ДОДАТОК А

Лістинг додатку

```
#include "Application.h"

#include "JsonItemsSerializer.h"
#include "checklistitem.h"
#include "checklistmodel.h"
#include "testingdocumentsmodel.h"
#include <QQmlContext>

namespace AppSetup
{

class App::AppImpl
{

public:
    void registerTypes()
    {
        ChecklistModel::registerQmlType();
        TestingDocumentsModel::registerQmlType();
    }

    void addContextProperties(QQmlApplicationEngine& _qmlEngine)
    {
        _qmlEngine.rootContext()->setContextProperty(
            "testingDocumentsModel",
m_testingDocumentsModel.get());
    }

    void initAppHandlers()
    {
        m_testingDocumentsModel =
std::make_unique<TestingDocumentsModel>();
        m_testingDocumentsModel->setDocumentSerializer(

std::make_unique<Config::ItemsExporter::JsonItemsSerializer>());
        test_addStructure();
    }

    void cleanupApplication()
    {
    }

private:
    void test_addStructure()
    {
```

```
        auto pHardwareChecklist = m_testingDocumentsModel-
>addTestingDocument("Hardware checks");
        pHardwareChecklist->addChecklistItem("USB шина", "Включить
коннектор, отключить коннектор");
        pHardwareChecklist->addChecklistItem(
            "CAN шина", "Включить коннектор в плату, проверить
логи");

        auto pSoftwareChecklist = m_testingDocumentsModel-
>addTestingDocument("Software checks");
        pSoftwareChecklist->addChecklistItem("UI/UX", "Проверить
положение кнопок на экране");
        pSoftwareChecklist->addChecklistItem("System config",
"Проверить конфиги для разных OS");
    }

private:
    std::unique_ptr<TestingDocumentsModel>
m_testingDocumentsModel;
};

App::App() : m_pAppImpl{std::make_unique<App::AppImpl>()}
{
}

App::~App() = default;

App& App::Instance()
{
    static App appInstance{};
    return appInstance;
}

void App::registerTypes()
{
    m_pAppImpl->registerTypes();
}

void App::addContextProperties(QQmlApplicationEngine& _qmlEngine)
{
    m_pAppImpl->addContextProperties(_qmlEngine);
}

void App::initAppHandlers()
{
    m_pAppImpl->initAppHandlers();
}

void App::cleanupApplication()
{
    m_pAppImpl->cleanupApplication();
}
```

```
} // namespace AppSetup


#include "checklistmodel.h"

ChecklistModel::ChecklistModel(QString title, QObject* parent)
    : QAbstractListModel{parent}, m_title{title}
{
}

int ChecklistModel::rowCount(const QModelIndex& parent) const
{
    if (parent.isValid())
        return 0;

    return m_suites.size();
}

QVariant ChecklistModel::data(const QModelIndex& index, int role)
const
{
    const bool isIndexInvalid = !index.isValid() || index.row() <
0 || index.row() >= rowCount();

    if (isIndexInvalid)
        return QVariant();

    const auto& item = m_suites.at(index.row()).get();
    switch (role)
    {
    case ChecklistItemId:
        item->getId();
        break;
    case ItemTitle:
        return item->getTitle();
        break;
    case ItemSteps:
        return item->getSteps();
        break;
    case IsPassed:
        return item->isPassed();
        break;
    default:
        return QVariant{};
        break;
    }

    return QVariant{};
}

void ChecklistModel::removeChecklistItem(quint64 itemId)
```

```
{

    m_suites.removeAt(itemId);
    emit dataChanged(index(0, 0), index(itemId, 0));
}
void ChecklistModel::addChecklistItem(QString itemTitle, QString
itemSteps)
{
    auto itemId{m_suites.size()};
    auto createdItem = std::make_shared<ChecklistItem>(itemId);

    QQmlEngine::setObjectOwnership(createdItem.get(),
QQmlEngine::CppOwnership);

    createdItem->setTitle(itemTitle);
    createdItem->setSteps(itemSteps);

    beginInsertRows(QModelIndex{}, m_suites.size(),
m_suites.size());
    m_suites.push_back(std::move(createdItem));
    endInsertRows();
    emit dataChanged(index(0, 0), index(m_suites.size() - 1, 0));
}

ChecklistModel::TRoleNames ChecklistModel::roleNames() const
{
    QHash<int, QByteArray> roles;
    roles[ChecklistItemId] = "checkItemId";
    roles[ItemTitle] = "checkTitle";
    roles[ItemSteps] = "stepsDescription";
    roles[IsPassed] = "isPassed";
    return roles;
}

void ChecklistModel::forEachTestCase(ChecklistModel::TEnumerator
enumerator)
{
    for (const auto& item : m_suites)
        enumerator(item);
}

#include "JsonItemsSerializer.h"
