УДК 681.3.06

# ВИЗУАЛЬНЫЙ УРОВЕНЬ ПРЕДСТАВЛЕНИЯ АЛГОРИТМОВ ФУНКЦИОНИРОВАНИЯ РАСПРЕДЕЛЁННЫХ СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ НА ЯЗЫКЕ СТРУКТУРНОГО МОДЕЛИРОВАНИЯ

В.К. Погребной

Институт «Кибернетический центр» ТПУ
E-mail: vk@ad.cctpu.edu.ru

*Предлагается развитие концепции модульной технологии проектирования распределённых систем реального времени в направлении разработки средств автоматизации процессов эволюции моделей систем. Вводится понятие языка структурного моделирования (языка SML) и соответствующей технологии структурного моделирования (SML-технологии). Инструментальные средства SML-технологии разрабатываются в соответствии с правилами семантической среды функционирования систем, представленными в базе знаний и определяющими область возможных эволюций моделей. Разработан визуальный уровень представления исходных моделей систем в форме графа потока данных на языке SML. Предложен ряд механизмов доопределения моделей по обновлению данных и доступу к ресурсам, которые дают представление о характере операций, выполняемых инструментальными средствами SML-технологии в процессе эволюции моделей.*

***Ключевые слова:***
*Система реального времени, модульная структура, граф потока данных, структурное моделирование, эволюционное проектирование, язык SML, SML-технология.*

## Введение в структурное моделирование

Концепция имитационного моделирования распределённых систем реального времени (СРВ), изложенная в работе [1], развивает технологию модульного проектирования в направлении создания моделей, выполняемых на виртуальной машине моделирования (ВММ), названных активными. Принцип активности моделей, наряду с принципами дискретности и графовой формы представления, создаёт условия для реализации эволюционного подхода к проектированию СРВ. Эффективность применения эволюционного подхода определяется функциональной полнотой имеющейся совокупности формализованных процедур по трансформации модели и может быть отражена соответствующим принципом модульной технологии – принципом эволюционности. Можно сказать, что данный принцип отражает стремление строить модель СРВ в таком виде, чтобы разработчику проекта максимально упростить составление задания на трансформацию модели, а все «заботы» по отработке этого задания переложить на инструментальную систему и ВММ.

Цикл эволюции модели в соответствии с предложенной концепцией в упрощенном варианте представлен на рис. 1.

Центральное место в цикле эволюции занимает модель СРВ, представленная модульной структурой в форме графа потока данных (ГПД) [2] и модель архитектуры и топологии сети вычислительной системы [3], реализующей прикладные функции СРВ. В первом блоке осуществляется анализ результатов моделирования, полученных при выполнении текущего варианта активной модели на ВММ (блок 3), и формируются задания на изменение условий функционирования СРВ в надежде на улучшение качества проекта. Внесение изменений в модульную структуру модели осуществляется разработчиком проекта в интерактивном режиме. При этом могут выполняться предварительные расчёты для принятия решений по изменению модели. Принятые решения отображаются непосредственно на ГПД модульной структуры либо в описателях его компонентов.

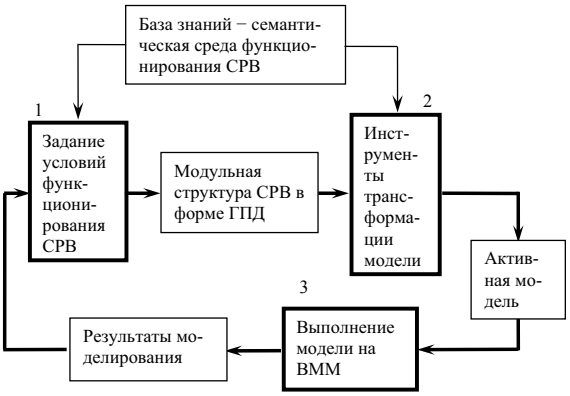


***Рис. 1.*** *Цикл эволюции модели*

Второй блок с помощью процедур трансформации преобразует внесённые в модульную структуру изменения в форму активной модели и настраивает её для выполнения на ВММ. Таким образом, модульная структура в форме ГПД выступает в роли промежуточного языка общения между разработчиком проекта (блок 1) и инструментальными средствами трансформации и настройки активной модели (блок 2).

