Form_n \le 40$ і її елементи ce_w^z не можуть посилатися самі на себе за допомогою «Контейнера рішень».

Визначимо детально підмножини, які входять в множину $Form_n$ і доведемо їх існування і основні властивості які вони описують.

Нехай P – кібернетична складова CPPS, що розробляється, отже, множина $Form_n \subseteq P$, якщо кожен елемент множини $Form_n$ є множиною P (рис. 4.2).

Теорема 2. $Form_n = P$ тоді і тільки тоді, коли одночасно $Form_n \subseteq P$ і $P \subseteq Form_n$, то $Form_n = P \Leftrightarrow Form_n \subseteq P$ і $P \subseteq Form_n$. Позначимо умову $P(Form_n)$, що визначається так: не існує жодного елемента $Form_n$, який задовольняв би умові теореми 2. Наприклад, $P(Form_n) = \{Form_n \neq P\}$. У контексті даної умови можна сказати, що всі елементи $(Form_n PE, CF_n) \in Form_n \neq P$. Виходячи з цього, для даної умови можна визначити множину P, яка не містить в собі жодного загального елемента $(Form_n PE, CF_n) \in Form_n$ та позначимо її як порожню множину $\varnothing$. В даному дослідженні це означатиме, що множина $Form_n$ не має ніяких $\xrightarrow{\Xi}$ в множині P і дана множина вважається надмірною і, отже, $Form_n = \varnothing$.

Для доказу виконання умов об'єднання і перетину запропонованих множин, що входять в P, необхідно, щоб вони задовольняли наступним властивостям:

- комутативністю 2 і $2'$;
- асоціативністю 3 і $3'$;
- дистрибутивністю 4 і $4'$.

Теорема 3. Нехай $Form_n$, $Form_n PE$ і CF_n довільна множина властивостей і параметрів, які входять в множину P. Тоді для них справедливі такі рівності:

$$1.\ Form_n \cup Form_n = Form_n;$$

$$2.\ Form_n \cup Form_n PE = Form_n PE \cup Form_n;$$

$$3.\ (Form_n \cup Form_n PE) \cup CF_n = Form_n \cup (Form_n PE \cup CF_n);$$

$$4.\ (Form_n \cup Form_n PE) \cap CF_n = (Form_n \cap CF_n) \cup (Form_n PE \cap CF_n); \quad (4.42)$$

$$1'.\ Form_n \cap Form_n = Form_n;$$

$$2'.\ Form_n \cap Form_n PE = Form_n PE \cap Form_n;$$

$3'.\ (Form_n \cap Form_nPE) \cap CF_n = Form_n \cap (Form_nPE \cap CF_n);$

$4'.\ (Form_n \cap Form_nPE) \cup CF_n = (Form_n \cup CF_n) \cap (Form_nPE \cup CF_n).$

Доказ. Кожне твердження цих властивостей випливає з визначення операцій над множинами і теореми 2. Для даних тверджень необхідно і достатньо довести 4-у властивість. Решта властивостей операцій доводяться аналогічно. Для спрощення математичних записів позначимо через Ψ ліву і Θ праву частина рівності 4. Покажемо, що обидві умови теореми 2 виконуються. Для цього спочатку доведемо, що $\Psi \subseteq \Theta$. Для цього візьмемо будь-який довільний елемент $\psi \in \Psi$. Тоді ψ одночасно належить $Form_n \cup Form_nPE$ і CF_n, з даної умови $\psi \in Form_n \cup Form_nPE$ випливає, що відповідно $\psi \in Form_n$ або $\psi \in Form_nPE$. Виходячи з цього якщо $\psi \in Form_n$, то $\psi \in Form_n \cap CF_n$. Якщо $\psi \in Form_nPE$, то $\psi \in Form_nPE \cap CF_n$. Отже, в будь-якому випадку $\psi \in Form_n \cap CF_n$. Значить $\psi \in \Theta$.

Доведемо, що виконується зворотне включення $\Theta \subseteq \Psi$. Візьмемо довільний $\theta \in \Theta$, тоді можемо записати:

$$\theta \in (Form_n \cap CF_n) \cup (Form_nPE \cap CF_n) \Rightarrow \theta \in Form_i \cap CF_n$$

чи

$$\theta \in Form_nPE \cap CF_n.$$

Якщо $\theta \in Form_n \cap CF_n \Rightarrow \theta \in Form_n$ і $\theta \in CF_z \Rightarrow \theta \in Form_i \cup Form_iPE$ та $\theta \in CF_n \Rightarrow \theta \in (Form_n \cup Form_nPE) \cap CF_n = \Psi$. Виходячи з цього, якщо $\theta \in Form_nPE \cap CF_n \Rightarrow \theta \in Form_nPE$ і $\theta \in CF_n \Rightarrow \theta \in Form_n \cup Form_nPE$ та $\theta \in CF_n \Rightarrow \theta \in (Form_n \cup Form_nPE) \cap CF_n = \Psi$. Звідси $\theta \in \Psi$. З теореми 2 випливає, що $\Psi = \Theta$. Решта властивостей операцій над запропонованими множинами перевіряються аналогічно.

Визначимо доповнення множини P через позначення $\overline{P}$ як різницю між універсальною множиною I і множиною P звідси

$\overline{P} = \{ Form_n PE : Form_n PE \in I$ і $Form_n PE \notin P \}$.

Теорема 4. Доведемо властивості різниці множин $Form_n PE$ і CF_n за умови, що $\{Form_n PE, CF_n\} \in Form_n$, тоді вони повинні виконувати наступні властивості:

$$
\begin{aligned}
&1.\ Form_n PE \cup \varnothing = Form_n PE;\\
&2.\ Form_n PE \cup I = I;\\
&3.\ \overline{Form_n PE \cup CF_n} = \overline{Form_n PE} \cap \overline{CF_n};
\end{aligned}
\qquad (4.43)
$$

$$
\begin{aligned}
&1'.\ Form_n PE \cap \varnothing = \varnothing;\\
&2'.\ Form_n PE \cap \mathrm{I} = P;\\
&3'.\ \overline{Form_n PE \cap CF_n} = \overline{Form_n PE} \cup \overline{CF_n}.
\end{aligned}
\qquad (4.44)
$$

Доказ. Спочатку необхідно довести, що 3, 3′ виконують умови. Позначимо Λ ліву і Δ праву частину рівності. Покажемо, що $\Lambda \subseteq \Delta$ і $\Delta \subseteq \Lambda$.

Для будь-якого $\lambda \in \Lambda$ виконується $\lambda \notin Form_n PE \cup CF_z \Rightarrow \lambda \notin Form_n PE$, $\lambda \notin CF_n \Rightarrow \lambda \in \overline{Form_n PE}$ і $\lambda \in \overline{CF_n} \Rightarrow \lambda \in \Delta$. Отже, $\Lambda \subseteq \Delta$.

Для будь-якого $\upsilon \in \Delta$ виконується $\vartheta \in \overline{Form_n PE}$, $\lambda \in \overline{CF_n} \Rightarrow \lambda \notin \overline{Form_n PE}$ і $\lambda \notin CF_n \Rightarrow \lambda \notin Form_n PE \cup CF_n \Rightarrow \lambda \in Form_n PE$. Отже, $\Delta \subseteq \Lambda$.

Використовуючи розроблену структуру P, представлену на рис. 4.2, на базі наведених вище доказів, зробимо узагальнений математичний опис $Form_n$ у вигляді:

$$
\begin{aligned}
Form_n \in \{ & Form_n PE \subseteq ((mp_1^n, ..., mp_x^n) \in MP_n \cup (pp_1^n, ..., pp_a^n) \in PP_n) \wedge\\
& \wedge CF_n \subseteq [((pc_1^y, ..., pc_m^y) \in PC_y^1 \cap (ce_1^z, ..., ce_w^z) \in CE_z^1) \in CD_1^n],\\
& \ldots, [((pc_1^y, ..., pc_m^y) \in PC_y^x \cap (ce_1^z, ..., ce_w^z) \in CE_z^x) \in CD_x^n]\}
\end{aligned}
\qquad (4.45)
$$

Можна помітити, що запис (4.45) не описує всі можливі параметри, які притаманні $Form_n$, а має на увазі їх присутність у вигляді множини наборів параметрів $mp_1^n,...,mp_x^n$ і множини значень $pp_1^n,...,pp_a^n$, які можуть приймати параметри. Залежно від середовища розробки та мови високого рівня програмування набір параметрів може змінюватися. Але, в більшості випадків, значення які вони можуть приймати, описуються або в десятковій системі зчислення і вимірюються в pixel (в яких задаються розміри графічного вікна, його місце розташування і розташування елементів всередині нього та їх розміри), або за допомогою логічних значень («True», «False», які, наприклад, описують чи відображається елемент, чи ховається за певних подій), а також за допомогою зарезервованих лінгвістичних слів, що дають можливість спростити виконання дій з візуалізації і відображення інтерфейсу користувача для зручності роботи («Align», «BorderStyle», «Caption» і. т.д.).

Отже, кожному параметру mp_x^k може відповідати одне або кілька значень pp_a^n (залежно від способу задання значень). Виходячи з цього, в даному методі буде запропоновано два окремих випадки відповідності.

Окремий випадок 1. Визначимо відповідність ζ між множинами MP_n і PP_n, як довільну підмножину їх добутків $MP_n \times PP_n$, тобто $\zeta \subseteq MP_n \times PP_n$. Зауважимо, що ця відповідність складається з упорядкованих пар. Кожна пара $(mp_x^n, pp_a^n) \in \zeta$ показує, що параметру $mp_x^n \in MP_n$ відповідає значення $pp_a^n \in PP_n$ при відповідності ζ. Приклад представлення відповідності для першого окремого випадку між параметром і значенням наведено на рис. 4.3.

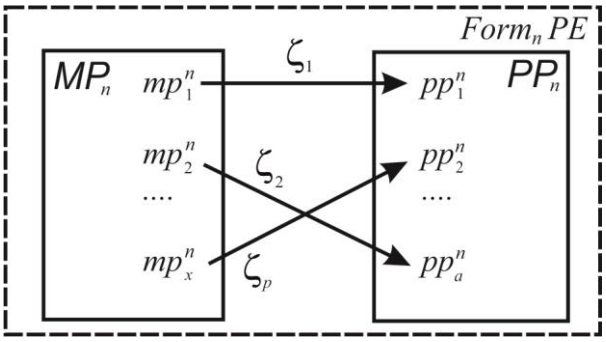


Рисунок 4.3 – Відповідність першого окремого випадку (коли параметру mp_x^n належить тільки одне значення pp_a^n)

Окремий випадок 2. Областю визначення відповідності ζ_1 назвемо множину $Dom\zeta = \{mp_x^n \in MP_n : де\ існує\ значення\ pp_a^n,\ що\ (mp_x^n, pp_a^n) \in \zeta\}$. Отже, дане визначення можна записати рекурсивно: для множини значень відповідності ζ, яка називає множину $\mathrm{Im}\,\zeta = \{pp_a^n \in PP_n : існує\ параметр\ mp_x^n \in MP_n,\ що\ (mp_x^n, pp_a^n) \in \zeta\}$.

На рис. 4.4 представлена відповідність для другого окремого випадку.

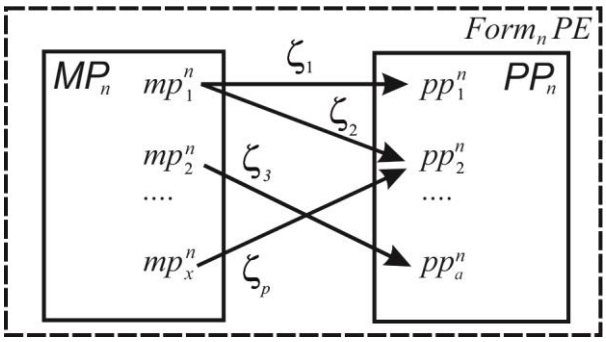


Рисунок 4.4 – Відповідність другого окремого випадку (коли параметру mp_x^n може належати багато значень pp_a^n).

Надалі позначимо відповідність між множиною параметрів $(mp_1^n, ..., mp_x^n) \in MP_n$ і множиною значень $(pp_1^n, .., pp_a^n) \in PP_n$ через такий запис:

- для першого окремого випадку

$$mp_1^n \xrightarrow{\zeta} pp_1^n \text{ або } \zeta : (mp_1^n) \in MP_n \to (pp_1^n) \in PP_n; \qquad (4.46)$$

- для другого окремого випадку

$$mp_1^n \xrightarrow{\zeta} (pp_1^n, pp_2^n, pp_{a-1}^n) \tag{4.47}$$

чи

$$\zeta : (mp_1^n) \in MP_n \to (pp_1^n, pp_2^n, pp_{a-1}^n) \in PP_n . \tag{4.48}$$

Визначимо відповідність ζ як таку, що виконує такі вимоги:

- усюди визначена, якщо $Dom\zeta = MP_n$;

- сюр'єктивна, якщо $\mathrm{Im}\,\zeta = PP_n$;

- однозначна, якщо кожному $mp_x^n \in Dom\zeta$ відповідає єдиний параметр pp_a^n з PP_n, тобто з $(mp_x^n, pp_a^n) \in \zeta$ і $(mp_x^n, pp_a^n) \in \zeta \Rightarrow pp_a^n = pp_1^n$;

- ін'єкційна, якщо різним параметрам $Dom\zeta$ відповідають різні параметри з PP_t, тобто $(mp_x^n, pp_a^n) \in \zeta$ і $(mp_x^n, pp_a^n) \in \zeta \Rightarrow mp_x^n = mp_1^n$. При виконанні цих вимог відображення матиме хоч одну відповідність (визначимо його частковою бієкцією).

Дане рішення може бути застосованим і достатнім для визначення множини значень $(pp_1^n,..., pp_a^n)$, які можуть приймати параметри $((mp_1^n ..., mp_x^n) \in MP_n)$ для завдання параметрів візуалізації $Form_n$, тобто всі необхідні параметри, які дають можливість створити «порожній» шаблон Windows Form ("вікна"), і не враховує подій і реакцій при взаємодії з $Form_n$, тобто такі дії як стандартні вбудовані події «Close», «Create» і реакції у вигляді «контейнера рішень», в якому розробником CPPS реалізовані варіанти взаємодії через GUI з даними, параметрами яких надходять з фізичної складової, або вирішують завдання функціоналу HMI на кібернетичному рівні у відповідності до вимог ТЗ і головної мети CPPS. Ґрунтуючись на вищесказаному, відповідності 1 і 2 окремих випадків для знаходження $mp_1^n \xrightarrow{\zeta} pp_1^n$ і $mp_1^n \xrightarrow{\zeta} (pp_1^n, pp_2^n, pp_{a-1}^n)$ опис $Form_n$ не підходять.

Для їх вирішення на рис. 4.2 визначені множини $(ME_n, EA_n) \in Form_n PE$, як невід'ємна частина множини $Form_n$. Розглядаючи запропоновану концепцію реалізації процесу розробки CPPS, визначимо призначення кожної множини:

$(me_1^n,...,me_h^n) \in ME_n$ – множина параметрів подій, які відповідають множині $Form_n PE$, де me_h^n описується лінгвістичної константою («Close», «Create», і т.д.) і може розширюватися в залежності від середовища розробки в рамках однієї мови об'єктно-орієнтованого програмування.

Множину $(ea_1^n,...,ea_z^n) \in EA_n$ введено для завдання природного лінгвістичного опису події, наприклад, ea_z^n можна описати простими словами або виразами: «Закрити форму», «Закрити форму зі збереженням даних», «Закрити вікно з діалоговим вікном» і т.д. Опис структури команд на природній мові представлено в розділі 5. Дане введення необхідно за твердженням $(mp_1^n...,mp_x^n) \in MP_n \notin (me_1^n,...,me_h^n) \in ME_n$, внаслідок того, що елементи $pp_a^n \neq ea_z^n$ відповідають за способом представлення значень опису.

Тоді виникає необхідність існування множини $(z_1^o, z_2^o,..., z_q^o) \in Z_o$, де Z_o – множина можливих фрагментів рішень у вигляді програмного коду виконання певної функції і процедури, яка містить «шаблон», що залежить від мови програмування і середовища розробки. Надалі, в рамках даних досліджень, визначимо його як «*Container Solution*», який визначається шляхом завдання унікального «*Linguistic Variable*» ea_z^n. Наведемо приклад взаємодії множини $(me_1^n,...,me_h^n) \in ME_n$ за допомогою «*Linguistic Variable*» $(ea_1^n,...,ea_z^n) \in EA_n$ і множини $(z_1^o, z_2^o,..., z_q^o) \in Z_o$ - «*Container Solution*» для середовища розробки Red Studio X3 (рисунок 4.5).

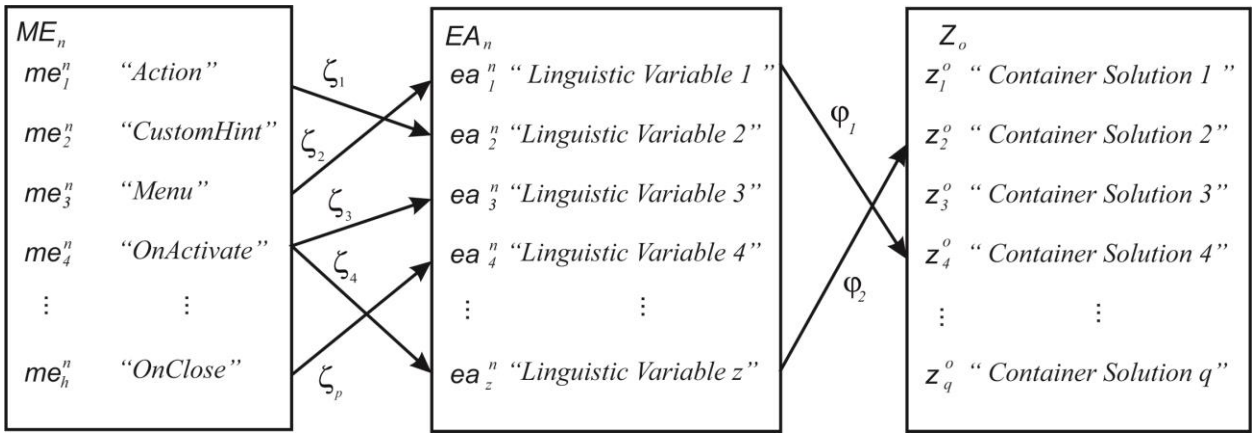


Рисунок 4.5 – Приклад взаємодії множини $(me_1^n,...,me_h^n) \in ME_n$ за допомогою «*Linguistic Variable*» $(ea_1^n,...,ea_z^n) \in EA_n$ і множини $(z_1^o, z_2^o,..., z_q^o) \in Z_o$ «*Container Solution*»

Визначимо існування зворотної відповідності до $\zeta \subseteq ME_n \times EA_n$ як $\zeta' = \{(ea_2^n, me_1^n) : (me_h^n, ea_z^n) \in \zeta\}$. Зауважимо, що $\zeta' \subseteq EA_n \times ME_n$, тому ζ' – це відповідність між EA_n і ME_n. На рисунку 4.6 показано зворотну відповідність ζ.

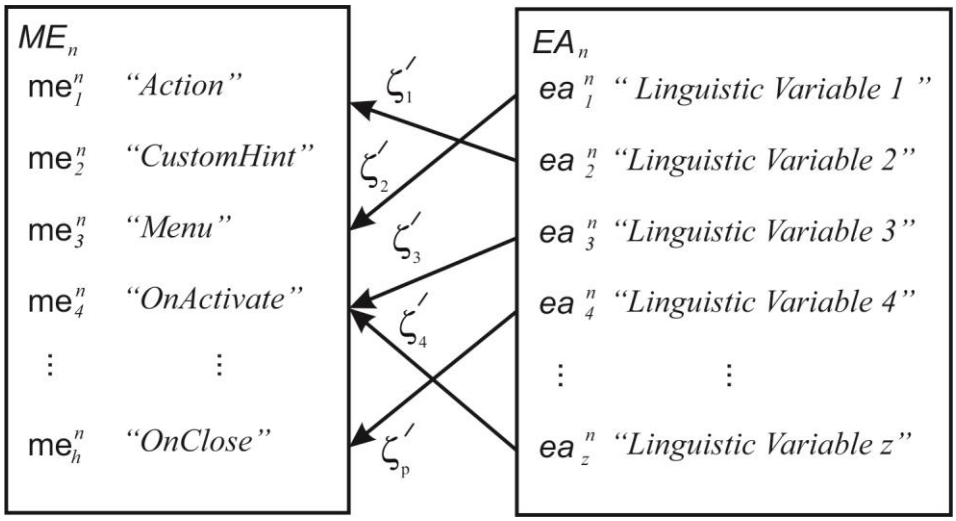


Рисунок 4.6 – Зворотна відповідність ζ

В даному випадку для рис. 4.6 математичний запис буде такий:

$$\zeta' = \{(ea_1^n, me_3^n), (ea_2^n, me_1^n), (ea_3^n, me_4^n), (ea_4^n, me_h^n), (ea_z^n, me_4^n)\} \quad (4.49)$$

Для доказу існування такого рішення необхідно визначити композиційну відповідність. Отже, з рис. 4.5 композиційною відповідністю $\zeta \subseteq ME_n \times EA_n$ і $\phi \subseteq EA_n \times Z_o$ назвемо таку відповідність $\chi \subseteq ME_n \times Z_o$, що $\chi = \{(me_h^n, z_q^o) : існує\ \ елемент\ \ ea_z^n \in EA_n,\ \ що\ \ (me_h^n, ea_z^n) \in \zeta\ \ \ i\ \ \ (ea_z^n, z_q^o) \in \varphi\}$. Надалі в даному дослідженні композиційну відповідність будемо записувати як $\chi = \zeta \circ \varphi$. Тоді композиційна відповідність для рис. 4.5 і є множиною, яку представимо у вигляді системи:

$$\begin{cases} \zeta_2 \circ \varphi_1 = \{(me_{h-1}^n, z_4^0)\} \\ \zeta_4 \circ \varphi_2 = \{(me_h^n, z_2^o)\} \end{cases} \tag{4.50}$$

Теорема 5. Про існування часткової бієкції ζ' і $\zeta \circ \varphi$. Якщо $\zeta \subseteq ME_n \times EA_n$ і $\varphi \subseteq EA_n \times Z_o$ – мають властивості часткової бієкції, то повинні відповідати таким властивостям:

1. ζ' є частковою бієкцією між EA_n і ME_n;

2. $\zeta \circ \varphi$ є частковою бієкцією між ME_n і Z_o.

Доказ:

1. Так як ζ частково безумовно, то ζ' сюр'єктивне, а отже, ζ' всюди визначено хоч одним зв'язком. Дві властивості, що залишаються перевіряються аналогічно.

2. Оскільки ζ і φ всюди частково визначені, то їх композиція $\zeta \circ \varphi$ буде визначена хоча б в одному параметрі me_h^n множини ME_n. Так як ζ діє на ea_z^n *(«Linguistic Variable»)* множини EA_n і φ – на всі можливі z_q^o *«Container Solution»* множини Z_o, то $\zeta \circ \varphi$ є сюр'єктивною відповідністю. Однозначність і ін'єктивність доводяться аналогічно.

Розширимо математичну множину $CF_n \in Form_n$ (рис. 4.2) і визначимо її як набір підмножин представлених на рис. 4.7.

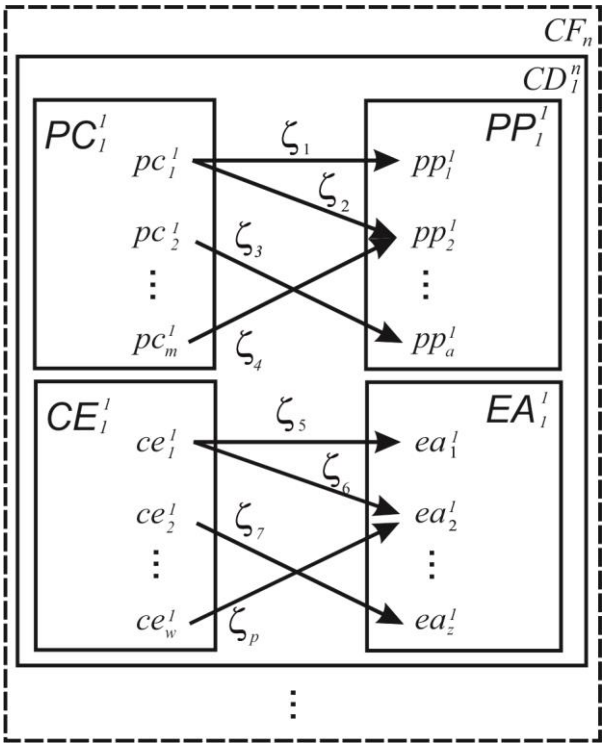


Рисунок 4.7 – Подання множини CF_n для опису властивостей елементів управління роботи з даними

Визначимо основні властивості множини CF_n як $(CD_1^n,...,CD_x^n) \in CF_n$, яка містить в собі певну кількість візуальних графічних елементів HMI, та є необхідною і достатньою для реалізації всіх функцій і операцій для досягнення головної мети розробки CPPS. Під візуальними компонентами будемо розуміти GUI елементи інтерфейсу («Grid», «Table», «Menu», «Button», і т.д.). Запис CD_1^n означає наступне: n – номер $CF_n \in Form_n$, якому належить візуальний компонент, а 1 – його нумерація в множині CF_n.

Це пояснюється тим, що множина опису візуальних компонентів $CD_1^n,...,CD_x^n$, які формують HMI взаємодії управління з даними ,які, в свою чергу, можуть мати різне призначення, а отже, різний набір параметрів як для графічного подання, так і для подій, які вони можуть обробляти. Зауважимо, що один елемент CD_x^n може бути використаний безліч разів, але при цьому мати різні призначення відповідно до логіки, закладеної розробником. Тому визначимо, що $CD_x^n \in (PC_y^x, PP_t^x, CE_z^x, EA_p^x)$ мають властивості, доведені в

теоремі 1-4 і так як:

$$CF_n \in \{CD_1^n \in (PC_y^1, PP_t^1, CE_z^1, EA_p^1), ..., CD_x^n \in (PC_y^x, PP_t^x, CE_z^x, EA_p^x)\},$$

є невід'ємною частиною $Form_n \in P$. Тому для спрощення реалізації даного методу і ґрунтуючись на особливостях задання основних геометричних параметрів опису візуалізації елементів форм, приймемо, що множина $CF_n \in (PP_t^n, EA_p^n) = Form_n PE(PP_t^n, EA_p^n)$ містить всі припустимі параметри для опису візуальних властивостей PP_t^n і «лінгвістичних імен» EA_p^n як форми ($Form_n PE$), так і компонентів CF_n, що дозволить в майбутньому використовувати одну базу знань. Використання даного рішення дозволяє на базі доведених двох окремих випадків (рис. 4.3 - 4.4) реалізувати зв'язок між елементами як:

$$\begin{aligned}\varepsilon = \{&(pc_1^n, pp_1^n), (pc_1^n, pp_2^n), (pc_2^n, pp_a^n), (pc_m^n, pp_2^n),\\ &(ce_1^n, ea_1^n), (ce_1^n, ea_2^n), (ce_2^n, ea_z^n), (ce_w^n, ea_z^n)\}\end{aligned} \quad (4.51)$$

за умови, що всі властивості ζ, які доведені вище зберігаються для ε. Внаслідок цієї взаємодії множини $(ce_1^n, ..., ce_w^n) \in CE_n$ за допомогою лінгвістичного «імені» $(ea_1^n, ..., ea_z^n) \in EA_n$ і множини $(z_1^o, z_2^o, ..., z_q^o) \in Z_o$ «Container Solution». Останнє буде таке ж, як представлене на рис. 4.5 і збереже всі властивості і визначення для CF_n.

Спираючись на запропоновані рішення, в рамках даних досліджень, визначимо наступну форму запису для множини $(Form_n PE, CF_n) \in Form_n$ і призначення кожної з її підмножин:

- математичний опис множини $Form_n PE$:

$$Form_n PE \in \underbrace{((mp_1^n, mp_2^n, ..., mp_x^n) \in MP_n}_{\text{Безліч параметрів}} \xrightarrow{\zeta_p} \underbrace{((pp_1^n, pp_2^n, ..., pp_a^n) \in PP_n)}_{\text{Безліч значень}} \wedge$$

$$\wedge \underbrace{((me_1^n, me_2^n, ..., me_h^n) \in ME_n}_{\text{Безліч подій}} \xrightarrow{\zeta_p} \underbrace{((ea_1^n, ea_2^n, ..., ea_z^n) \in EA_n)}_{\text{Безліч "лінгвістичних імен"}} \xrightarrow{\varphi_e} \tag{4.52}$$

$$\xrightarrow{\varphi_e} \underbrace{((z_1^o, z_2^o, ..., z_q^o) \in Z_o)}_{\text{Безліч "контейнерів рішень"}} \cdot$$

- математичний опис множини CF_n:

$$CF_n \in (\underbrace{CD_x^n}_{\textit{елемент}} \in [\underbrace{((pc_1^x, pc_2^x, ..., pc_m^x) \in PC_y^x)}_{\text{Безліч параметрів елемента}} \xrightarrow{\varepsilon_v} \underbrace{((pp_1^x, pp_2^x, ..., pp_a^x) \in PP_t^x}_{\text{безліч значень}} \wedge$$

$$\wedge \underbrace{((ce_1^z, ce_2^z, ..., ce_w^z) \in CE_z^x)}_{\text{Безліч подій}} \xrightarrow{\varepsilon_v} \underbrace{((ea_1^p, ea_2^p, .., ea_z^p) \in EA_p^x)}_{\text{Безліч "лінгвістичних імен"}} \xrightarrow{\varphi_e} \tag{4.53}$$

$$\xrightarrow{\varphi_e} \underbrace{((z_1^o, z_2^o, ..., z_q^o) \in Z_o)}_{\text{Безліч "контейнерів рішень"}}]) .$$

Виходячи з цього $Form_n$ можна представити наступним чином:

$$
\begin{aligned}
&Form_n^{master} \in \{Form_n PE \in ((mp_1^n, mp_2^n, ..., mp_x^n) \in MP_n) \xrightarrow{\xi} (pp_1^n, pp_2^n, ..., pp_a^n) \in PP_n)) \wedge ((me_1^n, me_2^n, ..., me_h^n) \in ME_n) \xrightarrow{\xi} \\
&\xrightarrow{\xi} ((me_1^n, me_2^n, ..., me_h^n) \in ME_n) \xrightarrow{\xi} ((ea_1^n, ea_2^n, ..., ea_z^n) \in EA_n) \xrightarrow{\varphi} ((z_1^o, z_2^o, ..., z_q^o) \in Z_o) \wedge \\
&\wedge CF_n \in (CD_x^n \in [((pc_1^y, pc_2^y, ..., pc_m^y) \in PC_y^x) \xrightarrow{\varepsilon} ((pp_1^t, pp_2^t, ..., pp_a^t) \in PP_t^x) \wedge ((ce_1^z, ce_2^z, ..., ce_w^z) \in CE_z^x) \xrightarrow{\varepsilon} \\
&\xrightarrow{\varepsilon} ((ea_1^p, ea_2^p, ..., ea_z^p) \in EA_p^x) \xrightarrow{\phi} ((z_1^o, z_2^o, ..., z_q^o) \in Z_o)]) \wedge (CD_{x+1}^n \in [((pc_1^y, pc_2^y, ..., pc_m^y) \in PC_y^{x+1}) \xrightarrow{\varepsilon} \\
&\xrightarrow{\varepsilon} ((pp_1^t, pp_2^t, ..., pp_a^t) \in PP_t^{x+1}) \wedge ((ce_1^z, ce_2^z, ..., ce_w^z) \in CE_z^{x+1}) \xrightarrow{\varepsilon} ((z_1^o, z_2^o, ..., z_q^o) \in Z_o)]) \wedge \\
&\wedge (CD_{x+2}^n \in [((pc_1^y, pc_2^y, ..., pc_m^y) \in PC_y^{x+2}) \xrightarrow{\varepsilon} ((pp_1^t, pp_2^t, ..., pp_a^t) \in PP_t^{x+2}) \wedge ((ce_1^z, ce_2^z, ..., ce_w^z) \in CE_z^{x+2}) \xrightarrow{\varepsilon} \\
&\xrightarrow{\varepsilon} ((ea_1^p, ea_2^p, ..., ea_z^p) \in EA_p^{x+2}) \xrightarrow{\varphi} ((z_1^o, z_2^o, ..., z_q^o) \in Z_o)]) \wedge (CD_{x+3}^n \in [((pc_1^y, pc_2^y, ..., pc_m^y) \in PC_y^{x+3}) \xrightarrow{\varepsilon} \\
&\xrightarrow{\varepsilon} ((pp_1^t, pp_2^t, ..., pp_a^t) \in PP_t^{x+3}) \wedge ((ce_1^z, ce_2^z, ..., ce_w^z) \in CE_z^{x+3}) \xrightarrow{\varepsilon} ((ea_1^p, ea_2^p, ..., ea_z^p) \in EA_p^{x+3}) \xrightarrow{\varphi} \\
&((z_1^o, z_2^o, ..., z_q^o) \in Z_o)]) \wedge \ldots \\
&\qquad\qquad \vdots \\
&\qquad\qquad \vdots \\
&\wedge CF_n \in (CD_x^{n+1} \in [((pc_1^y, pc_2^y, ..., pc_m^y) \in PC_y^x) \xrightarrow{\varepsilon_{v-1}} ((pp_1^t, pp_2^t, ..., pp_a^t) \in PP_t^x) \wedge \\
&\wedge ((ce_1^z, ce_2^z, ..., ce_w^z) \in CE_z^x) \xrightarrow{\varepsilon_v} ((ea_1^p, ea_2^p, ..., ea_z^p) \in EA_p^x) \xrightarrow{\varphi_e} ((z_1^o, z_2^o, ..., z_q^o) \in Z_o)])\}
\end{aligned}
\tag{4.54}
$$

Варто врахувати, що кібернетична складова CPPS складається з множини Windows Form, кожна з яких містить елементи GUI, сукупності яких реалізують HMI та дозволяють контролювати і маніпулювати великими обсягами виробничих параметрів. Отже, для зручності візуального подання інформації в сучасних CPPS необхідно використовувати групування даних за загальними логічними ознаками. Для цього застосовується багатовіконний інтерфейс, тобто набір множини $Form_n$. Такий підхід вимагає розробки графічного методу подання та опису зв'язків між формами. Ґрунтуючись на (4.39) і (4.40), необхідно розробити графічне представлення $Form_n$ її видів і типів зв'язків між ними.

В ході дослідження для вирішення даного завдання були проаналізовані наступні методи:

- структурний аналіз Гейне-Сарсона;
- структурний аналіз проєктування Йодана Де Марко;
- системи Джексона;
- структурні системи Варнье-Орра;
- аналіз і проєктування СРВ Уорда-Меллора і Хатлі;
- інформаційне моделювання Мартіна;
- інформація сигнально-кодової конструкції (СКК), методологія Константайна.

Ґрунтуючись на специфіці завдань дослідження і складності візуального представлення інформації та її кількості, запропоновано використовувати методологію СКК, яка представляється в блоковому вигляді, де кожен блок є модулем, тобто модель відносин ієрархії між формами кібер складової CPPS. При цьому циклічні і умовні виклики модулів моделюються спеціальними вузлами, тому потоки повинні проходити через ці вузли. Міжмодульні зв'язки за даними і управлінням також моделюються спеціальними вузлами, які прив'язані до потоків (тобто до викликів модулів, які в даному дослідженні позначено як $Form_n$ і є невід'ємною частиною P). Базовими елементами опису є наступні типи

модулів:

- модуль подання фрагменту зв'язку і його локалізацію на діаграмі;

- підсистема раніше визначених модулів, деталізованих за допомогою діаграм або математичного опису, які можуть повторюватися будь-яку кількість разів;

- бібліотека визначається поза проєктом даної системи;

- область даних використовується для визначення модулів, які не містять області глобальних і розподілених даних.

Обрана методологія СКК використовує власні елементи побудови та подання структурної карти у відповідності до стандартів IBM, ISO і ANSI. Для даної методології розроблені правила зв'язності і зчеплення модулів, що дає можливість спростити подання взаємодії між формами.

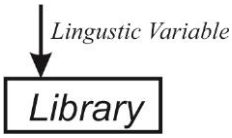
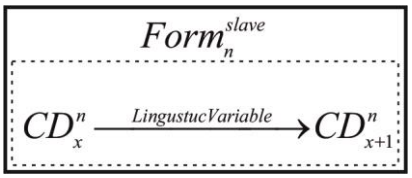
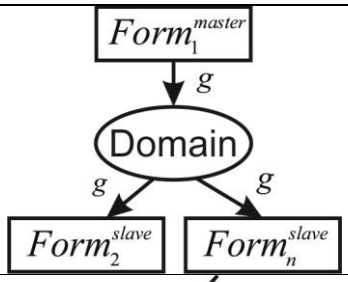
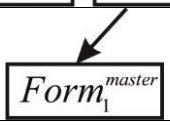
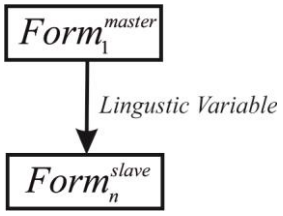
Виходячи з вищесказаного, в рамках даних досліджень, проведемо адаптацію методології Константайна. Це дозволить спростити процес управління розробкою CPPS і дасть можливість розробнику уявити візуальний опис і графічне представлення взаємодії між $Form_n$ на базі отриманих алгоритмів функціонування.

У таблиці 4.1 представлена модифікована модель графічного представлення методології Константайна для розробки візуалізації зв'язків між Windows Forms ghb розробки кібернетичної складової CPPS.

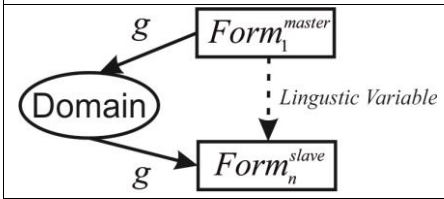
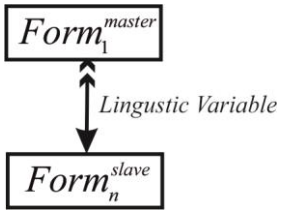
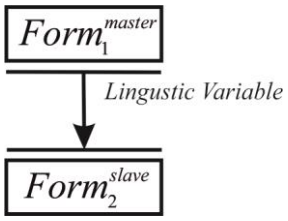
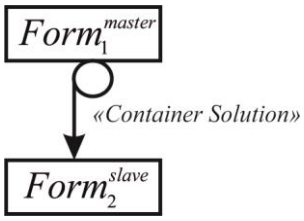
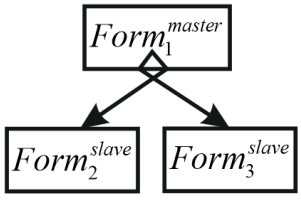
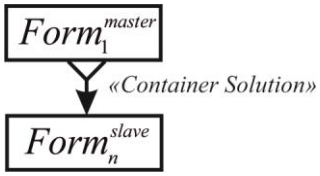
Таблиця 4.1 – Модифікована модель графічного представлення методології Константайна

Графічне представлення	Опис
1	2
$Form_1^{master}$ $Form_n^{slave}$	$Form_1^{master} \in [(Form_1 PE \in (MP_1 \in (mp_1^1,...,mp_x^1) \wedge$ $\wedge PP_1 \in (pp_1^1,...,pp_a^1) \wedge ME_1 \in (me_1^1,...,me_h^1)) \wedge$ $\wedge (CF_1 \in (CD_x^1 \in (PC_y^x (pc_1^y,...,pc_m^y) \wedge$ $\wedge PP_t^x \in (pp_1^t,...,pp_a^t) \wedge CE_z^x (ce_1^z,...,ce_w^z),\ldots))))]$

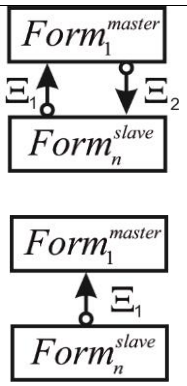
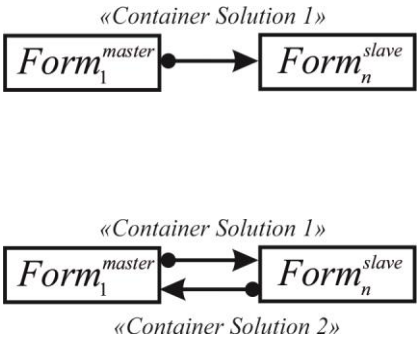
Продовження табл. 4.1

1	2
	$EA_n \in (ea_1^n,...,ea_z^n) \xrightarrow{\varphi} Z_o \in (z_1^o,...,z_q^o)$ як певний («Linguistic Variable») ea_z^n, якому належить множина або єдиний «Container Solution» $((z_1^o) \vee (z_2^o, z_3^o,..., z_q^o))$, що задовольняє умові φ
	$CF_n \in (CD_x^n \in (PC_y^x(pc_1^y,..., pc_m^y) \wedge$ $\wedge PP_t^x \in (pp_1^t,..., pp_a^t) \wedge CE_z^x(ce_1^z,...,ce_w^z)) \xrightarrow{\varphi_n}$ $\xrightarrow{\varphi_n} (CD_{x+1}^n \in (PC_y^{x+1}(pc_1^y,..., pc_m^y) \wedge PP_t^{x+1} \in$ $\in (pp_1^t,..., pp_a^t) \wedge CE_z^{x+1}(ce_1^z,...,ce_w^z)))$ при взаємодії $CD_x^n \to CD_{x+1}^n$, в рамках однієї з $Form_n^{master}$, за умов φ
	Набір глобальних змінних, які передають необхідні дані між $Form_1^{master} \xrightarrow{g} Form_2^{slave}$ за умови, що $P \in (Form_1^{master}, Form_2^{slave})$ є необхідними для подальшої роботи g.
	Ініціалізація $Form_1^{master}$ за умови, що ніякі дані на неї не передаються
	Посилання на елемент $CD_x^n \in (CE_z^x \in (ce_1^n,...,ce_w^n))$, де ce_w^n – подія яка є необхідною для вирішення поставленого завдання на базі «Linguistic Variable» $ea_z^p \in EA_p^x$ за допомогою z_q^o «Container Solution»
	Зв'язок за даними (змінними) Ξ, типи даних представлені в параметричній моделі
	Зв'язок за управлінням Ψ візуальними елементами подіями CD_x^n через «Container Solution»
	Потік – виклик (Ω) $Form_1^{master}$ в $Form_2^{slave}$. Використовується для графічного позначення виконання асинхронних і синхронних операцій, що дозволяє виконувати тривалі операції і паралельно з цим працювати з інтерфейсом $Form_1^{master}$.