#include <fmt/format.h>
#include <nlohmann/json.hpp>

#include <QJsonArray>

namespace Config::ItemsExporter
{

class JsonItemsSerializer::JsonSerializerImpl
{
```

```
public:
    void serializeTo(const QString& outputPath,
gsl::not_null<TestingDocumentsModel*> pModel)
    {
        QJsonArray rootArray;

        pModel->forEachDocument(
            [&rootArray, this](const
std::shared_ptr<ChecklistModel>& checkListModel) {

rootArray.push_back(SerializeCheckListModel(checkListModel));
            });

        auto correctedPath{QUrl(outputPath).toLocalFile()};
        qDebug() << QString::fromStdSt  ring(
            fmt::format("Corrected path is: {}",
correctedPath.toStdString()));
        qDebug() << QString::fromStdString(
            fmt::format("JSON is: ${}",
QJsonDocument(rootArray).toJson(QJsonDocument::Indented)));
        QFile file(correctedPath);
        if (!file.open(QIODevice::ReadWrite | QIODevice::Text))
            return;

        QTextStream out(&file);
        out <<
QJsonDocument(rootArray).toJson(QJsonDocument::Indented);
    }

private:
    QJsonObject SerializeCheckListModel(const
std::shared_ptr<ChecklistModel> checkListModel)
    {
        QJsonObject objectModel;
        objectModel["checklistTitile"] = checkListModel-
>getTitle();

        QJsonArray testCases;

        checkListModel->forEachTestCase(
            [&testCases](const std::shared_ptr<ChecklistItem>&
checkListItem) {
                QJsonObject caseItem;
                caseItem["caseTitle"] = checkListItem->getTitle();
                caseItem["testDescription"] = checkListItem-
>getSteps();
                caseItem["passed"] = checkListItem->isPassed();

                testCases.push_back(caseItem);
            });
        objectModel["testCases"] = testCases;
        return objectModel;
```

```
    }
};

JsonItemsSerializer::~JsonItemsSerializer() = default;

JsonItemsSerializer::JsonItemsSerializer() :
m_pImpl{std::make_unique<JsonSerializerImpl>()}
{
}
void JsonItemsSerializer::serializeTo(
    const QString& outputPath,
    gsl::not_null<TestingDocumentsModel*> pModel)
{
    m_pImpl->serializeTo(outputPath, pModel);
}

} // namespace Config::ItemsExporter

#include "Application.h"
#include <QGuiApplication>
#include <QQmlApplicationEngine>
int main(int argc, char* argv[])
{
#if QT_VERSION < QT_VERSION_CHECK(6, 0, 0)
    QCoreApplication::setAttribute(Qt::AA_EnableHighDpiScaling);
#endif

    qputenv("QT_QUICK_CONTROLS_STYLE", QByteArray("Material"));
    qputenv("QT_QUICK_CONTROLS_MATERIAL_THEME",
QByteArray("White"));

    QGuiApplication app(argc, argv);

    AppSetup::App::Instance().registerTypes();

    QQmlApplicationEngine engine;

    AppSetup::App::Instance().initAppHandlers();
    AppSetup::App::Instance().addContextProperties(engine);

    const QUrl url(QStringLiteral("qrc:/main.qml"));
    QObject::connect(
        &engine,
        &QQmlApplicationEngine::objectCreated,
        &app,
        [url](QObject* obj, const QUrl& objUrl) {
            if (!obj && url == objUrl)
                QCoreApplication::exit(-1);
        },
        Qt::QueuedConnection);
    engine.load(url);
```

```
    QObject::connect(&app, &QGuiApplication::aboutToQuit, [] {
        AppSetup::App::Instance().cleanupApplication();
    });

    return app.exec();
}

#include "testingdocumentsmodel.h"
#include <IModelItemsExporter.h>
#include <fmt/format.h>

TestingDocumentsModel::TestingDocumentsModel(QObject* parent) :
QAbstractListModel(parent)
{
}

int TestingDocumentsModel::rowCount(const QModelIndex& parent)
const
{

    if (parent.isValid())
        return 0;
    return m_checklists.size();
}

QVariant TestingDocumentsModel::data(const QModelIndex& index, int
role) const
{
    qDebug(fmt::format("TestingDocumentsModel:data, index:{0}",
index.row()).c_str());

    if (!index.isValid())
        return QVariant();
    QVariant retValue;
    if (auto it = m_checklists.find(index.row()); it !=
m_checklists.end())
    {
        auto pDocumentItem = it->second.get();
        switch (role)
        {
            break;
        case ChecklistTitleRole:
            return pDocumentItem->getTitle();
            break;
        case ChecklistModelRole:
            retValue.setValue(pDocumentItem);
            return retValue;