Учитывая, что общение разработчика проекта (блок 1) с инструментальной системой моделирования (блоки 2 и 3) осуществляется на структурном уровне представления модели в форме ГПД, данный язык будем именовать языком структурного моделирования (Structural Modelling Language – SML), а соответствующую технологию структурного моделирования – SML-технологией. Модульная структуризация модели в SML-технологии используется не только для её представления в форме

ГПД, но и при выполнении модели на ВММ. С этой целью ГПД модели системы преобразуется в специальную форму описания, названную SML-программой, где каждый модуль воспринимается как команда ВММ. Полные сведения об алфавите языка SML и синтаксических правилах описания условий функционирования СРВ содержатся в базе знаний, показанной на рис. 1 отдельным блоком.

База знаний включает описание совокупности объектов языка SML, их атрибутов, признаков, параметров, отношений, механизмов реализации отдельных правил, которые все вместе определяют семантическую среду (эволюционное пространство) функционирования СРВ. Здесь следует отметить, что разработчик проекта в основном имеет дело лишь со структурной составляющей семантической среды функционирования СРВ, используя визуальное представление модульной структуры модели в форме ГПД.

В итоге разработчик проекта, придерживаясь правил языка SML, с помощью программных средств, содержащихся в блоке 1, задаёт **что** должна делать СРВ, а блок 2 знает **как** это делать, трансформируя и доопределяя модель соответствующим образом. При этом разработчик проекта должен иметь в виду, что **знания** блока 2 ограничены содержанием семантической среды функционирования СРВ, заложенным в базе знаний.

Содержание базы знаний можно условно разделить на 2 части. В первую из них входят объекты языка SML, которые использует разработчик проекта при построении модульной структуры модели и задании условий функционирования СРВ. Вторая часть, напротив, содержит информацию для работы блока 2 при выполнении трансформаций и настройке активной модели. Ниже в статье рассматриваются основные объекты из первой части базы знаний, которые доступны разработчику проекта. Приводится также ряд скрытых от разработчика проекта механизмов по доопределению модели, относящихся к второй части базы знаний и используемых блоком 2 при выполнении своих функций.

### Модульная структура модели и её визуальное представление в форме ГПД

В соответствии с технологией модульного проектирования модель алгоритма функционирования системы представляется модульной структурой, получаемой путём композиции модулей, взятых из заранее созданной библиотеки. Библиотека создаётся под конкретную предметную область и при построении модульной структуры системы может пополняться недостающими модулями, как правило, в составе групп $A$ и $F$ [1]. Вопросы создания библиотеки модулей и композиции из них модульных структур в данной работе не рассматриваются. Основное внимание уделяется изложению правил языка SML по визуальному представлению модульной структуры в форме ГПД и отображению на нём условий функционирования системы.

На рис. 2 приведён пример модульной структуры модели, представленной в форме ГПД. Имитацию оборудования объекта осуществляют модули $A_1$ и $A_2$, реализующие алгоритмы функционирования соответствующих подсистем. Автономное управление технологическими параметрами первой подсистемы выполняют две прикладные функции. Первая из них выполняется модулями $F_5$ и $F_9$, вторая функция – модулями $F_6$, $F_{10}$, $F_{13}$. Автономное управление технологическими параметрами второй подсистемы выполняет третья функция – модули $F_8$, $F_{12}$, $F_{14}$. Четвёртая функция – модули $F_7$ и $F_{11}$, анализирует работу подсистем и управляет параметрами, оптимизирующими работу объекта в целом. Пятая функция – модули $T_4$, $H_{15}$, $S_{16}$, $H_3$, запускается циклически таймером $T_4$ и на основе анализа состояния оборудования объекта и параметров технологического процесса выполняет подстройку параметров оборудования.