Продовження табл. 4.1

1	2
	Паралельний виклик – робота з асинхронними операціями, що дозволяє при тривалих розрахунках взаємодіяти з $Form_1^{master}$, а результат виконання буде виведений в $Form_2^{slave}$.
	Послідовний виклик – робота з графічним інтерфейсом $Form_1^{master}$ блокується для користувача, поки тривала операція не буде виконана, результат виводитися в $Form_2^{slave}$. Після завершення роботи користувача з $Form_2^{slave}$ за його результатами якого вона буде закрита, користувач може продовжити роботу з $Form_1^{master}$.
	$Form_1^{master}$ викликає $Form_2^{slave}$ як співпрограму, яка використовує принцип багатопотокового коду, при цьому не вимагаючи наявності декількох потоків, а може виконуватися всередині одного потоку. В рамках даних досліджень, при розробці CPPS, співпрограми корисні для реалізації методів кінцевих автоматів, генераторів (ітераторів) тощо.
	Цикл – призначений для організації багаторазового виконання набору інструкцій, поки не будуть виконані умови. «Container Solution», що містить елемент циклу з умовою, що належить $Form_1PE$ або елементу CD_x^1, які по завершенню ініціалізують передачу результату в $Form_2^{slave}$.
	Розгалуження «Container Solution» забезпечує виконання певних наборів команд (команди) за умови істинності деякого логічного виразу. Може існувати з однією гілкою, з двома гілками, з декількома умовами. При виконанні «Container Solution», який належить $Form_1^{master}$ в залежності від виконання умов істинності може ініціалізувати передачу результату в $Form_2^{slave}$ або $Form_3^{slave}$.
	Одноразове виконання «Container Solution», яка належить $Form_1^{master}$, в строго заданій послідовності програмного коду викликає $Form_n^{slave}$.

Продовження табл. 4.1

1	2
	Графічне подання взаємодії (4.40): - двостороннє $\xrightarrow{Form_1^{mater}(значення)} \Xi \xleftarrow{Form_n^{slave}(значення)}$ - одностороннє $Form_1^{master}(значення)\Xi \xleftarrow{Form_n^{slave}(значення)}$ через глобальну змінну Ξ, в рамках одного P, яка визначена в будь-якій $Form_1^{master},...,Form_n^{slave}$ або належить CD_x^n.
	Зв'язок по управлінню (4.39) - односторонній через подію, що належить $Form_1^{master}$ на $Form_n^{slave}$ $Form_1^{master}[FP_1(ea_z^p \in EA_p^1 : me_h^d \in (ME_d^1)] \xrightarrow{\Psi_i^1}$ $\xrightarrow{\Psi_i^1} Form_n^{slave}[CF_z^n\{(ce_w^z \in CE_z^n : pc_l^f \in (PE_f^n)) \in CF_z^n\}]$ - двосторонній $Form_1^{master}[FP_1(ea_z^p \in EA_p^1 : me_h^d \in (ME_d^1)] \xrightarrow{\Psi_1^1} \Xi$ $\xleftarrow[\Psi_n^2]{} Form_n^{slave}[CF_z^n\{(ce_w^z \in CE_z^z : pc_l^f \in (PE_f^n)) \in CF_z^n\}]$ де Ψ_l^i – «Linguistic Variable», який вказує на той чи інший z_q^o «Container Solution».

Відповідно до таблиці 4.1, в даному методі пропонується виділити 2 основні групи зв'язків:

- група зв'язків всередині $Form_n$:

ζ_n – зв'язок існування, що показує, якому параметру або події $Form_n$ належить певне значення (чисельне, зарезервоване, логічне) або «Linguistic Variable»;

φ_n – зв'язок, що показує, якому певному «Linguistic Variable» належить «Container Solution» за умови існування ζ_n.

- група зв'язків між $Form_1^{master},...,Form_n^{slave}$ в рамках одного P:

Ξ – зв'язок за даними, здійснюється ведення глобальної змінної певного типу (int, char), яка містить в собі дані, що отримуються у результаті виконання «Container Solution» і слугують для подальшої обробки в

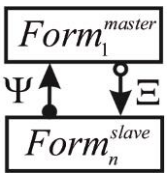
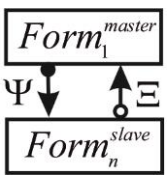
наступному «Container Solution»;

Ψ – зв'язок з управлінням через використання подій, притаманних елементам CD_x^n, які задаються за допомогою «Linguistic Variable» і містять певний «контейнер рішень» у вигляді програмного коду;

Ω – потік, виклик для вирішення тривалих «Container Solution» (при виконанні складних розрахункових завдань, обробки великих обсягів даних, виконання завдань на рівнях $AEofS_i$ і передачі їх на рівень декомпозиції вище), які необхідно реалізувати за допомогою асинхронних і синхронних операцій, в даному випадку використовується синтез «контейнерів рішень».

Необхідно врахувати можливість існування комбінованих типів зв'язків між $Form_1^{master},...,Form_n^{slave}$. Коли використовується Ξ для передачі даних в $Form_n$ викликається відповідна реакція у вигляді ініціалізації CD_x^n на $Form_{n+1}$ через Ψ. Графічне представлення комбінованих типів зв'язків представлено в таблиці 4.2.

Таблиця 4.2 – Модель графічного представлення комбінованих типів зв'язків

Графічне представлення	Опис
	$Form_1^{master}$ взаємодіє за допомогою глобальної змінної ϖ, яка містить числове або логічне значення (тип зв'язку Ξ) на $Form_n^{slave}$, яка після обробки в «контейнері рішень» повертає результат обробки «контейнера рішень» на $Form_1^{master}$ у вигляді візуального відображення або реакцій з елементу CD_x^1 (тип зв'язку Ψ).
	$Form_1^{master}$ через подію графічного елементу інтерфейсу CD_x^1 з «контейнера рішень» (тип зв'язку Ψ) викликає $Form_n^{slave}$ за певною подією, до якої належить «контейнер рішень». Після виконання отриманий результат повертається через глобальну змінну ϖ (тип зв'язку Ξ) на $pc_n^i \in PC_n^i$ (без використання елементів CD_{x+1}^1 на $Form_1^{master}$).

4.4. Формальне подання властивостей і подій форм та його компонентів

На базі запропонованого математичного опису множини $Form_n PE$ (4.52) як співвідношення $MP_n \xrightarrow{\zeta} PP_n$ і $ME_n \xrightarrow{\zeta} EA_n \xrightarrow{\varphi} Z_o$, а також графічних елементів CD_x^n (4.22) у вигляді співвідношень $PC_y^x \xrightarrow{\varepsilon} PP_t^x$ і $CE_z^x \xrightarrow{\varepsilon} EA_p^x \xrightarrow{\varphi} Z_o$ в рамках даних досліджень необхідно провести формальний опис значень і «лінгвістичних змінних», які відповідають опису властивостей і подій властивих формі і графічним елементам (GUI).

Для цього необхідно систематизувати типи подання значення і «лінгвістичні змінні», в залежності від їх задання, представлених в алгоритмі функціонування CPPS. В ході аналізу сучасних об'єктно-орієнтованих середовищ розробки було виділено такі типи представлення значень (табл. 4.3).

Таблиця 4.3 – Типи представлення значень PP_t^x, PP_n, EA_p^x, EA_n

Тип подання значень	Застосування	Приклад
1	2	3
Текстовий $(a,b,c,...,z)$ $(a,б,c,...,я)$	PP_t^x, PP_t	$pp_a^t \in PP_t^x \equiv PP_n = Name\ of\ the\ main\ form$ (назва форми для користувача задається розробником)
Лінгвістичний *(aaab bbcc,...,aabb)*	EA_p^n, EA_n	$ea_z^p \in EA_p^z =$"*AllCloseForm*" лінгвістичне ім'я контейнера рішень
Логічний (*true, false*)	PP_t^x, PP_n	$pp_a^t \in PP_t^x \equiv PP_n = false$ (невикористання певного параметра, обирається розробником)
Цілочисельний $(0,1,2,...,n)$ $(0,1,2,...,n)$	PP_t^x, PP_n	$pp_a^t \in PP_t^x \equiv PP_n = 380$ (розмір довжини $Form_n$ координати розташування)

Продовження табл. 4.3

1	2	3
Цілочисельний негативний (−1)	PP_t^x, PP_n	Застосовується для індексації, при значенні (−1) – індексація відсутня, а якщо є, то тип цілочисельний
Текстове слово, словосполучення ($aa, ab, ..., aabb$) (зарезервовані середовищем розробки)	PP_t^x, PP_n	$pp_a^t \in PP_t^x \equiv PP_n = clBtnFace$ (визначення значення кольору фону для $Form_n$ в середовищі RAD Studio XE3)

Текстовий тип використовується для визначення призначення параметрів, які в більшості випадків, при розробці інтерфейсу користувача носять інформаційний статичний характер.

Лінгвістичний – привласнення унікального визначення для «лінгвістичного імені» EA_n певній події, яку потрібно обробити за допомогою «контейнера рішень». Z_o задається розробником. «Лінгвістичне ім'я» неповторне в множині EA_n.

Булевий тип дозволяє розробнику вибрати активність / пасивність обраного параметра. У більшості випадків він використовується для задання значень параметрів $mp_x^n \in MP_n$ і $pc_m^y \in PC_y^n$ для опису $Form_n$ і елементів CD_x^n.

Цілочисельний тип використовується для опису значень, які задаються користувачем в *pix*. В основному служать для визначення розмірів $Form_n$, а також розмірів і координат розташування CD_x^n в рамках $Form_n$.

Цілочисельний негативний тип використовується для індексації кількості можливого вибору в реалізації графічного інтерфейсу користувача. В основному застосовується при нумерації image, зустрічається як значення параметрів $mp_x^n \in MP_n$ і $pc_m^y \in PC_y^n$.

Текстове слово або словосполучення – значення у вигляді слів або скорочень, які суворо фіксовані в середовищі розробки. Кожному певному

параметру $mp_x^n \in MP_n$, $pc_m^y \in PC_y^n$ може належати певний набір двох або більше значень pp_a^t.

Для спрощення формалізації подання значень PP_t^x, PP_n, EA_p^x, EA_n пропонується згрупувати в дві групи за ознаками опису параметрів форм MP_n і елементів PC_y^x, а також за подіями форм $me_h^n \in ME_n$ і подіями елементів $ce_w^z \in CE_z^n$:

- формальне подання значень PP_t^x, PP_n форм і графічних елементів:

1. Цілочисельне:

$$mp_x^n \vee pc_m^y = \begin{cases} a^1, \textit{якщо } a_i \le pp_a^n \le [\textit{значення}] \\ a^2, \textit{якщо } [\textit{значення}] \le pp_a^n \le [\textit{значення}] \\ \vdots \\ a^2, \textit{якщо } [\textit{значення}] \le pp_a^n \le a_o \end{cases} \quad (4.55)$$

де $mp_x^n \vee pc_m^y$ – позначення $a-ого$ параметра для MP_n і PC_y^x відповідно;

$a^1.a^2,\ldots,a^n$ – ідентифікатори діапазонів значень;

a_i, a_j – граничні значення, які: $a_i \to \min$; $a_j \to \max$;

$[\textit{значення}]$ – виділені порогові значення параметра.

2. Булеве:

$$mp_x^n \vee pc_m^y = \begin{cases} a^1, \textit{якщо } pp_1^n = [\,false] \\ a^2, \textit{якщо } pp_2^n = [true] \end{cases} \quad (4.56)$$

де $mp_x^n \vee pc_m^y$ – позначення $a-ого$ параметра для MP_n і PC_y^x відповідно;

$a^1.a^2$ – ідентифікатори діапазонів значень;

[*true, false*] – визначення логічного значення.

3. Текстове слово або словосполучення:

$$mp_x^n \vee pc_m^y = \begin{cases} a^1, \textit{якщо } pp_1^n = [\textit{слово, словосполучення}] \\ a^2, \textit{якщо } pp_2^n = [\textit{слово, словосполучення}] \\ \vdots \\ a^n, \textit{якщо } pp_a^n = [\textit{слово, словосполучення}] \end{cases} \quad (4.57)$$

де $mp_x^n \vee pc_m^y$ – позначення $a-ого$ параметра для MP_n і PC_y^x відповідно;

$a^1.a^2,\ldots,a^n$ – ідентифікатор значень;

[*слово, словосполучення*] – визначення тексту або словосполучення.

4. Цілочисельне негативне:

$$mp_x^n \vee pc_m^y = \begin{cases} a^1, \textit{якщо } a_i = pp_a^n = -1 \\ a^2, \textit{якщо } [1] \le pp_a^n \le [\textit{значення}] \\ \vdots \\ a^n, \textit{якщо } [\textit{значення}] < pp_a^n \le a_j \end{cases} \quad (4.58)$$

де $mp_x^n \vee pc_m^y$ – позначення $a-ого$ параметра для MP_n і PC_y^x відповідно;

$a^1.a^2,\ldots,a^n$ – ідентифікатори діапазонів значень;

a_i, a_j – граничні значення, які: $a_i = -1;\ a_j \to \max$;

$[\textit{значення}]$ – виділені порогові значення параметра.

- формальне подання подій форм ME_n і графічних елементів CE_z^x.

1. Лінгвістичне:

$$me_h^n \vee ce_w^z = \begin{cases} a^1, \textit{якщо } ea_1^p = [\textit{слово}] \\ a^2, \textit{якщо } ea_2^p = [\textit{слово}] \\ \vdots \\ a^n, \textit{якщо } ea_z^p = [\textit{слово}] \end{cases} \quad (4.59)$$

де $me_h^n \vee ce_w^z$ – позначення $n-oї$ події для ME_n і CE_z^x відповідно;

$a^1, a^2,\ldots,a^n$ – ідентифікатори діапазонів значень;

[*слово*] – визначення, «лінгвістичне ім'я» контейнера рішень, яке є неповторним і унікальним в EA_p.

На базі вищевказаного, наведемо приклад математичного опису та графічного представлення зв'язків з таблиць 4.1 і 4.2.

Нехай існує P як проєктований CPPS, який складається з двох діалогових форм. $Form_1^{master}$ виступає у вигляді головної форми розроблювального CPPS, яка представлена множиною $Form_1^{master} PE$ (описує необхідний і достатній набір параметрів ME_n і їх значень PP_t^n, які припустимі для кожного параметра $mp_1^n,..,mp_x^n$, при цьому кількість і назва параметрів залежить від обраного середовища розробки), якій відповідає певний $pp_1^t,...,pp_a^t$ набір, значення яких припустимо кожному mp_x^n. Отже, можна записати наступну відповідність:

$$MP_n \xrightarrow{\zeta_i} PP_n \text{ як } \underbrace{\begin{cases} mp_1^n(Caption) \xrightarrow{\zeta_1} \\ mp_2^n(ClientHeight) \xrightarrow{\zeta_2} \\ \vdots \\ mp_x^n(Visible) \xrightarrow{\zeta_n} \end{cases}}_{параметр} \underbrace{\begin{cases} pp_1^n(textname) \\ pp_2^n(1,2,..,1200\,pix) \\ \vdots \\ pp_a^n(true,\ false) \end{cases}}_{значення} \quad (4.60)$$

Для спрощення подання $MP_n \xrightarrow{\zeta_i} PP_n$ введемо поняття приналежності, з точки зору опису присвоєння значень, які задаються розробником залежно від вимог замовника. Введемо визначення «основних параметрів» mp_x^n, які заповнюються обов'язково за умови, що середовище розробки не зможе побудувати $Form_n$ і «не основних» $\overline{mp}_n^k$, в яких присутність значень pp_a^n можна не вказувати (вказується в разі потреби або формується середовищем розробки за замовчуванням). Приклад математичного запису представлений у (4.50) запишемо як:

$$MP_n \longrightarrow \begin{cases} mp_1^n \xrightarrow{\zeta_1 = textname} pp_1^n \\ mp_2^n \xrightarrow{\zeta_2 = 640\,pix} pp_2^n \\ \overline{mp}_3^n \xrightarrow{\zeta_3 = \text{bsSizeable}} pp_3^n \\ \dots \\ mp_x^n \xrightarrow{\zeta_n = true} pp_a^n \end{cases} PP_n \qquad (4.61)$$

Це дозволяє нам описати для $Form_n PE$ всі необхідні параметри реалізації графічного відображення $Form_n$, які можна задати наступним чином:

$$Form_1^{master}\left[Form_1 PE(MP_k^1) \xrightarrow{mp_1^k = textname, mp_2^k = 640, mp_3^k = beSizeable, \dots, mp_x^k = true} PP_1 \right] \qquad (4.62)$$

Ґрунтуючись на запропонованій формі запису (4.51), наведемо математичний опис основних параметрів, необхідних і достатніх для реалізації порожньої (без елементів $CD_x^n \in CF_n$ і подій ME_n) $Form_n$ для об'єктно-орієнтованого середовища розробки Rad Studio XE3.

$$Form_1PE[(MP_1) \xrightarrow{mp_3^1 = alNone;\ mp_4^1 = false;\ mp_5^1 = false;\ mp_6^1 = 255;\ mp_8^1 = false;} \wedge$$

$$\wedge \xrightarrow{mp_9^1 = false;\ mp_{10}^1 = bdLeftToRight; mp_{12}^1 = bsSizeable;} \wedge$$

$$\wedge \xrightarrow{mp_{13}^1 = 0;\ mp_{14}^1 = NameFormCaption; mp_{15}^1 = 212;\ mp_{16}^1 = 418;\ mp_{17}^1 = clBtnFace;} \wedge$$

$$\wedge \xrightarrow{mp_{19}^1 = true; mp_{20}^1 = crDefault; mp_{22}^1 = dmActiveForm; mp_{23}^1 = false;} \wedge$$

$$\wedge \xrightarrow{mp_{24}^1 = false; mp_{25}^1 = dkDra;\ mp_{26}^1 = dmManual; mp_{27}^1 = true; mp_{29}^1 = fsNormal;} \wedge$$

$$\wedge \xrightarrow{mp_{30}^1 = TGlassFrame; mp_{31}^1 = 250;\ mp_{32}^1 = 0; mp_{35}^1 = htContext; ...;\ mp_{44}^1 = Form_1;} \wedge$$

$$\wedge \xrightarrow{mp_{46}^1 = false; mp_{48}^1 = true; mp_{49}^1 = true;\ mp_{50}^1 = false; mp_{51}^1 = 96;\ mp_{53}^1 = pmNone;} \wedge$$

$$\wedge \xrightarrow{mp_{55}^1 = poDefaultPosOnly; mp_{56}^1 = po\,\mathrm{Pr}oportional; mp_{57}^1 = true;\ mp_{58}^1 = false;} \wedge$$

$$\wedge \xrightarrow{mp_{59}^1 = false;\ mp_{60}^1 = 10; mp_{62}^1 = 0;\ mp_{63}^1 = tipDontCare; mp_{64}^1 = 0; ...;\ mp_{66}^1 = false;} \wedge$$

$$\wedge \xrightarrow{mp_{67}^1 = clBlac; mp_{68}^1 = false;\ mp_{70}^1 = false;\ mp_{71}^1 = 434; mp_{73}^1 = wsNormal} (PP_1)]$$

Можна помітити, що параметри:

$$mp_4^1, mp_5^1, mp_8^1, mp_9^1, mp_{23}^1, mp_{24}^1, mp_{46}^1, mp_{50}^1, mp_{58}^1, mp_{59}^1, mp_{66}^1, mp_{68}^1, mp_{70}^1,$$
$$mp_{13}^1, mp_{32}^1, mp_{62}^1, mp_{64}^1, mp_{27}^1, mp_{48}^1, mp_{49}^1, mp_{57}^1$$

мають однакові значеннями, отже, в межах розроблюваного методу пропонується згрупувати параметри за однаковими значеннями. Це дає можливість спростить математичне подання опису до такого виду:

$$Form_1 \in [(MP_1 \in (mp_1^1, ...mp_x^1) \xrightarrow{\zeta} PP_1 \in (pp_1^1, ..., pp_a^1)) \in Form_1PE] \quad (4.63)$$

$$Form_1PE[(MP_1) \xrightarrow{mp_3^1 = alNone;\ mp_4^1, mp_5^1, mp_8^1, mp_9^1, mp_{23}^1, mp_{24}^1, mp_{46}^1, mp_{50}^1, mp_{58}^1,} \wedge$$

$$\wedge \xrightarrow{mp_{59}^1, mp_{66}^1, mp_{68}^1, mp_{70}^1 = false; mp_6^1 = 255; mp_{10}^1 = bdLeftToRight; mp_{12}^1 = bsSizeable;} \wedge$$

$$\wedge \xrightarrow{mp_{13}^1, mp_{32}^1, mp_{62}^1, mp_{64}^1 = 0;\ mp_{14}^1 = NameFormCaption; mp_{15}^1 = 212;\ mp_{16}^1 = 418;} \wedge$$

$$\wedge \xrightarrow{mp_{17}^1 = clBtnFace; mp_{19}^1, mp_{27}^1, mp_{48}^1, mp_{49}^1,\ mp_{57}^1 = true; mp_{20}^1 = crDefault;} \wedge$$

$$\wedge \xrightarrow{mp_{22}^1 = dmActiveForm; mp_{25}^1 = dkDra;\ mp_{26}^1 = dmManual; mp_{29}^1 = fsNormal;} \wedge$$

$$\wedge \xrightarrow{mp_{30}^1 = TGlassFrame; mp_{31}^1 = 250; mp_{35}^1 = htContext; ...;\ mp_{44}^1 = Form_1; mp_{51}^1 = 96;} \wedge$$

$$\wedge \xrightarrow{mp_{53}^1 = pmNone;\ mp_{55}^1 = poDefaultPosOnly;\ mp_{56}^1 = popopropoonal; mp_{60}^1 = 10;} \wedge$$

$$\wedge \xrightarrow{mp_{63}^1 = tipDontCare; ...;\ mp_{67}^1 = clBlac;\ mp_{71}^1 = 434; mp_{73}^1 = wsNormal} (PP_1)]$$

Розглянемо математичний опис подій $(me_1^n, \ldots, me_h^n) \in ME_n$ і «лінгвістичні змінні» $(ea_1^n, \ldots, ea_z^n) \in EA_n$, а також «контейнер рішень» $(z_z^o, z_2^o, \ldots, z_q^o) \in Z_o$, притаманні кожній $Form_i$ і приналежні множини $Form_1PE$.

$$ME_n \xrightarrow{\begin{cases} me_1^n \xrightarrow{\zeta_1 = ea_1^n \text{''лінгвістична змінна 1''} = \varphi_1} z_1^o \\ me_2^n \xrightarrow{\zeta_2 = ea_2^n \text{''лінгвістична змінна 2''} = \varphi_2} z_2^o \\ me_3^n \xrightarrow{\zeta_3 = ea_3^n \text{''лінгвістична змінна 3''} = \varphi_3} z_3^o \\ \ldots \\ me_h^n \xrightarrow{\zeta_n = ea_z^n \text{''лінгвістична змінна n''} = \varphi_n} z_q^o \end{cases}} Z_o$$

Врахуємо таку можливість як використання однієї «лінгвістичної змінної» ea_z^n для опису двох і більше подій $(me_1^n, \ldots, me_h^n) \in ME_n$. Для $Form_n$ кожної ea_z^n «лінгвістичній змінній» належить свій єдиний $z_q^o \in Z_o$ «контейнер рішень», який містить програмний код.

$$Form_1 PE[(ME_1) \xrightarrow{me_1^1, me_4^1 = "лінгвістична\ змінна\ 1", me_2^1 = "лінгвістична\ змінна\ 2"} (Z_o)] \quad (4.64)$$

Для опису параметрів PC_y^x і їх значень PP_t^x графічних елементів CD_x^n, які є невід'ємною частиною $Form_n$ пропонується використовувати тип запису представленого в (4.51), а опис взаємодії подій CE_z^x через множину «лінгвістичних імен» EA_n з «контейнером рішень» Z_o, відповідно до (4.63).

На базі розробленого математичного опису $Form_1$ представленого вище, параметрів і подій $Form_1 PE$ та елементів графічного управління даними CD_x^z у вигляді Button1 (CD_1^1) за подією ce_6^1, яка виконує процедуру закриття $Form_1$ через «лінгвістичну змінну 4», яка посилається на «контейнер рішень 1» ($z_1 \in Z_o$) і Button 2 CD_2^1, яка ініціалізує виклик $Form_2$ через подію ce_6^2, використовуючи «лінгвістичну змінну 5» наведену нижче:

$$
\begin{aligned}
&Form_1(Form_1PE[(MP_1)\xrightarrow{mp_4^1, mp_5^1, mp_8^1, mp_9^1, mp_{23}^1, mp_{24}^1, mp_{46}^1, mp_{50}^1, mp_{58}^1, mp_{59}^1, mp_{66}^1, mp_{68}^1, mp_{70}^1 = false;}\wedge\\
&\wedge\xrightarrow{mp_3^1 = alNone;\ mp_6^1 = 255; mp_{10}^1 = bdLeftToRight; mp_{12}^1 = bsSizeable; mp_{13}^1, mp_{32}^1, mp_{62}^1, mp_{64}^1 = 0;}\wedge\\
&\wedge\xrightarrow{mp_{14}^1 = NameFormCaption; mp_{15}^1 = 212; mp_{16}^1 = 418; mp_{17}^1 = clBtnFace; mp_{19}^1, mp_{27}^1, mp_{48}^1, mp_{49}^1,\ mp_{57}^1 = true;}\wedge\\
&\wedge\xrightarrow{mp_{20}^1 = crDefault; mp_{22}^1 = dmActiveForm; mp_{25}^1 = dkDra;\ mp_{26}^1 = dmManual; ontext;\ mp_{44}^1 = Form_1;}\wedge\\
&\wedge\xrightarrow{mp_{29}^1 = fsNormal; mp_{31}^1 = 250; mp_{30}^1 = TGlassFrame; mp_{35}^1 = htCmp_{35}^1 = htContext;\ mp_{44}^1 = Form_1;}\wedge\\
&\wedge\xrightarrow{mp_{51}^1 = 96; mp_{53}^1 = pmNone;\ mp_{55}^1 = poDefaultPosOnly; mp_{60}^1 = 10;\ mp_{63}^1 = tipDontCare; mp_{56}^1 = po\,\mathrm{Pr}\,oportional;}\wedge\\
&\wedge\xrightarrow{mp_{67}^1 = clBlac; mp_{71}^1 = 434; mp_{73}^1 = wsNormal}(PP_1)]\wedge[(ME_1)\xrightarrow{\mathrm{me}_1^1, \mathrm{me}_4^1 = "лінгвістична\ змінна\ 1";}\wedge\\
&\wedge\xrightarrow{me_2^1 == "лингвистическая\ переменная 2"; me_3^1 == "лінгвістична\ змінна\ 3"}(Z_0))])\wedge\\
&\wedge(CF_1[CD_1^1(PC_1^1)\xrightarrow{pc_2^1, pc_6^1, pc_{12}^1, pc_{14}^1, pc_{19}^1 = alNone; pc_3^1, pc_{43}^1, pc_{53}^1 = false; pc_7^1 = Close; pc_{10}^1 = crDefault;}\wedge\\
&\wedge\xrightarrow{pc_{13}^1, pc_{27}^1, pc_{41}^1, pc_{42}^1, pc_{46}^1 = -1; pc_{20}^1, pc_{35}^1, pc_{36}^1, pc_{37}^1, pc_{38}^1, pc_{39}^1, pc_{48}^1, pc_{51}^1 = true; pc_{15}^1 = crDrag; pc_{16}^1 = dkDrag;}\wedge\\
&\wedge\xrightarrow{pc_{17}^1 = dmManual; pc_{22}^1 = 25; pc_{23}^1, pc_{47}^1, pc_{49}^1 = 0; pc_{25}^1 = htContext; pc_{26}^1 = CloseForm1; pc_{28}^1 = iaLeft; pc_{27}^1 = 544;}\wedge\\
&\wedge\xrightarrow{pc_{33}^1 = mrNone; pc_{34}^1 = BoottonClose; pc_{44}^1 = bsPushButton; pc_{50}^1 = 328; pc_{52}^1 = 75}(PP_1^1)\wedge\\
&\wedge(CE_1^1)\xrightarrow{ce_6^1 = "лінгвістична\ змінна\ \ 4"}(Z_o)])\wedge[CD_2^1(PC_2^1)\xrightarrow{pc_2^2.pc_6^2, pc_{12}^2, pc_{14}^2, pc_{19}^2 = alNone; pc_3^2,}\wedge\\
&\wedge\xrightarrow{pc_{43}^2, pc_{53}^2 = false; pc_7^2 = Close; pc_{13}^2, pc_{41}^2, pc_{46}^2 = -1; pc_{10}^2 = crDefault; pc_{15}^2 = crDrag; pc_{16}^2 = dkDrag; pc_{49}^2 = 0;}\wedge\\
&\wedge\xrightarrow{pc_{34}^2 = BoottonOpen; pc_{44}^2 = bsPushButton; pc_{50}^2 = 328; pc_{52}^2 = 75}(PP_2^1)\wedge\\
&\wedge(CE_2^1)\xrightarrow{ce_6^2 = "лінгвістична\ змінна\ 5"}(Z_o)])
\end{aligned}
\tag{4.65}
$$

Фрагмент графічної моделі (таблиця 4.1, 4.2) взаємодії $Form_1^{master}$ і $Form_{2-9}^{slave}$ з використанням формального опису значень параметрів і подій представлений на рис. 4.8.

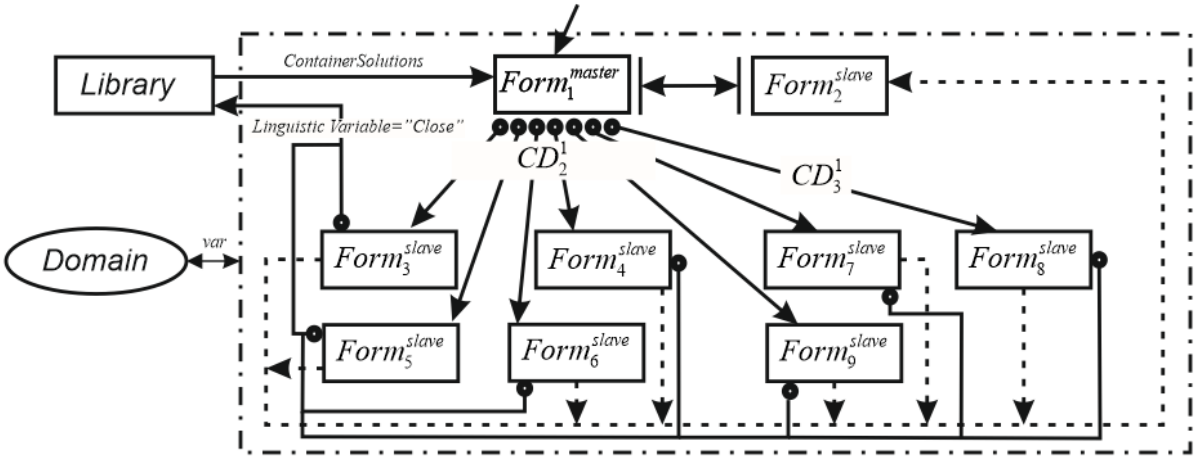


Рисунок 4.8 – Фрагмент графічної моделі взаємодії $Form_1^{master}$ і $Form_{2-9}^{slave}$.

РОЗДІЛ 5
РОЗРОБКА СИНТАКСИЧНОЇ І СЕМАНТИЧНОЇ МОДЕЛІ МОВИ ВИЗНАЧЕННЯ І ОПИСУ МОДЕЛЮВАННЯ КІБЕРФІЗИЧНИХ ВИРОБНИЧИХ СИСТЕМ

Запропоновано і формалізовано модель ЖЦ розробки CPPS «Jump» на базі синтезу візуальних компонентів і формальне подання властивостей і подій, що дозволяє автоматизувати процес управління розробкою CPPS на базі використання «контейнерів рішень» і «лінгвістичних змінних», представлених в 4 розділі.

Варто зауважити, що запропоновані рішення складні для розуміння користувачів і є математичним ядром, для роботи з яким необхідно розробити формальну мову опису структури і параметрів розроблюваного CPPS, який буде максимально близьким і інтуїтивно зрозумілим деякій підмножині природної мови і буде зрозумілий і простий в описі і застосуванні в предметній області (ПрО).

З одного боку було виявлено, що розробники CPPS використовують прив'язку до елементів інтерфейсу користувача і мінімізують використання програмного коду, який в більшості випадків слугує оброблювачем тієї чи іншої події, ініціалізованої користувачем або внутрішніми процесами, шляхом взаємодії з необхідними візуальними елементами для вирішення того чи іншого завдання. З іншого боку – користувачі, які є фахівцями ПрО, можуть сформулювати семантично правильну структуру і внутрішню ієрархію всіх елементів інтерфейсу з необхідними значеннями параметрів і подій, які необхідно реалізувати для вирішення поставлених завдань в ТЗ на CPPS.

Виникає необхідність розробити синтаксичну та семантичну моделі нової формальної мови, яка об'єднувала би в собі запропоновані в 2-4 розділі моделі і методи процесів управління розробкою, декомпозиції і формалізації CPPS в конструкцію побудови зрозумілого і простого як для фахівців, так і

для розробника CPPS, і близького деякій підмножині природної мови, що сприяє вирішенню завдань автоматичної трансформації семантичних правил опису процесу управління розробкою CPPS в синтаксично і термінологічно правильну структуру для реалізації її компіляції в необхідному середовищі розробки.

5.1 Розробка синтаксичної та семантичної мовної моделі.

Дамо визначення поняттю мовна модель (ММ) – це декларативна (непроцедурна) мова, призначенням якої є визначення та опис термінології, в основі яких покладено запропоновані поняття моделі ЖЦ процесу управління розробкою CPPS «Jump» і співвідношення між метаданими та даними предметної області і способів їх перетворення.

У даній роботі запропонована наступна специфікація мови моделей даних:

- **дозволені буквено-цифрові символи**, які підтримуються середовищами розробок для мов високого рівня програмування і відповідають таблиці ASCII-кодів: + - \ . , ! “ < > = ( ) $ % & ~ * _ & @ пробіл; { };

- **ключові слова**: базове поняття у вигляді слів зарезервованих в розроблюваній математичній моделі (ММ) і служать для опису ключових ознак представлених в таблиці 5.1

Таблиця 5.1 – Ключові слова.

Form	*ValueElement*
$Form^{master}$	*LingusticVariable*
$Form^{slave}$	*ContainerSolution*
ParameterForm	*parameter*
ElementForm	*value*
EvenForm	*name*
ParametrElement	*event*
EventElement	*cod*

- **ідентифікатори**, використовувані для позначення таких ознак:

- ознака приналежності параметрів та подій до доменного або недоменного типів: *domen*, *not_domen*. Домени відповідних характеристик (значень), що належать до перелічуваного (облікового) типу, який має можливість вибору із заздалегідь сформованого списку. Прикладом можуть виступати деякі параметри *ParameterForm*, відображення *Form*, які можуть набувати значень *true* чи *false*, параметр *Align*, що притаманні $dom(parameter^{z}_{p\_список})$ та $dom(parameter^{h}_{p\_список})$ з (4.27), який може набувати значень, фіксованих середовищем розробки CPPS і зазначених в ТЗ;

- тип даних значень (*value*), який визначає характеристику параметрів *ParameterForm* та *ParametrElement* (текстове, булеве, цілочисельне, цілочисельне негативне, текстове, словосполучення), опис яких представлено в таблиці 4.3;

- лінгвістичний опис ознаки посилання *LingusticVariable* (*name*) на *ContainerSolution*, який містить необхідний *cod*;

- базові поняття моделі ЖЦ «Jump», які дають можливість зв'язати події *ElementForm* і *EventElement*, містять набір певних *event*, що належать певному візуальному графічного елементу з *ContainerSolution* (*cod*) через *LingusticVariable*(*name*).

Як можна побачити, на відміну від ключових слів, запропоновані ідентифікатори теоретично можна перевизначити, але це призводить до виникнення помилок, внаслідок чого перераховані вище ідентифікатори входять в фіксований словник ключових слів.

- **літерали**, певний набір значень, які не представлені ідентифікатором.

Рядкові літерали представляються у вигляді послідовності дозволених символів з різним типом написання (великі та малі) літер. Наприклад, *name_form,* яка використовується в параметрах *Caption*, *Name* і т.д., а також привласнення унікального імені (*name*) для кожного *LingusticVariable*, яке містить певний фрагмент програмного коду. Приклад «зберегти в БД», «розрахувати результат», і т.д., які задаються кінцевим користувачем з метою зручності методології, що розробляється.

Алгебраїчні літери – літери, що являють собою опис простих логічних операцій типу *True, False*, які дозволяють задати значення (*value*) того чи іншого параметра (*parameter*), який належить *ParametrElement*, *ParameterForm* та є необхідним і достатнім для опису властивостей візуальних елементів CPPS або функції c-MES, відповідно до вимог ТЗ.

Зарезервовані літери являють слово, словосполучення або скорочення, яке дає можливість вибрати ту чи іншу властивість параметра, необхідне для досягнення умов заданих в алгоритмі функціонування. Прикладом може виступати властивість форми *WindowsState*, в середовищі розробки RadStudioXE16, якому можна обрати наступні зарезервовані слова скорочень, як *wsNormal,wsMinimized, wsMaximized*, тобто, при першому запуску розробник може задати вид відображення форми. *wsNormal* – відображення за замовчуванням, в тому вигляді, в якому вона створювалася на етапі проєктування, *wsMinimized* – форма, що відображається в мінімізованому вигляді на панелі завдань, *wsMaximized* – при запуску форма розгортається на весь розмір робочого столу.

Зарезервовані літери можуть бути загальними для *ParameterForm* і *ParametrElement*, а також спеціалізованими, тобто належати певній

візуальній формі, яка визначає специфіку того чи іншого елемента. Але, слід зауважити, що зарезервовані літери для визначення значень того чи іншого параметра візуальних компонентів, які мають однакове призначення, можуть виконувати різні задані функції і обробляти події в одному середовищі розробки.

Типи представлених значень, в яких містяться деякі параметри *ParameterForm* і *ParametrElement*, припустимі в області застосування (таблиця 4.3).

Цілочисельний тип даних (integer) дозволяє привласнити параметру *ParameterForm* і/або *ParametrElement* певне і необхідне цифрове значення розмірності або координат розміщення візуального елемента щодо *Form*. Використовується в основному для опису візуальних графічних елементів. Найменший логічний елемент двовимірного цифрового зображення в растровій графіці (pixel). Довжина рядка залежить від роздільної здатності екрану і вимог ТЗ, висунутого замовника розробнику. Формальне представлення наведено в (4.55).

Цілочисельний негативний дозволяє привласнити параметру певне значення, що входить в діапазон $(-1,1,2,3,...,n)$, яке належить виключно *ParametrElement* і описує нумерацію в даному контексті:

-1 – нумерація відсутня, параметр не задіяний;

$1,2,3,...,n$ – нумерація графічного зображення (іконки), яка належить певному параметру (*parameter*) для *ElementForm*. Формальне представлення наведено в (4.58).

Текстовий / лінгвістичний (char) дозволяє привласнити параметру логічне впорядкування значення символів, які містять необхідні для користувача пояснення або назву графічних елементів, які потрібні для зручності роботи з CPPS. Також даний тип подання значення використовується для завдання певного імені *LinguisticVariable*, що присвоюється події *EventForm*, *EventElement*. Формальне представлення надано в (4.59).