            break;
        default:
            return QVariant{};
        }
    }
```

```
    return QVariant();
}

TestingDocumentsModel::TRoleNames
TestingDocumentsModel::roleNames() const
{
    QHash<int, QByteArray> roles;
    roles[ChecklistModelIdRole] = "documentId";
    roles[ChecklistTitleRole] = "documentTitle";
    roles[ChecklistModelRole] = "checkListModel";

    return roles;
}

std::shared_ptr<ChecklistModel>
TestingDocumentsModel::addTestingDocument(QString documentName)
{
    beginInsertRows(QModelIndex{}, m_checklists.size(),
m_checklists.size());
    auto itemId = m_checklists.size();
    auto pTestingDocument =
std::make_shared<ChecklistModel>(documentName);
    QQmlEngine::setObjectOwnership(pTestingDocument.get(),
QQmlEngine::CppOwnership);
    m_checklists.insert({itemId, pTestingDocument});
    endInsertRows();

    emit dataChanged(index(0, 0), index(rowCount() - 1, 0));
    qDebug(fmt::format("Rows count:{0}", rowCount() - 1).data());
    return pTestingDocument;
}

void TestingDocumentsModel::setDocumentSerializer(
    std::unique_ptr<Config::ItemsExporter::IItemsSerializer>&&
documentsSerializer)
{
    m_documentsSerializer = std::move(documentsSerializer);
}

void
TestingDocumentsModel::forEachDocument(TestingDocumentsModel::TEnu
merator enumerator)
{
    for (const auto& [itemKey, itemValue] : m_checklists)
    {
        enumerator(itemValue);
    }
}

void TestingDocumentsModel::native_addTestingDocument(QString
documentName)
{
```

```
    auto pItem = addTestingDocument(documentName);
}

Q_INVOKABLE void TestingDocumentsModel::storeConfigTo(QString
filePath)
{
    if (m_documentsSerializer)
        m_documentsSerializer->serializeTo(filePath, this);
}

#ifndef APPLICATION_H
#define APPLICATION_H

#include <QQmlApplicationEngine>
#include <memory>

namespace AppSetup
{

class App
{

public:
    static App& Instance();

public:
    void registerTypes();

    void setPhonesCount(std::size_t _phonesCount);

    void addContextProperties(QQmlApplicationEngine& _qmlEngine);

    void initAppHandlers();

    void cleanupApplication();

private:
    App();
    ~App();

private:
    class AppImpl;
    std::unique_ptr<AppImpl> m_pAppImpl;
};

} // namespace AppSetup
#endif // APPLICATION_H


#ifndef CHECKLISTITEM_H
#define CHECKLISTITEM_H
```

```
#include <QObject>

class ChecklistItem : public QObject
{
    Q_OBJECT
public:
    explicit ChecklistItem(quint64 itemId, QObject* parent =
nullptr)
        : QObject(parent), m_id{itemId}, m_isPassed{false},
m_title{}, m_steps{}
    {
    }

    Q_PROPERTY(quint64 itemId MEMBER m_id READ getId CONSTANT)
    Q_PROPERTY(QString itemTitle MEMBER m_title READ getTitle
WRITE setTitle NOTIFY titleChanged);
    Q_PROPERTY(QString stepsDescription MEMBER m_stepsDescription
READ getSteps WRITE setSteps
                  NOTIFY stepsChanged);

public:
    QString getTitle() const noexcept
    {
        return m_title;
    }

    void setTitle(const QString& newTitle)
    {
        m_title = newTitle;
        emit titleChanged(newTitle);
    }

    QString getSteps() const noexcept
    {
        return m_steps;
    }

    void setSteps(const QString& newSteps)
    {
        m_steps = newSteps;
        emit stepsChanged(newSteps);
    }

    quint64 getId() const noexcept
    {
        return m_id;
    }

    bool isPassed() const noexcept
    {
        return m_isPassed;
    }
```

```
    void setPassed() noexcept
    {
        if (!m_isPassed)
        {
            m_isPassed = true;
            emit passedChanged(m_isPassed);
        }
    }

    void resetPassed() noexcept
    {
        if (m_isPassed)
        {
            m_isPassed = false;
            emit passedChanged(m_isPassed);
        }
    }

signals:
    void titleChanged(QString newTitle);
    void stepsChanged(QString newStepsDescription);
    void passedChanged(bool isPassed);

private:
    bool m_isPassed;
    quint64 m_id;
    QString m_title;
    QString m_steps;
};

#endif // CHECKLISTITEM_H

#ifndef CHECKLISTMODEL_H
#define CHECKLISTMODEL_H

#include <QAbstractListModel>
#include <QQmlEngine>
#include <QtCore>
#include <memory>
#include <vector>

#include "checklistitem.h"

class ChecklistModel : public QAbstractListModel
{
    Q_OBJECT

public:
    explicit ChecklistModel(QString title, QObject* parent =
nullptr);
```

```
public:
    enum Roles
    {
        ChecklistItemId = Qt::UserRole,
        ItemTitle,
        ItemSteps,
        IsPassed
    };
    Q_PROPERTY(
        QString documentTitle MEMBER m_title READ getTitle WRITE
setTitle NOTIFY titleChanged);

public:
    Q_INVOKABLE void removeChecklistItem(quint64 itemId);
    Q_INVOKABLE void addChecklistItem(QString itemTitle, QString
itemSteps);

public:
    int rowCount(const QModelIndex& parent = QModelIndex()) const
override;

    QVariant data(const QModelIndex& index, int role =
Qt::DisplayRole) const override;

    using TRoleNames = QHash<int, QByteArray>;
    TRoleNames roleNames() const override;

    static void registerQmlType()
    {
        qmlRegisterUncreatableType<ChecklistModel>(
            "app.models", 1, 0, "ChecklistModel", "interface");
        qRegisterMetaType<ChecklistModel*>("ChecklistModel *");
    }

    using TEnumerator = std::function<void(const
std::shared_ptr<ChecklistItem>&)>;
    void forEachTestCase(TEnumerator enumerator);

public:
    QString getTitle() const noexcept
    {
        return m_title;
    }

    void setTitle(const QString& newTitle)
    {
        m_title = newTitle;
        emit titleChanged(newTitle);
    }

signals:
    void titleChanged(QString newTitle);
```

```
private:
    using ChecklistItemsStorage =
QVector<std::shared_ptr<ChecklistItem>>;
    QString m_title;

    ChecklistItemsStorage m_suites;
};

#endif // CHECKLISTMODEL_H

#ifndef IMODELITEMSEXPORTER_H
#define IMODELITEMSEXPORTER_H
#include <QString>
#include <checklistitem.h>
#include <checklistmodel.h>
#include <gsl/pointers>
#include <memory>
#include <testingdocumentsmodel.h>
namespace Config::ItemsExporter
{

class IItemsSerializer
{

public:
    virtual void serializeTo(
        const QString& outputPath,
        gsl::not_null<TestingDocumentsModel*> pModel) = 0;
    virtual ~IItemsSerializer() = default;
};
} // namespace Config::ItemsExporter

#endif // IMODELITEMSEXPORTER_H

#ifndef JSONITEMSSERIALIZER_H
#define JSONITEMSSERIALIZER_H
#include <IModelItemsExporter.h>

#include <memory>
namespace Config::ItemsExporter
{

class JsonItemsSerializer : public IItemsSerializer
{
public:
    JsonItemsSerializer();
    ~JsonItemsSerializer();

public:
    void serializeTo(const QString& outputPath,
gsl::not_null<TestingDocumentsModel*> pModel)
```

```
        override;

private:
    class JsonSerializerImpl;
    std::unique_ptr<JsonSerializerImpl> m_pImpl;
};

} // namespace Config::ItemsExporter
#endif // JSONITEMSSERIALIZER_H

#ifndef TESTINGDOCUMENTSMODEL_H
#define TESTINGDOCUMENTSMODEL_H

#include "checklistmodel.h"
#include <QAbstractListModel>
#include <QQmlEngine>
#include <QtCore>
#include <map>
#include <memory>

namespace Config::ItemsExporter
{
class IItemsSerializer;
}

class TestingDocumentsModel : public QAbstractListModel
{
    Q_OBJECT

public:
    explicit TestingDocumentsModel(QObject* parent = nullptr);

    enum Roles
    {
        ChecklistModelIdRole = Qt::UserRole,
        ChecklistTitleRole,
        ChecklistModelRole
    };

    int rowCount(const QModelIndex& parent = QModelIndex()) const
override;

    QVariant data(const QModelIndex& index, int role) const
override;

    using TRoleNames = QHash<int, QByteArray>;
    TRoleNames roleNames() const override;

public:
    Q_INVOKABLE void native_addTestingDocument(QString
documentName);
    Q_INVOKABLE void storeConfigTo(QString filePath);
```

```
public:
    std::shared_ptr<ChecklistModel> addTestingDocument(QString
documentName);
    void setDocumentSerializer(
        std::unique_ptr<Config::ItemsExporter::IItemsSerializer>&&
documentsSerializer);

    using TEnumerator = std::function<void(const
std::shared_ptr<ChecklistModel>&)>;
    void forEachDocument(TEnumerator enumerator);

public:
    static void registerQmlType()
    {
        qmlRegisterUncreatableType<TestingDocumentsModel>(
            "app.models", 1, 0, "TestingDocumentsModel",
"interface");
    }

private:
    using TDocumentsStorage = std::map<quint64,
std::shared_ptr<ChecklistModel>>;
    std::unique_ptr<Config::ItemsExporter::IItemsSerializer>
m_documentsSerializer;
    TDocumentsStorage m_checklists;
};

#endif // TESTINGDOCUMENTSMODEL_H
```