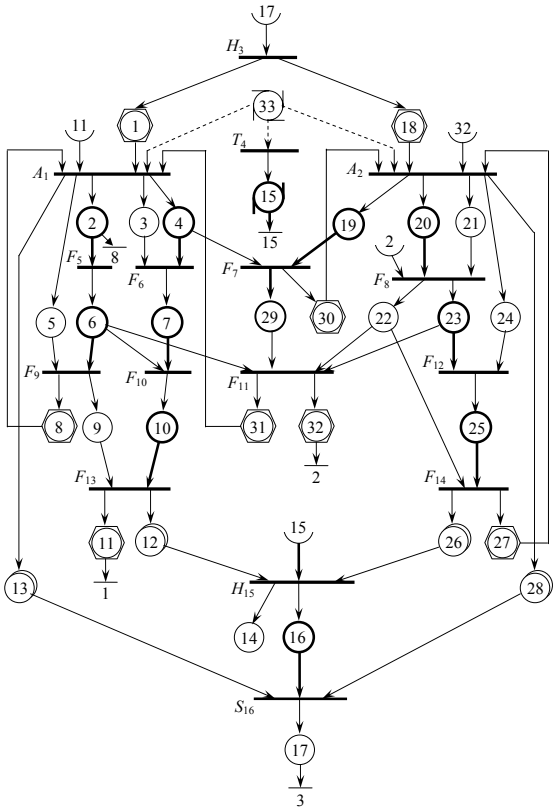


***Рис. 2.*** *Пример модульной структуры модели системы в форме ГПД*

Перечисленные прикладные функции на основе входных данных (показаний датчиков), полученных имитаторами подсистем объекта (модули $A_1$ и $A_2$), вычисляют управляющие воздействия (позиции $d_1$, $d_{18}$, $d_{30}$, $d_8$, $d_{31}$, $d_{32}$, $d_{27}$), которые поступают на исполнительные механизмы объекта. Таким образом, представление модульной структуры (рис. 2), которое в графовой форме отражает последовательность преобразования данных модулями прикладных функций, получило название графа потока данных (ГПД). Заметим, что ГПД в таком виде отражает функциональный уровень выполнения прикладных функций СРВ без учёта динамики их взаимодей-

ствия. При выполнении такого ГПД на ВММ можно проследить наличие и исполнение всех технологических алгоритмов по управлению объектом, предусмотренных техническим заданием.

Основными компонентами модульной структуры являются модули, позиции и дуги. Модули в ГПД, независимо от принадлежности к группам *A, F, H, G, T, S* [1], имеют одинаковое изображение в виде планок. Порядковые номера модулей проставляются рядом с планками и помечаются принадлежностью к одной из групп, например $A_1$, $T_4$, $F_5$, $H_{15}$. Позиции изображаются кружками и нумеруются независимо от модулей. Порядковые номера позиций проставляются внутри кружков. Как видно из рис. 2 позиции в ГПД имеют различные изображения в виде определённых дополнений к кружку. Эти дополнения в языке SML стандартизованы и облегчают разработчику проекта визуальное восприятие ГПД. Перечень видов изображений позиций с указанием их групп [1] и смыслового содержания дополнительных признаков приведено в таблице.

***Таблица.*** *Изображение позиций графа, рис. 2*

| № | Виды позиций | Смысловое содержание дополнительного признака | Группа позиции |
|---|---|---|---|
| 1 | ○, ○ | Состояние данных | D |
| 2 | ◯, ◯ | Совокупность присоединенных состояний данных | D |
| 3 | ○, ○ | Сигнальные данные | P |
| 4 | ○, ○ | Значение счетчика | K |
| 5 | ◯, ◯ | Выделенное показание таймера | D |
| 6 | ◯ | Управляющее воздействие | U |
| 7 | ◯ | Разделяемый ресурс | R |
| 8 | ◯ | Потребляемый ресурс | Q |
| 9 | ◯ | Общая область памяти | Z |
| 10 | ◯ | Буфер | B |
| 11 | ◯ | Начальные условия | D |
| 12 | ◯ | Сигнал запуска агрегатов и таймеров | P |

Первые пять видов позиций имеют по два варианта изображения. Первый из них является базовым и воспринимается модулем как информационный вход (выход). Позиция второго варианта изображения (выделена жирно) содержит дополнительный признак – требование на обработку, что соответствует входу запуска модуля.

Дуги в ГПД изображаются линиями трёх типов – сплошная тонкая, сплошная жирная и пунктирная. Тонкая линия соответствует информационным дугам, связывающим модули и позиции. Жирными линиями связываются входные позиции запуска с модулями, которые они запускают. Например, на рис. 2 позиция 4 является входом запуска для модуля $F_6$ и обычным информационным входом для модуля $F_7$. Пунктирными линиями изображаются дуги, входящие в модуль из позиций видов 7–12, таблица. На рис. 2 это дуги, выходящие из позиции 33.

Для улучшения визуального восприятия ГПД допускаются разрывы отдельных дуг. На рис. 3 представлены правила изображения разрыва дуги, выходящей из позиции $d_i$ в модуль $m_j$ (рис. 3, *а*) и дуги, выходящей из модуля $m_j$ в позицию $d_i$ (рис. 3, *б*). Разрывы дуг по правилу, показанному на рис. 3, *а*, можно видеть на рис. 2, например для дуги ($d_2$, $m_8$).