Логічний (булевий) може приймати тільки два значення: *true* (правда) або *false* (брехня) і виконує роль перемикача використання того чи іншого параметра в *ParameterForm* і *ParameterElement*. Формальне представлення надано в (4.56).

Текстове словосполучення (перелічуваний тип) – тип даних, заданий списком у вигляді домену (4.27)–(4.30), дозволяє задати список зарезервованих слів в середовищі розробки або скорочень, які можуть приймати той чи інший *parameter* для *ParameterForm* і *ParametrElement*. Формальне подання надано в (4.57).

Розділювачі – символьні позначення виділення основних елементів синтаксичної конструкції розроблюваної ММ.

<*Form*> (кутові дужки *Form*) – використовуються для вказівки ключового слова, яке показує початок метаопису тієї чи іншої *Form* в конструкції ММ.

</*Form*> (слеш кутові дужки *Form*) – використовується для вказання ключового слова, яке показує завершення метаопису тієї чи іншої *Form* в конструкції ММ.

Для запропонованої конструкції ключового слова, на початку і завершенні метаопису *Form* накладені такі обмеження: назва *Form* може мати нумерацію як *Form*1, або буквене визначення, наприклад, *Form_master* або *Form_add_operat*. При цьому обов'язково ключове слово початку метаопису має збігатися з ключовим словом завершення метаопису тієї чи іншої *Form* в конструкції ММ. При невиконанні цієї вимоги до конструкції, інтерпретатор ММ не зможе сприйняти вміст як метаопис всіх необхідних параметрів і подій, властивих даній *Form*.

{ (відкриваюча фігурна дужка) – обов'язковий символ початку рядка метаопису *Form* і *ElementForm*.

} (закриваюча фігурна дужка) – обов'язковий символ завершення рядка метаопису *Form* і *ElementForm*.

(решітка) – після цього символу конструкція інтерпретатора ММ сприймає початок опису графічних візуальних елементів інтерфейсу користувача (*ElementForm*).

/# (слеш решітка) – після даної комбінації символів інтерпретатор ММ вважає, що опис графічних візуальних елементів інтерфейсу користувача (*ElementForm*) завершено.

/ (слеш) – використовується для завдання ієрархії метаопису візуальних графічних елементів (*ElementForm*), відповідно до дерева побудови CPPS, і застосовується всередині # /# метаопису *Form* . *ElementFor m*1/*ElementFor m*2 – необхідно розуміти як *ElementForm*2, що знаходиться всередині *ElementForm*1 і є її невід'ємною частиною.

[ ] квадратні дужки – використовується для завдання метаопису необхідних параметрів і подій *ParameterForm* , *EventForm* , *ParameterElement* , *EventElement* .

; (крапка з комою) – обов'язковий символ конструкції ММ, який показує, що для даного *parameter* або *event* присвоєння *value* та *name* відповідно, завершено, застосовується всередині .

, , (перерахування через кому) – використовується для перерахування назв *parameter* для *ParameterForm* , *ParameterElement* , а також *e v e n* для *EventForm* , *EventElement* за умови, що для набору з декількох *parameter* або *event* значення *value* та *name* відповідно, однакове і застосовується всередині .

= (знак рівності) – присвоює *parameter* певне значення типу даних *value* і застосовується для вказівки події (*event*) певного *name* з *LinguistiVariable*, яке містить посилання на *cod* або його фрагмент в *ContainerSolution*. Варто врахувати, що в залежності від контексту (логіки й змісту виконуваних дій), даний знак можна трактувати і як інструкцію присвоєння, згідно з якою для зазначеного базового параметра визначається значення, яке йому належить.

Коментарі – всі символи і рядки, записані всередині даної конструкції інтерпретатором ММ, ігнорується і сприймаються як коментарі. Допустимі до використання буквено-цифрові символи національних алфавітів, підтримуються операційною системою і середовищем розробки. Обмеженням для коментарів служить те, що послідовність не повинна перевищувати 255 символів.

?** (знак питання з двома зірочками) – показує, що після заданих символів слідує коментар, який ігнорується інтерпретатором ММ.

**? (дві зірочки і знак питання) – показує, що після заданих символів закінчується коментар і далі йде текст, що не ігнорується інтерпретатором ММ.

Для адаптації розроблюваного синтаксису опису ММ, запропоновано використовувати форму Бекуса-Наура. Обґрунтуванням цього вибору служить те, що розширена форма Бекуса-Наура використовується для опису контекстно-вільних граматик [217] і дозволяє спростити і скоротити обсяг опису. Розширена форма Бекуса-Наура описана в міжнародному стандарті ISO/IEC -14977 [218]. Аналіз ISO / IEC -14977 показав, що розширена форма Бекуса-Наура дає можливість розробити інтуїтивно просту і адаптивну формальну мову подання та опису необхідних даних для розробки CPPS на базі підходів до об'єктно-орієнтованого програмування.

Ґрунтуючись на запропоновані вище специфікації мови моделей даних і основній концепції моделі ЖЦ проєктування CPPS «Jump», пропонується наступна синтаксична діаграма ММ CPPS, яка представлена на рисунку 5.1.

Синтаксична діаграма запропонованих в даному дослідженні типів представлення значень, які можуть належати ідентифікаторам представлена на рисунку 5.2 [219].

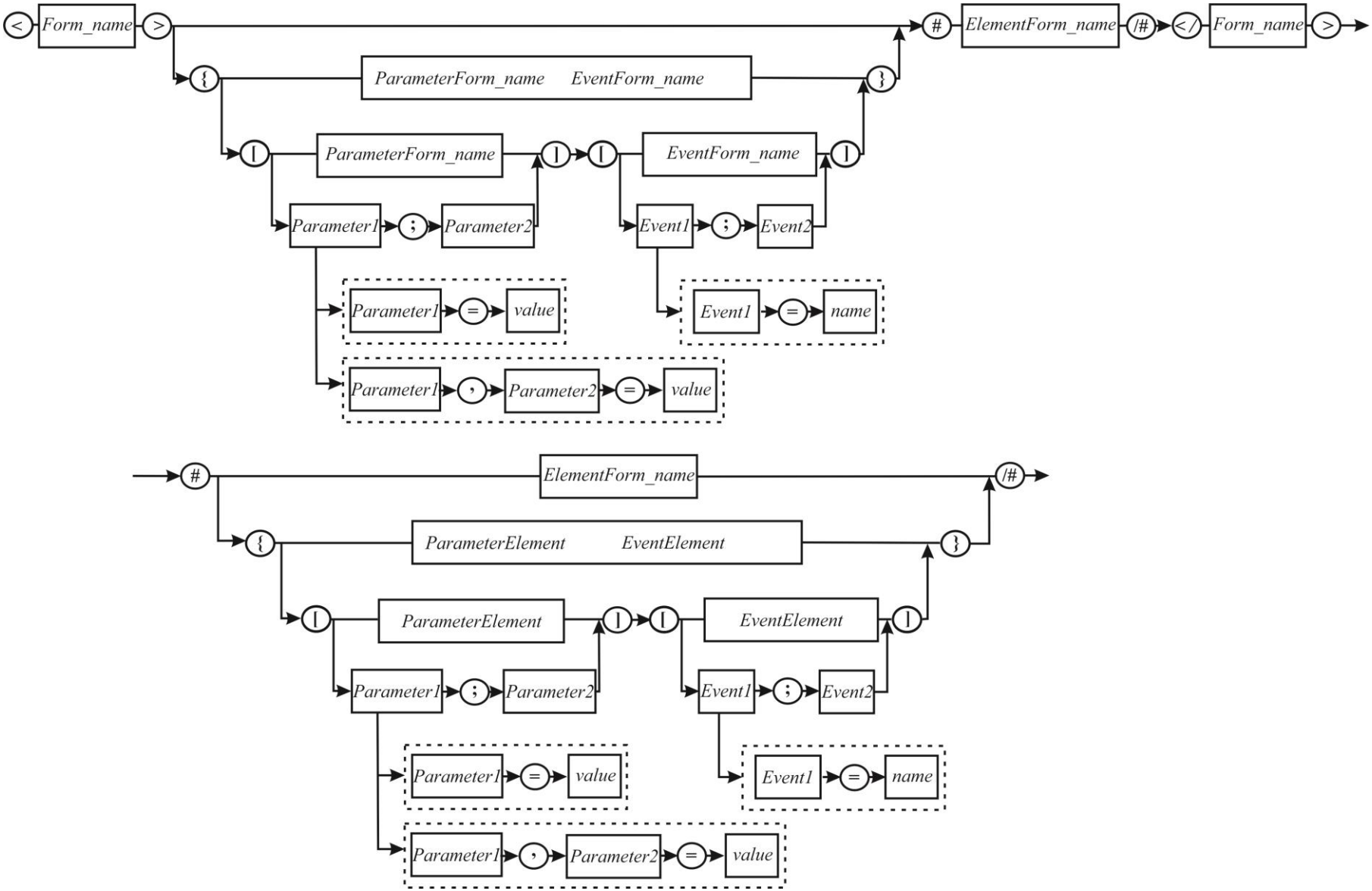


Рисунок 5.1 – Синтаксична діаграма ММ, що розроблюється

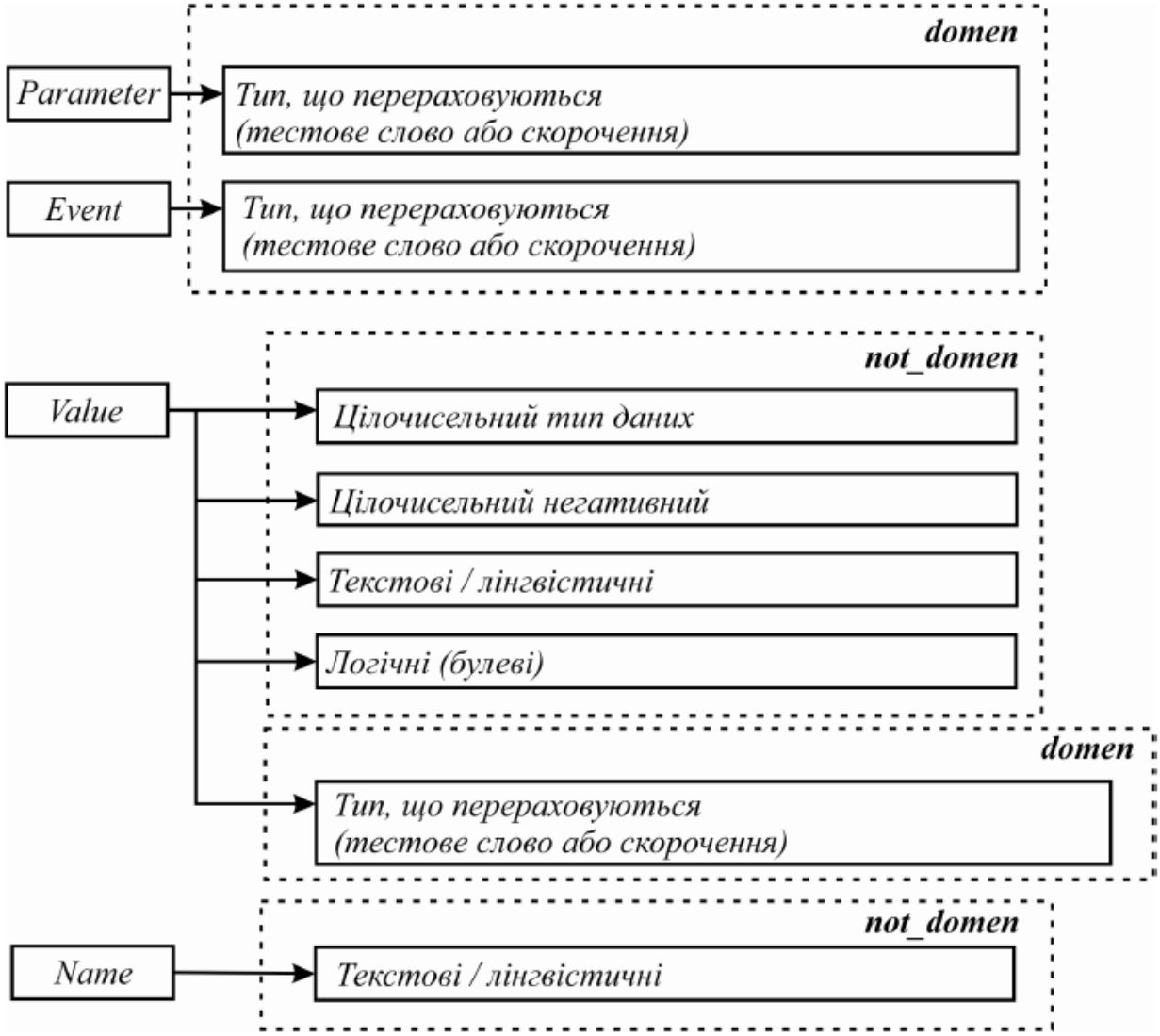


Рисунок 5.2 – Синтаксична діаграма типів представлення значень ідентифікаторів

Як можна бачити з рисунка 5.2 ідентифікатори *Parameter*, *Event* відноситься до *domen*-ого (спискового) типу і представляється у вигляді текстового слова або скорочення, які відносяться до *ParameterForm_name* та *EventForm_name*, а також *ParameterElement* и *EventElement*, відповідно до рисунка 5.1. Список параметрів (*parameter*) і подій (e*vent*), які належать *ParameterForm_name*, *EventForm_name*, в рамках одного середовища розробки, є постійним і незмінним. Для списку параметрів (*parameter*) і подій (e*vent*), які належать *ParameterElement*, *EventElement* відповідно,

однаковим є обмеження, що ці візуальні графічні елементи мають однакові призначення в рамках одного середовища розробки. Варто звернути увагу, що до даного типу відноситься *value* для *ParameterElement* і *ParameterForm_name*, які містять зарезервоване середовищем розробки текстове слово або скорочення.

Ідентифікатори *value* і *name* відносяться до *not_domen*-ого (не облікового) типу. Це обґрунтовано тим, що значення *value* можуть задаватися розробником залежно від вимог, що висуваються ТЗ на CPPS. Для ідентифікатора *name*, який входить *LinguisticVariable*, ім'я, яке посилається на *ContainerSolution*, що містить необхідний фрагмент або частину програмного коду (*cod*), задається користувачем, з урахуванням його логічних переваг і зручністю застосування.

Для зручності читання та подання розробленої декларативної мови (рисунок 5.1 і 5.2) необхідно щоб він мав якості розуміння і читання. Цього можна досягти, використовуючи мінімум три принципи подання мови [220], а саме щоб вона була:

- максимально лінійною;
- короткою;
- само-документованою.

Ґрунтуючись на запропонованих допущеннях і рекомендаціях для розроблюваної декларативної мови CPPS, пропонується наступний тип стилю запису мови моделі, який дає можливість спростити та стандартизувати код.

Приклад 5.1

< Form _ master >

{ ?** відкриття блоку опису параметрів і значень, а також подій та імен *LinguisticVariable* для *Form _ master* **?

[*parameter*1 = *value*; *parameter*2, *parameter*3 = *value*]

[$event1 = name; event2 = name$]

} ?** закриття блоку опису параметрів і значень, а також подій та імен $LinguisticVariable$ для $Form\_master$ **?

“ ім'я елемента в середовищі розробки ”

?** відкриття блоку опису візуальних графічних елементів $Form\_master$ **?

{ ?** блок опису $Element1\_Form\_master$ **?

[$parameter1 = value; parameter2, parameter3 = value$]

[$event1 = name; event2 = name$]

} ?** закриття блоку опису $Element1\_Form\_master$ **?

{ ?** блок опису $Element2\_Form\_master$ **?

[$parameter1 = value; parameter2, parameter3 = value$]

[$event1 = name; event2 = name$]

} ?** закриття блоку опису $Element2\_Form\_master$ **?

/# ?** закриття блоку опису візуальних графічних елементів $Form\_master$ **?

$< / Form\_master >$

При необхідності реалізації ієрархії (дерева побудови) приналежності візуальних графічних елементів $ElementForm1/ElementForm2$ пропонується наступний фрагмент структури мета опису:

Приклад 5.2.

“ ім'я елемента в середовищі розробки ” ?** відкриття блоку опису візуальних графічних елементів $Form\_master$ **?

{ ?** блок опису $Element1Form\_master$ **?

[$parameter1 = value; parameter2, parameter3 = value$]

[$event1 = name; event2 = name$]

} ?** закриття блоку опису $Element1Form\_master$ **?

/ “ ім'я елемента в середовищі розробки ”

{ ?** блок опису $Element2Form\_master$ **?

[$parameter1 = value; parameter2, parameter3 = value$]

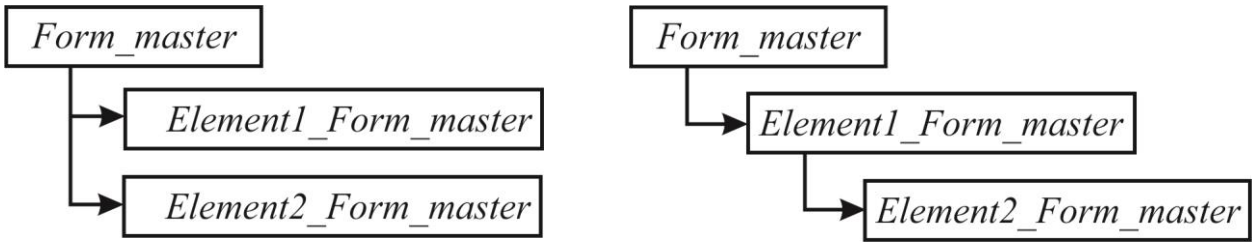
[$event1 = name; event2 = name$]

} ?** закриття блоку опису $Element\ 2Form\_master$ **?

/# ?** закриття блоку опису візуальних графічних елементів $Form\_master$ **?

Використання “/” (слеш) дозволить інтерпретатору ММ визначити ступінь вкладання (приналежності) візуального елемента в інший, тобто в середовищі розробки реалізувати дерево структури (Structure) CPPS. На рис. 5.3 наведено графічну побудову структури дерева побудови $Form\_master$ CPPS для прикладу 1 (а) і прикладу 2 (б).

Для вказівки відповідних значень ($value$) і імен ($name$), в наведених вище прикладах, в рамках $LinguisticVariable$ для $parameter$ та $event$, відповідно після знака рівності (=), задається тип значення, при відсутності значень або використанні значень за замовчуванням, зарезервованих середовищем розробки, даний параметр в метаописі не декларується (не позначається).



а) б)

а) - $Element1Form\_master$ и $Element2Form\_master$ рівносильно належить $Form\_master$;

б) - $Element2Form\_master$ належить $Element1Form\_master$;

Рисунок 5.3 – Графічне подання дерева побудови структури CPPS

На прикладі 5.3 показаний рядок метаопису створення порожньої форми в середовищі Rad Studio XE6 для VLC Form Application.

Приклад 5.3

$$
\begin{aligned}
&< Form1 > \\
&\quad \{ \\
&\quad\quad [Caption = example\ 1, ClientHeight = 464, ClientWidth = 687, \\
&\quad\quad Height = 503, Name = Form\_master, Width = 703] \\
&\quad \} \\
&< / Form1 >
\end{aligned}
\tag{5.1}
$$

На базі наведеного в (5.1) метаопису було згенеровано графічне представлення найпростішої користувальницької форми, яку представлена на рисунку 5.4.

Фрагмент метаопису додаткових візуальних графічних елементів виду *Standard- Button* (користувальницької кнопки, яка виконує певну подію) представлено в (5.2).

$$
\begin{aligned}
&\#\ \text{“Button\_close”} \\
&\quad\quad [Caption = Close, Height = 33, Top = 408, Left = 560, \\
&\quad\quad Name = Button\_close, Width = 91] \\
&/\#
\end{aligned}
\tag{5.2}
$$

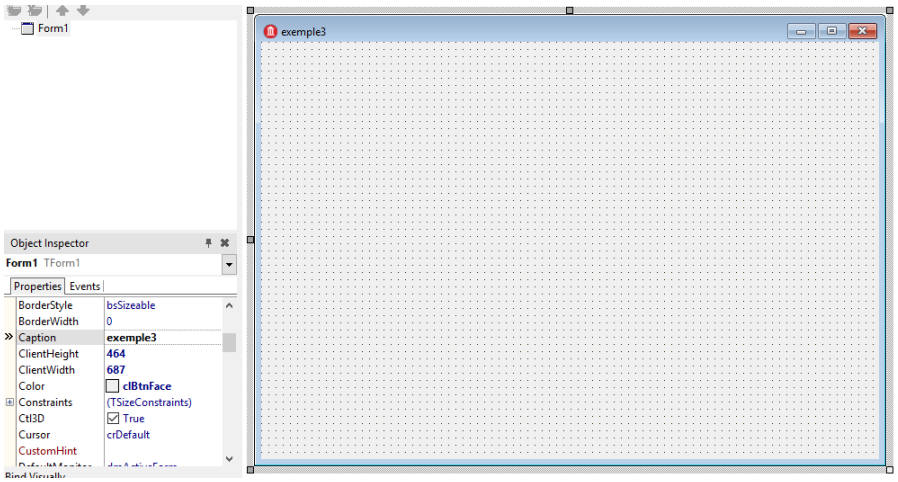


Рисунок 5.4 – Фрагмент середовища розробки Rad Studio XE6 найпростішої користувальницької форми

На рисунку 5.5 наведено приклад реалізації форми з елементом Button, метаопис якого наведено в (5.1) і (5.2) відповідно.

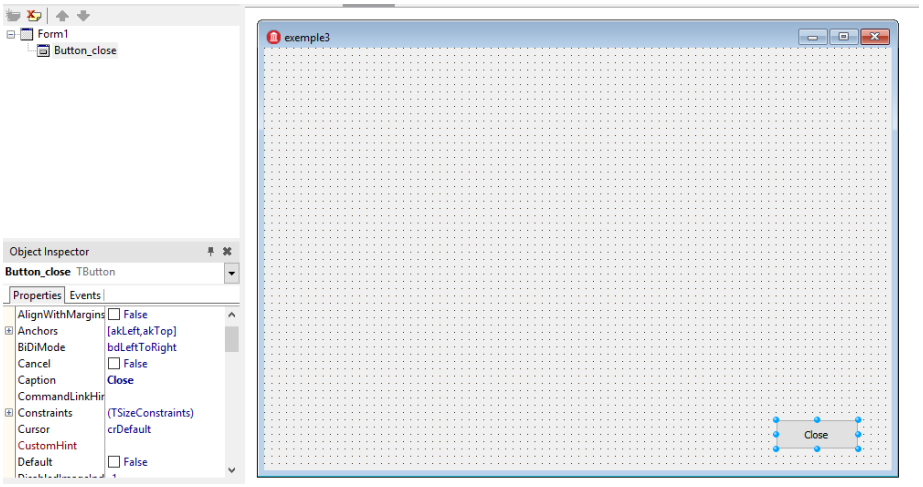


Рисунок 5.5 – Фрагмент середовища розробки з реалізацією форми і графічним елементом Button

Крім реалізації графічного візуального інтерфейсу, наведеного на рис. 5.4 - 5.5, на базі метаописів (5.1) – (5.2) був згенерований програмний код на мові Pascal, представлений на рис. 5.6.

```
unit Unit1;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs;

type
  TForm1 = class(TForm)
    Button_close: TButton;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

end.
```

Рисунок 5.6 – Програмний код на мові Pascal

Кожен елемент опису ММ, наведений в синтаксичній моделі (5.1)–(5.2), записується відповідно до синтаксичної діаграми, показаної на рис. 5.1, і діаграми типів подання значень ідентифікаторів, яка представлена на рисунку 5.2. Семантична модель ММ являє собою систему значень, приписуваних конструкціям і розробленій синтаксичній моделі мови (інтерпретації конструкції). Ця модель представляється в процесі тлумачення (розбору) запропонованих правил опису та подання специфікації ММ, символів і їх комбінацій.

Розглянемо метаопис прикладу (5.2), для визначення необхідності реалізувати вкладень (приналежності) одного візуального елемента в інший, як показано на рисунку 5.3,б. У відповідності до запропонованої синтаксичної моделі (рисунок 5.1), метаопис прийме наступний вигляд:

$$
\begin{aligned}
&\#\ \text{“}GrupBox1\text{”} \\
&\quad [Caption = GroupBox1, Height = 186, Top = 272, \\
&\quad Left = 4, Name = GroupBox1, Width = 678] \\
&/\ \text{“}Buttion1\text{”} \\
&\quad [Caption = Close, Height = 32, Top = 146, Left = 585, \\
&\quad Name = Buttion1, Width = 86]
\end{aligned}
\qquad (5.3)
$$

/#

У середовищі розробки даний метаопис дозволяє реалізувати ступінь вкладеності візуальних графічних елементів один в одного і побудувати «дерево» *Form*1, на базі якого розробляється інтерфейс користувача у відповідності з ТЗ на CPPS і алгоритмом функціонування. На рисунку 5.7 наведено фрагмент середовища розробки Rad Studio XE6 з генерованим інтерфейсом користувача відповідно до метаопису (5.3).

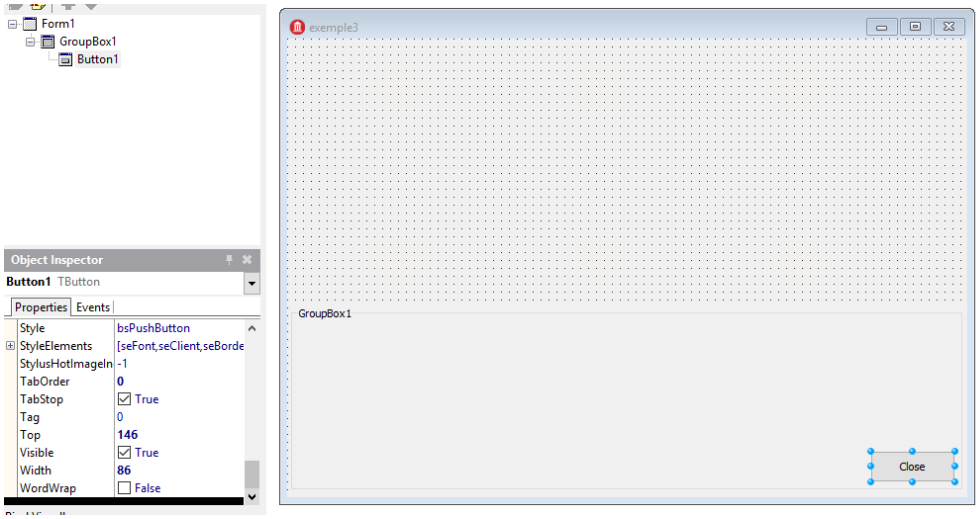


Рисунок 5.7 – Реалізація інтерфейсу *Form*1

Виходячи з наведених вище фрагментів метаописів (5.1)–(5.3) і рис. 5.3, можна задати будь-яку глибину вкладення графічних елементів інтерфейсу користувача. Це дає можливість реалізувати, за допомогою запропонованої синтаксичної діаграми ММ (рисунок 5.1), структуру CPPS будь-якого ступеня складності, і спростити процес розробки візуальної складової на базі об'єктно-орієнтованого підходу до програмування.

5.2 Розробка синтаксису метаопису подій

Ґрунтуючись на запропонованій синтаксичній діаграмі, представленій на рисунку 5.1, пропонується наступний метаопис подій (*event*) для *Form* та

ElementForm. На базі прикладу (5.1) наведемо приклад простого метаопису спрацьовування на елементі *Button* події *Onclick*, на відпрацювання *LinguisticVariable* з іменем *Close_All*.

Приклад 5.4

$$
\begin{aligned}
&<Form1>\\
&\quad\{\\
&\quad\quad[Caption = example\ 1, ClientHeight = 464, ClientWidth = 687,\\
&\quad\quad Height = 503, Name = Form\_master, Width = 703]\\
&\quad\quad\quad[Caption = Close, Height = 33, Top = 408, Left = 560,\\
&\quad\quad\quad Name = Button\_close, Width = 91, OnClick = Close\_All]\\
&\quad\}\\
&</Form1>
\end{aligned}
\tag{5.4}
$$

Як можна бачити з метаопису на прикладі (5.4), розробник описує існування події на елементі *Button* з ім'ям $Name = Button\_close$, *LinguisticVariable* під ім'ям $Close\_All$ на подію *Onclick*. Дане представлення дозволяє розробнику реалізувати модель ЖЦ процесу управління розробкою CPPS, яке представлене в 4 розділі, у вигляді послідовності зв'язків як:

$$event \rightarrow LinguisticVariable \rightarrow ContainerSolution \rightarrow cod \tag{5.5}$$

Внаслідок виконання метаопису (5.4) розробник отримує згенерований програмний код представлений на рисунку 5.8

```
procedure TMaster.Button_closeClick(Sender: TObject);
begin
Close;
end;
```

Рисунок 5.8 – Результат генерації програмного коду

Розглянемо фрагмент – приклад метаопису на прикладі (5.6) реалізації більш складної конструкції коду, який був згенерований в процесі проєктування «Автоматизована система нормування «НОРМА» (свідоцтво про реєстрацію авторського права на твір №57667 від 17.12.2014р.)

На $Form1$ існує елемент $DBGrid\_operac$ для відображення інформації з БД. В $Properties\_DBGrid\_operace$ вказана прив'язка до візуального випадаючого елементу інтерфейсу $PopupMenu\_operac$. Необхідно згенерувати програмний код видалення з БД обраного запису в $DBGrid\_operac$.

Приклад 5.5

$< Form1 >$

$\vdots$

$\#$ *"DBGrid_operace"*

{

$[DataSurce = Form1.IBDataSet\_Nak\_Operac,$

$Height = 120, Left = 344, Top = 24, Width = 320,$

$Name = DBGrid\_operace,$

$PopupMenu = PopupMenu\_operac]$

}

$\vdots$

$\#$ *"DBGrid_operace"*

{

$[Name = PopupMenu\_operac, Items = 28,]$

$[OnClick = N28Click]$

}

$\vdots$

"N28"

{

[*Caption* = *Delete*, *Name* = 28]

[*OnClick* = *Delete _ select _ operace*]

}

⋮

< / *Form*1 >

Приклад з згенерованого програмного коду реалізації події на елемент *PopupMenu _ operac*, при одноразовому натисканні, в випадаючому меню опції *Delete* з внутрішньою індексацією 28, спрацьовує вибір коду з *LingusticVariable* = *Delete _ select _ operace*. Приклад згенерованого програмного коду після редагування розробником представлений на рис. 5.9.

```
procedure TForm1.N28Click(Sender: TObject);
begin
if messageDlg('Удалить операцию из базы данных?',mtConfirmation,[mbYes,mbNo],0)=mrYes
then Form1.IBDataSet_NAK_Operac.Delete;
//Form1.IBDataSet_NAK_Operac.Post;
Form1.IBDataSet_NAK_Operac.Close;
Form1.IBDataSet_NAK_Operac.Open;
DataModule_redaktor.IBStoredProc_update_detal_norma.ParamByName('detal_id').AsInteger := IBDataSet_NAK_DETAL.fieldByName('id_detal').AsInteger;
DataModule_redaktor.IBStoredProc_update_detal_norma.ExecProc;
IBDataSet_NAK_DETAL.Refresh;
IBDataSet_NAK_Operac.Refresh;
DBGridEh1.Refresh;
end;
```

Рисунок 5.9 – Фрагмент коду після редагування розробником

Ґрунтуючись на прикладах наведених вище, розробник отримує можливість на базі запропонованої моделі та методів розробки CPPS запропонованих в 2-3 розділі, а також на базі розробленої моделі ЖЦ процесу управління розробкою CPPS в 4 розділі за допомогою метаопису створити і реалізувати кібернетичну складову CPPS з можливістю «часткової» генерації програмного коду. Повнота генерації програмного коду безпосередньо залежить від змісту DB *"Container Solutions"*, в якому знаходяться приклади реалізації подій (*cod*), які можуть виконуватися в процесі роботи з розробки CPPS. Дане рішення дозволяє адаптувати запропонований метаопис до будь-якої об'єктно-орієнтованої мови, а також дає можливість розробнику

розширювати DB новими *“Container Solutions”*, що в майбутньому дозволить скоротити витрати часу на етапі програмування.

РОЗДІЛ 6
РОЗРОБКА СИСТЕМИ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ УПРАВЛІННЯ РОЗРОБКОЮ КІБЕРНЕТИЧНОЇ СКЛАДОВОЇ CPPS ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

6.1. Структура системи автоматизації процесів управління розробкою кібернетичної складової CPPS

Ґрунтуючись на запропонованих моделях і методах процесу управління розробкою складних CPPS, наведених в 2-5 розділах, для підтвердження адекватності і правильності прийнятих наукових рішень і проведення експериментальних досліджень необхідно реалізувати їх в автоматизовану систему управління процесами розробки CPPS.

Виходячи з вищесказаного, для автоматизованої системи управління процесами розробки CPPS, що розроблюється, запропонована наступна структура, яка представлена на рисунку 6.1.

Дана структура складається з наступних блоків:

- текстовий редактор мови моделювання (ММ) / модуль аналізу вхідних даних з MS Excel (* .xls). Редактор призначений для введення даних в систему за допомогою розробленої ММ і дає можливість забезпечити зчитування вихідних даних альтернативним способом через MS Excel (* .xls), при умові необхідності розробки тільки кібернетичної складової CPPS. Формат вхідних даних має жорстку деревоподібну ієрархію, яка дає можливість відразу визначити структуру об'єктів візуалізації і їх взаємозв'язок, підпорядкованість в HMI на базі GUI;

- модуль аналізу синтаксису вхідних даних, який проводить: перевірку на адекватність і правильність задання вихідних даних; їх повноту, яка необхідна для реалізації. При виникненні критичних помилок

невідповідності логіки або браку даних, формується файл помилки, в якому відбувається запис про помилки або попередження.

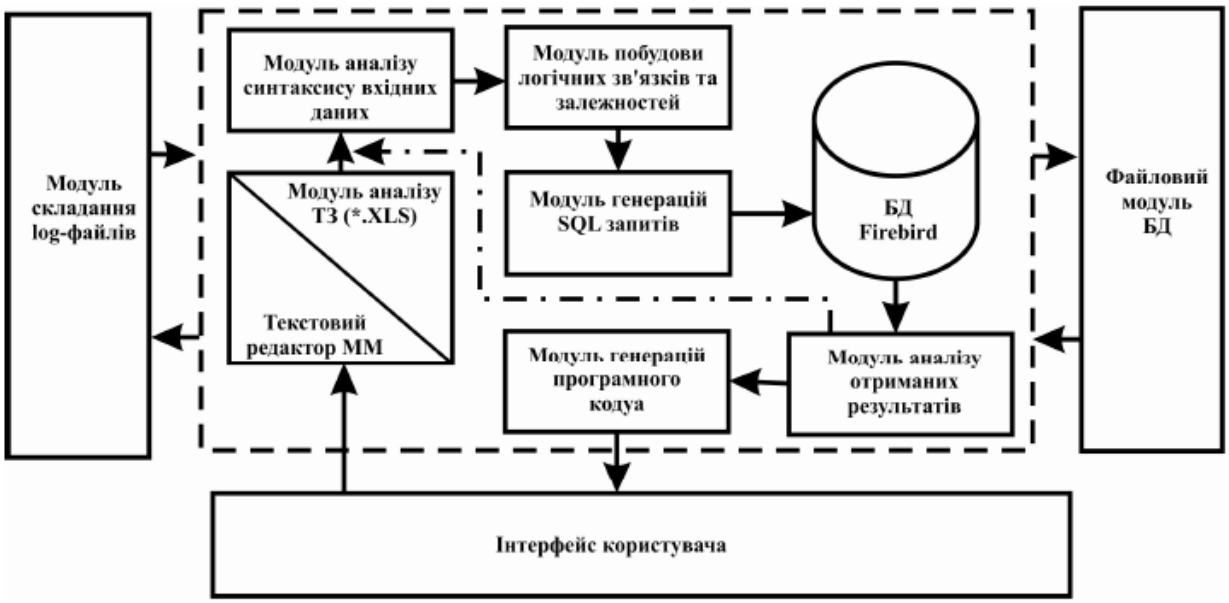


Рисунок 6.1. – Структура автоматизованої системи управління процесами розробки CPPS

- модуль побудови логічних зв'язків і залежності між візуальними компонентами інтерфейсу і подіями, які накладаються розробником на дії для даних подій. Перевірка проводитиметься з використанням контролю взаємозв'язків виконання відповідно алгоритмів функціонування та вимог ТЗ, обмеженням функціональних можливостей кожного компонента, які залежать від повноти бази даних (БД) і можуть бути доповнені;

- модуль генерації запитів до БД – генерує запити на мові SQL відповідно до використаних компонентів і подій (дій), які вони повинні виконувати. В ході виконання запиту формується список шаблонів програмного коду, що підходять під умови вибірки і логіки роботи модуля;

- модуль аналізу отриманих результатів з БД – проводить синтаксичний аналіз отриманих результатів, визначає максимально відповідний фрагмент програмного коду, який підходить під умови вибірки в залежності від заданих умов пошуку;

- модуль генерації вихідного файлу програмного коду – підключає необхідні бібліотеки, дотримує синтаксис структури, правил оформлення і представляє його у вигляді пакету файлів, які необхідні для відкриття розроблюваного програмного продукту в середовищі розробки, заданого користувачем;

- модуль складання log-файлів, які містять звіти по роботі кожного модуля на етапах проєктування, а саме містить в собі такі дані:

а) результати перевірки синтаксису і повноти опису розроблюваного CPPS, а також аналіз результатів запровадження даних альтернативним способом через MS Excel (* .xls). У звіті міститься номер рядка і номер комірки, в якій необхідно провести коригування даних або задати їх. Рядок може приймати два значення. Значення *error* – критична помилка, в ході якої система не може використовувати дані і *warning* – попередження користувача про можливість некоректного представлення даних, що може вплинути на результат генерації вихідного коду розроблюваного CPPS;

б) процентне співвідношення ймовірності генерації програмного коду, відповідно до тих чи інших завдань та в залежності від повноти БД за кожною подією (дією) над об'єктом графічного інтерфейсу.

- БД (Firebird) – вільна кросплатформенна реляційна система управління базами даних, в якій зберігається вся необхідна інформація і посилання на файли, які містять фрагменти програмного коду *("Container Solutions"*);

- файловий модуль БД, призначений для зберігання файлів *"Container Solutions"*.

6.2. Обґрунтування вибору середовища розробки та мови високого рівня

Для розробки автоматизованої системи управління процесами розробки CPPS був проведений аналіз мов високого рівня програмування і середовищ розробки. Основними критеріями для вибору середовища розробки послужили наступні параметри:

- підтримка роботи з розподіленими хмарними базами даних;

- робота з мінімальними втратами в машинних ресурсах при обробці великих обсягів інформації;

- можливість розробки своїх візуальних і бібліотечних компонентів;

Ґрунтуючись на вище перерахованих параметрах, був проведений аналіз середовищ розробки під ОС Windows, таких як Red Studio X3 (мова високого рівня програмування Pascal) і Visual C (мова програмування C ++) [223,224].

Це дозволить не тільки скоротити час реалізації розроблених моделей і методів, а й збільшити економічний ефект на етапі проєктування і програмування «Автоматизованої системи управління процесами розробки CPPS». Підбиваючи підсумки, можна з упевненістю сказати, що для проведення комп'ютерного експерименту з метою апробації прийнятих рішень для даної дослідницької роботи підходить середовище програмування Red Studio X3, яке повністю відповідає всім вимогам щодо реалізації доступу до баз даних.

6.3. Обґрунтування вибору баз даних Firebird

Для реалізації зберігання і роботи з великими обсягами інформації необхідно обрати сучасну базу даних, яка буде відповідати наступним

вимогам до розроблюваної «Автоматизованої системи управління процесами розробки CPPS»:

- можливість зберігання великих обсягів інформації;
- можливість додавання / видалення інформації;
- швидкість доступу до інформації;
- можливість віддаленого доступу до інформації.

В ході аналізу було обрано такі БД:

- реляційна база даних MS SQL;
- реляційна база даних Oracle;
- реляційна база даних PostgreSQL;
- реляційна база даних Interbase;
- не реляційна база даних MongoDB.

Перевагами розробки баз даних на реляційній моделі полягає в наступному:

- модель даних відображає інформацію в найбільш простій для користувача формі;
- заснована на розвиненому математичному апараті, який дозволяє досить лаконічно описати основні операції над даними;
- дозволяє створювати мови маніпулювання даними непроцедурного типу;
- маніпулювання даними на рівні вихідної БД і можливість зміни;
- сучасні автоматизовані системи та системи PLM використовують реляційні бази даних і побудовані на вільно розповсюджуваній системі управління базами даних (Firebird), що дає можливість зменшити загальну вартість «Автоматизованої системи управління процесами розробкою CPPS» як на момент її впровадження, так і безпосередньо під час її розробки.

До основних недоліків можна віднести наступне:

- трудомісткість розробки;
- невелику швидкість доступу до даних.