ДОДАТОК Б

Презентація

Автоматизація тестування спеціалізованої комп'ютерної системи на основі вбудованих RFID-датчиків

Виконав:
студент групи СКСм-20-1
Садкова Марія
Володимирівна

Керівник:
зав. каф. АПОТ
проф. Чумаченко С.В.

Харківський національний університет радіоелектроніки, Харків, Україна,
2021р

Зміст

- Актуальність проблеми
- Умови тестування
- Техніки тест-дизайну
- Тестові-артефакти
- Тестування апаратного додатку
- Реалізація моделі системи
- Висновки

Актуальність проблеми

- Тестування починається з організаційних документаційних робіт ще до створення продукту. Таким чином, зараз все активніше починають набувати популярність різноманітни баг-трекінгові системи та додатки для автоматизації тестування, які базуються на ручній роботі тестувальників та допомогають оптимізувати процес.
- Метою кваліфікаційної роботи є створення прототипу системи для автоматизації тестування апаратного додатку з вбудованим RFID датчиком, логування результатів роботи додатку, ведення тестової документації та безпосереднього створення тестових артефактів.

Основні види тестування

- функціональні;
- нефункціональні;
- пов'язані зі змінами.

Умови тестування

Визначення необхідних умов:

- Необхідними умовами істинності твердження А називаються умови, без дотримання яких А не може бути істинним;
- Достатніми називаються такі умови, при наявності (виконання, дотриманні) яких твердження А є істинним.

Техніки тест дизайну

- Еквівалентний Поділ;
- Аналіз Граничних Значень;
- Причина/Наслідок;
- Передбачення помилки;
- Вичерпне тестування;
- Парне тестування.

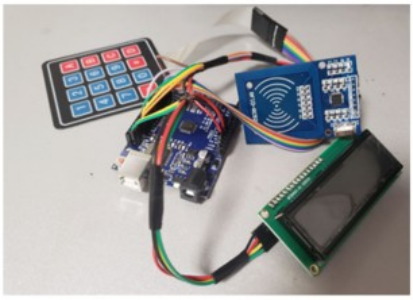
Тестові артефакти

- план тестування (Test Plan);
- Чек-ліст;
- Test Case;
- Баг репорт.

Тестовий спеціалізований апаратний додаток з вбудованим RFID-датчиком

Вимоги що до фукнціоналу додатку:

- реалізація зчитування поточного балансу карти;
- поповнення картки;
- зменшення балансу картки;
- скидання балансу картки до дефолтних установок;
- відображення балансу та операції на дісплеї терміналу;
- можливість живлення приладу від різних источників.



Техніка гранічних значень

Слід звернути увагу, що кнопки A та B мають у собі арифметичні функції додавання та віднімання, що еквівалентно поповненю та , відповідно, зняття умовних кошт з RFID-картки

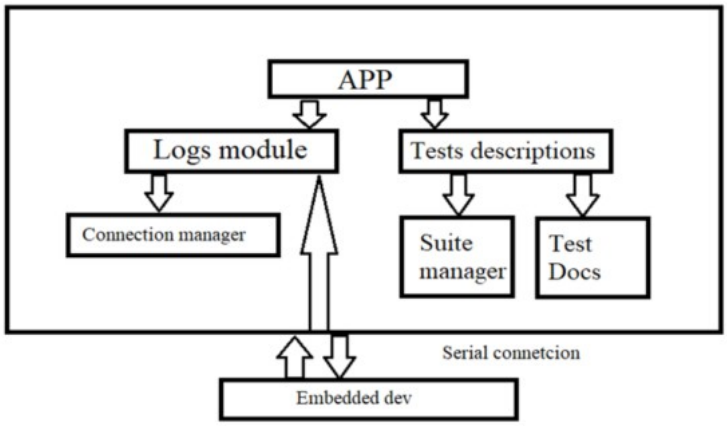
Стартовий баланс RFID-картки	Виконувана функція	Гранічне значення перевірки	Очікуваний результат
0	A	1	1
1	A	1000	1001
1	B	1	0
1	B	2	0

Баг-репорт

Текс

Заголовок	Блимання надписі після використання терміналу
Проект	Апаратний додаток з вбудованим RFID-датчиком
Важливість	Minor
Приорітет	Medium
Статус	Новий
Шаги відтворення	1)Підключити термінал до живлення 2)Ввести на клавіатурі число 1 3)Натиснути кнопку у A 4)Прикласти до RFID-датчика RFID-карту 5)Натиснути кнопку C 6) Прикласти RFID-карту з пункту 4
Очікуваний результат	На дісплеї після шагу 3 відобразився надпис «Replenished Ok» Після шагу 5 на дісплеї віобразився надпис «Balance:1»
Фактичний результат	На дісплеї після шагу 3 відобразився надпис «Replenished Ok» Після шагу 5 на дісплеї віобразився надпис «Balance:1» Після цього відобразився надпис «Hello Player!» Актуально для дій A,B,C,#