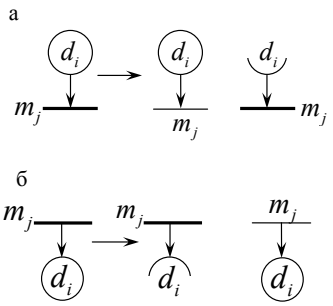


***Рис. 3.*** *Изображение разрыва дуги*

Согласно [1] структура модуля включает три блока: операционный блок, блок состояний и блок адаптации модуля. Каждый блок содержит описание на языке SML параметров модуля, относящихся к соответствующему блоку, условий совместной работы модулей в модели и условий функционирования объекта управления, влияющих на динамику работы модуля. Для удобства описание параметров операционного блока включается в описание блока адаптации, а описание блока состояний структурировано на описатели отдельных позиций. Это позволяет разработчику проекта визуально для любого модуля ГПД кликом вызывать и редактировать содержание соответствующего блока адаптации и, аналогично, для любой позиции ГПД – её описатель.

При построении ГПД все позиции изображаются как состояния данных без требования на обработку (вид 1 в таблице). Изображение позиций с дополнительными признаками происходит автоматически после указания этого признака в описателе позиции. Аналогично, в исходном варианте ГПД все дуги изображаются тонкими линиями. Жирные и пунктирные дуги появляются в ГПД после введения соответствующих признаков в описатели позиций.

Приведённый на рис. 2 ГПД отображает на позициях и дугах ряд признаков, которые автоматически перенесены из описателей позиций и блоков адаптации модулей. На последующих этапах доопределения модели часть сведений из описателей блоков модулей автоматически или вручную также может быть отображена на ГПД. Но такие этапы доопределения модели не связаны с её структурными свойствами и в данной работе не рассматриваются. Поэтому ниже в качестве примеров излагаются только те механизмы доопределения моделей, которые не отображаются на ГПД, т. е. являются скрытыми от разработчика проекта, но описываются средствами языка SML на структурном уровне.

**Механизмы обновления выходных данных с присоединением состояний**

Операционный блок модуля при однократном исполнении получает одно состояние совокупности выходных данных $d_i$. В общем случае, при выполнении модуля $K$ раз, для каждого данного $d_i$ будет получена совокупность состояний $\{d_{i1},...,d_{ik},..,d_{iK}\}$. Число сохраняемых состояний из данной совокупности не зависит от числа срабатываний модуля и обычно ограничивается одним или несколькими последними состояниями. При сохранении ограниченного числа состояний после очередного срабатывания модуля происходит обновление одного из сохраняемых состояний на вновь полученное. Для случаев сохранения одного состояния различается три условия его обновления:

- ОС – обновление текущего состояния на вновь полученное (С – обновление);
- ЦО – циклическое обновление текущего состояния с временем цикла срабатывания модуля $\Delta T$ (Ц – обновление);
- ТО – обновление состояния в момент времени $t$, удовлетворяющий условию $t–t^*\le\Delta t$, где $t^*$ – момент поступления данных с требованием на обработку с целью получения состояния, а $\Delta t$ – допустимый интервал времени на получение состояния (Т – обновление).

Если необходимо сохранять несколько последних состояний, то выполняется обновление с присоединением очередного состояния к сохраняемой совокупности. В этом случае возможны следующие условия обновления с присоединением:

- ЦП – циклическое присоединение $K$ состояний, $K$=const (Ц – присоединение);
- КП – присоединение $K$ состояний, $K$=const (К – присоединение);
- ТП – присоединение состояний, полученных за интервал времени $\Delta t$, $\Delta t$=const (Т – присоединение);
- ВП – присоединение $B$ состояний, $B\ne$const, полученных за интервал времени, определяемый числом $K(m_j)$, $K(m_j)$=const, последних запусков некоторого модуля $m_j$ в ГПД (В – присоединение).

Отличие Ц – обновления от С – обновления заключается в том, что при С – обновлении момент получения нового состояния совпадает с моментом завершения работы модуля, а при Ц – обновлении момент получения состояния связывается с моментом окончания цикла $\Delta T$, независимо от реального момента завершения работы модуля в цикле $\Delta T$.

Условие Т – обновления предполагает контроль момента получения состояния. При моделировании такой контроль выполняется практически всегда и осуществляется с целью анализа работоспособности системы. Функции контроля реализуются на структурном уровне и являются скрытыми от разработчика проекта. Если при этом момент времени $t$ и $t^*$ и, соответственно величина $\Delta t$, должны оцениваться более точно, чем это позволяют такты моделирования, то вводятся более точные таймеры. На рис. 4, *а*, показан ГПД, реализующий функцию контроля для Т – обновления. Здесь модуль $m_{j1}$ формирует состояние позиции $i_1$ с требованием на обработку для получения модулем $m_{j2}$ управляющего воздействия (позиция $i_2$) через 0,8 такта моделирования. Скрытая часть ГПД включает две позиции $t_k$, формируемые ВММ, состояния которых обозначают моменты завершения выполнения соответствующих модулей. Позиция $t$ показывает время, которое отсчитывает таймер $T$ с начала получения управляющего воздействия. Модуль синхронизации $S$ выдаёт информацию (позиция $\Delta$) для ВММ в случае, если время получения управляющего воздействия превышает установленные 0,8 единиц.