Незважаючи на недоліки, які були виявлені в ході аналізу сучасних баз даних, для вирішення поставленого завдання розробки «Автоматизованої системи управління процесами розробки CPPS» було прийнято рішення реалізувати базу даних на основі Firebird і в якості СУБД використовувати графічний інтерфейс IBExpert.

Обґрунтуванням вибору БД Firebird і СУБД IBExpert слугувало те, що даний вид БД успішно застосовується провідними виробниками ПЗ з вирішення задач розробки інформаційно-аналітичних модулів для CAx-систем та модулів CAE-аналізу SolidWorks.

Дана СУБД легко імпортується і інтегрується з роботою сучасних реляційних баз даних. Виходячи з всього вище вказаного пропонується використовувати для розробки БД – Firebird 3.0.2 і IBExpert Version 2017.04.24.

6.4. Розробка логічної і фізичної моделі бази даних

Виходячи з розроблених математичних моделей і методів, а також запропонованої технології, необхідно оперувати великими масивами інформації, які мають взаємопов'язану структуру з інформаційними блоками прийняття рішень, візуалізацією і складно-підлеглими залежностями в ієрархії побудови і генерації вихідного програмного коду.

Для спрощення і систематизації зберігання інформації, необхідної для вирішення поставленого завдання, було прийнято рішення розбити логічну структуру БД на модулі, які будуть об'єднані за «сутностями», в яких зберігається взаємозв'язок. Це буде реалізовано з використанням «зовнішніх ключів» типу підключення «один-нескінченність». Запропонована логічна структура БД представлена на рисунку 6.2.

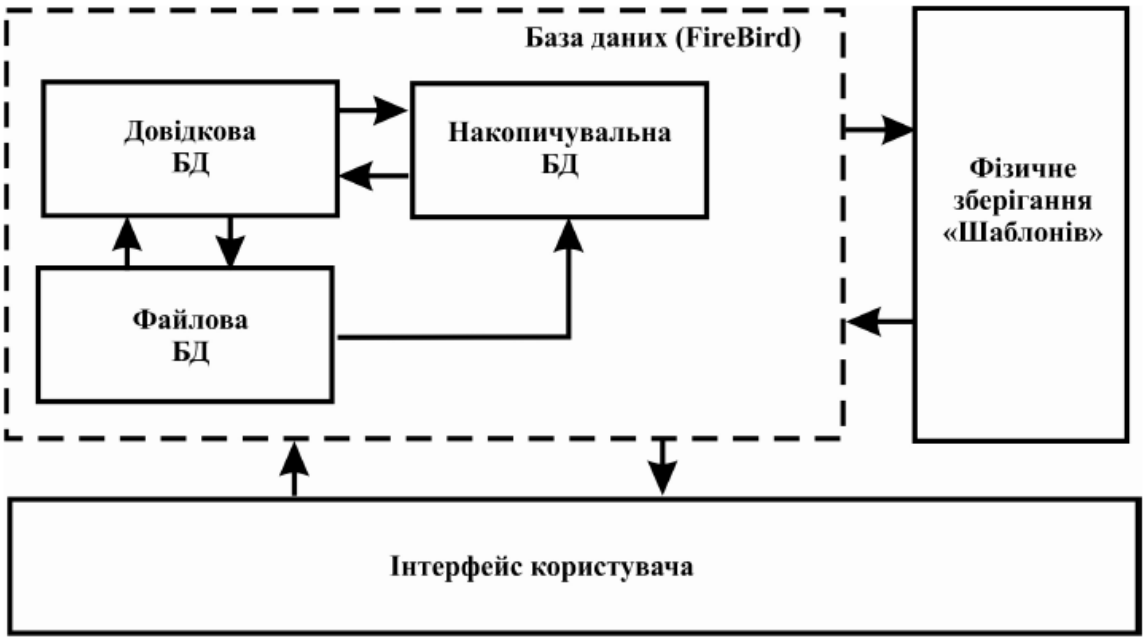


Рисунок 6.2 – Логічна структура БД

Логічна структура БД представлена у вигляді модулів, які виконують такі функції:

- довідкова БД являє собою ієрархію таблиць, в яких міститься: інформація про мови програмування; структуру коду; бібліотеки і взаємозв'язок їх використання; необхідні процедури і функції, які можна використовувати; події і можливі обробники дій над графічними елементами інтерфейсу; вбудовані процедури. Всі збережені дані в довідковій БД підкорюються критерію вибору середовища розробки та мови програмування. Довідкова БД реалізована з урахуванням розширення і адаптації до того чи іншого середовища розробки, користувач з правами адміністратора може додавати нові шаблони коду і прив'язувати їх до подій або до внутрішніх процедур або функцій.

- накопичувальна БД призначена для зберігання поточних проєктів та проєктів, реалізованих в даній системі. Структура накопичувальної БД є жорстко структурованою залежністю від даних замовника; назви проєкту і технічного завдання; згенерованого вихідного коду і інтерфейсу користувача; можливості редагування будь-якого елементу конструкції і шаблонів, які застосовані в даному проєкті. Користувач «Автоматизованої

системи управління процесами розробки CPPS» може редагувати визначення і сутності даного проєкту, змінювати взаємозв'язок і послідовність вставки кожного шаблону (контейнера) з програмним кодом в залежності від вимог замовника.

- фізичне зберігання «шаблонів» представляє собою сукупність сутностей, що підкоряються довідковій базі даних. Ця сукупність містить в собі посилання на фізичну папку, в якій знаходяться файли з мінімальною і необхідною структурою кожної мови програмування для компіляції порожнього проєкту.

Представимо кожен модуль у вигляді фізичної моделі БД, яка реалізована в «Автоматизованій системі управління процесами розробки CPPS». Запропонована фізична структура побудови зв'язків таблиць заснована на використанні металінгвістичної формули Бекуса-Наура (мова БНФ). Вона дає можливість систематизувати мови високого рівня програмування у вигляді блоків, заголовків програми і тіла програм як «сутності» (елементи мови, організацію дій над даними, організацію даних і субпідлеглих «сутностей»: алфавіт, лексеми, синтаксис, оформлення програмного коду, елементи введення-виведення даних (елементи інтерфейсу, роботи з БД, робота з файлами), обробка даних (події над елементами форм (операції і вирази, оператори, організація і використання підпрограм).

Інформаційна модель взаємозв'язків основних «сутностей» довідкової БД представлена на рисунку 6.3

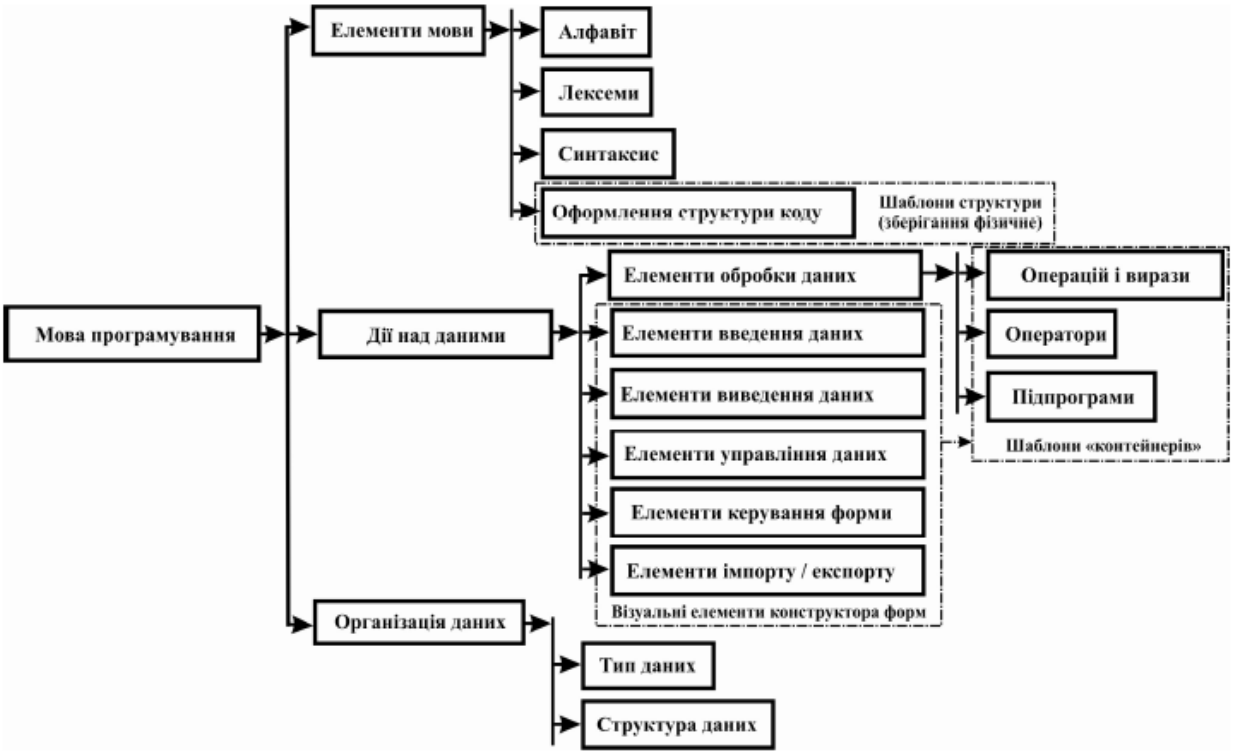


Рисунок 6.3 – Інформаційна модель взаємозв'язків основних «сутностей» довідкової БД

Структура фізичної моделі довідкової БД за зв'язками сутностей: «Мова програмування», «Елементи мови», «Алфавіт», «Лексеми», «Синтаксис», «Оформлення структури коду» представлена на рисунку 6.4.

На рисунку 6.5 представлена фізична модель довідкової БД за зв'язками сутностей: «Дія над даними»→«Елементи введення даних», «Елементи виведення даних», «Елементи управління даними», «Елементи управління форми», «Елементи імпорту / експорту», а також зв'язкам: «Елементи обробки даних», «Операції і вирази», «Оператори», «Підпрограми».

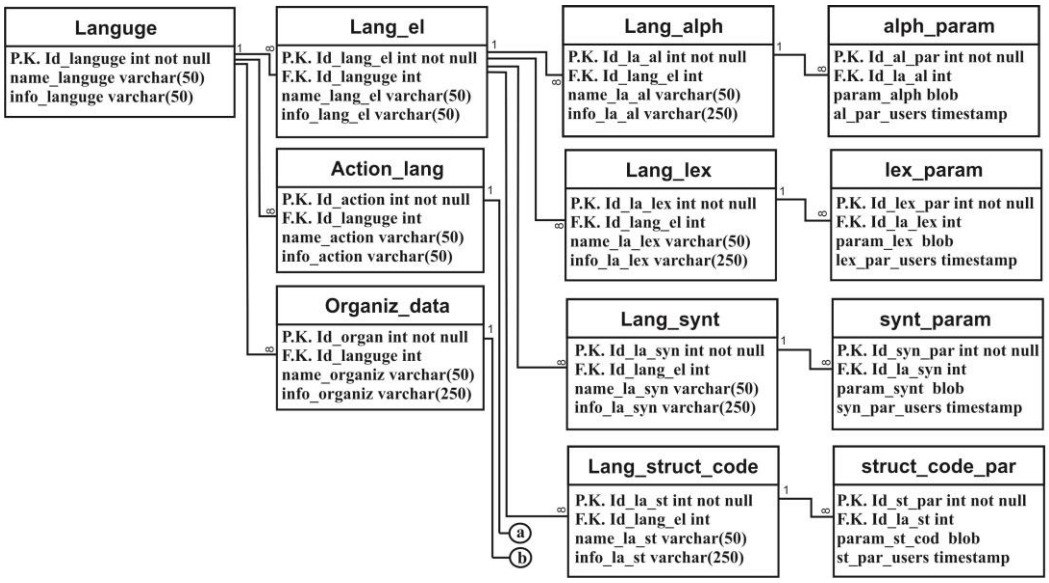


Рисунок 6.4 – Фрагмент структури фізичної моделі довідкової БД за зв'язками сутностей: «Мова програмування», «Елементи мови», «Алфавіт», «Лексеми», «Синтаксис», «Оформлення структури коду»

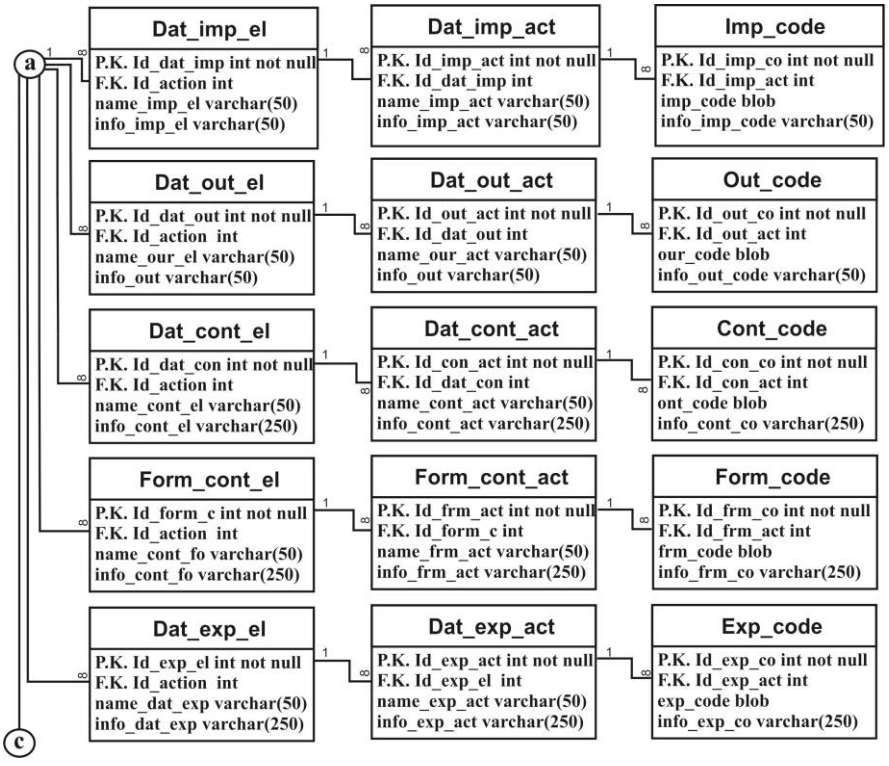


Рисунок 6.5 – Фрагмент структури фізичної моделі довідкової БД за зв'язками сутностей: «Дія над даними»→«Елементи введення даних», «Елементи виведення даних», «Елементи управління даними», «Елементи управління формами», «Елементи імпорту / експорту»

Реалізацію фрагменту зв'язків зберігання інформації в довідковій БД за сутностями «Елементи обробки даних»→«Операції і вирази», «Оператори», «Підпрограми» представлено на рисунку 6.6.

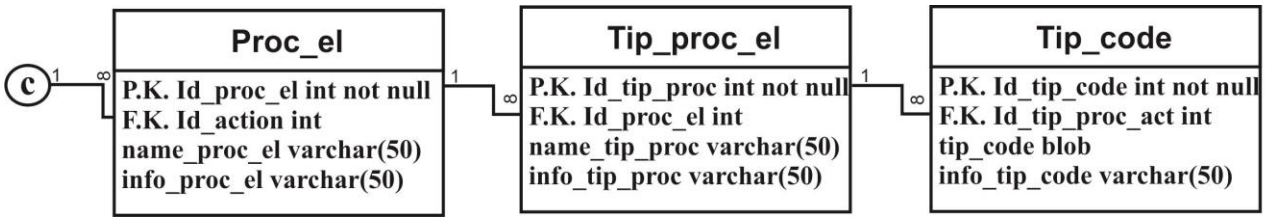


Рисунок 6.6 – Фрагмент зв'язків зберігання інформації в довідковій БД по сутностей «Елементи обробки даних»→«Операції і вирази», «Оператори».

Фрагмент структури фізичної моделі довідкової БД для зв'язків сутностей «Організація даних»→«Тип даних», «Структура даних» представлена на рисунку 6.7.

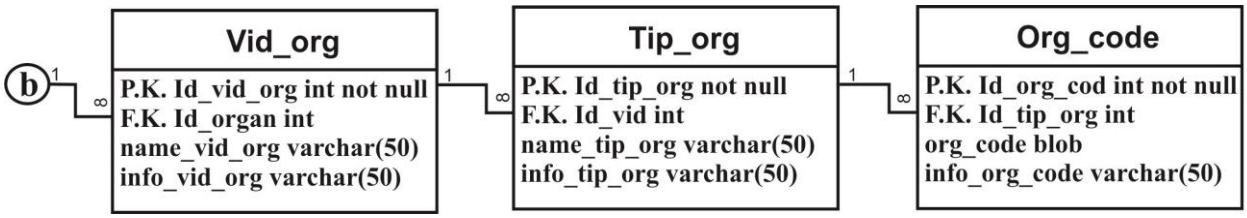


Рисунок 6.7 – Фрагмент структури фізичної моделі довідкової БД для зв'язків сутностей «Організація даних»→«Тип даних», «Структура даних»

Для зберігання довідкової інформації «шаблонів» і синтаксису мови SQL в дослідженні пропонується наступна інформаційна модель (рис. 6.8).

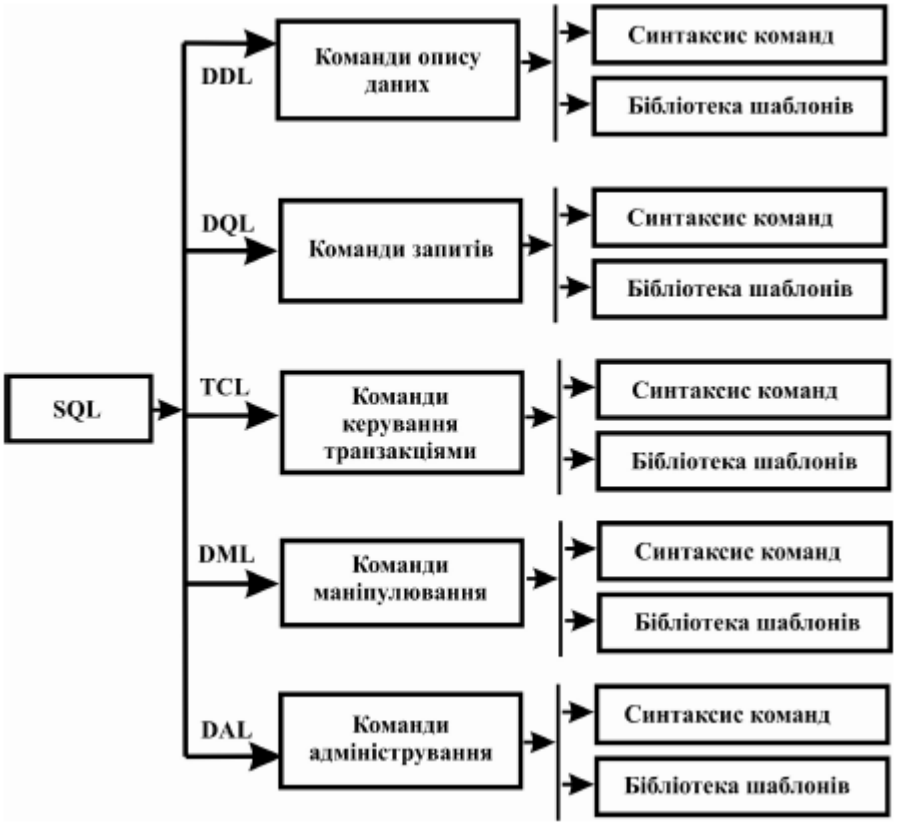


Рисунок 6.8 – Інформаційна модель довідкової БД для зберігання SQL

На базі запропонованої інформаційної моделі розроблена фізична модель БД, яка складається з наступної послідовності зв'язків сутностей: «SQL»→«Тип команд»→«Синтаксис команди»→«Шаблони коду». Дане рішення дозволить користувачу систематизувати «контейнери» під загальний синтаксис команд мови SQL.

Поле «Id_action» в таблиці «Tip_sq» зберігає значення первинного ключа таблиці «Action_lang» і дає можливість зв'язати команди SQL з візуальними елементами і дії над даними.

Структура фізичної моделі представлена на рисунку 6.9.

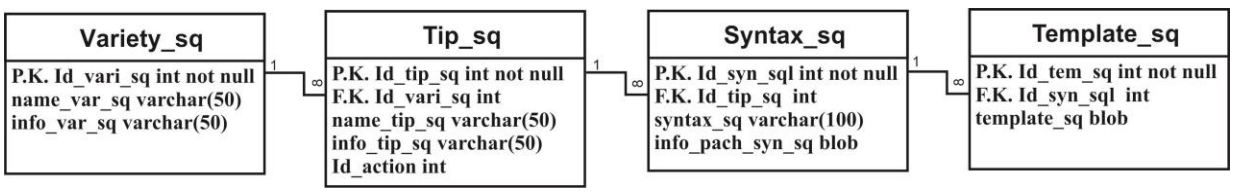


Рисунок 6.9 – Фізична модель довідкової БД для зберігання шаблонів команд SQL

Запропонована фізична структура довідкової БД реалізована у вигляді 37 таблиць, що містять більше 110 полів для зберігання і взаємодії інформаційних залежностей, які повністю відповідають законам нормалізації, в них відсутня надмірність і неузгодженість між сутностями. Все це робить БД гнучкою і адаптивною в роботі.

Для «Автоматизованої системи управління процесами розробки CPPS» запропонована накопичувальна база даних, яка реалізує функції роботи із замовником. У ній зберігається інформація про поточні проєкти CPPS, що дозволяє проводити коригування вихідного коду, а також використовувати накопичену базу в якості шаблонів для створення нових проєктів з мінімальним коригуванням. Це дозволить скоротити час проєктування і домогтися максимальної економічної ефективності використання «Автоматизованої системи управління процесами розробки CPPS». Інформаційна модель накопичувальної БД представлена на рисунку 6.10.

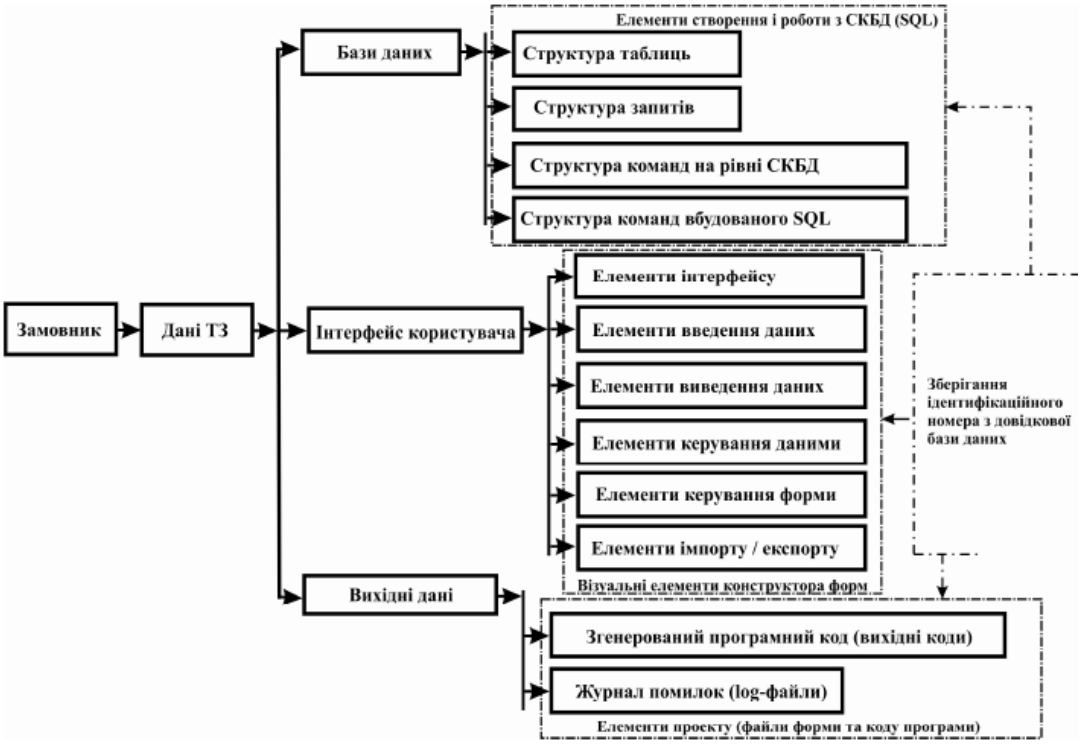


Рисунок 6.10. – Інформаційна модель накопичувальної БД

Спроєктована фізична модель накопичувальної БД представлена на рисунку 6.11.

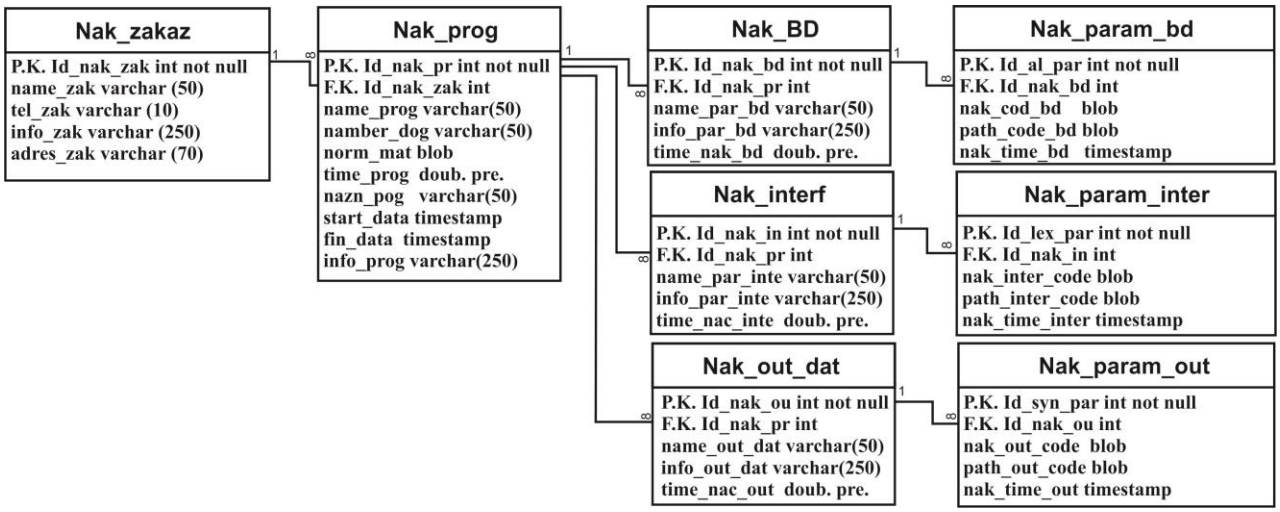


Рисунок 6.11 – Фізична модель накопичувальної БД

6.5. Розробка системи автоматизації процесу управління розробкою кібернетичної складової CPPS

Для апробації запропонованих методологій процесу управління розробкою складних CPPS, а також їх програмної реалізації у вигляді системи автоматизації процесів управління розробкою кібер складової CPPS розроблено інтерфейс користувача. Він реалізований з урахуванням ергономічності і інтуїнтивності розуміння користувачем. Враховуючи, що користувачу необхідно одночасно оперувати великими об'ємами інформації, запропоновано використовувати метод MDI інтерфейсу, який дозволить контролювати велику кількість інформації і максимально логічно провести її групування для зручності доступу і контролю.

На рисунку 6.12 представлене вікно ініціалізації і завантаження бібліотек для запуску системи автоматизації процесу управління розробкою кібер складової CPPS

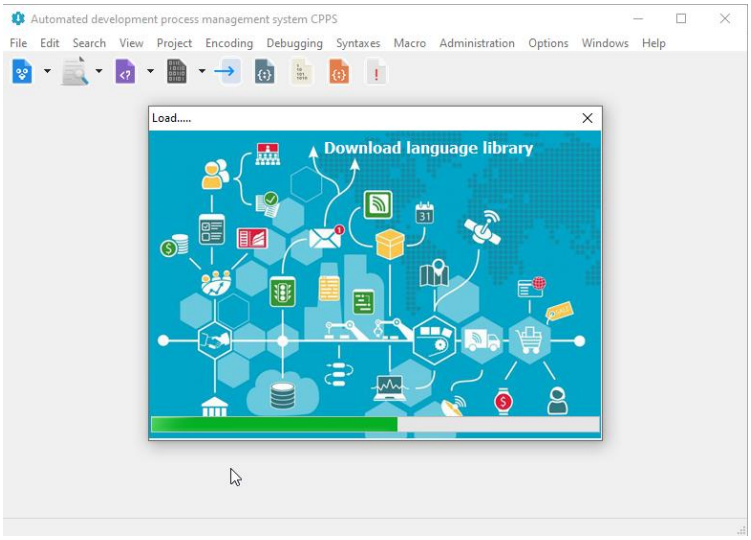


Рисунок 6.12 – Вікно ініціалізації і завантаження бібліотек з БД

Після успішної ініціалізації і підключення бібліотек: «Language library», «CPPS Structure», параметрів «Physical world parameters» і «Cyber world parameters», а також «Software modules» і «Language compilers», інформація для яких має зберігатися в БД, користувачу надається Select menu для вибору основних функцій системи автоматизації процесу управління розробкою кібер складової CPPS, яку представлено на рис. 6.13

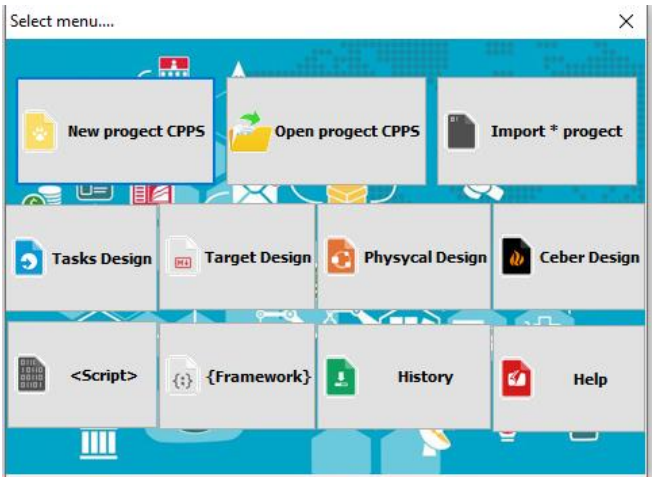


Рисунок 6.13 – Select menu для вибору основних функцій системи автоматизації процесу управління розробкою кібер складової CPPS

До основних функцій системи автоматизації процесу управління розробкою кібер складової CPPS, представленим в Select menu, відносяться:

- New project CPPS – модуль створення нового проєкту з п'ятого рівня опису головної мети розробки CPPS, а також цілей на кожному рівні розробки і відповідність завдань, вирішення яких дозволяють досягти головної мети. Послідовність процесу управління розробкою CPPS представлений на рисунку 2.1;

- Open project CPPS – відкриття вже існуючого проєкту, який було збережено у вигляді окремих файлів. В контексті даної реалізації пропонуються наступні розширення файлів доступні до завантаження в систему:

- * .tas – файл містить інформацію про цілі розробки CPPS (Tasks Design), опис якого представлено в підрозділі 2.2;

- * .tar – файл містить інформацію про завдання розробки CPPS (Target Design), опис якого представлено в підрозділі 2.2;

- * .phy – файл містить інформацію про фізичну складову CPPS (Physical Design), опис представлено в підрозділі 2.3;

- * .cyb – файл містить інформацію про кібернетичну складову CPPS (Cyber Design), опис представлено у підрозділі 2.4 та у розділі 4;

- * .scr – файл містить інформацію про синтаксичну та семантичну мовні моделі CPPS (<Script>), опис яких представлено у 5 розділі;

- * .frm – файл визначає структуру і правила побудови архітектури процесу управління розробкою CPPS, який задає поведінку за умовчанням, на початковому етапі розробки (рис. 2.1) і ускладнюється з кожною ітерацією відповідно до запропонованої технології управління розробкою CPPS, яку представлено на рисунку 2.7. {Framework}.

- модуль Import * project – дає можливість проводити імпорт даних для реалізації візуалізації інтерфейсу користувача, в рамках кібер складової CPPS, з альтернативних джерел з «жорсткою» структурою

подання інформації (* .xml - Imports from XML, *. xls- Imports from Excel 2003 and under, *. xlsx - Imports from Excel 2007 and over).

Дане рішення обумовлене виникненням завдання швидкої розробки прототипу візуалізації інтерфейсу кібер складової CPPS при безпосередній участі замовника. Це дає можливість вже на ранньому етапі процесу управління розробкою CPPS отримати вихідні дані для чіткого розуміння вимог до графічного інтерфейсу.

Варто зауважити, що розроблений метод (в підрозділі 6.6) подання інформації для розробки інтерфейсу користувача має один істотний недолік, який полягає у необхідності розробки «жорстко» фіксованої, підпорядкованої структури представлення параметрів і значень візуальних форм і об'єктів інтерфейсу користувача у залежності від мови та середовища розробки CPPS.

- модуль Tasks Design і Target Design дають можливість розробити і визначити цілі розробки CPPS на кожному рівні процесу управління, а також обрати необхідні завдання та вимоги до параметрів і характеристик для їх досягнення;

- Physical Design – модуль, який дозволяє користувачу описати послідовність процесу управління розробкою і вибору параметрів і зв'язків на функціональному і організаційно-технічному етапі, а також визначає технічні характеристики атомарних елементів (фізичні і електричні параметри датчиків і виконавчих пристроїв відповідно до їх специфікації), які дозволять досягти вимоги до завдань, що ставляться перед ними;

- модуль Cyber Design – дає можливість на базі обраних і реалізованих параметрів фізичної складової CPPS розробити інфологічну і інформаційну структуру управління. Це дасть можливість отримати алгоритми роботи і управління на кожному рівні CPPS, на базі яких можна реалізувати автоматизацію отримання прототипу інтерфейсу користувача і його програмний код для обраного середовища розробки;

- модуль <Script> – представляє собою текстове середовище розробки кібер складової CPPS, на базі запропонованих синтаксичних правил і семантичної мовної моделі, фрагменти якої представлені в 5 розділі;

- {Framework} – модуль («каркас») архітектури процесу управління розробкою CPPS, на якому користувач може бачити не тільки процес розробки, а й взаємозв'язки між усіма етапами і рівнями, перевірки цілісності досягнення головної мети розробки CPPS. Основним завданням модуля {Framework} є послідовна візуалізація зв'язків від початкового етапу процесу управління розробкою до отримання алгоритмів функціонування і на їх базі реалізація візуальних компонентів і генерація програмного коду розроблюваного CPPS;

- модуль History містить текстову інформацію з даними про типи та види внесених змін в проєкти CPPS, час внесення і ім'я користувача, що дає можливість відстежувати зміни, а також некоректні правки, щоб зробити відкат до останньої робочої версії;

- модуль Help містить повну довідкову інформацію про архітектуру, послідовності, правила роботи з системою.

Розглянемо основні модулі, які необхідні для реалізації процесу управління розробкою кібер складової CPPS. На рис. 6.14 представлено вікно модуля «New project CPPS».

Як можна бачити, на даному етапі розробник CPPS задає такі дані: «Project number», «Name», «Customer». У випадаючому меню «Level» розробник обирає етап розробки, який може бути: «Target» – етапи цілей і завдань, «Physical» – етапи функціонального рівня і організаційно-технічної структури, «Cybernetic» – етапи інфологічної і інформаційної структури управління, а також алгоритмів функціонування, «Full life cycle "Jump"» – процес управління розробкою CPPS з «нуля»; «Specificity» – містить найменування і типи ЧПК обладнання, опис головної мети і завдань, які повинна вирішувати CPPS; «Language» і «Development

environment» – вибір мови та середовища розробки; вибір типу БД відбувається у випадаючому меню «DB type», а вибір системи управління баз даних (СУБД) у «DBMS» відповідно.

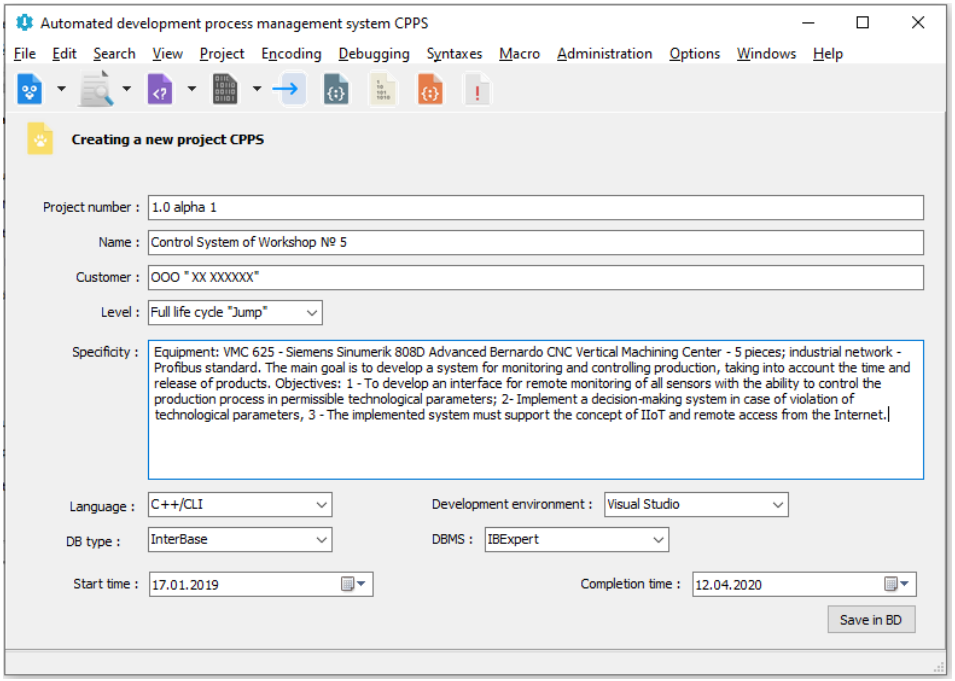


Рисунок 6.14 – Вікно модуля «New project CPPS»

Після збереження нового проєкту в БД, за допомогою кнопки «Save in BD» відкривається вікно, представлене на рисунку 6.15, яке дозволяє розробнику обрати етап «Level»=«Full life cycle "Jump"». Отже, розробнику доступні в «Project tree» всі етапи і рівні процесу управління розробкою CPPS. Обираючи необхідний етап і рівень розробник може використовувати текстове середовище розробки <Script>, в даному випадку це кібер складова CPPS на етапі «Control algorithms» на рівні «Decomposition level 0», в якій за допомогою семантичної мовної моделі описуються всі необхідні параметри і зв'язки для реалізації графічного інтерфейсу користувача CPPS.

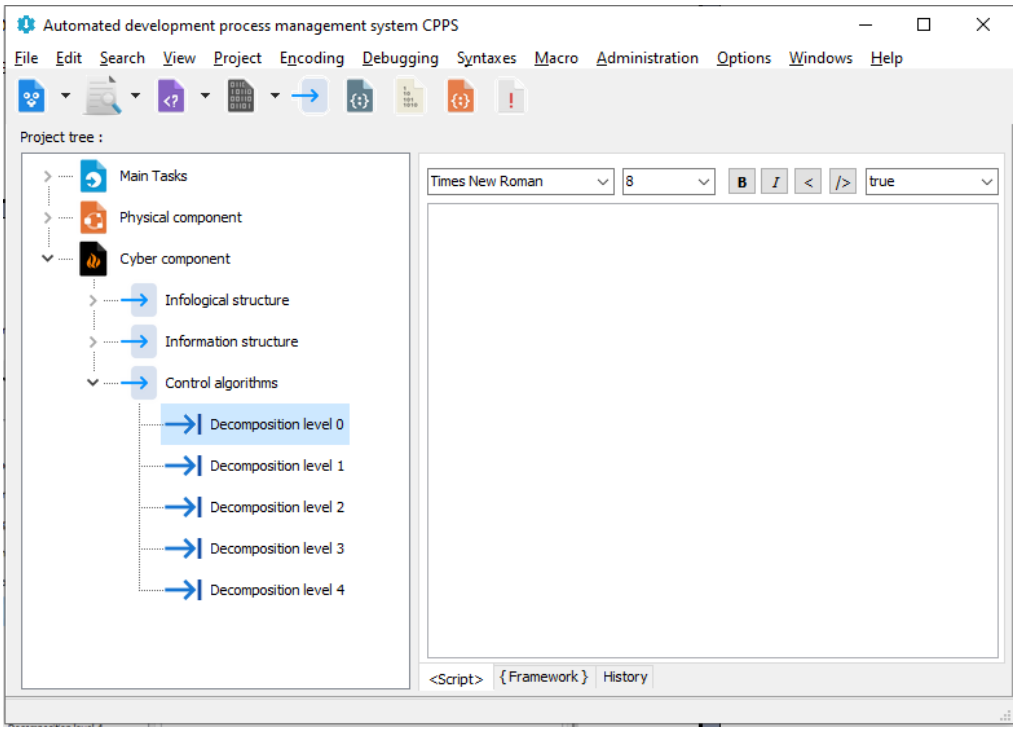


Рисунок 6.15 – Вікно розробки CPPS на етапі «Control algorithms» рівня «Decomposition level 0».

