УДК 62—50:007:57

В. В. ТИЩЕНКО, И. В. ГОЛИУС

ПРИКЛАДНЫЕ АСПЕКТЫ ИМИТАЦИОННОЙ МЕТОДИКИ ПРОЕКТИРОВАНИЯ МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ КИБЕРНЕТИЧЕСКИХ СИСТЕМ

Современное имитационное моделирование, положенное в основу методики проектирования, является важнейшим элементом создания новых систем. Оно охватывает все стадии проектирования последних и служит экспериментальной базой разработки проекта, в основу которого положены бионические методы анализа.

Сам процесс проектирования — малоформализованный, творческий процесс, для которого не существует достаточно определенных правил и методик, позволяющих управлять ходом разработки больших кибернетических систем.

Поэтому методически будет правильно, если придерживаться бионического анализа в системном подходе к разработке новой системы и выделить три неформальные стадии проектирования: концептуальную, логическую и программную. На концептуальной

стадии проектирования предусматривают разработку задания, выработку спецификаций и концепций системы. Стадия логического проектирования находит свое отражение в графической форме представления моделей систем в виде схем, графов, сетей Петри и другом виде математического аппарата систем. Стадия программного проектирования является практическим шагом проектирования, выражающимся в фактическом построении имитационных моделей и планировании эксперимента с моделью проекта. Эта стадия состоит из этапов программирования, отладки совместно с тестированием и верификацией и сопровождения, включающего коррекцию программ и адаптацию к пользователю.

Заканчивается постадийное проектирование систем проведением имитационного эксперимента, который в случае неудачи может образовать цикл повторения неформальной методики разработки проектных работ.

В аспекте разработки математического обеспечения кибернетических систем (МОКС) на концептуальной стадии проектирования используют понятие процесса, как основного концепта МОКС. Интерпретация процесса на логической стадии осуществляется в статике — графами $G(x, y)$ и $C(x, y)$, а в динамике — сетями Петри $S(x, y)$. В принятом обозначении x — множество вершин, y — комплект направленных дуг графа.

Традиционно полагают, что элементарным процессом являются вершины графа $G(x, y)$ или дуги графа $C(x, y)$ или позиции и хотя бы один переход в сети Петри $S(x, y)$. Такое представление называют формальными процессами [1], что послужило основой для анализа их схем.

Реальный процесс и его отображение в вычислительной системе — это комплекс элементарных процессов управления, обработки и распределения ресурса. Отображение реального процесса в терминах сети Петри $S=(P, T, I, Q)$, где $P=\{p_1, p_2 \dots p_n\}$ — конечное множество позиций $n \geqslant 0$; $T=\{t_1, t_2 \dots t_n\}$ — конечное множество переходов $m>0$; $I: T \to P^\infty$ — входная функция-отображение из переходов в комплекты позиций; $Q: T \to P^\infty$ — выходная функция-отображение из переходов в комплекты позиций [2], назовем ее имитационной моделью процесса.

Опишем простейшую имитационную модель вычислительного процесса

$$t_1(p_0 \wedge p_3 \wedge p_4) \vdash p_1(t_1);$$

$$t_2(p_6 \wedge p_1) \vdash p_2(t_2) \wedge p_3(t_2);$$

$$t_3(p_1 \wedge p_5) \vdash p_9(t_3);$$

$$t_4(p_3 \wedge p_8) \vdash p_{10}(t_4);$$

$$t_5(p_7 \wedge p_{11}) \vdash p_3(t_5),$$

где p_0, p_1, p_2, p_3 — позиции, моделирующие состояние элементов вычислительного процесса; p_4, p_5, p_6, p_7, p_8 — позиции, моделирующие семафоры управления процессом; p_9, p_{10}, p_{11} — позиции, мо-

делирующие вспомогательные элементы процесса; t_1, t_2, t_3, t_4, t_5 — переходы, моделирующие фрагменты вычислительного процесса.

Динамика функционирования имитационной модели состоит в последовательно-параллельном перемещении фишки через переход t_1, где $i=1...5$. Причем движение фишки может быть разрешено только в том случае, когда каждая позиция перехода t_i имеет хотя бы по одной фишке.

Каждый переход в имитационной модели процесса отражает специфические особенности вычислительного процесса. Например, переход является переходом, определяющим готовность процесса. Вектор состояния t_1 определяется множествами нормальных состояний $C_{G1}=\{C_{G10}, C_{G11}\}$, где C_{G10} — исходное состояние, C_{G11} — готовность процесса и ошибочных состояний $C_{G2}=\{C_{G20}, C_{G21}\}$, где C_{G20}, C_{G21} — блокировки по одному и двум факторам управления соответственно. Аналогично, спецификации перехода t_2 отражают завершение процесса, t_3 — связывается с аварийным завершением, а t_4, t_5 — с управлением ресурсами. Наличие или отсутствие фишек в позициях любого перехода определяют состояние перехода, а в целом состояния переходов обусловливают состояние процесса. Динамика всех переходов модели процесса за определенный квант времени определяет пространство состояний модели процесса, которое можно представить как таблицу пространства состояний, прокомпозированную по кодам состояний.