Механизм обновления по условиям ЦП и КП показан на рис. 4, *б*. Модуль $F$ запускается позицией $i_2$ и включает очередное состояние, полученное модулем $m_{j1}$, в совокупность состояний, определяемую позицией $i_2^*$. Состояние, включаемое в совокупность, сопровождается временем получения $t_k$, а состояние с наименьшим временем получения $t_k$ исключается из совокупности так, чтобы в ней всегда сохранялось не более $K$ состояний доступных модулю $m_{j2}$. Для случаев, когда модуль $m_{j2}$ дол-

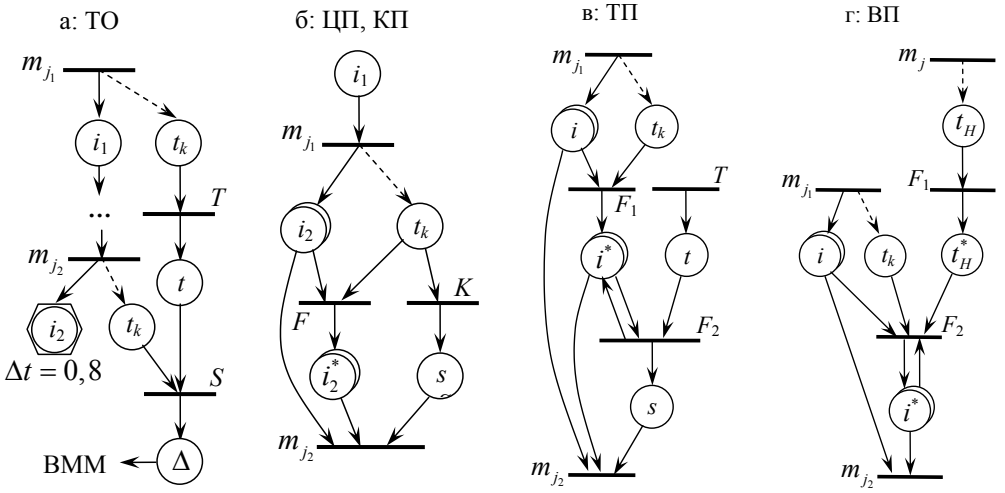


***Рис. 4.*** *Структурный уровень представления механизмов обновления*

жен запускаться в $r$ раз реже, чем модуль $m_{j1}$, скрытая часть ГПД дополняется счётчиком $K$ и позицией $s$. Счётчик $K$ запускается позицией $t_k$ и после $r$ запусков формирует сигнал в позиции $s$, которая запускает модуль $m_{j2}$ и разрешает доступ к совокупности состояний в позиции $i_2^*$. Заметим также, что дуга ($i_2$, $m_{j2}$) в работе рассматриваемого механизма обновления не участвует.

Обновление по условию ТП представлено на рис. 4, *в*. Скрытая часть ГПД представлена модулями $F_1$, $F_2$, таймером $T$ и позициями $t_k$, $i^*$, $t$, $s$. Модуль $F_1$ запускается позицией $t_k$ и включает состояние позиции $i$, получаемое модулем $m_{j1}$, в совокупность состояний, определяемую позицией $i^*$. Модуль $F_2$ запускается в каждом такте моделирования, генерируемым таймером $T$, и исключает из совокупности (позиция $i^*$) состояние с моментом получения $t_k \le t+1-\Delta t$. Позиция $s$, состояние которой формируется модулем $F_2$, в каждый такт моделирования разрешает модулю $m_{j2}$ доступ к совокупности состояний, полученных за текущий интервал времени $\Delta t$. Если модуль $m_{j2}$ необходимо запускать через некоторое установленное число тактов, например 12, то модуль $F_2$ дополняется функцией счётчика так, чтобы сигнал в позиции $s$ формировался через 12 тактов.

Сказанное выше иллюстрируется на рис. 5 диаграммой работы модуля $m_{j1}$ и вариантами совокупностей состояний, полученных для $\Delta t$=20 тактов. Здесь же показаны варианты совокупностей доступных модулю $m_{j2}$ через 12 тактов.