При виконанні команди «Debugging» в системі була реалізована синтаксична і логічна перевірка написаного коду з виділенням рядків, де сталася критична помилка, або видається попередження про можливу логічну невідповідність вимогам, яка в подальшому позначиться при загальній компіляції проєкту розробки CPPS .

Обираючи закладку {Framework} розробник стає доступною візуалізація зв'язків від початкового етапу процесу управління розробкою до отримання алгоритмів функціонування, з точною прив'язкою до рівня «Decomposition level 0». Це дозволяє в процесі управління розробкою візуально спостерігати зміну кількості зв'язків їх характеристик і прив'язок по всій архітектурі процесу управління розробкою CPPS.

Детальніше принцип роботи в системі автоматизації процесу управління розробкою кібер складової CPPS буде приведено в розділі 6.7, в якому будуть проводитись експериментальні дослідження з розробки

фрагменту реалізації кібер складової CPPS у відповідності з господарським договором.

6.6 Розробка структури представлення даних для створення прототипу графічного інтерфейсу CPPS

Одним із завдань в процесі управління розробкою CPPS є створення візуального графічного інтерфейсу користувача у вигляді прототипу із зазначенням всіх необхідних і достатніх об'єктів відображення інформації. Розроблюваний прототип повинен відповідати вимогам інформативності та зручності подання даних, ергономічності і usability.

Для вирішення даного завдання рекомендується запрошувати представника замовника CPPS і обговорювати всі вимоги до розміщення графічних елементів інтерфейсу, їх тип і функції, які вони повинні виконувати на рівнях SCADA, MES і ERP-систем. Одним зі складних моментів у роботі з замовником є швидкість розробки прототипу інтерфейсу і внесення змін та коригувань в режимі реального часу й демонстрація отриманого результату.

Виходячи їх цього, в даній роботі розроблена структура представлення даних для створення прототипу графічного інтерфейсу, яка реалізована в модулі Import * project (рис. 6.12). Ґрунтуючись на позиції, що структура представлення даних для створення прототипу графічного інтерфейсу повинна дотримуватися всіх правил і вимог до об'єктно-орієнтованого програмування і прив'язки до мови та середовища розробки, робимо припущення, що структура повинна мати «жорстко» підпорядковану ієрархію представлення даних і властивостей, а також подій по кожному графічному об'єкту інтерфейсу.

Виходячи з висунутих припущень пропонується використовувати наступні формати файлів, які дозволяють реалізувати вимоги:

- * .xml – Imports from XML;

- *. Xls – Imports from Excel 2003 and under;

- *. xlsx – Imports from Excel 2007 and over.

Варто зауважити, що інформація про дані, властивості, події і значення по кожному графічному об'єкту інтерфейсу для різних середовищ розробки може відрізнятися, отже, це говорить необхідність створення шаблонних структур для кожного середовища розробки.

Структура представлення даних для створення прототипу графічного інтерфейсу повинна повністю описувати його функціональність, специфіку і додаткові функції, які можна буде інтерпретувати як запити до БЗ (бази знань) за допомогою запропонованих математичних моделей і методів опису та прийняття рішень. Виходячи з цього, була розроблена структура представлення даних (підрозділи 4.3-4.4) у вигляді множини взаємно залежних параметрів.

Дані параметри в сумі дають можливість окреслити програмний продукт, що розроблюється, на базі простого природної мови опису даних в легкодоступній і інтуїтивно зрозумілій формі, використовувати стандартні прикладні програмні рішення, що відповідають міжнародним форматам експорту / імпорту даних і при цьому мати структуроване подання даних.

Структуризацію властивостей, параметрів і подій відповідно до запропонованих рішень можна представити як складно підпорядковану структуру з прив'язкою до подій, які виникають при роботі з інтерфейсом залежно від дій користувача.

На рисунку 6.16 представлена графічна структура зв'язків властивостей і параметрів, подій і дій для $Form_1^{master}$.

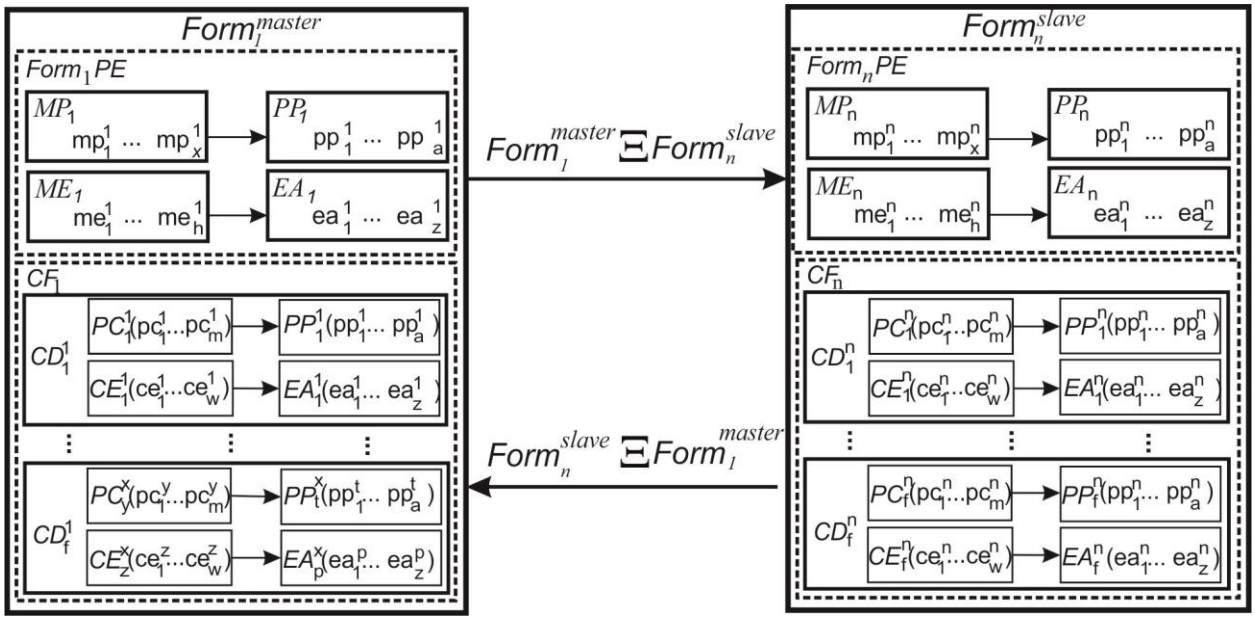


Рисунок 6.16 – Графічна структура зв'язків властивостей, параметрів, подій і дій для $Form_1^{master}$

Представлена концепція побудована на застосуванні графічних елементів інтерфейсу як базових носіїв інформації про розроблюваний CPPS.

Під $Form_1^{master}$ розуміється головна форма кібер складової CPPS, яка характеризується сукупністю основних блоків: $Form_1 PE$, CF_1. Блок $Form_1 PE$ складається із взаємозв'язаних структурних елементів MP_1: параметри $mp_1^1,...,mp_t^1 \to PP_1$: значення $pp_1^1,...,pp_q^1$. Кожному mp_t^1 відповідає одне pp_q^1, яке може набувати значень цифрових, логічних операцій (false, true) або зарезервованих параметрів опису значень pp_q^1 під певну мову високого рівня програмування (Top і т.д.). Набір таких параметрів суворо обмежений і несе в собі інформацію про умови візуального відображення $Form_1^{master}$ для користувача (по центру робочого столу, розгорнута на весь робочий стіл, і т.д.). Блок ME_1 являє собою сукупність подій $me_1^1,...,me_h^1$, які можна накласти на $Form_1^{master}$, при обробці дій у вигляді «Create form»,

«Close form» і т.д. Набір подій і параметрів у вигляді програмного коду строго визначений для кожної мови і середовища розробки. Кожній з подій $me_1^1,...,me_h^1$ відповідають $ea_1^1,...,ea_z^1$ блоку EA_1, які можуть набувати значень у вигляді функцій і процедур взаємодії з користувачем, описаних у вигляді фрагментів програмного коду.

Кожна $Form_1^{master}$ має нескінченний набір CF_1, який представляє собою структуру $Form_1^{master}$ у вигляді дерева побудови, де кожен елемент дерева – набір візуальних компонентів, призначених для роботи з даними і елементами групування (елементи введення (Edit) і відображення даних (Grid), елементи інтерфейсу (Menu), елементи групування (GroupBox)).

Кожен елемент представлений в блоці CD_1^1 є взаємозв'язком $PC_1^1 \to PP_1^1$. Ґрунтуючись на запропонованій структурі PC_1^1, яка володіє такими ж властивостями як елемент MP_1, але при цьому CE_1^1 володіє нескінченною варіацією дій EA_1^1, які можуть співпадати з EA_1 (звернення до БД, розрахунки, закриття форми, і т.д.) при виклику певної події (натискання на компонент «Button», наведення мишки, і т.д.).

Залежно від загальної структури побудови графічного інтерфейсу CPPS (скільки елементів типу, $Form_1^{master}...Form_n^{slave}$, буде використовуватися) необхідно врахувати передачу глобальних змінних і функцій переходу між вікнами інтерфейсу. Внаслідок цього взаємодію між елементами $Form_1^{master} \Xi Form_n^{slave}$ має бути враховано в рамках розроблюваного CPPS, у середньому кількість елементів типу $Form_1^{master}$ і $Form_n^{slave}$ може коливатися від 1 ... 25-30 і вище, вони можуть викликатися спливаючим всередині головної $Form_1^{master}$ і посилатися на неї.

Ґрунтуючись на запропонованій графічній структурі зв'язків властивостей і параметрів, подій і дій для $Form_1^{master}$, розробимо структуру представлення даних в пакеті Excel 2003, для створення прототипу

графічного інтерфейсу CPPS мовою програмування Pascal у середовищі розробки Red Studio X5, яку представлено на рисунку 6.17.

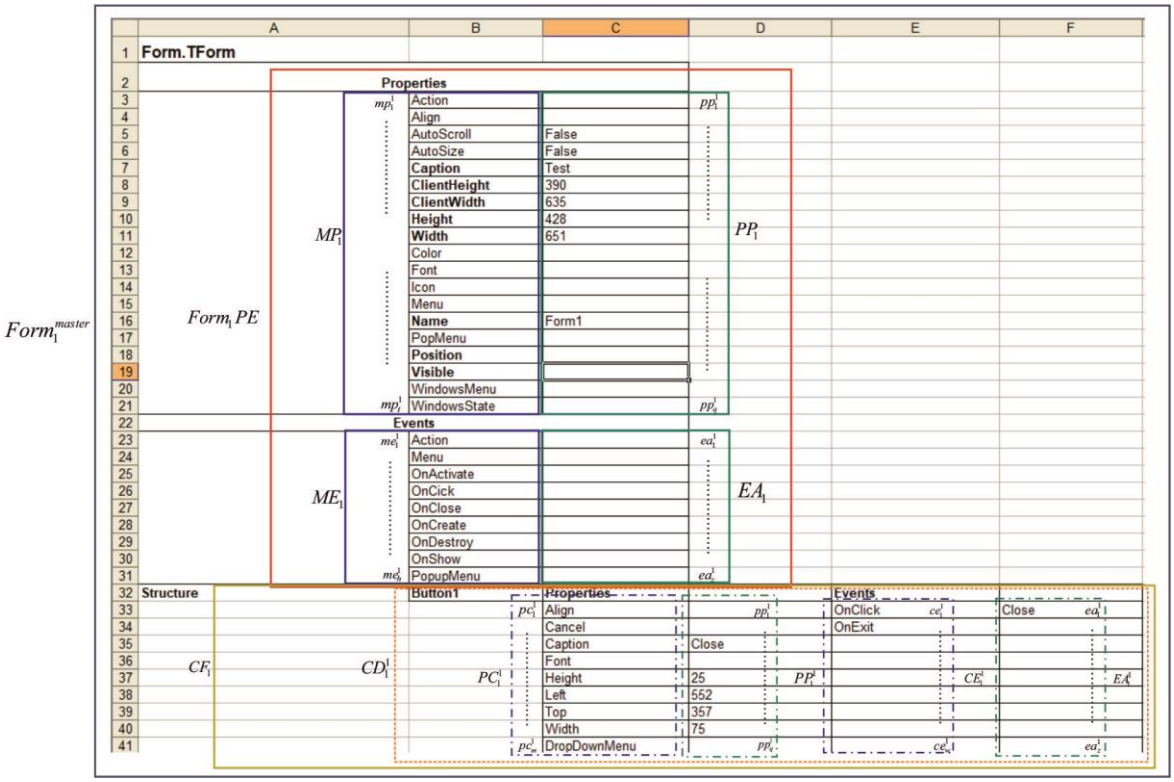


Рисунок 6.17 – Приклад структури представлення даних в пакеті Excel 2003 мовою Pascal в середовищі розробки Red Studio X5 для створення прототипу інтерфейсу користувача через модуль Import * project

Фрагмент заповненої структури представлення даних з створення прототипу $Form_1^{master}$ у блоці $Form_1 PE$ із зв'язками $MP_1 \rightarrow PP_1$ і $ME_1 \rightarrow EA_1$ наведений на рисунку 6.18.

1	**Form1.TForm**		
2	**Properties**		
3		Action	
4		Align	
5		AutoScroll	False
6		AutoSize	False
7		**Caption**	Test
8		**ClientHeight**	390
9		**ClientWidth**	635
10		**Height**	428
11		**Width**	651
12		Color	
13		Font	
14		Icon	
15		Menu	
16		**Name**	Form1
17		PopMenu	
18		**Position**	
19		**Visible**	
20		WindowsMenu	
21		WindowsState	
22	**Events**		
23		Action	
24		Menu	
25		OnActivate	
26		OnCick	
27		OnClose	
28		OnCreate	
29		OnDestroy	
30		OnShow	
31		PopupMenu	

Рисунок 6.18 – Фрагмент заповненої структури представлення даних з створення прототипу $Form_1^{master}$

Приклад наведеного фрагменту структури представлення даних (рис. 6.18) дозволить в модулі Import * project згенерувати програмний код мовою Pascal для середовища розробки Red Studio X5 для головного вікна із заданими розмірами і параметрами відповідно вимог замовника. Згенерований програмний код відкривається в середовищі розробки і уточняються вимоги до візуального графічного інтерфейсу.

Для створення об'єктів управління або візуалізації інформації до блоку CF_1 (рис. 6.17) додається елемент управління або візуалізації CD_1^1. Фрагмент заповненої структури представлення даних по створенню візуального графічного елементу «Button1», а також елемент групування візуальних елементів «GroupBox1», до якого входить елемент вводу текстової інформації «Edit1», представлено на рисунку 6.19.

У стовпець «Events», який є блоком CE_1^1, в рядку «OnClick» (ce_1^1), реалізована подія на одноразове натискання графічного елементу «Button» (ea_1^1), при якому викликається «*LingusticVariable*» з ім'ям «Close», яка посилається на «*ContainerSolutions*», який містить готовий програмний код

функції яка закриває $Form_1^{master}$. Більш детально математичний опис механізму взаємодії описаний в (4.52) і (4.53).

32	**Structure**	**Button1**	**Properties**		**Events**		
33			Align		OnClick	Close	
34			Cancel		OnExit		
35			Caption	Close			
36			Font				
37			Height	25			
38			Left	552			
39			Top	357			
40			Width	75			
41			DropDownMenu				
42		**GroupBox1**	**Properties**		**Events**		
43			Align		OnClick		
44			Cancel		OnExit		
45			Caption	Test			
46			Font				
47			Height	343			
48			Left	8			
49			Top	8			
50			Width	619			
51			DropDownMenu				
52			TabOrder	1			
53			**Edit1**	**Properties**		**Events**	
54				Align	alNone	OnClick	
55				Cancel			
56				Text	Search		
57				Font			
58				Height	21		
59				Left	16		
60				Top	24		
61				Width	585		

Рисунок 6.19 – Фрагмент заповненої структури представлення даних блоку CF_1, який містить елементи CD_1^1 «Button1» і CD_2^1 «GroupBox1»

У залежності від вибраної мови і середовища розробки кібер складової CPPS, буде змінюватися назва і призначення елементів в блоках $MP_1(mp_1^1,...,mp_t^1)$ і $ME_1(me_1^1,...,me_h^1)$, тому це твердження правомірно для елементів блоків $CD_1^1,...,CD_f^1$. Але при цьому структура представлення даних для створення прототипу графічного інтерфейсу кібер складової CPPS буде однакова за умови, що будуть використовуватися об'єктно-орієнтовані мови програмування.

6.7 Експериментальні дослідження і аналіз отриманих результатів

Для проведення експериментальних досліджень і перевірки правильності запропонованих математичних моделей і методів в даному дослідженні розроблено структурну схему системи моніторингу та візуалізації даних для кіберфізичних виробничих систем. Ця система

призначена для отримання мережевих даних та їх основі надання графічного інтерфейсу користувачеві. На рисунку 6.20 зображена загальна система для подібних систем.

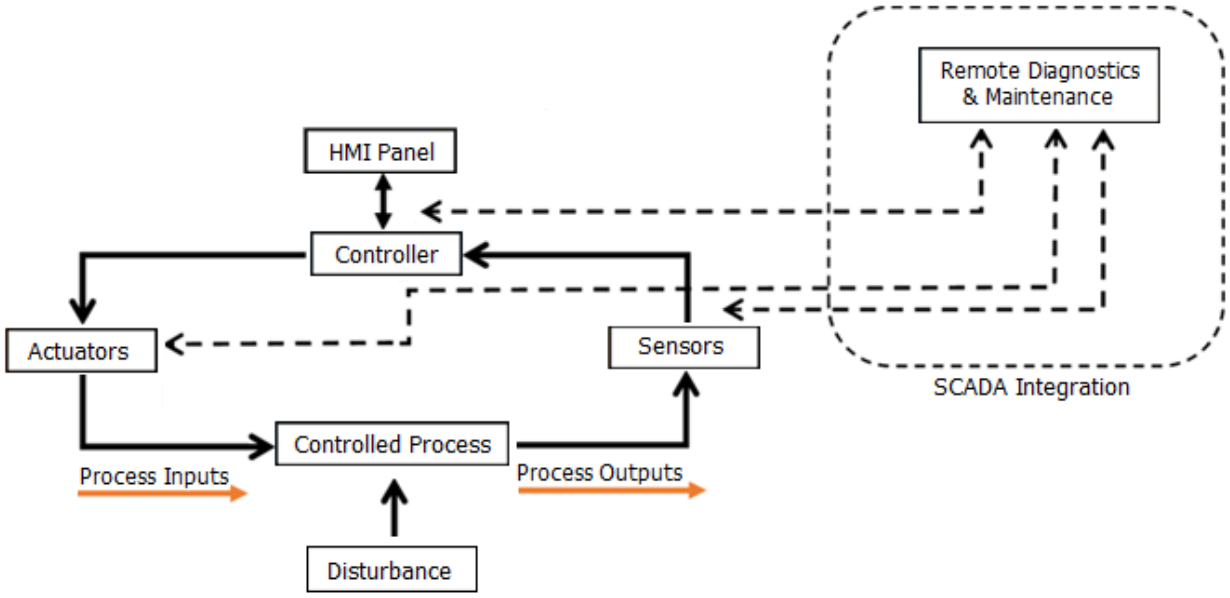


Рисунок 6.20 – Загальна структурна схема SCADA/HMI систем.

Структурну схема розроблюваної системи представлено на рис. 6.21.

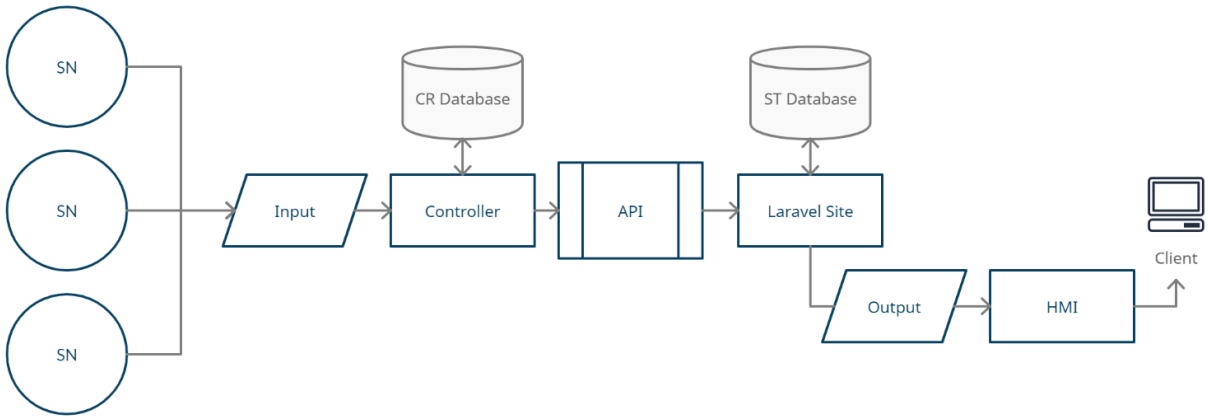


Рисунок 6.21 – Структура розроблюваної системи

На отриманій структурній схемі зображені всі основні елементи системи такі як SN – датчики підключені до системи, Contoreller – мікрокомп'ютер Raspberry Pi, який отримує дані з датчиків зберігає їх у базу даних CR Database, після чого відправляє дані через API на створену кінцеву

точку у Laravel Site. Після отримання даних з Contoreller у backend частині сайту дані опрацьовуються, зберігаються у базі даних ST Database та виводяться на екран користувача.

Для розробки системи моніторингу та візуалізації даних для кіберфізичних виробничих систем було створено інформаційну модель системи (рис. 6.22).

Інформаційна модель — система сигналів, що свідчать про динаміку об'єкта керування, умови зовнішнього середовища та стан самої системи керування. Інформаційною моделлю можуть слугувати наочні зображення, знаки, графічні моделі і комбіновані зображення.

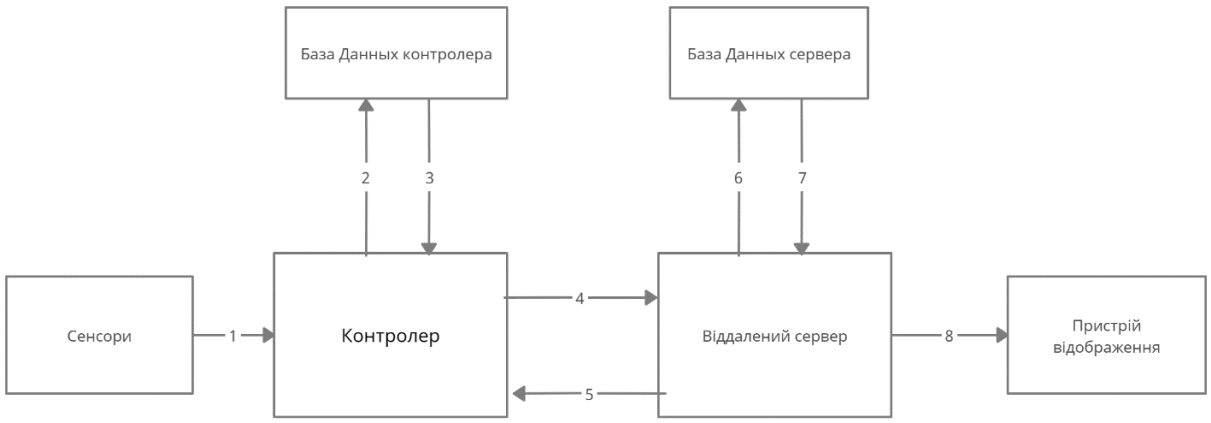


Рисунок 6.22 – Інформаційна модель системи

Розшифрування даних, які передаються у ході процесу використання системи, що передаються. Стрілка під номером 1 передає об’єкт даних з датчиків до контролера, у який входить інформація, що це за сенсор, отримані значення, локація та одиниці вимірювання. Після отримання об’єкту даних контролер записує дані у базу даних (2), крім вже існуючих даних, додається час збереження та унікальний ідентифікатор для цих даних. Після збереження дані з усіх об’єктів збираються (3) у один великий об’єкт та відправляться до віддаленого серверу (4). Віддалений сервер повідомляє контролер про отримання об’єкта (5), у випадку не отримання цієї відповіді контролер повторює операції 3 та 4, доки не отримає відповідь. Після

розшифрування та обробки отриманого об’єкта, дані з нього завантажуються до бази даних (6) для аналізу даних з сенсорів за довгий час. Після чого дані з бази даних структуруються (7) та відображаються на пристрої користувача (8).

Після створення інформаційної моделі було розроблено архітектуру системи з урахуванням ділення системи на окремі модулі, а саме модуля моніторингу та візуалізації. Кожний з модулів був проаналізований та розбитий на основні компоненти та їх складові. Архітектура зображена на рисунку 6.23.

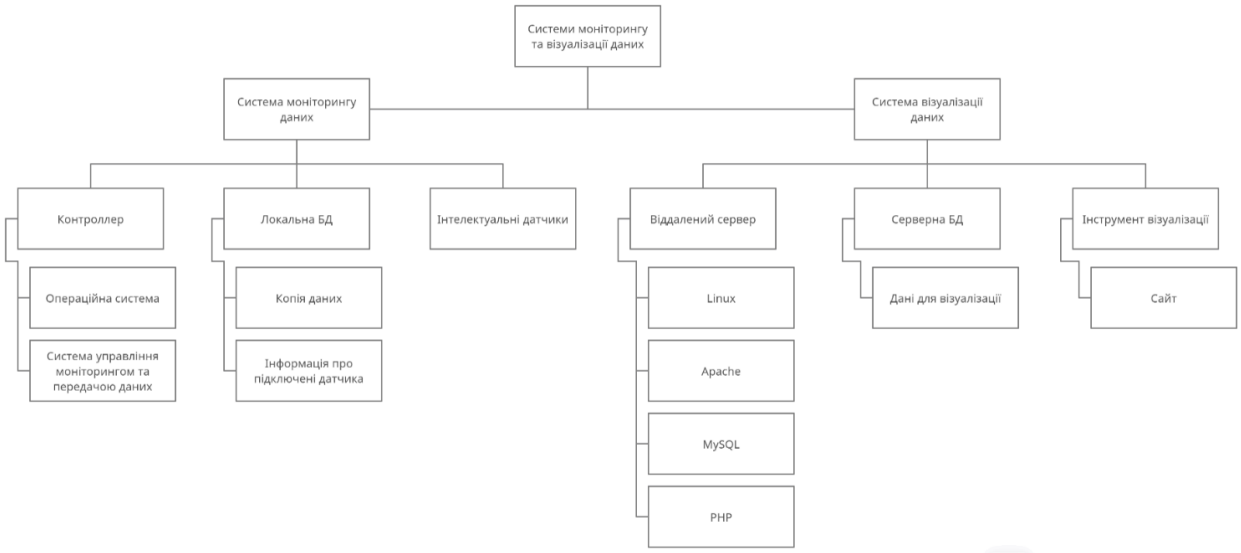


Рисунок 6.23 – Архітектура розроблюваної системи.

Наступним кроком є аналіз та вибір апаратних модулів для реалізації системи моніторингу та візуалізації даних для кіберфізичних виробничих систем. Одноплатний комп'ютер (SBC, single-board computer) представляє собою комп'ютер, на якому всі основні компоненти розміщені на одній платі. У більшості випадків одноплатні комп'ютери зустрічаються як промислові комп'ютери чи як вбудовані або комплексні офісні комп'ютери. Такий спосіб використання застосовується у системах, до яких застосовуються підвищені заходи безпеки, або у системах що повинні бути максимально компактними.

Особливістю одноплатного комп'ютера, на відміну від настільного персонального комп'ютера, є те, що подібні комп’ютери не покладаються на

слоти розширення для периферійних функцій. Побудова подібних комп'ютерів заснована на використанні широкого спектру мікропроцесорів. Використовуються такі комп’ютери у випадках простих конструкцій, побудовані комп'ютерними любителями, для яких часто використовують статичну оперативну пам'ять та недорогі 8-бітні або 16-бітні процесори. Інші типи, такі як блейд-сервери, працювали б аналогічно серверному комп'ютеру, лише у більш компактному форматі [25].

Зовнішній вигляд одноплатного комп'ютера Raspberry Pi 4 Model B представлено на рисунку 6.24.

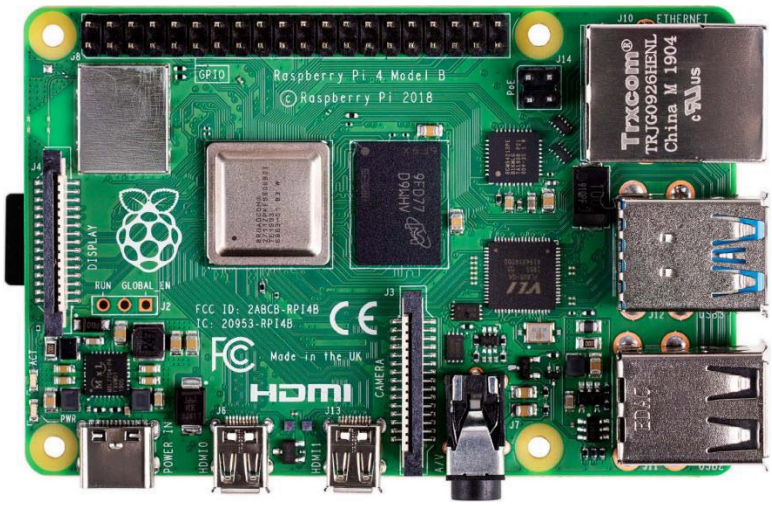


Рисунок 6.24 – Зовнішній вигляд одноплатного комп'ютера Raspberry Pi 4 Model B

Основні характеристика Raspberry Pi 4 Model B вказані у таблиці 6.1.

Таблиця 6.1 – Характеристика одноплатного комп'ютера Raspberry Pi 4 Model B [27]

Параметри	Характеристики
1	2
Процесор	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-біт SoC @ 1,5 ГГц
Оперативна пам’ять (RAM)	1-8Гб
Сховище eMMC	microCD

Продовження таблиці 6.1

1	2
GPIO	40
Мережеві можливості	Ethernet 10 / 100 / 1000 Wi-Fi 2,4 ГГц / 5,8 ГГц, 300 Мб/с
Живлення	5 В, 3 А
Частота процесора	1,5 ГГц
Робоча температура	від 0 °С до + 50 °С

Raspberry Pi 4 Model B в порівнянні з попереднім поколінням Raspberry PI3 Model B + отримала досить багато удосконалень, такі як підвищення швидкості роботи процесора, поліпшена продуктивність мультимедійного модуля, збільшення кількості і швидкості оперативної пам'яті і оновлені мережеві модулі, при цьому збережено рівень енергоспоживання і сумісність периферійних модулів. Крім того, ця плата отримала додаткове оновлення для багатьох елементів плати, серед яких підтримка одночасного підключення двох 4 К дисплеїв, 1 Гбіт/с Ethernet, 2 USB 3.0 порту.

Ключовими характеристиками даного продукту є новий, високопродуктивний 64-розрядний чотирьох ядерний процесор, підтримка двох дисплеїв з роздільною здатністю до 4 К через пару micro-HDMI портів, апаратне декодування відео, від 1 Гб до 8 Гб оперативної пам'яті, двохдіапазонна бездротова мережа на 2,4 ГГц та 5,0 ГГц, Bluetooth 5.0, Gigabit Ethernet, два порти USB 3.0 і PoE.

Для живлення моделі потрібний блок 5 В, 3 А з можливістю використання блоку живлення на 2,5 А при споживанні підключеної периферії не більше 500 мА сумарно.

Інтелектуальний датчик – це сенсорний пристрій, який здатний виконувати низку інтелектуальних функцій у рамках свого завдання чи обов'язку. Інтелектуальний датчик здатний до самотестування та пристосуватися самостійно, а також самоідентифікації. Ці датчики розуміють середовище, в яке вони потрапляють, і вони можуть керувати широким спектром умов. Інтелектуальний датчик здатний керувати своїми функціями

внаслідок подразника від зовнішніх функцій. Це показує, що інтелектуальний датчик має архітектуру вдосконаленого навчання, адаптації та обробки сигналів, все в одному інтегральному ланцюзі. Для інтелектуального датчика потрібен спеціалізований апарат, який називається схемою кондиціонування сигналу, щоб контролювати та керувати собою та іншими приладами .

Інтелектуальні датчики використовуються для моніторингу різних промислових процесів, збору даних, проведення вимірювань і надсилання даних на централізовані хмарні обчислювальні платформи, де інформація збирається та аналізується за закономірностями. Ці дані можуть контролювати ключові особи, які приймають рішення у будь-який час. Існує 4 основні типи інтелектуальних датчиків, які використовуються і забезпечують можливість впровадження Industry 4.0 на виробництвах.

Датчики рівня використовуються для вимірювання в режимі реального часу контейнерів, бункерів і резервуарів, передачі інформації в режимі реального часу в системі управління запасами та системи контролю процесів. Вони використовуються в усьому: від утилізації відходів до зрошення, вимірювання дизельного палива тощо.

Датчики температури є одними з найпоширеніших датчиків, що використовуються в промислових умовах. Часто використання подібних інтелектуальних датчиків потрібне для виявлення та запобігання перегріву, також для своєчасного інформування персоналу про необхідність технічного обслуговування або вимкнення.

Датчики тиску використовуються для моніторингу трубопроводів і сповіщення централізованої обчислювальної системи про витоки або порушення, які сповіщають керівників про необхідність технічного обслуговування та ремонту.

Інфрачервоні інтелектуальні датчики однаково багатоцільові та використовуються в дуже різних галузях. Вони використовуються в медицині для відстеження біологічних функцій, таких як кровотік під час операції,

вони використовуються в архітектурі, інженерії та будівництві для моніторингу витоків тепла в будівлях і промислових об’єктах.

В залежності від задачі підприємства щодо розроблюваної системи можливо підключити будь який з представлених типів датчиків. В рамках дослідження розроблюваної системи буде використовуватися датчик температур з додаванням Wi-Fi модуля 2,4 ГГц ESP-01s ESP8266 та Arduino Mega 2560 у якості контролера для інтелектуального датчики.

TMP35, TMP36 і TMP37 – низьковольтні прецизійні стоградусні датчики температури (рис 6.25).

Рисунок 6.25 – Зовнішній вигляд температурного датчика TMP36gz

Вони забезпечують вихідну напругу, лінійно пропорційну температурі за шкалою Цельсія. TMP35/TMP36/TMP37 не потребує будь-якого зовнішнього калібрування для забезпечення типової точності ± 1 °C при + 25 °C та ± 2 °C у діапазоні температур від – 40 °C до + 125 °C. Низький вихідний імпеданс TMP35/TMP36/TMP37, лінійний вихід та точне калібрування спрощують взаємодію зі схемою контролю температури та аналого-цифровими перетворювачами. Всі три пристрої призначені для роботи з однополярним живленням від 2,7 В до 5,5 В максимум. Струм живлення значно нижче 50 мкА, що забезпечує дуже низький самонагрів, менше 0,1 °C в нерухомому повітрі. Крім того, передбачено функцію відключення, що дозволяє скоротити струм живлення до рівня менше 0,5 мкА. TMP35 функціонально сумісний з LM35/LM45 та забезпечує вихідну напругу 250 мВ

при 25 °C. TMP35 зчитує температуру від 10 °C до 125 °C. TMP36 працює в діапазоні від - 40 °C до + 125 °C.

ESP8266 ESP-01s це модуль на основі 32-біт низькоспоживчого мікроконтролера з Wi-Fi модулем, інтегрованим TCP/IP стеком та управлінням AT-командами. Для програм передбачено 4 Мб SPI-flash. Модуль із високим ступенем інтеграції. Він споживає дуже мало енергії, забезпечуючи бездротове підключення до Інтернету для пристроїв, що вбудовуються. Він може бути легко інтегрований у пристрої з обмеженими ресурсами через його малий форм-фактор всього 24,5 мм × 14 мм. На рисунку 6.26 зображений зовнішній вигляд модуля.

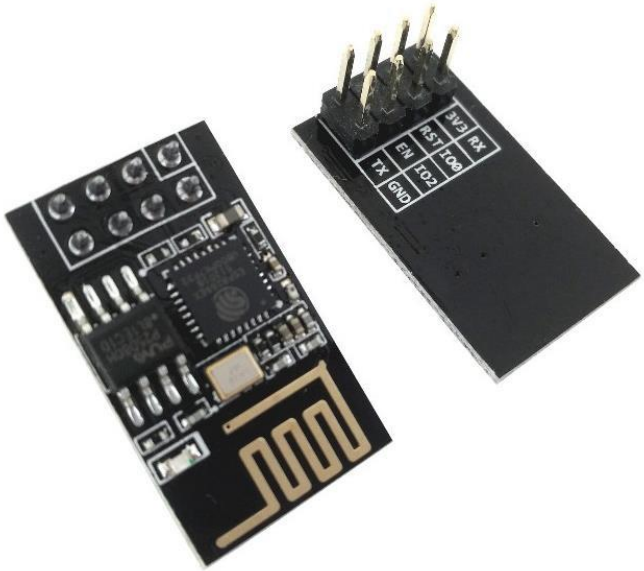


Рисунок 6.26 – Зовнішній вигляд Wi-Fi модуля ESP-01s ESP8266

Повний технічний опис модуля:

– підтримка Wi-Fi протоколів 802.11;

– Wi-Fi Direct (P2P), soft-AP;

– вбудований стек TCP/IP;

– вбудований TR перемикач, LNA, підсилювач потужності та відповідність мережі;

– вбудований PLL, регулятори, та система управління живленням;

– вихідна потужність +19,5 дБм у режимі 802.11b;

– SDIO 2.0, SPI, UART;

– STBC, 1×1 MIMO, 2×1 MIMO;

– пробудження та відправлення пакетів до 22 мс;

– споживання в режимі Standby до 1,0 мВт;

– розміри: 24,5 мм × 14 мм.

Варіанти застосування модуля:

– інтеграція до системи розумного будинку;

– домашня метеостанція з переглядом показів онлайн;

– облік показань лічильників води, електролічильників та перегляд показань онлайн;

– керована по Wi-Fi розетка, люстра або інші електроприлади;

– вбудовування у вимикачі з локальним та віддаленим керуванням.

Для подальшої побудови системи було розроблено загальний алгоритм системи. Розроблений алгоритм має загальний характер і охоплює всі етапи роботи системи. Узагальнений алгоритм роботи макету системи моніторингу та візуалізації даних для кіберфізичних виробничих систем представлено на рис. 6.27.

Рисунок 6.27 – Узагальнений алгоритм роботи макету системи моніторингу та візуалізації даних для кіберфізичних виробничих систем

У відповідності до узагальненого алгоритму роботи макету першим етапом алгоритму роботи буле підключення датчиків, обраних у другому розділі. Для виконання цього етапу дані про датчики будуть занесені до бази даних, розміщеній на контролері. Після чого використовуючи ці дані буде проводитися підключення датчиків через бездротове з’єднання.

Другий етап представляє собою перевірку підключення між контролером та датчиком шляхом відправки декількох сигналів до датчиків та відповідей на них. Крім того, перевірка стабільності Інтернет підключення.

API адреси так само, як і дані про датчики, зберігаються у локальній базі даних. На четвертому та п’ятому етапах ці адреси зчитуються з базі даних та перевіряються їх активність, тобто наявність доступу до збережених адресів. У разі успішного проходження всіх попередніх етапів починається роботи основної частини алгоритму роботи системи.

Процес збору даних з датчиків поділений на два етапі. Перший, це сам процес отримання необроблених даних з датчиків, а до другого етапу відноситься первинна обробка отриманих даних та збереження їх до локальної бази даних. Це необхідно, щоб у разі тимчасової відсутності з’єднання з віддаленими серверами дані не були втрачені.

Наступним етапом нові дані з бази даних зберігаються у форматі JSON (JavaScript Object Notation). Цей формат представляє собою текстовий формат обміну даними між комп'ютерами. Формат дає змогу описувати об'єкти та інші структури даних. Використовується переважно для передачі структурованої інформації через мережу [33]. Після чого отриманий набір даних відправляється за вказаними у базі даних API адресами.

На етапі отримання даних сервером пакет даних, отриманий з контролера, розшифровується та зберігається у базу даних сервера з точним часом, коли ці дані були зафіксовані датчиками.

Останнім етапом є зчитування даних з бази даних та відображення їх на пристрої користувача за допомогою веб-сайту, основними інструментами

для чого буде фреймворк для PHP Laravel, який буде відповідати за обробку та зберігання даних, та фреймворк для JavaScript Vue.js, який буде відповідати за візуальну частину веб-сайту та можливість динамічного відображення даних без необхідності перезавантаження сторінки.