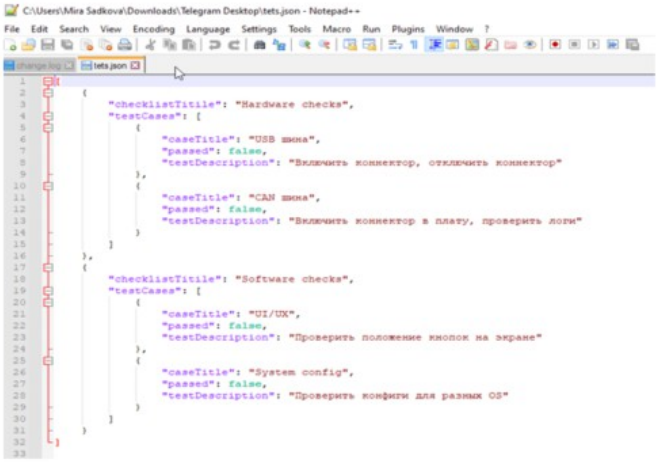
Модель прототипу системи



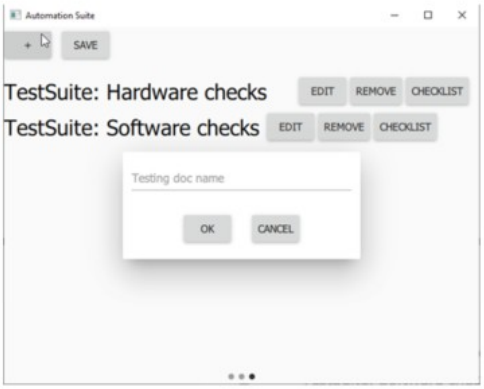
Фрагмент реалізації головного вікна програми


```
ApplicationWindow{
    width: 640
    height: 480
    visible: true
    title: qsTr("Automation Suite")
    id:rootWindow

    SwipeView{
        id: mainView
        currentIndex: 0
        anchors.fill: parent

        MainPageSelector
        {

        }
        LoggerComponentView
        {id:loggerView}

        PageTestingDocumentsView
        { id: testingDocsView}}}
```

Експорт файлу створеної документації



Модальне вікно додавання чек–ліста




```
RowLayout
{
    spacing:28
    Layout.fillWidth: true;
    Layout.alignment: Qt.AlignHCenter;
    Button
    {
        id: acceptButton;
        text: "OK"
        onClicked:
        {
            if(inputText.text!=="")
                addNewItemRequested(inputText.text);
        }
    }
    Button
    {
        id: cancelButton;
        text: "Cancel"
        onClicked:
        {
            addNewDocumentSetPopup.close();
        }
    }
}
}
```

Заключення

- Була розроблена модель автоматизації функціонального тестування;
- Дана модель являє собою програмний додаток та дозволяє вирішити логування роботи апаратного додатку та проблему ведення тестової документації ;
- Логування роботи апаратного додатку дає можливість точного сберігати логи, які використовуються для подальшого розвитку продукту, а також додавати до баг-репортів як низкорівневий доказ наявності несправності;
- Разроблена модель дає можливість вирішити проблему ведення тестової документації для зручного збереження артефактів та обновлення їх до актуальної версії продукту, що дозволяє більш структуровано зберігати інформацію.

Перспективи дослідження

- додавання окремого функціоналу баг-трекінгу;
- інтеграція з іншими системами ведення менежменту проектів;
- створення мобільної версії додатку;
- оптимізація існуючої кодової бази;
- додавання різних форматів експорту тестових артефактів.

ДОДАТОК В

В.1 Публікація на тему “ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ ЯЗЫКОВ ОПИСАНИЯ АППАРАТУРЫ VHDL И VERILOG”

ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ ЯЗЫКОВ ОПИСАНИЯ АППАРАТУРЫ VHDL И VERILOG

Садковая М.В.
Научный руководитель – Филиппенко И.В.
Харьковский национальный университет радиоэлектроники
(61166, Харьков, пр. Науки, 14, каф. АПВТ, тел. (050)861-15-14)
e-mail: mariia.sadkova@nure.ua

VHDL and Verilog are hardware descriptive languages. These languages are designed for simulate electronic circuits at the level of valve, register transmissions, microcircuit cases. Therefore, they can be called languages of through fbnunctional and logical design. However, they have a number of differences, which we will consider in this article.

Языки VHDL и Verilog относятся к языкам описания аппаратуры. Эти языки предназначены для моделирования электронных схем на уровнях вентилей, регистровых передач, корпусов микросхем. Поэтому их можно назвать языками сквозного функционально-логического проектирования. Однако они имеют ряд отличий, которые необходимо учитывать при проектировании различных цифровых устройств.

VHDL (Very high speed integrated circuits Hardware Description Language) – данный язык предназначен для описания проектируемых систем на схемотехническом уровне проектирования и замены классического подхода к схемотехническому проектированию на уровне отдельных элементов. Язык позволяет описывать цифровые системы на алгоритмическом уровне. При помощи специального программного обеспечения описание на языке VHDL преобразовывается в схему на уровне простейших элементов цифровой электроники.

Verilog – это язык описания аппаратуры, используемый для разработки и моделирования электронных систем. Этот язык (также известный как Verilog HDL) позволяет осуществить проектирование, верификацию и реализацию (например, в виде СБИС) аналоговых, цифровых и смешанных электронных систем на различных уровнях абстракции.

По сравнению с Verilog, VHDL более богатый и строго типизованный и строго детерминистический язык, более детализированный. В результате проекты, написанные на VHDL, считаются самодокументированными. Синтаксис сильно отличается от стиля языка С, и инженеры, работающие в VHDL, постоянно сталкиваются с необходимостью явного преобразования из одного типа данных в другой. VHDL часто сразу показывает ошибки, которые пропускает Verilog, а так же имеет подчеркнуто однозначно недвусмысленную семантику, и поэтому легче переносится между разными системами разработки (в том смысле, что перенос точнее переносит все тонкости работы исходного проекта).

Verilog выглядит более гармонично и удобочитаемо. Что достигается отсутствием длинных названий типов данных и строк объявления сигналов и регистров. Так же проще и аккуратнее реализована запись векторов.