Скрытая часть ГПД, реализующая обновление по условию ВП, представлена на рис. 4, *г*, и включает модули $F_1$, $F_2$ и позиции $t_k$, $t_н$, $i_н^*$, $i^*$. В механизме обновления по условию ВП формирование совокупности состояний, осуществляемое модулем $m_{j1}$, ставится в зависимость от запусков некоторого модуля $m_j$. Модуль $m_{j1}$ должен присоединять все состояния, полученные не ранее $K(m_j)$ последних запусков модуля $m_j$. С этой целью модуль $F_1$ запускается позицией $t_н$, в которой с помощью ВММ фиксируется очередной момент запуска модуля $m_j$, и формируется позиция $i_н^*$. Состояние позиции $i_н^*$ указывает наиболее ранний запуск модуля $m_j$ из $K(m_j)$ последних запусков. В начале моделирования состояние позиции $i_н^*$ принимается равным 0. Это значение сохраняется в позиции до $K(m_j)$ запусков модуля $m_j$, после чего $i_н^*$ заменяется на момент первого запуска. При следующем запуске $i_н^*$ становится равным моменту второго запуска и т. д. Модуль $F_2$ запускается позицией $t_k$, которая фиксирует момент завершения работы модуля $m_{j1}$, и если $t_k > i_н^*$, то очередное состояние позиции $i$ заносится в совокупность состояний, определяемую позицией $i^*$, и из неё исключаются все состояния с моментами получения $t_k \le i_н^*$. В рассматриваемом ГПД, также как и в ГПД на рис. 4, *в*, дуга ($i$, $m_{j2}$) в работе не участвует. Пример формирования совокупностей состояний по условию ВП для случая $K(m_j)$=3 показан на рис. 5 в виде диаграммы, расположенной ниже оси времени.

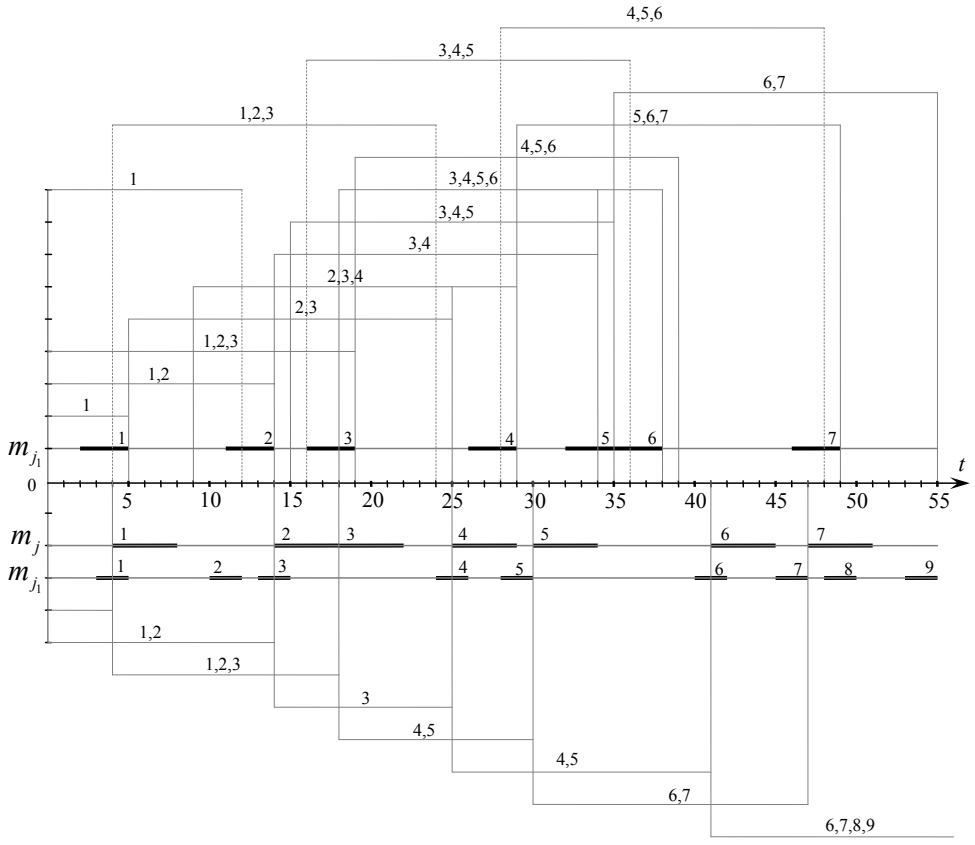


***Рис. 5.*** *Иллюстрации к обновлению по условиям Т-присоединения (выше оси времени) и В-присоединения (ниже оси времени)*

### Механизмы доступа модулей к ресурсам

Необходимость организации доступа к потребляемому $Q$ и разделяемому $R$ ресурсам возникает всякий раз, когда несколько модулей (больше одного) обращаются к одному ресурсу. Ресурс $Q$ является потребляемым и расходуется модулем в заранее оговоренных объёмах и не возвращается. Поэтому потребление данного ресурса предполагает постоянный контроль его наличия и при необходимости пополнения.