На першому етапі було створено програму для підключення датчиків для отримання даних з них та модуль для бездротової передачі даних. У даній програмі для сумісної роботи всіх модулів було обрано метод передачі даних через UART. Для цього були підключені відповідні піни RX та TX, та задано швидкість передачі даних. Після чого проведено підключення Wi-Fi модуля до Wi-Fi. За допомогою створеної функції getTemp() отримуються дані з датчика та перетворюються їх у цілочисельне значення температури. Після чого створюється масив значень у форматі JSON. Наступними командами відбувається підключення модуля до віддаленого серверу. У разі успішного виконання підключення відбувається побудова HTTP POST запиту, за допомогою якого дані передаються до наступного модуля системи, а саме модуля візуалізації. Після чого відбувається закриття з’єднання та програма повертається на етап отримання даних з датчика (рис 6.28).

```
10
11  void setup() {
12    Serial.begin(115200);
13    Serial1.begin(115200);
14    sendCommand("AT", 5, "OK");
15    sendCommand("AT+CWMODE=1", 5, "OK");
16    sendCommand("AT+CWJAP=\""+ AP +"\",\""+ PASS +"\"", 20, "OK");
17  }
18
19  void loop() {
20    if (Serial.available()) { Serial1.write(Serial.read()); }
21    getTemp();
22    valSensor = temp;
23    String jSON = "{\"sensor_id\":\"1\",\"value\":\""+ String(temp) +"\"}";
24    unsigned int l = String(temp).length();
25    unsigned int i = 200- (190 + l);
26    sendCommand("AT+CIPMUX=0",5,"OK");
27    sendCommand("AT+CIPSTART=\"TCP\",\""+ HOST +"\","+ PORT,16,"OK");
28    sendCommand("AT+CIPSEND=200", 4,">");
29    Serial1.println("POST /api/sensors-data-store HTTP/1.1");
30    Serial1.println("Host: "+ HOST +":"+ PORT +"");
31    Serial1.println("User-Agent: insomnia/2022.6.0");
32    Serial1.println("Content-Type: application/json");
33    Serial1.println("Accept: */*");
34    Serial1.println("Content-Length: 35");
35    Serial1.println("");
36    Serial1.println(jSON);
37    while(i > 0) {
38      Serial1.println(""); i--;
39    }
40    delay(4000);
41    countTrueCommand++;
42    sendCommand("AT+CIPCLOSE=0",5,"OK");
43  }
44
45  void getTemp(){
46    int reading = analogRead(A5);
47    float voltage = reading * 5.0;
48    voltage /= 1024.0;
49    Serial.print(voltage); Serial.println(" volts");
50    float temperatureC = (voltage - 0.5) * 100 ;
51    Serial.println("Temperature = " + String(temperatureC));
52    Serial.println("Volts = " + String(voltage));
53    temp =  temperatureC + 0.5;
54  }
```

Рисунок 6.28 – Код отримання даних з датчика та відправки до системи

Створення API точок. На цьому етапі були створені всі головні точки доступу до системи. Вони поділяються на точки для роботи з процесами, сенсорами та окремо даними з датчиків. В ході роботи були створені точки для створення, редагування, показу та видалення даних з бази даних за допомогою них. На рисунку 6.29 зображено код оголошення всіх точок.

```php
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\Api\v1\ProcessesController;
use App\Http\Controllers\Api\v1\SensorsController;
use App\Http\Controllers\Api\v1\SensorsDataStoreController;

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| is assigned the "api" middleware group. Enjoy building your API!
|
*/

Route::get('process/{id}/{range}/range', [ProcessesController::class, 'showSensorsRangeByProcessId'])->name('process.showSensorsRangeByProcessId');
Route::get('process/{id}/sensors', [ProcessesController::class, 'getSensorsById'])->name('process.getSensorsById');
Route::resource('process', ProcessesController::class);
Route::get('sensor/{id}/{range}/range', [SensorsController::class, 'showRangeById'])->name('sensors.showRangeById');
Route::resource('sensor', SensorsController::class);
Route::get('sensors-data-store/{range}/range', [SensorsDataStoreController::class, 'showRange'])->name('sensors-data-store.showRange');
Route::resource('sensors-data-store', SensorsDataStoreController::class);
```

Рисунок 6.29 – Код оголошення API точок системи

Функція store необхідна для отримання даних, відправлених через API. У цій функції відбувається валідація отриманих даних, в результаті якої у разі не проходження відправляється інформація про помилку та помилковий статус, а у разі успішного проходження валідації відбувається збереження даних та відправлення відповіді з позитивним статусом. На рисунку 6.30 зображено код функції store.

```php
    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $validator = Validator::make(
            $request->all(), [
                'sensor_id' => ['required'],
                'value' => ['required'],
            ]
        );

        if ($validator->fails()){
            return [
                'status' => false,
                'errors' => $validator->messages()
            ];
        }

        $sensorsDataStore = SensorsDataStore::create([
            'sensor_id' => $request->sensor_id,
            'value' => $request->value,
        ]);

        return [
            'status' => true,
            'process' => $sensorsDataStore
        ];
    }
```

Рисунок 6.30 – Код функції store

Логіка цієї функції включає в себе запит на отримання даних з бази даних згідно з параметрами вказаними у API точці. До цих параметрів входить унікальний ідентифікатор процесу, на якому ці дані отримані, а також кількість значень, які необхідно отримати. Сам запит складається з декількох частин, а саме, отримання всіх сенсорів, що використовуються у процесі, після чого згідно їх унікального ідентифікатора отриманя необхідної кількості значень для кожного датчика. На рисунку 6.31 зображено код функції showSensorsRangeByProcessId.

```php
/**
 * Display the range of resource by sensor.
 *
 * @param  int  $id
 * @param  int  $range
 * @return \Illuminate\Http\Response
 */
public function showSensorsRangeByProcessId($id, $range)
{
    $sensors = Process::findOrFail($id)->processSensors()->get();
    $dataList = collect([]);
    foreach ($sensors as $sensor) {
        $dataList = $dataList->merge( [
            [
                $sensor->name, Sensor::findOrFail($sensor->id)->sensorData()->latest()->take($range)->get()
            ]


        ]);
    }
    return $dataList;
    //return Process::findOrFail($id)->sensorsData()->latest()->take($range)->get();
}
```

Рисунок 6.31 – Код для отримання даних датчиків по одному процесу

Функція візуалізації даних представляє собою скрипт для отримання і виводу даних та є основною для системи візуалізації, складається з декількох простіших за своїм призначенням методів.

Метод loadChartData виконує функцію отримання даних з системи через запит до спеціальної api адреси, у якому необхідно вказати кількість значень, що необхідно отримати та id процесу, дані з якого потрібно візуалізувати.

Метод createChart перетворює отримані дані у лінійну діаграму, яка відображається у системі. На рисунку 6.32 зображено код всієї функції

```
<script>
    import axios from 'axios';

    export default {
        name: 'Index',
        data() {
            return {
                range: 10,
                process_id: 1,
                timer: '',
                chartData: []
            }
        },
        created () {
            this.fetchEventsList();
            this.timer = setInterval(this.fetchEventsList, timeout: 5000);
        },
        methods: {
            fetchEventsList () {
                console.log('load');
                this.loadChartData();
                this.createChart();
            },
            loadChartData () {
                axios.get('/api/process/'+ this.process_id +'/' + this.range + '/range')
                .then(res => {
                    this.chartData = res.data;
                })
            },
            createChart () {
                var chartsData = [];
                for (let k = 0; k < this.chartData.length; k++){
                    let chartLine = [];
                    for (let i = 0; i < this.chartData[k][1].length; i++){
                        chartLine.push(this.chartData[k][1][i].value)
                    }
                    chartLine.reverse()
                    chartLine.unshift(this.chartData[k][0])
                    chartsData.push(chartLine)
                }

                bb.generate( config: {
                    data: {
                        columns: chartsData,
                        type: "line",
                    },
                    bindto: "#lineChart"
                });
            },
        }
    }
</script>
```

Рисунок 6.32– Код функції візуалізації даних.

Розробка логічної та фізичної моделі бази даних. Логічна модель даних, або логічна схема, представляє собою модель даних для предметної області, створюється незалежно від конкретного продукту керування базами

даних або технології зберігання, але в термінах структур даних, таких як реляційні таблиці та колонки, об'єктно-орієнтовані класи чи теги XML. Вона є протилежністю концептуальній моделі даних, яка описує семантику організації без посилання на технологію.

В ході роботи над системою було розроблено комплекс баз даних, яка складається з двох баз даних. Одна з яких буде встановлена на Raspberry Pi 4, як система управління та друга, яка буде встановлена на віддаленому сервері та буде взаємодіяти з системою візуалізації. Взаємодія між частинами комплексу буде відбуватися через REST API з конвертацією даних у JSON формат для передачі їх між частинами системи. На рисунку 6.33 зображено комплексну логічну модель баз даних розроблюваної системи.

Рисунок 6.33 – Комплексна логічна модель баз даних розроблюваної системи

Логічну модель бази даних для Raspberry Pi 4 можна побачити на рисунку 6.34. До даної моделі входять декілька таблиць, які відповідають за інформацію про датчики, процеси на підприємстві, на яких потрібні датчики, інформація про необхідні API адреси, на які будуть відправлятися дані для візуалізації, та загальна таблиця з даними з датчиків, крім того, до кожного запису з датчика додається дані про процес, на якому дані було отримано, та датчик, який отримав ці данні.

Рисунок 6.34 – Логічна модель бази даних для Raspberry Pi 4.

Логічну модель бази даних для віддаленого серверу можна побачити на рисунку 6.35. Дана модель є похідною від моделі для системи управління, але в ній є деякі відмінності. Для коректної візуалізації інформації було додано таблицю з даними про припустимі значення для того чи іншого датчика на конкретному процесі. Ці данні дозволять розроблювати більш гнучку систему візуалізації з можливістю використовувати ті самі датчики на різних процесах, навіть, якщо допустимі значення в них сильно відрізняються. Крім того, ця модель не потребує таблиці зі списком API адрес.

Рисунок 6.35 – Логічна модель бази даних для віддаленого серверу.

Фізична модель даних представляє собою спосіб подання дизайну даних як вже реалізованого, чи призначеного для реалізації, у системі керування базами даних. У більшості життєвих циклів проєктів подібні моделі походять від логічних моделей даних, але в деяких випадках може бути зворотно розроблена з даної реалізації бази даних. Завершена фізична модель даних включатиме всі артефакти бази даних, необхідні для створення відношень між таблицями чи для досягнення мети продуктивності, наприклад, індексів, визначень обмежень, зв'язаних і секціонованих таблиць або кластерів.

На рисунку 6.36 зображено комплексну фізичну модель баз даних розроблюваної системи.

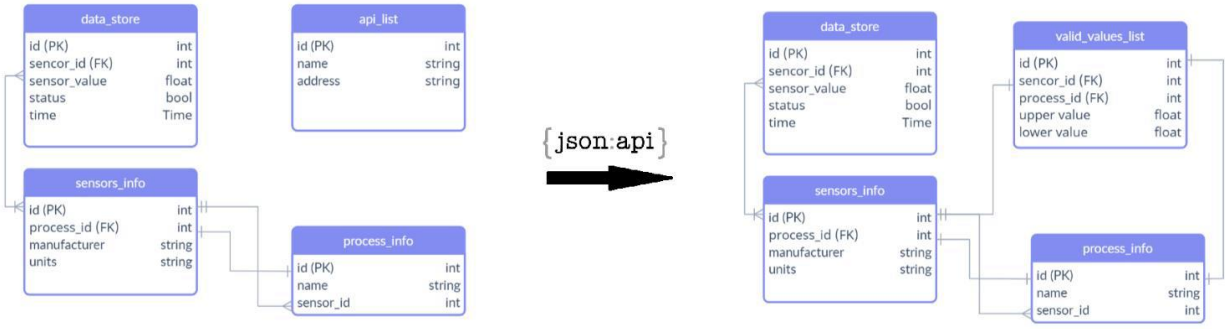


Рисунок 6.36 – Комплексна фізична модель баз даних для розроблюваної системи

Фізичну модель бази даних для Raspberry Pi 4 можна побачити на рисунку 6.37.

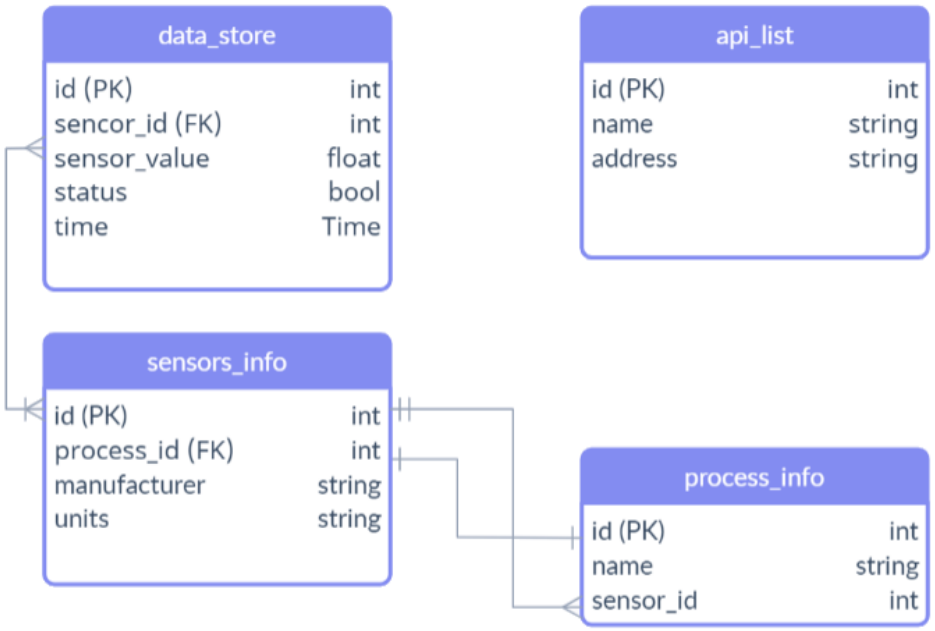


Рисунок 6.37 – Фізична модель бази даних для Raspberry Pi 4

Фізичну модель бази даних для віддаленого серверу можна побачити на рисунку 6.38

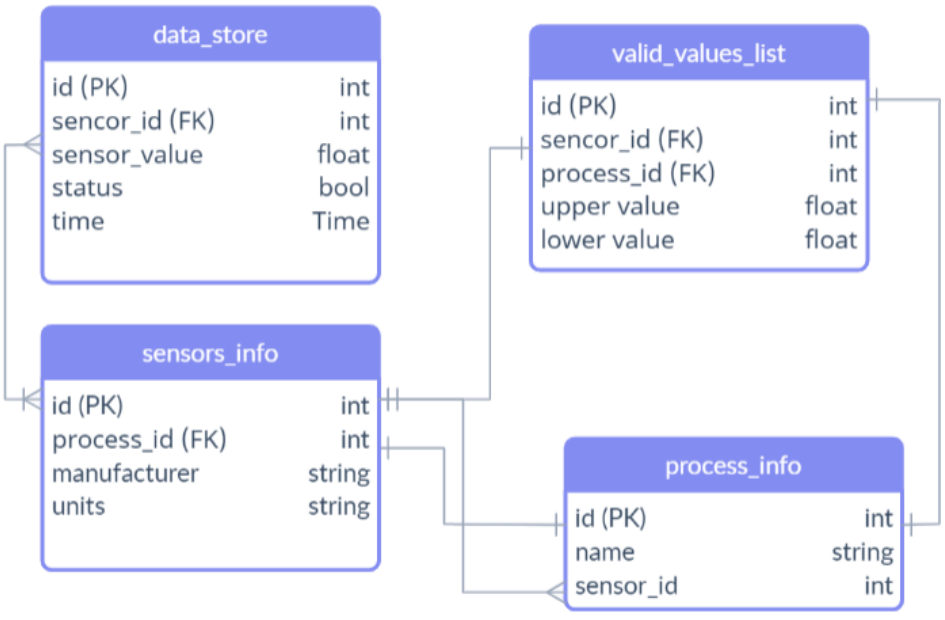


Рисунок 6.38 – Фізична модель бази даних для віддаленого серверу

На рисунку 6.39 зображено всі класи для створення таблиць бази даних у системі. Всі класи складаються з двох методів: up, що відповідає за

створення чи модифікацію таблиці, та down, що виконується у випадку видалення таблиці. Таблиця створюється за допомогою статичного метода create, у параметри якого вписується назва таблиці та безіменна функція, у якій виконується ініціювання всіх полів таблиць та зв'язки між ними. Метод id вказує на унікальний ідентифікатор запису, string та integer створюють поля з відповідними типами даних, а unsignedBigInteger вказує на поле, яке буде використовуватися для зовнішнього ключа. Для коректного підключення зовнішнього ключа також потрібно додати функцію foreign, у якій вказуються взаємозв’язок між таблицями та конкретними полями.

```php
class CreateProcessesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('processes', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('processes');
    }
}
```

```php
class CreateSensorsDataStoresTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('sensors_data_stores', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('sensor_id');
            $table->integer('value');
            $table->timestamps();

            $table->foreign('sensor_id')->references('id')->on('sensors');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('sensors_data_stores');
    }
}
```

```php
class CreateSensorsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('sensors', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('process_id');
            $table->string('name');
            $table->string('unit');
            $table->timestamps();

            $table->foreign('process_id')->references('id')->on('processes');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('sensors');
    }
}
```

```php
class CreateValidValuesListTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('valid_values_list', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('process_id');
            $table->unsignedBigInteger('sensor_id');
            $table->integer('upper_value');
            $table->integer('lower_value');
            $table->timestamps();

            $table->foreign('process_id')->references('id')->on('processes');
            $table->foreign('sensor_id')->references('id')->on('sensors');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('valid_values_list');
    }
}
```

Рисунок 6.39 – Класи міграцій таблиць процесів, датчиків, даних з датчиків та припустимих значень

Для даної системи буде реалізовано клієнт-серверну архітектуру на базі фреймворків Laravel та Vue JS з можливістю користувача звертатися до тонкого клієнта для отримання даних з датчиків у реальному часі.

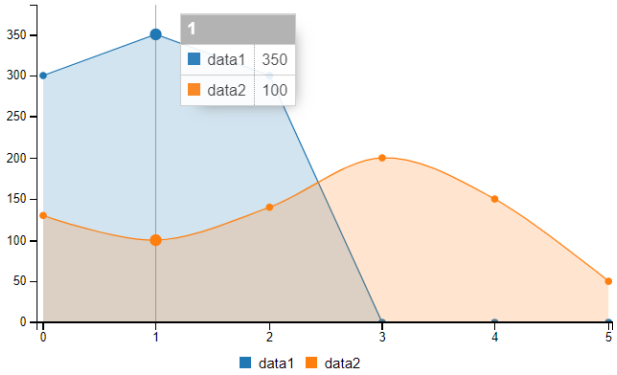
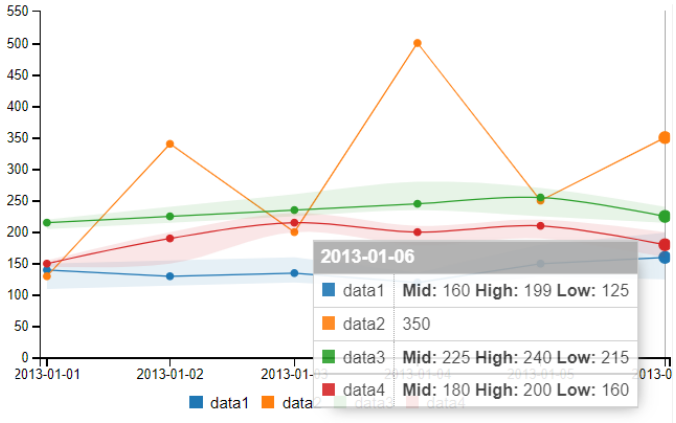
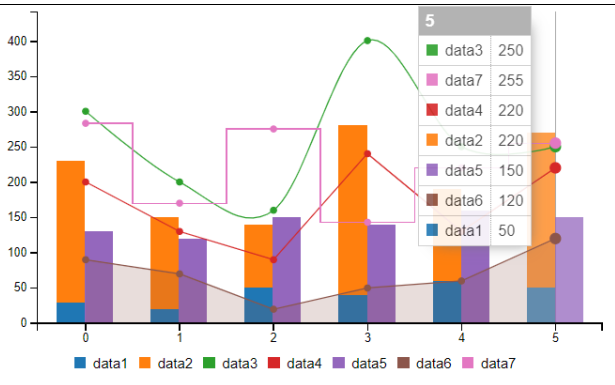
“Клієнт — сервер” (client–server) мережева архітектура, основною ідеєю якої є розподілення завдань чи мережевого навантаження між постачальниками послуг, званими серверами, та замовниками послуг, званими клієнтами. У більшості випадків програми розташовані на різних обчислювальних машинах і взаємодіють між собою через обчислювальну мережу за допомогою мережевих протоколів, але можуть бути розташовані також і на одній машині. Програми-сервери очікують від клієнтських програм запити та надають їм свої ресурси у вигляді даних або у вигляді сервісних функцій. Однак програма-сервер може виконувати запити від багатьох програм-клієнтів, її розміщують на спеціально виділеній обчислювальній машині, налаштованій особливим чином, як правило, спільно з іншими програмами-серверами, тому продуктивність цієї машини повинна бути високою. Через особливу роль такої машини в мережі, специфіки її обладнання та програмного забезпечення її також називають сервером, а машини, що виконують клієнтські програми, відповідно, клієнтами [34].

Тонкий клієнт – це у більшості випадків низькопродуктивний комп’ютер, оптимізований для встановлення віддаленого з’єднання з обчислювальним середовищем на основі сервера. Також може називатися мережевими комп'ютерами або нульовими клієнтами. Сервер виконує більшу частину роботи, яка може включати запуск програм, виконання обчислень і зберігання даних [35].

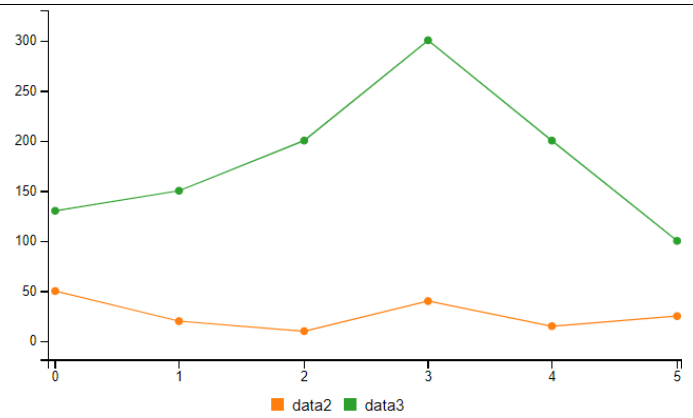
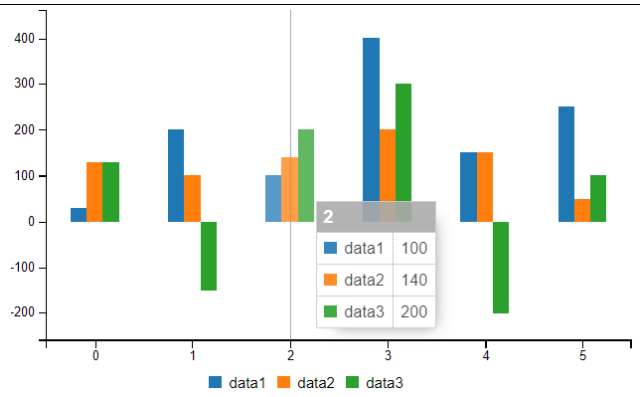
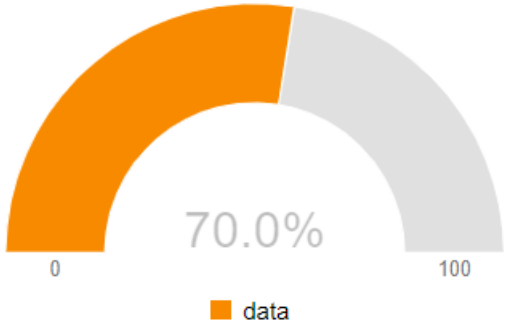
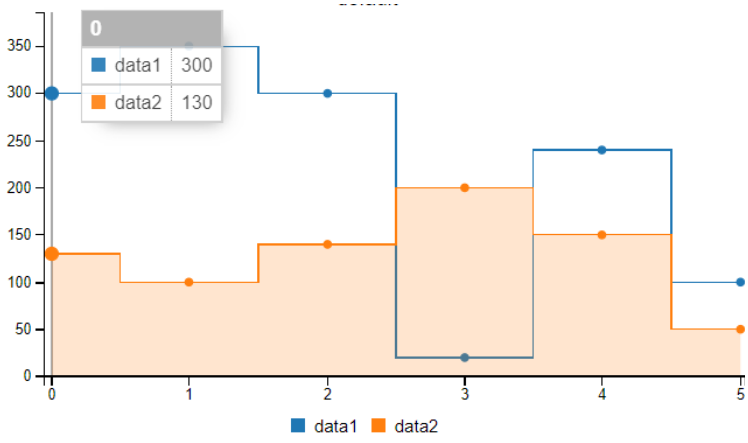
Візуалізація інформації у даній системі буде виконуватися за допомогою бібліотеки “billboard.js” на базі фреймворку Vue JS. Ця бібліотека була обрана, бо вона дає багато інструментів, які забезпечують адаптивний та інтерактивний вид готової системи. У таблиці 6.2 зображені основі види візуалізації інформації з бібліотеки, які можуть бути використані у

розроблюваній системі. Окрім лінійних діаграм є можливість використання секторних та кільцевих діаграми, гістограм, діаграм з областями, точкових та бульбашкових діаграм, поверхневих діаграм, пелюсткових діаграми та інших. Подібний широкий набір інструментів візуалізації дозволяє налаштувати системи під потреби більшості підприємств.

Таблиця 6.2 – Основі види візуалізації інформації з бібліотеки billboard.js [36]

Опис діаграм	Зображення діаграм
1	2
Діаграми з областями, використовуються для зображення зміни кількісних значень в певному інтервалі або за певний період часу. Найчастіше ці діаграми використовують для демонстрації тенденцій, а не конкретних значень.	
Діаграми площин – це лінійні діаграми, що відображають діапазон між нижчим і вищим значенням для кожної точки. Діаграми площин зазвичай використовуються для візуалізації діапазону, який змінюється з часом.	
Комбінована діаграма порівнює дані в кількох різних категоріях за певний період часу. Загалом існує два або три типи діаграм, об'єднаних разом, щоб показати зв'язок між точками даних. Як правило, гістограма та лінійна діаграма використовуються разом.	

Продовження таблиці 6.2

1	2
Лінійна діаграма відображає інформацію як серію точок даних, з'єднаних відрізками. Точки впорядковують за однією віссю. Лінійна діаграма часто використовується для візуалізації тенденції в даних через інтервали часу.	
Стовпчикова діаграма — це графік, який представляє згруповані дані за допомогою прямокутних стовпців, довжини яких пропорційні значенням, які вони представляють.	
Циферблатні діаграми, використовуються для відображення інформації як показання на циферблаті. Цей тип діаграми часто використовується у звітах на інформаційній панелі для відображення ключових індикаторів системи.	
Покрокова діаграма – це лінійна діаграма, яка не використовує найкоротшу відстань для з'єднання двох точок даних. Замість цього використовуються вертикальні та горизонтальні лінії для утворення ступінчастої прогресії. Вертикальні частини покрокової діаграми позначають зміни в даних та їх величину.	

Розроблюваний макет системи складається з двох основних модулів, а саме, модуля моніторингу та візуалізації. Модуль моніторингу представляє собою інтелектуальний датчик на основі температурного датчика TMP36gz, Arduino Mega 2560 та з Wi-Fi модулем ESP8266 ESP-13. Схема підключення представлена на рисунку 6.40.

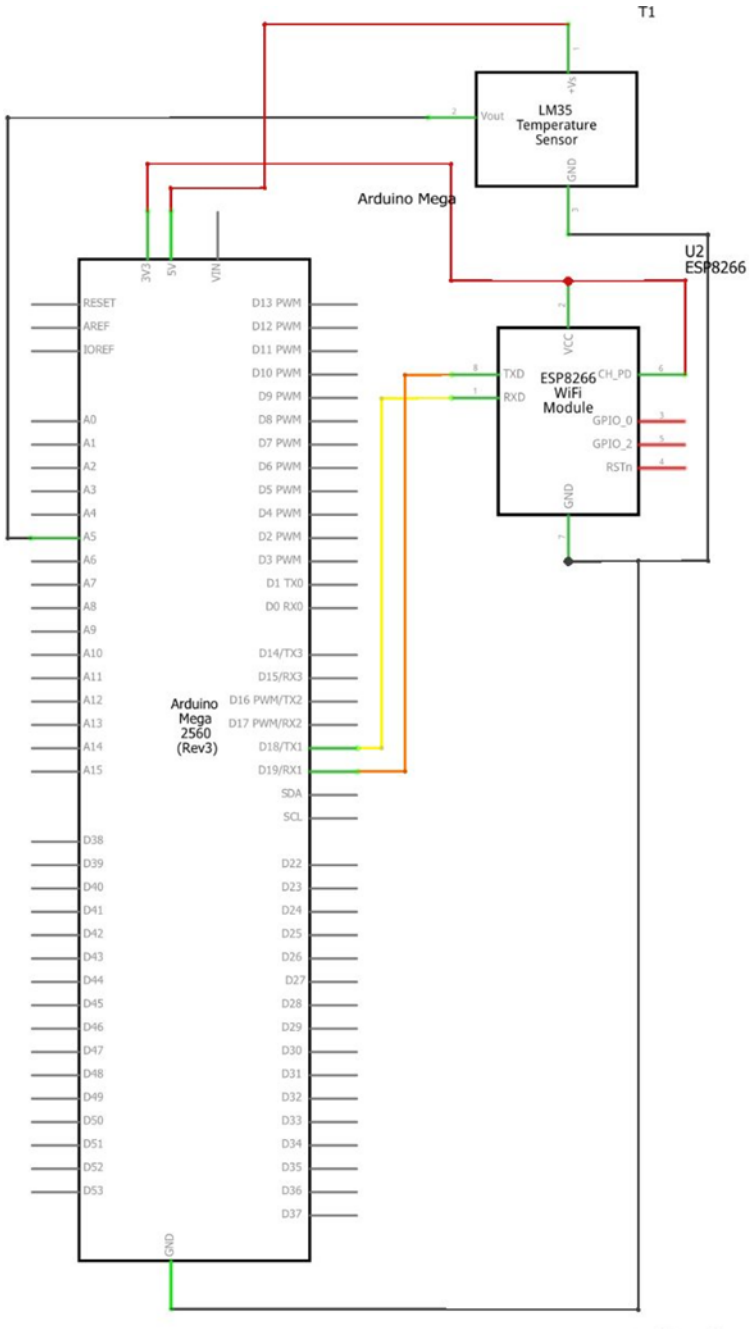


Рисунок 6.40 – Схема підключення модуля моніторингу

На рисунку 6.41 представлено схематичне підключення компонентів модуля.

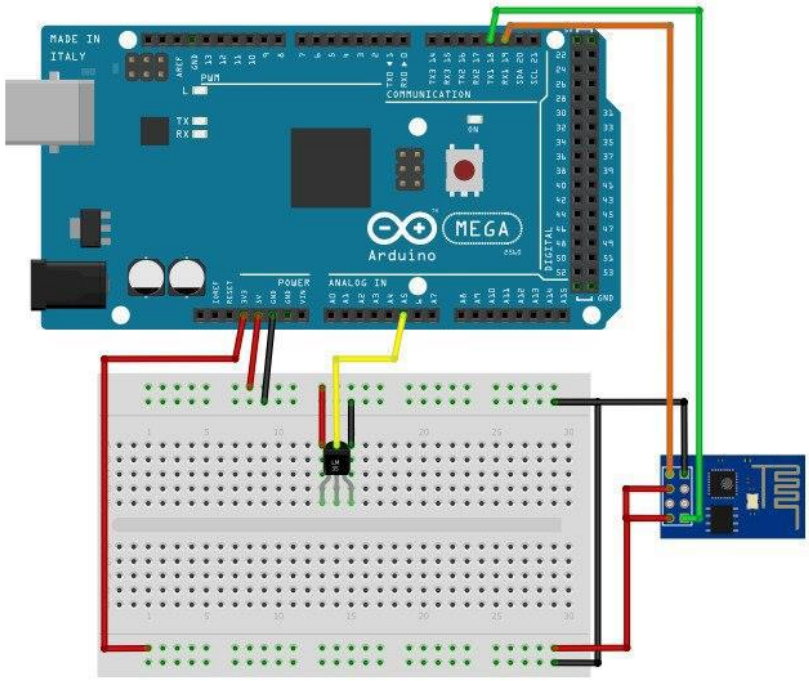


Рисунок 6.41 – Схематичне підключення компонентів модуля моніторингу

На рисунку 6.42 представлено зовнішній вигляд модуля.

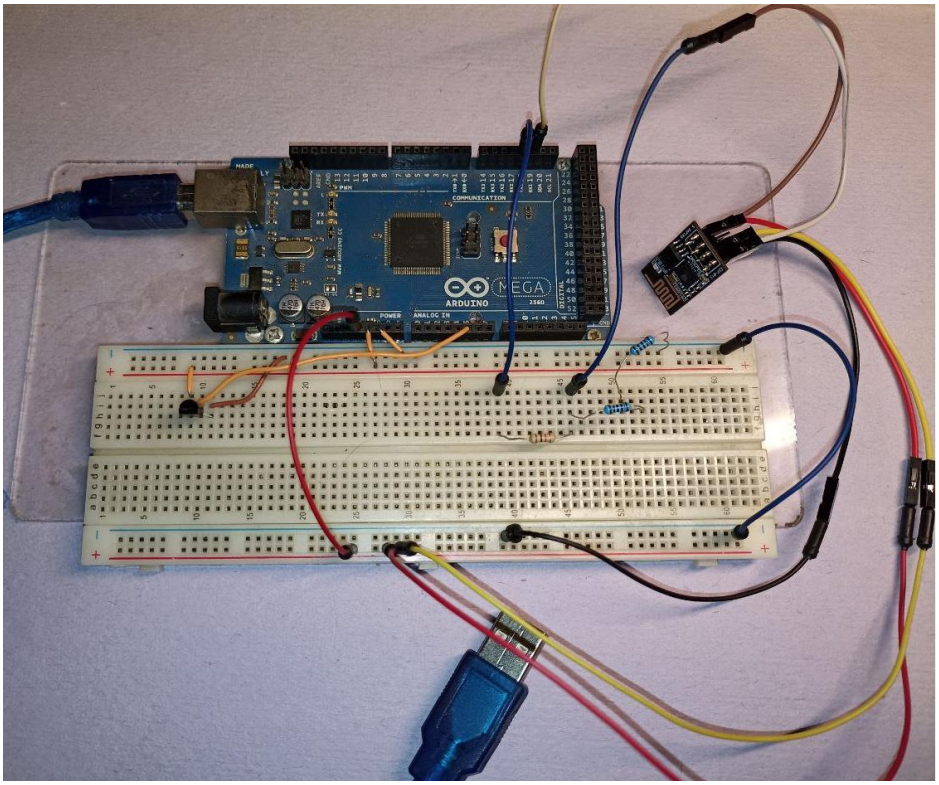


Рисунок 6.42 – Зовнішній вигляд модуля моніторингу

Модуль візуалізації представляє собою одноплатний комп'ютер Raspberry Pi 4 Model B, на який було встановлено операційну систему Linux, а саме дистрибутив Debian, сервер Apache2, базу даних MySQL та мову програмування PHP. Для перевірки коректності роботи модуля до нього також було підключено периферійні пристрої, а саме, дисплей, клавіатура та миша. Загальний вид Raspberry Pi 4 Model B представлений на рис. 6.42.

Рисунок 6.42 – Зовнішній вигляд модуля візуалізації

Після збирання та підключення усіх елементів системи було проведено експеримент. Завданням експерименту є перевірка швидкості передачі інформації в часі в залежності від відстані від датчика до сервера. Ця задача обумовлена модулюванням ситуації на підприємстві, коли датчики знаходяться на відстані від контролера та потрібно передавати дані до нього з мінімальною затримкою. Тобто у експерименті було перевірено швидкість реакції HMI на надходження інформації після спрацювання датчика. У якості інструменту візуалізації даних було використано лінійну діаграму, яка демонструє зміну значень датчика у реальному часі. На рис. 6.44 представлено вигляд розробленого HMI.

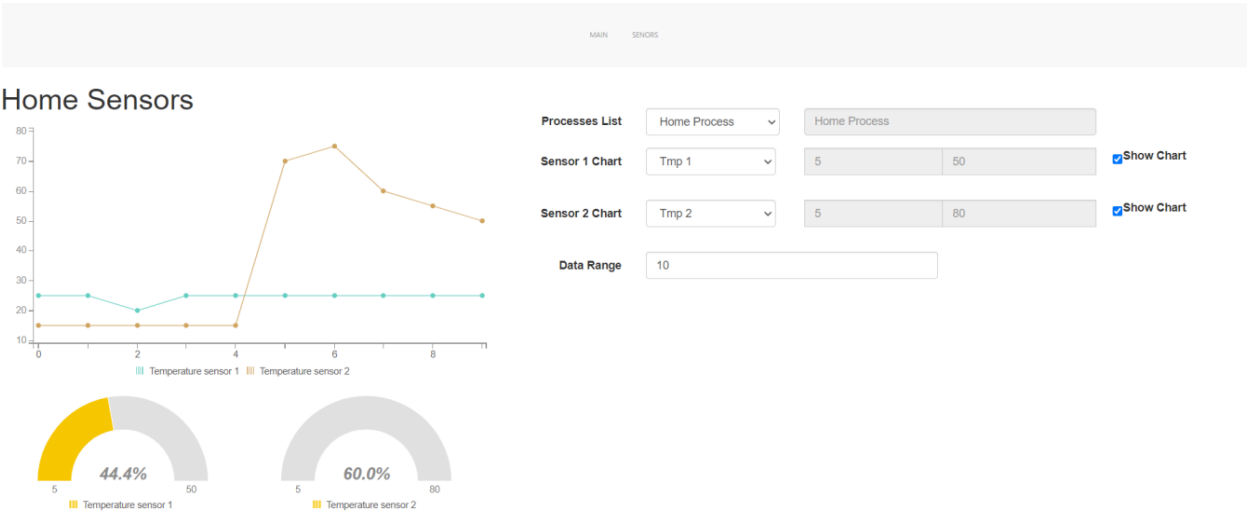


Рисунок 6.44 – Вигляд розробленого HMI

ВИСНОВКИ

У монографії наведено результати розробки нових моделей та методів кіберфізичних виробничих систем в концепції Industty 4.0. Аналіз розвитку сучасних технологій організації виробництва високтехнологічних виробів показав гостру необхідність створення нових підходів до їх струкутри і реалізації. Створення майбутніх технологій здійснюється на цифрових, інтелектуальних, підключених і автономних фабриках і виробничих мережах, що працюють в режимі реального часу. Для досягнення цих цілей протягом останніх 6 років у всьому світі, активно розроблюються, впроваджуються та використовуються CPPS. Використання CPPS дозволяє скоротити собівартість виробництва і збільшити економічну привабливість, а також домогтися мінімізації втрат з точки зору концепту LP.

Однак варто зауважити, що розробка нових або вдосконалення існуючих CPPS, залежать від їх архітектури, типу і виду обладнання, параметрів ТПВ виробів на всіх етпапах їх ЖЦ, які впливають на структуру інформаційних потоків, їх HMI в кібернетичній складовій і реалізацію зворотного зв'язку. Результати аналізу показали, що на даний момент часу існують етанлонні архітектури RAMI 4.0, 5C, 8C, CPPS-IoE на базі архітектури 5C, які несуть декларативно-рекомендаційний характер, що недозволяє розробнику CPPS оцінити складність, обсяг, врахувати всі елементи необхідні і достатні для успішної реалізації проекту.

В результаті проведених наукових дослідження були отримані такі наукові і практичні результати.

1. Проведено критичний аналіз існуючих архітектур, методів та моделей керування процесами в складних виробничих об'єктах та виявлено, що найбільш перспективним в рамках концепції Industry 4.0 є використання кіберфізичних систем. Встановленно основні протиріччя, що дозволили визначити наукову проблему.

2. Вперше розроблено архітектурно-логічну модель представлення керування процесами в складних виробничих об'єктах на базі кіберфізичних систем, яка базується на науково обґрунтованих теоріях мультисистем і моносистем та методах формалізованого представлення систем, що дозволило об'єднати стратегічні, фізичні та кібернетичні складові системи у єдиний інформаційний простір.

3. Вперше розроблено логічно узгоджені послідовності взаємопов'язаних методів прийняття рішень на кожному етапі архітектурно-логічної моделі, що дозволило реалізувати технологію «Digital Twins» та самоадаптацію елементів кібер-фізичних виробничих систем керування.

4. Вперше запропоновано технологію розробки кіберфізичних виробничих систем, яка реалізована на базі теоретико-множинного представлення інформаційних блоків за кожним етапом і рівнем архітектурно-логічної моделі, а також методах їх структуризації, що дозволило реалізувати гнучкість керування процесами в виробничих об'єктах.

5. Удосконалено метод подання структурних системних моделей кіберфізичного керування процесами в виробничих об'єктах, що дозволило формалізувати алгоритм функціонування на базі теорії апарату регулярних схем і алгоритмічних алгебр та побудувати структурні і подієві моделі функціонування кіберфізичних виробничих систем.