Модель вычислительного процесса в моменты функционирования (имитации) может последовательно замыкать цепочку состояний путем перехода из состояния в состояние. В этой имитационной модели процесса рассматриваются три схемы смены состояний:

нормальная

$$C_{G10} \to C_{G11} \to C_{S31} \to C_{G10},$$

нормальная, но с задержкой в функционировании

$$C_{G10} \to C_{G20} \to C_{G21} \to C_{G11} \to C_{S31} \to C_{G10},$$

ошибочная

$$C_{G10} \to C_{G11} \to C_{S40} \to C_{A51} \to C_{G10}.$$

Последняя схема, как правило, завершается аварийно и в исходной состояние устанавливается принудительно.

Модели вычислительных процессов принято рассматривать в зависимости от функций, выполняемых в системе, как процессы входные, выходные, вычислительные и др. Каждый из видов процессов различаются спецификациями и, следовательно, структурой сети Петри. Пространство состояний таких процессов будет не идентичным, но определив базовой моделью процессов вычислительный процесс, все другие процессы можно получить из этой модели, если применить следующие процедуры генерации:

вставки

$$t_1\{(p_0 \wedge p_3 \wedge p_4)\} \perp t_1(\{t_6(p_{12} \wedge p_{13}) \vee t_7(p_{14} \wedge p_{15})\} \wedge p_3 \wedge p_4,$$

развития

$$p_{10}(t_4) \rightarrow \{t_6(p_{12} \wedge p_{13}) \vee t_7(p_{14} \wedge p_{10})\},$$

свертки $\{t_1(p_0 \wedge p_3 \wedge p_4) \vdash p_1/t_1)\} \leftarrow \{t_2(p_1)\} \vdash p_1(t_1)$ и другие где $\perp$, $\leftarrow$, $\rightarrow$ знаки процедур, $\{\}$ — участок, подлежащий обработке процедурой.

Если соединить модели процессов по входным и выходным позициям модели, то можно образовать цепочки процессов, если процессы последовательные, или сети процессов, если процессы последовательно-параллельные. В этом случае необходима иерархическая модель управления, использующая один из последовательных факторов управления: конвейерный, параллельный, распределенный и осуществляющая управление ресурсами на базе распределения или кооперирования. Назовем системы управления одного уровня локальными, а каждого старшего уровня — глобальными. Все системы управления одного уровня могут реализовывать различные принципы управления ресурсами и процессами, но все уровни принципиально одинаковы. В процессе имитации в соответствии с описанием образовать цепочку системы управления, которая может управлять отдельным участком сети моделей вычислительного процесса.

Имитационное управление отдельной моделью процесса функционально полезно расчленить на системные элементы: принятие решения, включающее выработку управляющих воздействий и инициализацию процесса; исполнение решения — идентификация, классификация и фиксация состояний, а также выделение ресурса; контроль состояния — чтение и анализ состояния.

Все позиции участка сетевой модели процессов можно сгруппировать в форме вектора индикации состояний модели процессов. Динамика состояний процесса образует дерево пространства состояний (ДПС), тогда ДПС отражает свойства модели процессов на уровне системы. Назовем вектор индикации состояния процесса блоком состояния процесса (БСП), который содержит поля состояния процесса, состояния ресурса, а также поля иерархии власти управления.

Управление моделью процессов путем изменения БСП осуществляется локальной системой управления, которая на основе анализа БСП вносит в него изменения с помощью языка инструкций.

Оценка эффективности разрабатываемой модели МОКС играет немаловажную роль, так как для каждого класса процессов управление может быть эффективным или нет в зависимости от специализации системы управления. Для этой цели используем триплет эффективности МОКС $Q=<A, B, C>$, где A — коэффициент эффективности по управлению (КЭУ); B — коэффициент эффективности по обслуживанию (КЭО); C — коэффициент эффективности по ресурсам (КЭР).

КЭУ оценивается как $A=A_i/A_0+A_i$, где $A_i=a_{i1}+a_{i2}$ — текущий параметр количества асинхронных и синхронных процессов пользователя; A_0 — нормативный параметр количества системных про-

цессов. КЭУ определяет эффективность системы управления для определенного класса задач и его значение находится в интервале $0<A\leqslant 1$.

О функционировании системы для определенного класса задач можно судить по КЭО, который находим из соотношения $B=B_i/B_0+B_i$, где $B_i=b_{i1}+b_{i2}+b_{i3}$ — текущий параметр количества активных, готовых к выполнению и заблокированных процессов в очереди; B_0 — общая длина очереди системных и пользовательских процессов по всем очередям в задаче. КЭО нормирован по времени и его значение находится в интервале $0<B\leqslant 1$.

КЭР оценивает использование памяти для определенного класса задач для данной системы управления и определяется $C=C_1/C_2$, где C_1 — общее количество процессов, размещенных на нормированном участке памяти; $C_2=C_{i1}C_{i2}$ — количество нормированных фрагментов памяти (количество страниц, разделов) для данного класса задач.

КЭУ, КЭО, КЭР оцениваются как средние коэффициенты для класса задач и могут служить приближенной оценкой целесообразности использования МОКС. Практически триплет эффективности МОКС представляет функцию, выраженную в форме графика в трехмерном пространстве с оптимальным значением $A=B=C=1$.

Список литературы: 1. *Питерсон Дж.* Теория сетей Петри и моделирование систем. М., 1984. 264 с. 2. *Котов В. Е.* Сети Петри. М., 1984. 160 с. 3. *Валях Е.* Последовательно-параллельные вычисления. М., 1985. 456 с.

Поступила в редколлегию 15.03.88