Возможное время использования ресурса $R$ модули делят между собой. Принимается, что модуль захватывает ресурс $R$ в момент запуска и освобождает его в момент завершения работы. Ресурсная позиция группы $R$ может описывать один разделяемый ресурс или несколько, например, $r$ единиц. В последнем случае использовать ресурс $R$ могут одновременно до $r$ модулей. При этом предполагается, что все $r$ единиц ресурса являются равноценными.

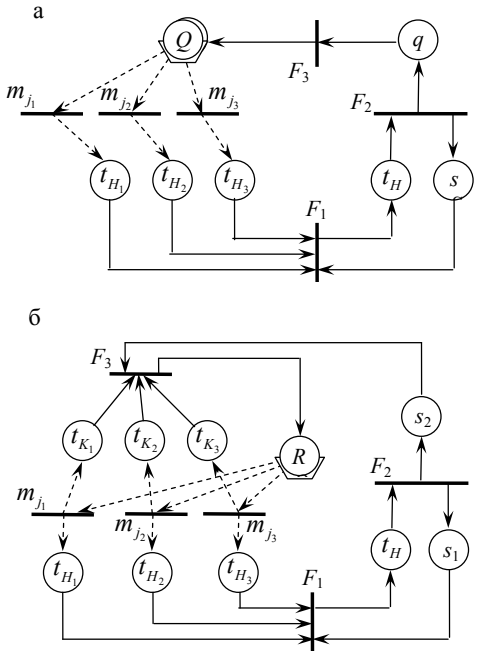


***Рис. 6.*** *Механизмы доступа к ресурсам Q и R*

На рис. 6, *а*, представлен ГПД, реализующий доступ к потребляемому ресурсу $Q$ модулей $m_{j1}$, $m_{j2}$, $m_{j3}$. Скрытая часть ГПД реализуется модулями $F_1$, $F_2$, $F_3$. Модуль $F_1$ запускается позицией $s$, состояние которой сигнализирует о том, что выделение ресурса предыдущему модулю завершено. Начальное состояние позиции $s$ содержит сигнал запуска модуля $F_1$. Учитывая, что в данной схеме доступа наличие ресурса всегда гарантировано, модуль $F_1$ реализует простой алгоритм выбора модуля для предоставления ресурса. Первым ресурс получает модуль с наиболее ранним началом выполнения. Исходя из этого модуль $F_1$ после запуска ждёт появления в позициях $t_{н1}$, $t_{н2}$, $t_{н3}$ сигнала о начале работы, выбирает соответствующий модуль и указывает его в позиции $t_н$. Модуль $F_2$ выделяет ресурс выбранному модулю и в позицию $s$ помещает сигнал о запуске модуля $F_1$. В модуле $F_2$ также принимается решение о необходимости пополнения ресурса и сообщение об этом помещается в позиции $q$. Пополнение ресурсов в позиции $Q$ осуществляет модуль $F_3$.

Скрытый ГПД, реализующий механизм доступа к разделяемому ресурсу $R$, представлен на рис. 6, *б*. Модули $m_{j1}$, $m_{j2}$, $m_{j3}$ потребляют ресурс $R$ по очереди, если он имеется в одном экземпляре. Если ресурс $R$ содержит $r$ равноценных единиц, то воспользоваться ресурсом может несколько модулей одновременно. Предлагаемая схема реализует обе ситуации по составу ресурса $R$.

Модуль $F_1$ в ждущем режиме опрашивает позиции $t_{н1}$, $t_{н2}$, $t_{н3}$ и выбирает модуль для выделения ресурса и информацию о нём помещает в позицию $t_н$. Модуль $F_2$ запускается позицией $t_н$ и выполняет следующие действия: проверяет наличие ресурса для выделения очередному модулю; уменьшает ресурс в позиции $R$, выделяя его модулю, или фиксирует недостаточное количество единиц (отсутствие) ресурса; формирует сигнал в позиции $s_2$ о возможности возвращения ресурса и сигнал запуска модуля $F_1$ в позиции $s_1$. Сигнал в позиции $s_2$ запускает модуль $F_3$, который ждёт появления момента завершения работы модуля в позициях $t_{к1}$, $t_{к2}$, $t_{к3}$ и по информации позиции $s_2$ освободившийся ресурс возвращает в позицию $R$.

При счётном ресурсе $R$ позиция $s_2$ выступает в роли буфера, накапливающего информацию от модуля $F_2$ о начале работы модулей и их потребностях в ресурсе, а модуль $F_3$ эту информацию извлекает и возвращает соответствующий ресурс.