6. Удосконалено метод синтезу алгоритмів функціонування кіберфізичних виробничих систем, що дозволило мінімізувати кількість операторів і спростити структуру системи функціонування виробничого об'єкту.

7. Удосконалено модель життєвого циклу керування виробничим об'єктом на базі кіберфізичних виробничих систем, що дозволило автоматизувати процес розробки кібернетичної складової на основі синтезованих блоків функціонування.

8. Вперше розроблено модель формалізації кібернетичної складової виробничого об'єкта у вигляді взаємопов'язаних багаторівневих GUI елементів, що дозволило автоматизувати реалізацію функцій відповідно до вимог HMI кіберфізичних виробничих систем.

9. Вперше розроблено математичний опис зв'язків між GUI як основних елементів HMI, що дозволило реалізувати структурне подання кібернетичної складової виробничого об'єкта за рахунок реалізації подій GUI елементів у вигляді фрагментів програмного коду.

10. Отримала подальший розвиток методологія Константайна, на базі якої запропоновано метод графічного представлення конструкції кібернетичної складової виробничого об'єкта, що дозволило досягнути редукцію розробки структури.

11. Вперше розроблено синтаксичну і семантичну моделі декларативної мови визначення і опису моделювання кібернетичної складової для керування процесами в складних виробничих об'єктах на базі кіберфізичних виробничих систем за розширеною формою Бекуса-Наура, що дозволило істотно спростити процес керування виробничим об'єктом.

Наукове значення роботи полягає у подальшому розвитку теорії керування складними організаційно-технічними об'єктами та комплексами на базі кіберфізичних виробничих систем, в рамках концепції Industry 4.0 та Smart Factory.

Отримані наукові результати в даному дослідженні дозволяють автоматизувати процес управління розробкою або удосконалення існуючих CPPS, які будуть відповідати вимогою технології «Digital Twins», що дозволить реалізувати концепції LP і Industry 4.0.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ercan Oztemel, Samet Gursev Literature review of Industry 4.0 and related technologies// Journal of Intelligent Manufacturing January 2020, Volume 31, Issue 1, pp 127–182 (DOI https://doi.org/10.1007/s10845-018-1433-8)

2. Alejandro GermánFrank, Lucas Santos Dalenogare, Néstor Fabián Ayala Industry 4.0 technologies: Implementation patterns in manufacturing companies // International Journal of Production Economics Volume 210, April 2019, Pages 15-26 (https://doi.org/10.1016/j.ijpe.2019.01.004)

3. Muhammad Haseeb, Hafezali Iqbal Hussain 1O, Beata Ślusarczyk,Kittisak Jermsittiparsert Industry 4.0: A Solution towards Technology Challenges of Sustainable Business Performance // Social Sciences 2019, 8(5), 154 (https://doi.org/10.3390/socsci8050154)

4. V.Alcácera, V.Cruz-Machado Scanning the Industry 4.0: A Literature Review on Technologies for Manufacturing Systems // Engineering Science and Technology, an International Journal Volume 22, Issue 3, June 2019, Pages 899-919 (https://doi.org/10.1016/j.jestch.2019.01.006)

5. Y. Lu Industry 4.0: A survey on technologies, applications and open research issues J. Ind. Information Integr. 6 (2017), pp. 1-10 (https://doi.org/10.1016/j.jii.2017.04.005)

6. Abid Haleem, Mohd Javaid Additive Manufacturing applications in Industry 4.0: A review // Journal of Industrial Integration and Management ; (https://doi.org/10.1142/S2424862219300011)

7. Daniel Alejandro Rossit, Fernando Tohmé, Mariano Frutos Industry 4.0: Smart Scheduling, International Journal of Production Research, Vol. 57, No. 12, 2019, pp.3802-3813 (https://doi.org/10.1080/00207543.2018.1504248)

8. Giovanna Culot, Guido Nassimbeni, Guido Orzes, Marco Sartora Behind the definition of Industry 4.0: Analysis and open questions // International Journal

of Production Economics Available online 10 January 2020 (https://doi.org/10.1016/j.ijpe.2020.107617)

9. Isabel Castelo-Branco, Frederico Cruz-Jesus, Tiago Oliveira Assessing Industry 4.0 readiness in manufacturing: Evidence for the European Union //Computers in Industry Volume 107, May 2019, Pages 22-32 (https://doi.org/10.1016/j.compind.2019.01.007)

10. Federal Ministry of Education and Research Germany (BMBF). The new High Tech Strategy - Innovations for Germany. Berlin. 2014.

11. Federal Ministry for Economic Affairs and Energy Germany (BMWi). Industrie 4.0 – Checkliste: Kommt Industrie 4.0 für unser Unternehmen in Frage. Berlin. 2017.

12. Хаустова В. Є., Крамарев Г. В., Зінченко В. А. Інноваційно-технологічне забезпечення модернізації пріоритетних галузей промисловості України //ЕКОНОМІКА. ЕКОНОМІКА ПРОМИСЛОВОСТІ, БІЗНЕСІНФОРМ № 3 ’2019 с. 218-228 (DOI: 10.32983/2222-4459-2019-3-218-228)

13. Ufuk Cebeci The Project Management of Industry 4.0 Strategy for Software Houses // Agile Approaches for Successfully Managing and Executing Projects in the Fourth Industrial Revolution 2019, 228- 241 (DOI: 10.4018/978-1-5225-7865-9.ch012)

14. Emmanuel Nowakowski, Matthias Farwick, Thomas Trojer, Martin Häusler, Johannes Kessler, Ruth Breu An Enterprise Architecture Planning Process for Industry 4.0 Transformations // In Proceedings of the 21st International Conference on Enterprise Information Systems (ICEIS 2019), pages 572-579 (DOI: 10.5220/0007680005720579)

15. Alexandre Moeuf, Samir Lamouri, Robert Pellerin, Simon Tamayo-Giraldo, Estefania Tobon-Valencia, Romain Eburdy Identification of critical success factors, risks and opportunities of Industry 4.0 in SMEs // Journal International Journal of Production Research (2019) (https://doi.org/10.1080/00207543.2019.1636323)

16. Anderl R, Fleischer J. Guidline Industrie 4.0 – Guiding principles for the implementation of Industry 4.0 in small and medium sized business. Frankfurt a. M: VDMA Forum Industrie; 2015

17. Ubaidullah Mohammad, Cheng Yee Low, Ramhuzaini Abd Rahman, Muhammad Akmal Johar, Ching Theng Koh, Roman Dumitrescu, Martin Rabe, Laban Asmar Smart Factory Reference Model for Training on Industry 4.0 // Journal of Mechanical Engineering Vol 16(2), 129-144, 2019.

18. Maximilian Both, Jochen Müller, Björn Kämper Development of Industry 4.0 models and their applicability for BIM // Revista Romana de Inginerie Civila; Bucharest Том 10, Изд. 3, (2019): 215-222.

19. Z Bradac, P Marcon, F Zezulka, J Arm, T Benesl Digital Twin and AAS in the Industry 4.0 Framework // IOP Conf. Series: Materials Science and Engineering 618 (2019) (doi:10.1088/1757-899X/618/1/012001)

20. ZVEI. Details of the Asset Administration Shell. Available online: URL: https://www.zvei.org/en/press-media/publications/details-of-the-asset-administration-shell/ (accessed on 27.01.2023)

21. Xun Ye; Seung Ho Hong Toward Industry 4.0 Components: Insights Into and Implementation of Asset Administration Shells // IEEE Industrial Electronics Magazine, Volume: 13, Issue: 1, March 2019, pp: 13-25 (DOI: 10.1109/MIE.2019.2893397)

22. Marketsandmarkets.com. Market Study: Industry 4.0 Market by Technology (Industrial Robotics, Cyber Security, Internet of Things, 3D Printing, Advanced Human-Machine Interface, Big Data, Augmented Reality & Virtual Reality, Artificial Intelligence) - Global Forecast to 2022. Available online: URL: https://www.marketsandmarkets.com/Market-Reports/industry-4-market-102536746.html. (accessed on 28.01.2020)

23. Luis Norberto López de Lacalle, Jorge Posada Special Issue on New Industry 4.0 Advances in Industrial IoT and Visual Computing for Manufacturing Processes // Applied sciences 2019, *9*(20), 4323 (https://doi.org/10.3390/app9204323)

24. Madakam S., Uchiya T. (2019) Industrial Internet of Things (IIoT): Principles, Processes and Protocols. In: Mahmood Z. (eds) The Internet of Things in the Industrial Sector. Computer Communications and Networks. Springer, Cham (https://doi.org/10.1007/978-3-030-24892-5_2)

25. Radanliev, Petar and Roure, David C. De and Nurse, Jason and Montalvo, Rafael Mantilla and Burnap, Pete, Supply Chain Design for the Industrial Internet of Things and the Industry 4.0 (March 4, 2019). (DOI: 10.13140/RG.2.2.36311.32160)

26. Philipp Osterrieder, LukasBudde, ThomasFriedli The smart factory as a key construct of industry 4.0: A systematic literature review // International Journal of Production Economics Available online 27 August 2019 (https://doi.org/10.1016/j.ijpe.2019.08.011)

27. Giacomo Büchi, Monica Cugno, Rebecca Castagnoli Smart factory performance and Industry 4.0 // Technological Forecasting and Social Change Volume 150, January 2020, 119790 (https://doi.org/10.1016/j.techfore.2019.119790)

28. Digital Manufacturing and Assembly Systems in Industry 4.0. Edited by Kaushik Kumar, Divya Zindani, J. Paulo Davim //2020 by Taylor&Francis, LLC. ISBN: 978-1-138-61272-3

29. Florian Maurer, Jens Schumacher evolving towards a smart factory of the future within supply chains: selected cases out of the alpine space // Proceedings of the 24th International Symposium on Logistics (ISL 2019) Supply Chain Networks vs Platforms: Innovations, Challenges and Opportunities Würzburg, Germany 14th – 17th July 2019, p.750, pp: 197-205

30. Nist.gov. Smart Manufacturing. Available online: URL: https://www.nist.gov/manufacturing-innovation-blog-categories/smart-manufacturing. (accessed on 28.01.2020)

31. Edward R. Griffor, David A. Wollman, Martin J. Burns, Joe Manganelli, Ronald Boring, Stephen Gilbert, Yi-Ching Lee, Dan Nathan-Robers Elaborating

the Human Aspect of the NIST Framework for Cyber-Physical Systems// Volume: 62 issue: 1, page(s): 450-454 (https://doi.org/10.1177/1541931218621103)

32. Giuseppe Aceto; Valerio Persico; Antonio Pescapé A Survey on Information and Communication Technologies for Industry 4.0: State-of-the-Art, Taxonomies, Perspectives, and Challenges // IEEE Communications Surveys & Tutorials, Volume: 21 , Issue: 4 , Fourthquarter 2019, pp: 3467 – 3501 (DOI: 10.1109/COMST.2019.2938259)

33. M.Faheem, S.B.H.Shah, R.A.Butt, B.Raza, M.Anwar, M.W.Ashraf, Md.A.Ngadi, V.C.Gungorb Smart grid communication and information technologies in the perspective of Industry 4.0: Opportunities and challenges // Computer Science Review Volume 30, November 2018, Pages 1-30 (https://doi.org/10.1016/j.cosrev.2018.08.001)

34. Yang Lu Industry 4.0: A survey on technologies, applications and open research issues // Journal of Industrial Information Integration Volume 6, June 2017, Pages 1-10 (https://doi.org/10.1016/j.jii.2017.04.005)

35. Qinglin Qi, Fei Tao Digital Twin and Big Data Towards Smart Manufacturing and Industry 4.0: 360 Degree Comparison // IEEE Access, Volume:6, 2018, pp: 3585 – 3593 (https://doi.org/ 10.1109/ACCESS. 2018.2793265)

36. Xiang T. R. Kong, Hao Luo, George Q. Huang, Xuan Yang Industrial wearable system: the human-centric empowering technology in Industry 4.0 // Journal of Intelligent Manufacturing. December 2019, Volume 30, Issue 8, pp 2853–2869 (https://doi.org/10.1007/s10845-018-1416-9)

37. Tortorella, G., Giglio, R. and van Dun, D. (2019), "Industry 4.0 adoption as a moderator of the impact of lean production practices on operational performance improvement", International Journal of Operations & Production Management, Vol. 39 No. 6/7/8, pp. 860-886. (https://doi.org/10.1108/IJOPM-01-2019-0005)

38. Qiushi Cao, Franco Giustozzi, Cecilia Zanni-Merk, François de Bertrand de Beuvron, Christoph Reich Smart Condition Monitoring for Industry 4.0

Manufacturing Processes: An Ontology-Based Approach // Journal Cybernetics and Systems, An International Journal, Volume 50, 2019 - Issue 2: adding smartness to systems with case studies and applications, Pages 82-96 (https://doi.org/10.1080/01969722.2019.1565118)

39. Omid Givehchi, Klaus Landsdorf, Pieter Simoens, Armando Walter Colombo Interoperability for industrial cyber-physical systems: An approach for legacy systems// IEEE Transactions on Industrial Informatics, Volume: 13 , Issue: 6 , Dec. 2017, Page(s): 3370 – 3378 (DOI: 10.1109/TII.2017.2740434)

40. David Romer, Peter Bernus Ovidiu NoranJohan Stahre Åsa Fast-Berglund The operator 4.0: human cyber-physical systems & adaptive automation towards human-automation symbiosis work systems// Advances in Production Management Systems. Initiatives for a Sustainable World. APMS 2016. IFIP Advances in Information and Communication Technology, vol 488. Springer, Cham pp 677-686 (https://doi.org/10.1007/978-3-319-51133-7_80)

41. Paulo Leitão, Armando Walter Colombo, Stamatis Karnouskos Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges// Computers in Industry Volume 81, September 2016, Pages 11-25 (https://doi.org/10.1016/j.compind.2015.08.004)

42. L.Monostori, B.Kádár, T.Bauernhansl, S.Kondoh, S.Kumara,G.Reinhart, O.Sauer, G.Schuh, W.Sihn, K.Ueda Cyber-physical systems in manufacturing//CIRP Annals Volume 65, Issue 2, 2016, Pages 621-641 (https://doi.org/10.1016/j.cirp.2016.06.005)

43. Xinghuo Yu ; Yusheng Xue Smart Grids: A Cyber–Physical Systems Perspective // Proceedings of the IEEE (Volume: 104 , Issue: 5 , May 2016) 1058 – 1070 (DOI: 10.1109/JPROC.2015.2503119)

44. Stefano Zanero Cyber-Physical Systems// Computer, Volume: 50, Issue: 4 , April 2017) Page(s): 14 – 16 (DOI: 10.1109/MC.2017.105)

45. Henry Muccini, Mohammad Sharaf, Danny Weyns Self-adaptation for cyber-physical systems: a systematic literature review // SEAMS '16: Proceedings of the 11th International Symposium on Software Engineering for Adaptive and

Self-Managing SystemsMay 2016 Pages 75–81 (https://doi.org/10.1145/2897053.2897069)

46. Sergii Iarovyi ; Wael M. Mohammed ; Andrei Lobov ; Borja Ramis Ferrer ; Jose L. Martinez Lastra Cyber–Physical Systems for Open-Knowledge-Driven Manufacturing Execution Systems// Proceedings of the IEEE (Volume: 104 , Issue: 5 , May 2016) Page(s): 1142 – 1154 (DOI: 10.1109/JPROC.2015.2509498)

47. X Yao, J Zhou, Y Lin, Y Li, H Yu, Y Liu Smart manufacturing based on cyber-physical systems and beyond // Journal of Intelligent Manufacturing volume 30(2019), pages 2805–2817 (DOI: https://doi.org/10.1007/s10845-017-1384-5)

48. Thomas Bangemann, Matthias Ried, Mario Thron, Christian Diedrich Integration of classical components into industrial cyber–physical systems // Proceedings of the IEEE, Volume: 104 , Issue: 5 , May 2016, Page(s): 947 – 959 (DOI: 10.1109/JPROC.2015.2510981)

49. V Jirkovský, M Obitko, V Mařík Understanding data heterogeneity in the context of cyber-physical systems integration// IEEE Transactions on Industrial Informatics (Volume: 13 , Issue: 2 , April 2017) Page(s): 660 – 667 (DOI: 10.1109/TII.2016.2596101)

50. István Mezgár, Gianfranco Pedone Cloud-Based Manufacturing (CBM) Interoperability in Industry 4.0 // Technological Developments in Industry 4.0 for Business Applications 2019, pp: 171-198 (DOI: 10.4018/978-1-5225-4936-9.ch008)

51. Fei Tao, Qinglin Qi, Lihui Wang, A.Y.C.Nee Digital Twins and Cyber–Physical Systems toward Smart Manufacturing and Industry 4.0: Correlation and Comparison // Engineering, Volume 5, Issue 4, August 2019, Pages 653-661 (https://doi.org/10.1016/j.eng.2019.01.014)

52. Gang Li, Jianlong Tan, Sohail S. Chaudhry Industry 4.0 and big data innovations // Journal Enterprise Information Systems,Volume 13, 2019 - Issue 2: Industry 4.0 and Big Data Innovations, pp: 145-147 (https://doi.org/10.1080/17517575.2018.1554190)

53. Yuqian Lu, Xun Xu Cloud-based manufacturing equipment and big data analytics to enable on-demand manufacturing services // Robotics and Computer-Integrated Manufacturing, Volume 57, June 2019, Pages 92-102 (https://doi.org/10.1016/j.rcim.2018.11.006)

54. Silva J., Gaitán M., Varela N., Lezama O.B.P. Engineering Teaching: Simulation, Industry 4.0 and Big Data // ICCVBIC 2019: Computational Vision and Bio-Inspired Computing pp 226-232 (https://doi.org/10.1007/978-3-030-37218-7_26)

55. Bogoviz, A., Lobova, S., Karp, M., Vologdin, E. and Alekseev, A. (2019), "Diversification of educational services in the conditions of Industry 4.0 on the basis of AI training", On the Horizon, Vol. 27 No. 3/4, pp. 206-212. (https://doi.org/10.1108/OTH-06-2019-0031)

56. Ellefsen A.P.T., Oleśków-Szłapka J., Pawłowski G., Toboła A., 2019. Striving for excellence in AI implementation: AI Maturity Model framework and preliminary research results. LogForum 15 (3), 363-376, (http://doi.org/10.17270/J.LOG.2019.354)

57. Antonio Celesti Guest Editorial Special Section on Cloud Computing, Edge Computing, Internet of Things, and Big Data Analytics Applications for Healthcare Industry 4.0 // IEEE Transactions on Industrial Informatics, Volume: 15 , Issue: 1 , Jan. 2019, pp: 454 – 456 (https://doi.org/10.1109/TII.2018.2883315)

58. Inés Sittón-Candanedo, Ricardo S. Alonso, Sara Rodríguez-González, José Alberto García Coria, Fernando De La Prieta (2020) Edge Computing Architectures in Industry 4.0: A General Survey and Comparison. In: Martínez Álvarez F., Troncoso Lora A., Sáez Muñoz J., Quintián H., Corchado E. (eds) 14th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2019). SOCO 2019. Advances in Intelligent Systems and Computing, vol 950 pp 121-131 (https://doi.org/10.1007/978-3-030-20055-8_12)

59. Alexander Kropp, Robert-Steve Schmoll, Giang T. Nguyen, Frank H. P. Fitzek Demonstration of a 5G Multi-access Edge Cloud Enabled Smart Sorting

Machine for Industry 4.0 //16th IEEE Annual Consumer Communications & Networking Conference (CCNC) 11-14 Jan. 2019, Las Vegas, NV, USA (https://doi.org/10.1109/CCNC.2019.8651732)

60. Klingenberg, C., Borges, M. and Antunes Jr, J. (2019), "Industry 4.0 as a data-driven paradigm: a systematic literature review on technologies", Journal of Manufacturing Technology Management, Vol. ahead-of-print No. ahead-of-print. (https://doi.org/10.1108/JMTM-09-2018-0325)

61. Sundarakani, B., Kamran, R., Maheshwari, P. and Jain, V. (2019), "Designing a hybrid cloud for a supply chain network of Industry 4.0: a theoretical framework", Benchmarking: An International Journal, Vol. ahead-of-print No. ahead-of-print. (https://doi.org/10.1108/BIJ-04-2018-0109)

62. C. Occhiuzzi, S. Amendola, S. Nappi, N. D'Uva, G. Marrocco RFID Technology for Industry 4.0: Architectures and Challenges // IEEE International Conference on RFID Technology and Applications (RFID-TA) 5-27 Sept. 2019 Pisa, Italy (https://doi.org/10.1109/RFID-TA.2019.8892049)

63. Leal P., Madeira R.N., Romão T. (2019) Model-Driven Framework for Human Machine Interaction Design in Industry 4.0. In: Lamas D., Loizides F., Nacke L., Petrie H., Winckler M., Zaphiris P. (eds) Human-Computer Interaction – INTERACT 2019. INTERACT 2019. Lecture Notes in Computer Science, vol 11749. Springer, Cham, pp 644-648 (https://doi.org/10.1007/978-3-030-29390-1_54)

64. Aitor Ardanza, Aitor Moreno, Álvaro Segura, Mikel de la Cruz, Daniel Aguinaga Sustainable and flexible industrial human machine interfaces to support adaptable applications in the Industry 4.0 paradigm // Journal International Journal of Production Research, Volume 57, 2019 - Issue 12: Special Issue: Sustainable Cybernetic Manufacturing, pages: 4045-4059 (https://doi.org/10.1080/00207543.2019.1572932)

65. Roda-Sanchez L., Olivares T., Garrido-Hidalgo C., Fernández-Caballero A. (2019) Gesture Control Wearables for Human-Machine Interaction in Industry 4.0. In: Ferrández Vicente J., Álvarez-Sánchez J., de la Paz López F., Toledo

Moreo J., Adeli H. (eds) From Bioinspired Systems and Biomedical Applications to Machine Learning. IWINAC 2019. Lecture Notes in Computer Science, vol 11487. Springer, Cham pp: 99-108 (https://doi.org/10.1007/978-3-030-19651-6_10)

66. Visit Hirankitti An Intelligent Agent Framework for SCADA // 2019 First International Conference on Digital Data Processing (DDP), 15-17 Nov. 2019, London, United Kingdom (https://doi.org/10.1109/DDP.2019.00024)

67. Luo Pan, Wu Zhizheng, Chen Fan Design of SCADA System for CNC Grinder Workshop Based on SIMATIC NET // 2019 IEEE International Conference on Smart Manufacturing, Industrial & Logistics Engineering (SMILE) 20-21 April 2019, Hangzhou, China (https://doi.org/10.1109/SMILE45626.2019.8965296)

68. Felipe Orellana, Romina Torres From legacy-based factories to smart factories level 2 according to the Industry 4.0 // Journal International Journal of Computer Integrated Manufacturing, Volume 32, 2019 - Issue 4-5: Smart Cyber-Physical System Applications in Production and Logistics, pages 441-451 (https://doi.org/10.1080/0951192X.2019.1609702)

69. Clemens Fallera, Max Höftmanna Service-oriented communication model for cyber-physical-production-systems// Procedia CIRP Volume 67, 2018, Pages 156-161 (https://doi.org/10.1016/j.procir.2017.12.192)

70. Francisco Almada-Lobo The Industry 4.0 revolution and the future of Manufacturing Execution Systems (MES)// Journal of Innovation Management (JIM) 3, 4(2015) pp: 16-21 (DOI: 10.24840/2183-0606_003.004_0003)

71. Lei Yue, Linkun Wang, Pengfei Niu, Nan Zheng Building a reference model for a Manufacturing Execution System (MES) platform in an Industry 4.0 context// Journal of Physics: Conference Series, Volume 1345(2019) (DOI https://doi.org/10.1088/1742-6596/1345/6/062002)

72. Jiří Vyskočil, Petr Kadera Plan Executor MES: Manufacturing Execution System Combined with a Planner for Industry 4.0 Production Systems // International Conference on Industrial Applications of Holonic and Multi-Agent

Systems (HoloMAS 2019) , Lecture Notes in Computer Science, vol 11710, 02 August 2019. Springer, Cham, pp 67-80 (DOI: https://doi.org/10.1007/978-3-030-27878-6_6)

73. Mantravadi Soujanya, Møller Charles An Overview of Next-generation Manufacturing Execution Systems: How important is MES for Industry 4.0? // Procedia Manufacturing 30 (2019) 588–595, (https://doi.org/10.1016/j.promfg.2019.02.083)

74. Mahyar Azarmipour, Haitham Elfaham, Caspar Gries, Ulrich Epple PLC 4.0: A Control System for Industry 4.0 // IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society, 14-17 Oct. 2019 Lisbon, Portugal (https://doi.org/10.1109/IECON.2019.8927026)

75. Markus Schäfer, Patrick Moll, Lukas Brocke, Sven Coutandin, Jürgen Fleischer Model for Web-Application based Configuration of Modular Production Plants with automated PLC Line Control Code Generation // Procedia CIRP, Volume 83, 2019, Pages 292-297 (https://doi.org/10.1016/j.procir.2019.03.126)

76. The Role of Smart Sensors in Production Processes and the Implementation of Industry 4.0 [Текст] / I. Karabegovic, E. Karabegovic, M. Mahmic, E. Husak // Журнал інженерних наук. - 2019. - Т. 6, № 2. - С. B8-B13. (https://doi.org/10.21272/jes.2019.6(2).b2)

77. Maurizio Galetto, Alessandro Schiavi, Gianfranco Genta, Andrea Prato, Fabrizio Mazzoleni Uncertainty evaluation in calibration of low-cost digital MEMS accelerometers for advanced manufacturing applications // CIRP Annals, Volume 68, Issue 1, 2019, Pages 535-538 (https://doi.org/10.1016/j.cirp.2019.04.097)

78. Mechatronics 2019: Recent Advances Towards Industry 4.0 //Editors: Roman Szewczyk, Jiří Krejsa, Michał Nowicki, Anna Ostaszewska-Liżewska, Advances in Intelligent Systems and Computing, Springer Nature Switzerland AG 2020, pages 515 ISBN 978-3-030-29992-7 (https://doi.org/10.1007/978-3-030-29993-4)

79. Kirchmer M., Franz P. (2019) Value-Driven Robotic Process Automation (RPA). In: Shishkov B. (eds) Business Modeling and Software Design. BMSD 2019. Lecture Notes in Business Information Processing, vol 356. pp 31-46 Springer, Cham (https://doi.org/10.1007/978-3-030-24854-3_3)

80. Andreas M. Radke, Minh Trang Dang, Albert Tan Using robotic process automation (RPA) to enhance item master data maintenance process // Scientific Journal of Logistics 2020, 16 (1), pp:129-140 (http://doi.org/10.17270/J.LOG.2020.380)

81. LilianaZarco, JörgSiegerta, Thomas Bauernhansl Software Model Requirements Applied to a Cyber-Physical Modular Robot in a Production Environment // Procedia CIRP Volume 81, 2019, Pages 352-357 (https://doi.org/10.1016/j.procir.2019.03.061)

82.

83. Mehrpouya, M.; Dehghanghadikolaei, A.; Fotovvati, B.; Vosooghnia, A.; Emamian, S.S.; Gisario, A. The Potential of Additive Manufacturing in the Smart Factory Industrial 4.0: A Review. Appl. Sci. 2019, 9, 3865 (https://doi.org/10.3390/app9183865)

84. Dilberoglu, U.M.; Gharehpapagh, B.; Yaman, U.; Dolen, M. The role of additive manufacturing in the era of industry 4.0.// Procedia Manufacturing, Volume 11, 2017, Pages 545-554 (https://doi.org/10.1016/j.promfg.2017.07.148)

85. DIN SPEC 91345:2016-04 (E) Reference Architecture Model Industrie 4.0 (RAMI4.0) Publication date 2016-04 (https://dx.doi.org/10.31030/2436156)

86. Kunath M., Winkler H. (2019) Adaptive Assistenzsysteme zur Entscheidungsunterstützung für die dynamische Auftragsabwicklung: Konzeptionelle Überlegungen und Anwendungsszenarien unter Berücksichtigung des Digitalen Zwillings des Produktionssystems. In: Obermaier R. (eds) Handbuch Industrie 4.0 und Digitale Transformation. Springer Gabler, Wiesbaden. P.1623., pp 269-294, ISBN 978-3-658-24575-7 (https://doi.org/10.1007/978-3-658-24576-4_12)

87. IEC PAS 63088:2017 (E) Smart manufacturing-Reference Architecture Model Industrie 4.0 (RAMI4.0), page 34. ISBN 978-2-8322-4053-3

88. DIN SPEC 16593-1:2018-04 (E) RM-SA - Reference Model for Industrie 4.0 Service Architectures - Part 1: Basic Concepts of an Interaction-based Architecture, page 48 (https://dx.doi.org/10.31030/2838942)

89. PD IEC/TS 62832-1:2016 Industrial-process measurement, control and automation. Digital factory framework. General principles, page 52. ISBN 978 0 580 90649 7

90. IEC 62264-1:2013 Enterprise-control system integration — Part 1: Models and terminology. Page 154.

91. IEC 62541-100:2015 OPC Unified Architecture - Part 100: Device Interface. Page 121.

92. i40.semantic-interoperability.org. Industry 4.0 Standards. Available online: URL: http://i40.semantic-interoperability.org/. (accessed on 30.01.2020)

93. Yang Liu ; Yu Peng ; Bailing Wang ; Sirui Yao; Zihe Liu Review on cyber-physical systems// IEEE/CAA Journal of Automatica Sinica, Volume: 4, Issue: 1 , Jan. 2017. pp: 27 – 40 (DOI: 10.1109/JAS.2017.7510349)

94. Paulo Leitão ; Stamatis Karnouskos ; Luis Ribeiro ; Jay Lee ; Thomas Strasser ; Armando W. Colombo Smart Agents in Industrial Cyber–Physical Systems// Proceedings of the IEEE (Volume: 104 , Issue: 5 , May 2016) Page(s): 1086 – 1101. (DOI: 10.1109/JPROC.2016.2521931)

95. Kazi Masudul Alam; Abdulmotaleb El Saddik C2PS: A digital twin architecture reference model for the cloud-based cyber-physical systems// IEEE Access. Volume: 5. Page(s): 2050 – 2062 (DOI: 10.1109/ACCESS.2017.2657006)

96. P.Hehenberger, B.Vogel-Heuser, D.Bradley, B.Eynard, T.Tomiyama, S.Achichef Design, modelling, simulation and integration of cyber physical systems: Methods and applications // Computers in Industry Volume 82, October 2016, Pages 273-289 (https://doi.org/10.1016/j.compind.2016.05.006)

97. Ilge Akkaya ; Patricia Derler ; Shuhei Emoto ; Edward A. Lee Systems Engineering for Industrial Cyber–Physical Systems Using Aspects // Proceedings

of the IEEE (Volume: 104 , Issue: 5 , May 2016) Page(s): 997 – 1012 (DOI: 10.1109/JPROC.2015.2512265)

98. Ezio Bartocci, Jyotirmoy Deshmukh, Alexandre Donzé, Georgios Fainekos,Oded Maler, Dejan Ničković, Sriram Sankaranarayanan Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications// Lectures on Runtime Verification. Lecture Notes in Computer Science, vol 10457. Springer, Cham, pp 135-175 (DOI: https://doi.org/10.1007/978-3-319-75632-5_5)

99. Li Da Xu, Lian Duan Big data for cyber physical systems in industry 4.0: a survey // Journal Enterprise Information Systems Volume 13, 2019 - Issue 2: Industry 4.0 and Big Data Innovations Pages 148-169 (https://doi.org/10.1080/17517575.2018.1442934)

100. BorjaBordel, RamónAlcarria, TomásRobles, DiegoMartín Cyber–physical systems: Extending pervasive sensing from control theory to the Internet of Things // Pervasive and Mobile Computing Volume 40, September 2017, Pages 156-184 (DOI: https://doi.org/10.1016/j.pmcj.2017.06.011)

101. Gtai.de. Why Germany is the place to be for Industrie 4.0. Available online: URL: https://www.gtai.de/gtai-en/invest/industries/industrie-4-0. (accessed on 30.01.2020)

102. Mckinsey.com. The next horizon for industrial manufacturing: Adopting disruptive digital technologies in making and delivering. Available online: URL: https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/the-next-horizon-for-industrial-manufacturing. (accessed on 30.01.2020)

103. Bcg.com. Industry 4.0: The Future of Productivity and Growth in Manufacturing. Available online: URL: https://www.bcg.com/publications/2015/engineered_products_project_business_industry_4_future_productivity_growth_manufacturing_industries.aspx. (accessed on 30.01.2020)

104. Pericles Loucopoulos, Evangelia Kavakli, Natalia Chechina Requirements Engineering for Cyber Physical Production Systems// Advanced

Information Systems Engineering. CAiSE 2019. Lecture Notes in Computer Science, vol 11483. Springer, Cham pp 276-291 (DOI: https://doi.org/10.1007/978-3-030-21290-2_18)

105. O Cardin Classification of cyber-physical production systems applications: Proposition of an analysis framework// Computers in Industry Volume 104, January 2019, Pages 11-21 (https://doi.org/10.1016/j.compind.2018.10.002)

106. S. Vijayakumar, N. Dhasarathan, P. Devabalan, C. Jehan Advancement and Design of Robotic Manipulator Control Structures on Cyber Physical Production System// Journal of Computational and Theoretical Nanoscience, Volume 16, Number 2, February 2019, pp. 659-663(5) Publisher: American Scientific Publishers (https://doi.org/10.1166/jctn.2019.7786)

107. HermannMeissner, Jan C. Auricha Implications of cyber-physical production systems on integrated process planning and scheduling// Procedia Manufacturing Volume 28, 2019, Pages 167-173 (https://doi.org/10.1016/j.promfg.2018.12.027)

108. Fazel Ansari, Robert Glawar, Tanja Nemeth PriMa: a prescriptive maintenance model for cyber-physical production systems // Journal International Journal of Computer Integrated Manufacturing Volume 32, 2019 - Issue 4-5: Smart Cyber-Physical System Applications in Production and Logistics Pages 482-503 (https://doi.org/10.1080/0951192X.2019.1571236)

109. Luis Alberto Cruz Salazar, Daria Ryashentseva, Arndt Lüder, Birgit Vogel-Heuser Cyber-physical production systems architecture based on multi-agent's design pattern—comparison of selected approaches mapping four agent patterns// The International Journal of Advanced Manufacturing Technology volume 105, pages4005–4034(2019) (https://doi.org/10.1007/s00170-019-04226-8.)

110. Pericles Loucpoulos, Evangelia Kavakli, Natalia Chechina Requirements Engineering for Cyber Physical Production Systems // Advanced

Information Systems Engineering: 31st International Conference CAiSE 2019, LNCS 11483 pp.276-291 (https://doi.org/10.1007/978-3-030-21290-2_18)

111. Dawn M. Tilbury Cyber-Physical Manufacturing Systems // Annual Review of Control, Robotics, and Autonomous Systems Vol. 2:427-443 (https://doi.org/10.1146/annurev-control-053018-023652)

112. Sven Hoffmann, Aparecido Fabiano Pinatti de Carvalho, Darwin Abele, Marcus Schweitzer, Peter Tolmie, Volker Wulf Cyber-Physical Systems for Knowledge and Expertise Sharing in Manufacturing Contexts: Towards a Model Enabling Design // Computer Supported Cooperative Work (CSCW) June 2019, Volume 28, Issue 3–4, pp 469–509 (doi:10.1007/s10606-019-09355-y)

113. Hyoung Seok Kang, Ju Yeon Lee, Sang Do Noh A dynamic processing methodology of manufacturing data for the automated throughput analysis in cyber-physical production environment // Journals Concurrent Engineering Volume: 27 issue: 2, page(s): 155-169 (https://doi.org/10.1177%2F1063293X19842264)

114. Frank Allgöwer, João Borges de Sousa, James Kapinski, Pieter Mosterman, Jens Oehlerking, Patrick Panciatici, Maria Prandini, Akshay Rajhans, Paulo Tabuada, Philipp Wenzelburger Position paper on the challenges posed by modern applications to cyber-physical systems theory // Nonlinear Analysis: Hybrid Systems Volume 34, November 2019, Pages 147-165 (https://doi.org/10.1016/j.nahs.2019.05.007)

115. Gianfranco E.Modoni, Enrico G.Caldarola, Marco Sacco, Walter Terkaj Synchronizing physical and digital factory: benefits and technical challenges //12th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 18-20 July 2018, Gulf of Naples, Italy, Volume 79, 2019, Pages 472-477 (https://doi.org/10.1016/j.procir.2019.02.125)

116. Christos Emmanouilidis, Petros Pistofidis, Luka Bertoncelj, Vassilis Katsouros, Apostolos Fournaris, Christos Koulamas, Cristobal Ruiz-Carcel Enabling the human in the loop: Linked data and knowledge in industrial cyber-

physical systems // Annual Reviews in Control Volume 47, 2019, Pages 249-265 (https://doi.org/10.1016/j.arcontrol.2019.03.004)

117. A V Gurjanov, D A Zakoldaev, A V Shukalov, I O Zharinov Formation principles of digital twins of Cyber-Physical Systems in the smart factories of Industry 4.0 //IOP Conf. Series: Materials Science and Engineering 483 (2019) (doi:10.1088/1757-899X/483/1/012070)

118. Industrial Agents: Emerging Applications of Software Agents in Industry // Edited by Paulo Leitão, Stamatis Karnouskos 2015 Elsevier. Inc. p. 447 ISBN: 978-0-12-800341-1

119. Plattform-i40.de. Agent-Based Networks for Cyber–Physical Production Systems (CPPS). Cyber–Physical Production Systems: Intelligent Software for Production Systems in Industrie 4.0 Available online: URL: https://www.plattform-i40.de/PI40/Redaktion/EN/Use-Cases/265-agent-based-networks-for-cyber%E2%80%93physica-production-systems-tu-muenchen/article-agent-based-networks-for-cyber%E2%80%93physica-production-systems-tu-muenchen.html. (accessed on 03.02.2020)

120. Ribeiro L, Hochwallner M (2018) On the design complexity of cyber-physical production systems. Complexity 2018:1–13. https://doi.org/10.1155/8503

121. Haoues M, Sellami A, Ben-Abdallah H, Cheikhi L (2017) A guideline for software architecture selection based on ISO 25010 quality related characteristics. Int J Syst Assur Eng Manag 8:886–909

122. Farid AM, Ribeiro L (2015) An axiomatic design of a multiagent reconfigurable mechatronic system architecture. IEEE Trans Ind Informatics 11:1142–1155. https://doi.org/10.1109/TII.2015.2470528

123. Cruz SLA, Mayer F, Schütz D, Vogel-Heuser B (2018) Platform independent multi-agent system for robust networks of production systems. IFAC-PapersOnLine 51:1261–1268. https://doi.org/10.1016/j.ifacol.2018.08.359

124. Fischer J, Marcos M, Vogel-Heuser B (2018) Model-based development of a multi-agent system for controlling material flow systems. Autom 66:438–448

125. Lüder A, Calá A, Zawisza J, Rosendahl R (2017) Design pattern for agent based production system control—a survey. In: 13th IEEE conference on automation science and engineering, CASE. pp 717–722

126. Rehberger S, Spreiter L, Vogel-Heuser B (2017) An agent-based approach for dependable planning of production sequences in automated production systems. At-Automatisierungstechnik 65:766–778

127. L. Ribeiro, M. Hochwallner (2018) On the design complexity of cyber-physical production systems. Complexity 2018:1–13. (https://doi.org/10.1155/2018/4632195)

128. Cruz SLA Vogel-Heuser B (2017) Comparison of agent oriented software methodologies to apply in cyber physical production systems. In: 15th international conference on industrial informatics, INDIN. IEEE. Emden, Germany, pp 65–71. https://doi.org/10.1109/INDIN.2017.8104748

129. Lüder A, Schleipen M, Schmidt N, et al (2018) One step towards an industry 4.0 component. In: 13th IEEE conference on automation science and engineering, CASE. pp 1268–1273

130. A. ISA, “ISA-95.00.01-2000: enterprise-control system integration part 1: models and terminology,” Technical report, ISA, The Instrumentation, Systems, and Automation Society, 2000.

131. Platform Industrie 4.0 (I4.0) (2018) The structure of the administration shell: trilateral perspective from France, Italy and Germany. 64

132. L. Ribeiro, “Cyber-physical production systems’ design challenges,” in 2017 IEEE 26th International Symposium on Industrial Electronics (ISIE), pp. 1189–1194, Edinburgh, UK, 2017.