Алфавит моделирования в VHDL включает в себя 9 значений: {'U', 'X', '0', '1', 'Z', 'W', 'L', 'H', '-'}. Verilog же имеет всего 4 значения: {'0', '1', 'X', 'Z'}.

Verilog слабо проверяет типы, и более краток, с эффективной нотацией. Он также детерминистический. Все типы данных заранее определены в Verilog, и каждый из них имеет битовое представление, по сравнению с VHDL, в котором пользователь имеет возможность вводить свои типы данных. Синтаксис похож на С. Из-за своей структуры VDHL отлавливает больше ошибок уже на ранних стадиях процесса разработки. С другой стороны Verilog позволяет инженерам быстро описывать модели.

Так же в VHDL, для расширения языка, позволяется использовать внешние пакеты и библиотеки, в то время как, Verilog не имеет подобной возможности. Из-за своей структуры VDHL отлавливает больше ошибок уже на ранних стадиях процесса разработки. С другой стороны Verilog позволяет инженерам быстро описывать модели.

Таким образом, VHDL является более академичным, многословным и сложным языком. Требуется написание большего объема кода, но строгость означает, что он с большей вероятностью будет работать. Verilog проще для типичного цифрового дизайна, но, соответственно, упрощает создание сложных ошибок. Выбор одного или другого зависит от используемых инструментов. Например, некоторые из популярных инструментов FPGA лучше работают с VHDL, когда популярные инструменты ASIC улучшают работу с Verilog

Список источников:

1. microsin URL: http://microsin.net/programming/xilinx/difference-between-vhdl-verilog-systemverilog.html

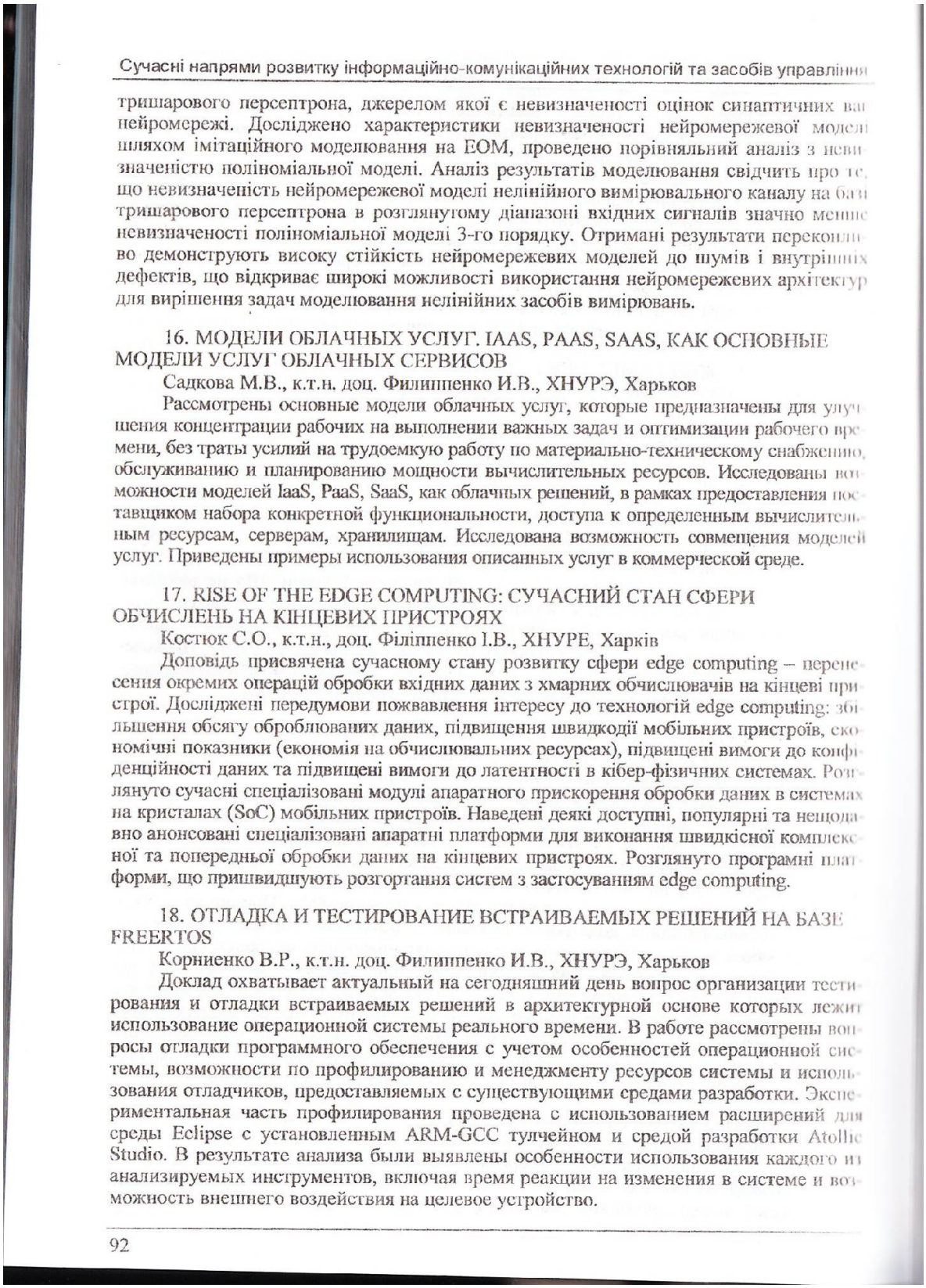
2. VHDL с нуля. // easyelectronics URL: http://we.easyelectronics.ru/plis/vhdl-s-nulya.html