133. Syed Imran Shafiq, Edward Szczerbicki, Cesar Saninc Proposition of the methodology for Data Acquisition, Analysis and Visualization in support of Industry 4.0 // Procedia Computer Science, Volume 159, 2019, Pages 1976-1985 (https://doi.org/10.1016/j.procs.2019.09.370)

134. Sebastian Thiede, Artem Turetskyy, Arno Kwade, Sami Kara, Christoph Herrmann Data mining in battery production chains towards multi-

criterial quality prediction // CIRP Annals, Volume 68, Issue 1, 2019, Pages 463-466 (https://doi.org/10.1016/j.cirp.2019.04.066)

135. Guejong Jo, Su-Hwan Jang, Jongpil Jeong Design and Implementation of CPPS and Edge Computing Architecture based on OPC UA Server // Procedia Computer Science, Volume 155, 2019, Pages 97-104 (https://doi.org/10.1016/j.procs.2019.08.017)

136. Malhotra J., Iqbal F., Sahu A.K., Jha S. (2019) A Cyber-Physical System Architecture for Smart Manufacturing. In: Shunmugam M., Kanthababu M. (eds) Advances in Forming, Machining and Automation. Lecture Notes on Multidisciplinary Industrial Engineering. Springer, Singapore. pp 637-647 (https://doi.org/10.1007/978-981-32-9417-2_53)

137. Christine Schulze, Sebastian Thiede, Bastian Thiede, Denis Kurle, Stefan Blume, Christoph Herrmann Cooling tower management in manufacturing companies: A cyber-physical system approach // Journal of Cleaner Production, Volume 211, 20 February 2019, Pages 428-441 (https://doi.org/10.1016/j.jclepro.2018.11.184)

138. Kashif Mahmood, Tatjana Karaulova, Tauno Otto, Eduard Shevtshenko Development of cyber-physical production systems based on modelling technologies // Proceedings of the Estonian Academy of Sciences, 2019, 68, 4, 348–355 (https://doi.org/10.3176/proc.2019.4.02)

139. Sergei Kaganski, Jüri Majak, Kristo Karjust, Silver Toompalu Implementation of key performance indicators selection model as part of the Enterprise Analysis Model. Procedia CIRP, 2017, 63, 283−288 (https://doi.org/10.1016/j.procir.2017.03.143)

140. O.Mörth, C.Emmanouilidis, N.Hafner, M.Schadler Cyber-Physical Systems for Performance Monitoring in Production Intralogistics // Computers & Industrial Engineering, Available online 1 February 2020, 106333 (https://doi.org/10.1016/j.cie.2020.106333)

141. João Vicente Cardoso Faria, Ivandro Cecconello Evaluation of OPC UA technology in order to minimize systems unavailability by improving M2M

connectivity // Scientia cum Industria, Vol 7, No 2 (2019), pp. 40 – 51 (http://dx.doi.org/10.18226/23185279.v7iss2p40)

142. R. van de Sand, S. Schulz, J. Reiff-Stephan (2019) Smart Process Communication for Small and Medium-Sized Enterprises. In: Popplewell K., Thoben KD., Knothe T., Poler R. (eds) Enterprise Interoperability VIII. Proceedings of the I-ESA Conferences, vol 9. Springer, Cham. pp 411-420. (https://doi.org/10.1007/978-3-030-13693-2_34)

143. Hammer M. (2019) Digitization Perspective: Impact of Digital Technologies in Manufacturing. In: Management Approach for Resource-Productive Operations. Industrial Management. Springer Gabler, Wiesbaden. pp. 27-68 (https://doi.org/10.1007/978-3-658-22939-9_3)

144. Chiara Cimini, Fabiana Pirola, Roberto Pinto, Sergio Cavalieri A human-in-the-loop manufacturing control architecture for the next generation of production systems // Journal of Manufacturing Systems, Volume 54, January 2020, Pages 258-271 (https://doi.org/10.1016/j.jmsy.2020.01.002)

145. Tobias Wagner, Christoph Herrmann, Sebastian Thiede Industry 4.0 Impacts on Lean Production Systems // Procedia CIRP, Volume 63, 2017, Pages 125-131 (https://doi.org/10.1016/j.procir.2017.02.041)

146. A. Syberfeldt, M. Holm, O. Danielsson, L. Wang, R.L. Brewster Support systems on the industrial shop-floors of the future – operators' perspective on augmented reality Procedia Cirp, 44 (2016), pp. 108-113, (10.1016/j.procir.2016.02.017)

147. S. Wang, J. Wan, D. Li, C. Zhang Implementing Smart Factory of Industrie 4.0: An Outlook Int J Distrib Sens Netw Int J Distrib Sens Netw, 2016 (2016), Article e3159805 (https://doi.org/10.1155/2016/3159805)

148. H.S. Kang, J.Y. Lee, S. Choi, H. Kim, J.H. Park, J.Y. Son, et al. Smart manufacturing: past research, present findings, and future directions Int J Precis Eng Manuf-Green Technol, 3 (2016), pp. 111-128, (10.1007/s40684-016-0015-5)

149. L.D. Xu, E.L. Xu, L. Li Industry 4.0: state of the art and future trends Int J Prod Res, 56 (2018), pp. 2941-2962, (10.1080/00207543.2018.1444806)

150. Rainer Stark, Carina Fresemann, Kai Lindow Development and operation of Digital Twins for technical systems and services // CIRP Annals, Volume 68, Issue 1, 2019, Pages 129-13 (https://doi.org/10.1016/j.cirp.2019.04.024)

151. Tortorella, G., Miorando, R., Meiriño, M. and Sawhney, R. (2019), "Managing practitioners' experience and generational differences for adopting lean production principles", The TQM Journal, Vol. 31 No. 5, pp. 758-771. (https://doi.org/10.1108/TQM-02-2019-0041)

152. Weber, C., Königsberger, J., Kassner, L., Mitschang, B. (2017). M2DDM – A maturity model for data-driven manufacturing, Procedia CIRP, Vol. 63, 173-178, (doi: 10.1016/j.procir.2017.03.309)

153. M. Resman, M. Pipan, M. Šimic, N. Herakovič A new architecture model for smart manufacturing: A performance analysis and comparison with the RAMI 4.0 reference model // Advances in Production Engineering & Management Volume 14, Number 2, June 2019, pp153–165 (https://doi.org/10.14743/apem2019.2.318)

154. Zezulka, F., Marcon, P., Vesely, I., Sajdl, O. (2016). Industry 4.0 – An introduction in the phenomenon, IFACPapersOnLine, Vol. 49, No. 25, 8-12, (https://doi.org/10.1016/j.ifacol.2016.12.002)

155. Phoenix contact. RAMI 4.0 and IIRA reference architecture models – A question of perspective and focus. Available online: URL: https://www.mynewsdesk.com/material/document/56241/download?resource_type=resource_document. (accessed on 05.02.2020)

156. Schweichhart, K. Reference architecture model Industrie 4.0 (RAMI 4.0). Available online: URL: https://ec.europa.eu/futurium/en/system/files/ged/a2-schweichhart-reference_architectural_model_industrie_4.0_rami_4.0.pdf. (accessed on 05.02.2023)

157. IEC 62264-1:2013 Enterprise-control system integration — Part 1: Models and terminology. Available online: URL: https://www.iso.org/standard/57308.html. (accessed on 05.02.2020)

158. IEC 61512-3:2008 Batch control Part 3: General and site recipe models and representation. Available online: URL: https://joinup.ec.europa.eu/solution/iec-61512-32008-batch-control-part-3-general-and-site-recipe-models-and-representation/about. (accessed on 05.02.2020)

159. M.A. Pisching, M.A.O. Pessoa, F.Junqueira, D.J. dos Santos Filho, P.E. Miyagi (2018). An architecture based on RAMI 4.0 to discover equipment to process operations required by products, Computers & Industrial Engineering, Vol. 125, 574-591, (https://doi.org/10.1016/j.cie.2017.12.029)

160. Elder Hernández, Pedro Senna, Daniela Silva, Rui Rebelo, Ana C. Barros, César Toscano Implementing RAMI4. 0 in Production-A Multi-case Study// International Conference of Progress in Digital and Physical Manufacturing, ProDPM 2019: Progress in Digital and Physical Manufacturing, Springer, Cham, pp 49-56 (https://doi.org/10.1007/978-3-030-29041-2_6)

161. Alexandros Bousdekis, Katerina Lepenioti, Dimitrios Ntalaperas, Danai Vergeti, Dimitris Apostolou, Vasilis Boursinos A RAMI 4.0 View of Predictive Maintenance: Software Architecture, Platform and Case Study in Steel Industry// Advanced Information Systems Engineering Workshops. CAiSE 2019. Lecture Notes in Business Information Processing, vol 349. Springer, Cham, Pages 95-106 (DOI: https://doi.org/10.1007/978-3-030-20948-3)

162. Václav Kaczmarczyk, Tomáš Benešl, Zdeněk Bradáč, Petr Fiedler, Zuzana Kaczmarczyková SkuBATCH - System for control of technological processes // IFAC-PapersOnLine, Volume 52, Issue 27, 2019, Pages 477-483 (https://doi.org/10.1016/j.ifacol.2019.12.709)

163. Llamuca J.D., Garcia C.A., Naranjo J.E., Rosero C., Alvarez-M E., Garcia M.V. (2020) Integrating ISA-95 and IEC-61499 for Distributed Control System Monitoring. In: Fosenca C E., Rodríguez Morales G., Orellana Cordero M., Botto-Tobar M., Crespo Martínez E., Patiño León A. (eds) Information and Communication Technologies of Ecuador (TIC.EC). TICEC 2019. Advances in Intelligent Systems and Computing, vol 1099. pp. 66-80 (https://doi.org/10.1007/978-3-030-35740-5_5)

164. Bernhard Wally, Jirí Vyskočil, Petr Novák, Christian Huemer, Radek Šindelar, Petr Kadera, Alexandra Mazak, Manuel Wimmer Flexible Production Systems: Automated Generation of Operations Plans Based on ISA-95 and PDDL // IEEE Robotics and Automation Letters, Volume: 4, Issue: 4, Oct. 2019, pp: 4062 – 4069 (DOI: 10.1109/LRA.2019.2929991)

165. Rafal Cupek, Adam Ziebinski, Marek Drewniak, Marcin Fojcik Knowledge integration via the fusion of the data models used in automotive production systems // Journal Enterprise Information Systems, Volume 13, 2019 - Issue 7-8: Computational Collective Intelligence for Enterprise Information. Pages 1094-1119 (https://doi.org/10.1080/17517575.2018.1489563)

166. Formalizing ISA-95 Level 3 Control with Smart Manufacturing System Models // By Leon F. McGinnis. National Institute of Standards and Technology, November 2019. PP- 86. (https://doi.org/10.6028/NIST.GCR.19-022)

167. Jehn-Ruey Jiang An improved cyber-physical systems architecture for Industry 4.0 smart factories//Advances in Mechanical Engineering 2018, Vol. 10(6) 1–15 (https://doi.org/10.1177/1687814018784192)

168. Lee J., Bagheri B., Kao H.-A., A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems, Manufacturing Letters 3 (2015) 18–23, (https://doi.org/10.1016/j.mfglet.2014.12.001).

169. Usharani Hareesh Govindarajan, Amy J.C. Trappey, Gopal Kumar Latent Dirichlet Allocation modeling for CPS patent topic discovery // Proceedings of the 2018 International Conference on Industrial Enterprise and System Engineering (IcoIESE 2018), Atlantis Highlights in Engineering (AHE), Volume 2, pp: 31- 36 (https://doi.org/10.2991/icoiese-18.2019.6)

170. Qingmeng Tan, Yifei Tong, Shaofeng Wu, Dongbo Li Modeling, planning, and scheduling of shop-floor assembly process with dynamic cyber-physical interactions: a case study for CPS-based smart industrial robot production// The International Journal of Advanced Manufacturing Technology, December 2019, Volume 105, Issue 9, pp 3979–3989 (https://doi.org/10.1007/s00170-019-03940-7)

171. Martin Frické, The Knowledge Pyramid: the DIKW Hierarchy // Ko knowledge organization, Seite 33 – 46, KO, Jahrgang 46 (2019), Heft 1, ISSN print: 0943-7444, ISSN online: 0943-7444, (https://doi.org/10.5771/0943-7444-2019-1-33)

172. D. Nell, E.H. Mathews, P. Maré Industry 4.0 Roll-out Strategy for Dynamic Mine Heat Load Management // South African Journal of Industrial Engineering, vol.30 n.3 Pretoria Nov. 2019 (http://dx.doi.org/10.7166/30-3-2232)

173. Yucong Duan, Zhihui Lu, Zhangbing Zhou, Xiaobing Sun, Jie Wu Data Privacy Protection for Edge Computing of Smart City in a DIKW Architecture // Engineering Applications of Artificial Intelligence, Volume 81, May 2019, Pages 323-335 (https://doi.org/10.1016/j.engappai.2019.03.002)

174. Jan Gabriel Pretorius, Marc John Mathews, Philip Maré, Marius Kleingeld, Johann van Rensburg Implementing a DIKW model on a deep mine cooling system // International Journal of Mining Science and Technology, Volume 29, Issue 2, March 2019, Pages 319-326 (https://doi.org/10.1016/j.ijmst.2018.07.004)

175. Patrick Tinz, Janik Tinz, Stefan Zander Knowledge Management Models for the Smart Factory: A Comparative Analysis of Current Approaches // In Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2019), pages 398-404 (DOI: 10.5220/0008348803980404)

176. H. Gao, Y. Duan, L. Shao, et al. Transformation-based processing of typed resources for multimedia sources in the IoT environment. Wireless Netw (2019) (https://doi.org/10.1007/s11276-019-02200-6)

177. Petar Radanliev, David De Roure, Razvan Nicolescu, Michael Huth A reference architecture for integrating the Industrial Internet of Things in the Industry 4.0 // Computers and Society (cs.CY) 2019. (DOI: 10.13140/RG.2.2.26854.47686)

178. Andreas Schumacher, Tanja Nemetha, Wilfried Sihna Roadmapping towards industrial digitalization based on an Industry 4.0 maturity model for

manufacturing enterprises // 12th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 18-20 July 2018, Volume 79, 2019, Pages 409-414 Pages 1-734 (2019) (https://doi.org/10.1016/j.procir.2019.02.110)

179. Jung K, Kulvatunyou B, Choi S, Brundage MP. An overview of a smart manufacturing system readiness assessment. IFIP - Advances in Information and Communication Technology 2016;488:705-712. (https://doi.org/10.1007/978-3-319-51133-7_83)

180. Jodlbauer H, Schagerl M. Reifegradmodell Industrie 4.0. Informatik 2016; 892:1473–1487.

181. Liebrecht C, Burgin J, Benterbusch J, Kiefer C, Lanza G. Shopfloor-getriebene Einführung von Industrie 4.0. Werkstattstechnik online 2016; 106(7/8):539–543. ISSN: 1436-5006, 1436-4980, KITopen-ID: 1000073998

182. Schmitz S. Industrie-4.0-Reifegradindex zur Standortbestimmung der Unternehmen. Unternehmen der Zukunft - Zeitschrift für Betriebsorganisation und Unternehmensentwicklung 2016;17(1):31–33.

183. De Carolis A, Macchi M, Negri E, Terzi S. A Maturity Model for Assessing the Digital Readiness of Manufacturing Companies. In: Lödding H, Riedel R, Thoben KD, Cieminski G, Kiritsis D, editors: Advances in Production Management Systems. The Path to Intelligent, Collaborative and Sustainable Manufacturing. Springer International Publishing; 2016. p.13-20. (DOI: 10.1007/978-3-319-66923-6_2)

184. Rößler MP, Haschemi M. Smart Factory Assessment (SFA): Eine Methodik zur integralen Reifegradbewertung von Produktion und Logistik hinsichtlich Lean und Industrie 4.0. ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb 2017;112(10):699–703. (DOI: 10.3139/104.111800)

185. Rajnai Z, Kocsis I. Assessing Industry 4.0 Readiness of Enterprises. Presented at IEEE 16th World Symposium on Applied Machine Intelligence and Informatics (SAMI). Kosice-Slovakia 2018. (DOI:10.1109/sami.2018.8324844)

186. Schott P, Lederer M, Niedermaier S, Bodendorf F, Hafner M. A Maturity Model to Organize the Multidimensionality of Digitalization in Smart

Factories. In: Yilmaz ÖF, Tüfekçí S, editors. Handbook of Research on Applied Optimization Methodologies in Manufacturing Systems. Nürnberg: IGI Global; 2018. p.354 -374 (DOI: 10.4018/978-1-5225-2944-6)

187. Nemeth T, Ansari F, Sihn W, Haslhofer B, Schindler A. PriMa-X: A Reference Model for Realizing Prescriptive Maintenance and Assessing its Maturity Enhanced by Machine Learning. Procedia CIRP 2018;72:1039-1044. (https://doi.org/10.1016/j.procir.2018.03.280)

188. Weber C, Königsberger J, Kassner L, Mitschang B. M2DDM–a maturity model for data-driven manufacturing. Procedia CIRP 2017;63:173-178. (doi: 10.1016/j.procir.2017.03.309)

189. Leyh C., Schäffer T., Bley K., Forstenhäusler S. (2017) Assessing the IT and Software Landscapes of Industry 4.0-Enterprises: The Maturity Model SIMMI 4.0. In: Ziemba E. (eds) Information Technology for Management: New Ideas and Real Solutions. ISM 2016, AITM 2016. Lecture Notes in Business Information Processing, vol 277. Springer, Cham. pp 103-119 (https://doi.org/10.1007/978-3-319-53076-5_6)

190. Klötzer C, Pflaum A. Toward the Development of a Maturity Model for Digitalization within the Manufacturing Industry Supply Chain. Proceedings of the 50th Hawaii International Conference on System Sciences 2017;4210-4219 (DOI: 10.24251/HICSS.2017.509)

191. M.Colli, U.Berger, M.Bockholt, O.Madsen, C.Møller, B. Vejrum Wæhrens A maturity assessment approach for conceiving context-specific roadmaps in the Industry 4.0 era // Annual Reviews in Control Volume 48, 2019, Pages 165-177 (https://doi.org/10.1016/j.arcontrol.2019.06.001)

192. Albert Albers, Bartosz Gladysz, Tobias Pinner, Viktoriia Butenko, Tobias Stürmlinger Procedure for defining the system of objectives in the initial phase of an Industry 4.0 project focusing on intelligent quality control systems // Procedia CIRP 52 (2016). 262 – 267 (https://doi.org/10.1016/j.procir.2016.07.067)

193. A. Gavriluță1, C. A. Gavriluță, I. Pascu and C. Neacșu The development of a methodology of learning to use simulation in the analysis of

production system performances // IOP Conf. Series: Materials Science and Engineering 564 (2019) (doi:10.1088/1757-899X/564/1/012100)

194. ISO 18828-5:2019 Industrial automation systems and integration — Standardized procedures for production systems engineering — Part 5: Manufacturing change management. [Electronic resource]. URL: https://www.iso.org/standard/ 69231.html.

195. Mykola Fisun, Mykhailo Dvoretskyi, Hlib Horban, Myroslav Komar Knowledge management applications based on user activities feedback // International Journal of Computing, 18(1) 2019, 32-44. Print ISSN 1727-6209

196. Turlakova, S. S. (2019). Information and communication technologies for the development of "smart" industries. Econ. promisl., 1(85), pp. 101-123. (doi: http://doi.org/10.15407/econindustry2019.01.101)

197. Nevliudov I., Yevsieiev V., Omarov M., Bronnikov A., Liashenko V. (2020). Method of Algorithms for Cyber-Physical Production Systems Functioning Synthesis. International Journal of Emerging Trends in Engineering Research (IJETER), Volume 8. No.10. PP. 7465–7473. DOI:10.30534/ijeter/2020/ 1278102020.

198. Pedro Daniel Urbina Coronado, Roby Lynn, Wafa Louhichi, Mahmoud Parto, Ethan Wescoat&Thomas Kurfess. (2018) Part data integration in the Shop Floor Digital Twin: Mobile and cloud technologies to enable a manufacturing execution system. Journal of Manufacturing Systems. Volume 48, Part C, P.25–33. DOI: 10.1016/j.jmsy.2018.02.002.

199. Yevsieiev V., Miliutina S., Kollesnyk K. (2015). Software development Life Cycle Model . Warsaw University of techology, Instiyte of Design Fundamenals XXIII Polish-Ukrainin conference CAD in MACHINERY DESIGN (CADMD 2015), P.19–20.

200. Євсєєв В.В., Демська А.І. (2017). Розробка моделі життєвого циклу розробки програмних продукту та програмних модулів для КІС ТПВ. Технология приборостроения. №1. С.12–16.

201. Nevlyudov I., Yevsieiev V., Miliutina S. (2014). Program Project Development Life Cycle Model. Комп'ютерні системи проектування теорія і практика. № 808. С. 26–30.

202. Nevlyudov I., Yevsieiev V., Miliutina S., Kolesnyk K. (2015). High – Level Programming Language Decomposition Parametric Model. Machine Dynamics Research, Warsaw University of Technology. Vol. 39. No 1. P.81–91.

203. Nevlyudov I., Yevsieiev V., Miliutina S., Kollesnyk K. (2017). Object semantic model for life cycle model "Jamp". CAD in Machinery Design. Implementation and Educational Issues. 25 Proceedings of Polish- Ukrainian Conference (CADMD'2017). P. 31–32.

204. Yevsieiev V. (2018). Visual objects interaction mathematical presentation to solve the problem of software design automation for computer information systems of technological production preparation. Наукові праці Донецького національного технічного університету. Серія: «Обчислювальна техніка та автоматизація». № 1(31). С. 24–31

205. Yevsieiev V. (2018). Conceptual scheme and basic concepts graphic representation of software and modules visual elements description in CIS TPP design automation problem solution. Наукові нотатки. Міжвузівський збірник (за галузями знань «Технічні науки»). Випуск 61. С. 40–47.

206. Yevsieiev V. (2018). Visual components formal description development for the automated design of software products and modules for computer-integrated production technological preparation systems. Вчені записки таврійського національного університету імені В.І. Вернадського Серія: Технічні науки. Том 29 (68) № 1 Частина 1. С.143–147, DOI: 10.31474/20754272-2018-1-31-24-31.

207. ISO/IEC 27001:2013. Information technology -Security techniques - Information security management systems – Requirements. [Electronic resource]. URL:https://www.iso.org/standard/54534.html. (Дата звернення: 02.09.2019).

208. Невлюдов І., Євсєєв В., Демська А. (2018). Розробка синтаксичної та семантичної моделі мови визначення і опису даних предметної області.

Manufacturing & Mechatronic Systems 2018: Proceedings of IIst International Conference (M&MS 2018). P. 48–53.

209. Nevliudov I., Yevsieiev V., J. H. Baker, Ahmad M. A., Lyashenko V. (2021). Development of a cyber design modeling declarative language for cyber physical production systems. Journal of Mathematical and Computational Science. No.1. P.520–542, DOI:10.28919/jmcs/5152

210. Yevsieiev V., Bronnikov A. (2020). Development of databases interconnection “essences” information model for cyber-physical production systems additive cyber design creation automation. Збірник наукових праць національного університету кораблебудування ім. адмірала Макарова. №3(481). P. 56–62. DOI: 10.15589/znp2020.3(481).7.

211. Yevsieiev V.(2017). Program code automated system development at early stage of software life cycle. Наукові праці Донецького національного технічного університету. Серія: «Обчислювальна техніка та автоматизація». №1(30). С.69–77. DOI: 10.31474/2075-4272-2018-1-31-24-31.

212. Nevliudov,I., Yevsieiev, V., Demska,N., Novoselov, S. (2020). Development of a software module for operational dispatch control of production based on cyber-physical control systems. Innovative Technologies and Scientific Solutions for Industries. No.4(14), P.155–168. DOI:10.30837/ITSSI.2020.14.155.

213. Khalid, M. S., Yevsieiev, V., Nevliudov, I. S., Lyashenko, V., & Wahid, R. (2022). HMI Development Automation with GUI Elements for Object-Oriented Programming Languages Implementation. International Journal of Engineering Trends and Technology, 70.1, 139-145.

214. Nevliudov, I., & et al.. (2021). Development of a cyber design modeling declarative Language for cyber physical production systems, J. Math. Comput. Sci., 11(1), 520-542.

215. Nevliudov, I., & et al.. (2021). GUI Elements and Windows Form Formalization Parameters and Events Method to Automate the Process of Additive CyberDesign CPPS Development. Advances in Dynamical Systems and Applications, 16(2), 441-455.

216. Невлюдов І.Ш. Автоматизована система керування технологічними процесами в SCADA системі TRACE MODE 6: Навчальний посібник / І.Ш. Невлюдов, А.О. Андрусевич, В.В. Євсєєв, С.С. Максимова, М.Г. Стародубцев, В.В.Невлюдова. Кривий Ріг: Криворізький коледж НАУ, 2018. 320 с.

217. Viktoriia Bortnikova, Vladyslav Yevsieiev, Iryna Botsman, Igor Nevliudov, Kostiantyn Kolesnyk, Nazariy Jaworski. Queries classification using machine learning for implementation in intelligent manufacturing // Chapter 6 in Monograph «Methods and tools in CAD – selected issues». – Białystok (Poland): Publishing House of Bialystok University of Technology. – 2021. – PP. 63-74.

218. Automation of Flexible HMI Interface Development for Cyber-Physical Production Systems / I. Nevliudov, V. Yevsieiev, N. Starodubcev, N. Demska // International periodic scien-tific journal SWorldJournal. – Issue No9, Part 1. – 2021. – P. 11-27.

219. Невлюдов, И., Стародубцев, Н., Евсеев, В., & Демская, Н. (2021). Automation of Flexible HMI Interface Development for Cyber-Physical Production Systems. SWorldJournal, 1(09-01), 11–27. (https://doi.org/10.30888/2663-5712.2021-09-01-009)

Додаток А
Граф приналежності

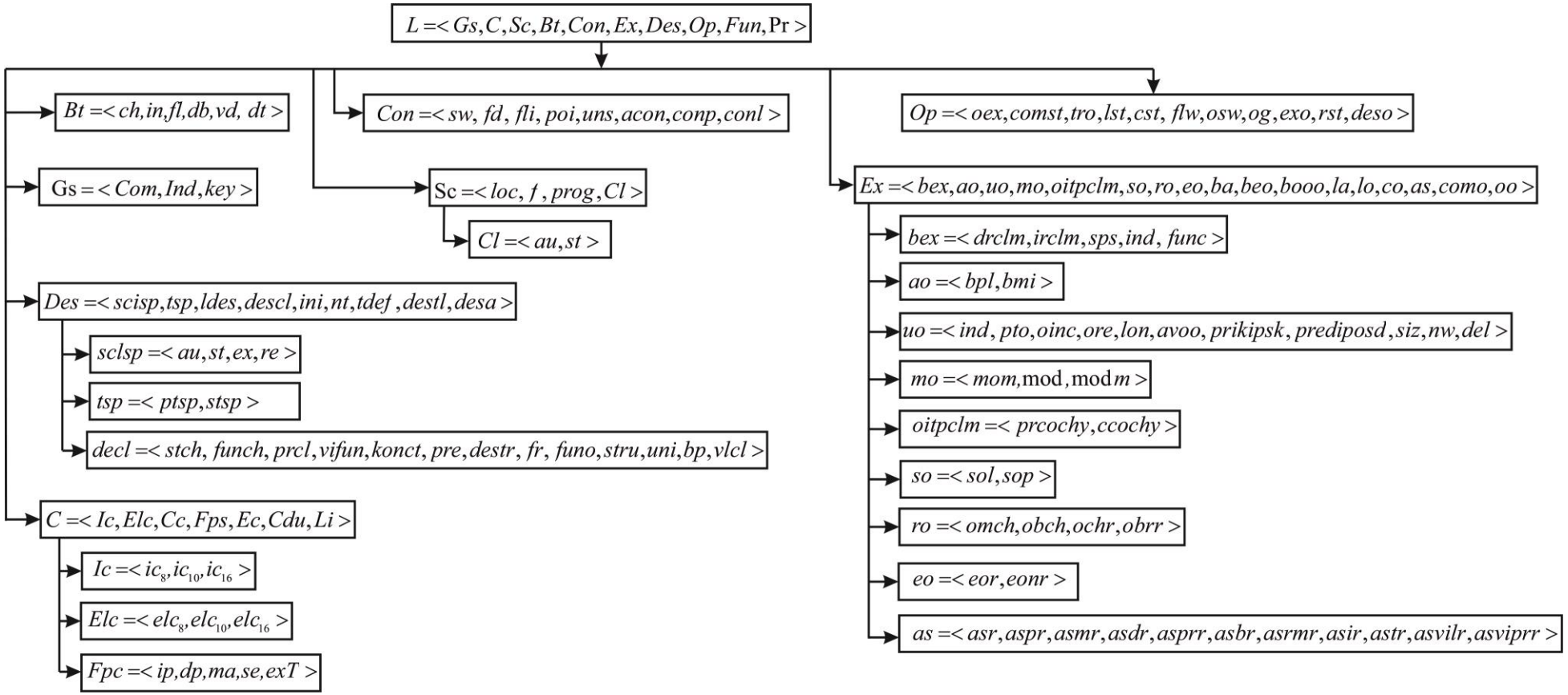


Рисунок А.1 - Граф приналежності

Додаток Б

Фрагмент реалізації Lingustic Variable →Container Solution

Таблиця Б.1 – Фрагмент реалізації Lingustic Variable →Container Solution

	Lingustic Variable	Container Solution	
		cod (Delphi XE3)	cod (Microsoft Visual Studio 2019)
1	2	3	4
Реалізація HMI на базі подій GUI елементів			

1	form_display	if Form_name.ShowModal = mrOk then
		begin
		Form_name.Visible:=true;
		end;
			private void Button1_Click(Object sender, EventArgs e
			)
			{
			Form1 myForm = new Form1();
			myForm.Show();
			}

2	panel_show	Panel_name.visible:=true;	Panelname.Visible = true;
3	dialog_window	if MessageDlg('your text',mtConfirmation, [mbYes,mbNo],0)=mrYes then ShowMessage('your text') else ShowMessage('your text');	var result = MessageBox.Show(‘message’, ‘caption’, MessageBoxButtons.YesNo,MessageBoxIcon.Question); if (result == DialogResult.Yes { MessageBox.Show (‘message’)} else MessageBox.Show (‘message’);
4	switching_ panels	Panel_name1.Visible:=True; Panel_name2.Visible:=False;	Panelname.Visible = true; Panelname.Visible = false;
5	message_box	ShowMessage('your text');	MessageBox.Show (‘message’)
6	button_message _box	MessageBox(Form_name.Handle, 'your text',' Error', MB_ICONERROR + MB_OK);	var result = MessageBox.Show(‘message’, ‘Error, MessageBoxButtons.YesNo, MessageBoxIcon.Error);
7	close	Close;	Form.Close();

8	run_file	ShellExecute(form_name.Handle,nil,'help.hlp',nil,nil,SW_SHOW);	ShellExecute shellExecute = new ShellExecute();
			shellExecute.Path = file;
			shellExecute.Execute();

Продовження таблиці Б.1

1	2	3	4

9	open_file_memo	if OpenTextFileDialog_name.Execute then
		Memo_name.Lines.LoadFromFile(OpenTextFileDialog1.FileName);
			if (openFileDialog1.ShowDialog() ==
			DialogResult.Cancel)
			return;
			string filename = openFileDialog1.FileName;
			string fileText =
			System.IO.File.ReadAllText(filename);
			textBox1.Text = fileText;

10	start_timer	Timer_name.Enabled := true;	myTimer.Enabled = true;

11	filling_progress_bar	if countTime < 20 then
		begin
		label_name.Caption := loadingMessage[countTime];
		inc(countTime);
		ProgressBar_name.Position:=countTime;
		Application.ProcessMessages;
		end
		else
		begin
		countTime:=0;
		end;
			if (countTime < 20)
			{label_name.Caption = loadingMessage[countTime];}
			countTime++;
			ProgressBar_name.Positin = countTime;
			Application.DoEvents();}
			else countTime =0;

12	delete_message	if Key = 46 then
		if (Application.MessageBox('Your text?',' Delete',
		MB_OKCANCEL+MB_ICONEXCLAMATION) = mrOk)
		then
		(Sender as TDBGrid).DataSource.DataSet.Delete;
			if (MessageBox.Show("Do you want to delete this row
			?", "Delete", MessageBoxButtons.YesNo) ==
			DialogResult.Yes)
			{
			dataGridView1.Rows.RemoveAt(dataGridView1.Selec
			tedRows[0].Index);
			sAdapter.Update(sTable);}

Продовження таблиці Б.1

1	2	3	4

13	choice_combobox	var
		s:string;
		begin
		s:=ComboBox_vibor1.Items[ComboBox_vibor1.ItemIndex];
		if s=' condition text 1'
		then
		begin
		// execution of the selected condition 1
		end ;
		if s=' condition text 2'
		then
		begin
		// execution of the selected condition 2
		end
			int selectedIndex = comboBox1.SelectedIndex;
			Object selectedItem = comboBox1.SelectedItem;
			if (comboBox1.SelectedIndex ==1) // execution of the selected condition 1
			else if(comboBox1.SelectedIndex ==2) // execution of the selected condition 2
			....

14	upload_picture	if OpenDialog_name.Execute then
		OpenDialog_name.DefaultExt:=GraphicExtension(TBitmap);
		Image_name.Picture.LoadFromFile(OpenDialog_name.FileName);
			var dialog = new OpenFileDialog();
			dialog.Title = "Open Image";
			dialog.Filter = "bmp files (*.bmp)|*.bmp";
			if (dialog.ShowDialog() == DialogResult.OK)
			{
			var PictureBox1 = new PictureBox;
			PictureBox1.Image(dialog.FileName);
			}
			dialog.Dispose();

Продовження таблиці Б.1

1	2	3	4

15	tabcontrol	if tabcontrol_name.Tabindex=0 then
		Panel_name1.Visible:=true
		else
		Panel1.Visible:=false;
		if tabcontrol_name.Tabindex=1 then
		Panel_name2.Visible:=true
		else
		Panel_name2.Visible:=false;
			if (TabControl1.SelectedIndex = 0){
			panelName1.Visible = true;}
			else panelName1.Visible = false;
			if (TabControl1.SelectedIndex = 1){
			panelName2.Visible = true;}
			else panelName2.Visible = false;

16	tree	if TreeView1.Selected <> nil then
		begin
		TreeView1.Items.AddChild(TreeView1.Selected,
		Edit1.Text);
		TreeView1.Selected.Expanded := True;
		end
		else
		TreeView1.Items.Add(nil, Edit1.Text);
		end;
			if (mySelectedNode != null &&
			mySelectedNode.Parent != null)
			{
			treeView1.SelectedNode = mySelectedNode;
			treeView1.LabelEdit = true;
			treeView1.SelectedNode.Expand();
			if(!mySelectedNode.IsEditing)
			{
			mySelectedNode.BeginEdit();
			}

Реалізація роботи з БД			

17	join db	if IBDatabase_ BD_name.Connected then
		IBDatabase_ BD_name.Close;
		p:= ExtractFilePath(Application.ExeName);
		p:= p + 'bd\BD_name.GDB';
		if not EMBEDDED then
		p:= 'localhost:' + p;
		IBDatabase_ name.DatabaseName := p;
		IBDatabase_ name.Open;
			string connString = @"Data
			Source="+datasource+";Initial Catalog="
			+database+";Persist Security
			Info=True;User
			ID="+username+";Password="+password;
			
			SqlConnection conn = new
			SqlConnection(connString);
			
			return conn;

Продовження таблиці Б.1

1	2	3	4

18	Open db	IBDatabase_ name.Open;
		IBTransaction_ name.StartTransaction;
			SqlConnection conn;
			conn = new SqlConnection(constr);
			conn.Open();

19		IBDataSet_BD_name.Open;	DataSet dataSet = new DataSet("Suppliers"); adapter.Fill(dataSet);
20	request_select_ all	Query_name.SQL.Clear; Query_name.SQL.Add('select * from table_1, table_2'); Query_name.SQL.Add('where table_1.id= table_2.id'); Query_name.Active:=true;	SqlCommand cmd = new SqlCommand("SELECT * FROM table_1,table_2 WHERE table_1.id= table_2.id'", connection)) using (SqlDataAdapter adapter = new SqlDataAdapter(cmd)) { adapter.Fill(table);}
22	update_table	DBGrid_name.Refresh;	datagridview1.Refresh();

23	add_new_entry	if messageDlg('Add?',mtConfirmation,[mbYes,mbNo],0)=mrYes
		then if (Edit_name.Text<>'')then
		IBDataSet_name.Open;
		IBDataSet_name.Append;//добавить новую запись
		IBDataSet_name['bd_field']:=Edit_name.Text;
		IBDataSet_name.Post;
		Edit_name.Text:='';
			var result = MessageBox.Show(“Add?”, “Add new record”, MessageBoxButtons.YesNo, MessageBoxIcon.Question); if (result == DialogResult.Yes) { DataSet dataSet = new DataSet("Suppliers"); DataTable dt = new DataTable("Table"); dt.Columns.Add(new DataColumn("id",typeof(int))); dt.Columns.Add(new DataColumn("bd_field", typeof(string))); DataRow dr = dt.NewRow(); dr["id"] = 123; dr["bd_field"] = Edit_name.Text; dt.Rows.Add(dr); ds.Tables.Add(dt); Edit_name.Text = “”;}

24	delete_entry	if messageDlg(' Delete?',mtConfirmation,[mbYes,mbNo],0)=mrYes then IBDataSet_name.Edit; IBDataSet_name.Delete; DBGrid_name.Refresh;	var result = MessageBox.Show(“Delete?”, “Delete entry”, MessageBoxButtons.YesNo, MessageBoxIcon.Question); if (result == DialogResult.Yes) { DataSet dataSet = new DataSet("Suppliers"); ds.Tables[0].Rows[rowindex].Delete(); ds.AcceptChanges();}

Продовження таблиці Б.1

1	2	3	4
25	edit_entry	IBDataSet_name.Edit; if MessageDlg(' Edit?',mtConfirmation, [mbYes,mbNo],0)=mrYes IBDataSet_name.Edit;	var result = MessageBox.Show(“Edit?”, “Edit entry”, MessageBoxButtons.YesNo, MessageBoxIcon.Question); if (result == DialogResult.Yes) { foreach(DataRow dr in table.Rows) { if(dr["Product_id"] == 123) { dr["Product_name"] = "one"; } }
26	add_field_memo	IBDataSet_name.Edit; IBDataSet_name[' field'] := Memo1.Lines.Text; IBDataSet_name.Post;	DataSet ds = new DataSet("Suppliers"); DataTable dt = new DataTable("Table"); DataRow dr in dt.Rows dr["Product_name"] = TextBox1.Text; ds.AcceptChanges();
27	message_caption	if MessageDlg('your text',mtConfirmation, [mbYes,mbNo],0)=mrYes then ShowMessage('your text') else ShowMessage('your text');	var result = MessageBox.Show(“your text”, “Caption”, MessageBoxButtons.YesNo, MessageBoxIcon.Question); if (result == DialogResult.Yes) {string message = "Message"; MessageBox.Show(message);} else { string message = "Message"; MessageBox.Show(message);}

28	message_add
	_bd
		if messageDlg('your text',mtConfirmation,[mbYes,mbNo],0)=
		mrYes
		then if (Edit_add_name1.Text<>'')or
		(Edit_add_name2.Text<>'') then
		begin
		IBDataSet_name.Open;
		IBDataSet_name.Open;
		IBDataSet_name.Append;
		IBDataSet_name['field name]:=Edit_add_name1.Text;
		IBDataSet_ name ['field name ']:= Edit_add_name2.Text;
		IBDataSet_name;
		end
		else
		IBDataSet_name.Edit;
			var result = MessageBox.Show(“your text”, “Caption”, MessageBoxButtons.YesNo, MessageBoxIcon.Question); if (result == DialogResult.Yes) { DataTable table = new DataTable(); DataColumn column; DataRow row; DataView view; row = table.NewRow(); row["field name 1"] = TextBox1.Text; row["field name 1"] = TextBox2.Text; table.Rows.Add(row); view = new DataView(table); dataGrid1.DataSource = view; }

Продовження таблиці Б.1

1	2	3	4
29	refresh_add	IBDataSet_name.Refresh;	ds.AcceptChanges();
30	transaction_save	Form_name.IBTransaction_name.CommitRetaining; Close;	qlConnection conn; … conn.Close();

31	append_info
	_bd
		IBDataSet_name1.Append;
		IBDataSet_name1['name_field1']:=IBDataSet_name2['name_field2'];
		IBDataSet_name1.Post;
			DataRow row;
			row = table.NewRow();
			row["field name 1"] = TextBox1.Text;
			row["field name 1"] = TextBox2.Text;

32	search_edit	var
		fre:boolean;
		begin
		if Edit_name.Text<>''then
		fre:=table_name.Locate('field',string(Edit_name.Text),
		[loCaseInsensitive,loPartialKey]);
		if fre<>true then
		MessageDlg('Your text',mtError,[mbOK],0)
		else
		form_name.Visible:=true
			bool fre;
			if (TextBox1.Text != “”)
			{ foreach(DataRow dr in table.Rows)
			{ if(dr["field”] == TextBox1.Text)
			{ String s = dr["field”];
			s = s.ToLower();
			dr["field _name"] = s; } } if (fre) {string message = "Message"; MessageBox.Show(message);} } else { base.OnVisibleChanged(e); this.Visible = true; }