3. Исследование комбинационных устройств: // URL: http://we.easyelectronics.ru/plis/vhdl-s-nulya.html

4. Особенности языков описания архитектуры // parallel URL: https://parallel.ru/fpga/hdl.html

5. habr URL: https://habr.com/ru/post/191606/

В.2 Публікація на тему “МОДЕЛИ ОБЛАЧНЫХ УСЛУГ: IAAS, PASS, SAAS, КАК ОСНОВНЫЕ МОДЕЛИ УСЛУГ ОБЛАЧНЫХ СЕРВИСОВ ”



тришарового персептрона, джерелом якої є невизначеності оцінок синаптичних ва нейромережі. Досліджено характеристики невизначеності нейромережевої моделі шляхом імітаційного моделювання на ЕОМ, проведено порівняльний аналіз з невизначеністю поліноміальної моделі. Аналіз результатів моделювання свідчить про те, що невизначеність нейромережевої моделі нелінійного вимірювального каналу на бази тришарового персептрона в розглянутому діапазоні вхідних сигналів значно менше невизначеності поліноміальної моделі 3-го порядку. Отримані результати переконливо демонструють високу стійкість нейромережевих моделей до шумів і внутрішніх дефектів, що відкриває широкі можливості використання нейромережевих архітектур для вирішення задач моделювання нелінійних засобів вимірювань.

16. МОДЕЛИ ОБЛАЧНЫХ УСЛУГ. IAAS, PAAS, SAAS, КАК ОСНОВНЫЕ МОДЕЛИ УСЛУГ ОБЛАЧНЫХ СЕРВИСОВ

Садкова М.В., к.т.н. доц. Филиппенко И.В., ХНУРЭ, Харьков

Рассмотрены основные модели облачных услуг, которые предназначены для улучшения концентрации рабочих на выполнении важных задач и оптимизации рабочего времени, без траты усилий на трудоемкую работу по материально-техническому снабжению, обслуживанию и планированию мощности вычислительных ресурсов. Исследованы возможности моделей IaaS, PaaS, SaaS, как облачных решений, в рамках предоставления поставщиком набора конкретной функциональности, доступа к определенным вычислительным ресурсам, серверам, хранилищам. Исследована возможность совмещения моделей услуг. Приведены примеры использования описанных услуг в коммерческой среде.

17. RISE OF THE EDGE COMPUTING: СУЧАСНИЙ СТАН СФЕРИ ОБЧИСЛЕНЬ НА КІНЦЕВИХ ПРИСТРОЯХ

Костюк С.О., к.т.н., доц. Філіппенко І.В., ХНУРЕ, Харків

Доповідь присвячена сучасному стану розвитку сфери edge computing – перенесення окремих операцій обробки вхідних даних з хмарних обчислювачів на кінцеві пристрої. Досліджені передумови пожвавлення інтересу до технологій edge computing: збільшення обсягу оброблюваних даних, підвищення швидкодії мобільних пристроїв, економічні показники (економія на обчислювальних ресурсах), підвищені вимоги до конфіденційності даних та підвищені вимоги до латентності в кібер-фізичних системах. Розглянуто сучасні спеціалізовані модулі апаратного прискорення обробки даних в системах на кристалах (SoC) мобільних пристроїв. Наведені деякі доступні, популярні та нещодавно анонсовані спеціалізовані апаратні платформи для виконання швидкісної комплексної та попередньої обробки даних на кінцевих пристроях. Розглянуто програмні платформи, що пришвидшують розгортання систем з застосуванням edge computing.

18. ОТЛАДКА И ТЕСТИРОВАНИЕ ВСТРАИВАЕМЫХ РЕШЕНИЙ НА БАЗЕ FREERTOS

Корниенко В.Р., к.т.н. доц. Филиппенко И.В., ХНУРЭ, Харьков

Доклад охватывает актуальный на сегодняшний день вопрос организации тестирования и отладки встраиваемых решений в архитектурной основе которых лежит использование операционной системы реального времени. В работе рассмотрены вопросы отладки программного обеспечения с учетом особенностей операционной системы, возможности по профилированию и менеджменту ресурсов системы и использования отладчиков, предоставляемых с существующими средами разработки. Экспериментальная часть профилирования проведена с использованием расширений для среды Eclipse с установленным ARM-GCC тулчейном и средой разработки Atollic Studio. В результате анализа были выявлены особенности использования каждого из анализируемых инструментов, включая время реакции на изменения в системе и возможность внешнего воздействия на целевое устройство.