### Язык SML и автоматизация процесса эволюции модели

Изложенные выше правила языка SML по визуальному представлению модели СРВ в форме ГПД отражают структурную составляющую описания алгоритмов функционирования модели. Текстовая составляющая описания модели, сосредоточенная в блоках адаптации модулей и описателях позиций, в данной статье не рассматривается. Вместе с тем результаты, полученные при разработке технологии структурного моделирования уже на данном этапе исследований позволяют сделать ряд обобщений относительно роли языка SML и соответствующих инструментальных средств в решении проблемы автоматизации процесса эволюции модели.

Технология структурного моделирования (SML-технология) основана на систематическом использовании принципов модульного проектирования распределённых СРВ и в первую очередь ориентирована на развитие эволюционного подхода к проектированию. Принципиально важным для SML-технологии является то, что эволюция модели происходит в рамках заранее разработанной семантической среды, определяющей область допустимых условий функционирования СРВ. Инструментальные программные средства поддержки

SML-технологии включают редактор языка SML, систему автоматизации трансформаций модели, систему формирования SML-программы, операционную систему ВММ (блок управления процессом моделирования), ВММ для выполнения SML-программы. Участие разработчика проекта в эволюционном процессе ограничивается составлением и, с помощью редактора SML, отображением на уровне ГПД очередного варианта условий функционирования модели. Инструментальные средства поддержки SML-технологии трансформируют модель под эти условия и преобразуют её в активную форму (SML-программу), а ВММ под управлением операционной системы выполняет активную модель, имитируя функционирование СРВ.

Таким образом, в каждом цикле эволюции модель СРВ последовательно изменяет своё состояние. Выделяются три основных состояния, которые с помощью инструментальных средств SML-технологии преобразуются из одного в другое. Исходное состояние модели формируется разработчиком проекта с помощью редактора языка SML. Оно включает визуальное представление структурной составляющей модели в форме ГПД и исходные состояния описателей блоков адаптации модулей и позиций. Здесь же задаются исходные условия функционирования модели в динамике, а также параметры исходного варианта архитектуры и топологии сети вычислительной системы, на которой будет реализована СРВ.

Второе состояние получается с помощью системы автоматизации трансформаций, которая выполняет множество операций по доопределению и настройке моделей, в том числе скрытых от разработчика проекта, формированию процессов и операций контроля возможности их совместной работы. Выполняя эти операции, система следует предписаниям блоков адаптации модулей и описателей позиций по условиям функционирования модели в динамике. В итоге модель преобразуется в активную форму в виде совокупности параллельных процессов с непротиворечивой системой отношений по взаимодействию. Третье состояние в виде SML-программы получается после выполнения настроек и контрольных функций по подготовке активной модели к выполнению на ВММ.

Выделение трёх состояний модели в значительной степени является условным, т.к. после построения исходной модели в форме ГПД выполняется большое число преобразований, последовательность выполнения которых не всегда предопределена. В результате с каждым преобразованием полнота и детальность описания модели возрастает и постепенно приближается к получению SML-программы. Средства языка SML позволяют представлять модель на любом уровне детализации описания. В этом смысле язык SML обладает ярко выраженным свойством иерархии, когда задание некоторого условия на верхнем уровне иерархии предполагает переход к более детальному его описанию с учётом других условий функционирования СРВ. В сравнении с известными языками моделирования, в том числе с универсальным языком моделирования UML [4], язык SML не является языком программирования модели. Это язык визуального представления модели на структурном уровне и задания на ней условий функционирования СРВ в рамках заранее разработанной замкнутой семантической среды, определяющей область возможных эволюций моделей при проектировании распределённых и жёстких СРВ.

Принципиально важным для языка SML является также и то, что разработчик проекта оперирует самым верхним слоем иерархии языка и, в основном, на визуальном уровне структурного представления модели в форме ГПД. Все последующие более детальные описания состояний модели формируются инструментальными средствами SML-технологии в соответствии с правилами семантической среды функционирования СРВ, представленной в базе знаний.

**СПИСОК ЛИТЕРАТУРЫ**

1. Погребной В.К. О построении активных моделей, распределённых систем реального времени // Известия Томского политехнического университета. – 2008. – Т. 312. – № 5. – С. 78–84.
2. Погребной В.К. Системы реального времени. Моделирование и автоматизированное проектирование. – Томск: Изд-во ТПУ, 2006. – 209 с.
3. Погребной А.В. Математические и программные средства построения архитектуры и топологии сети вычислительной системы для управления территориально распределёнными объектами: Автореф. дис. ... канд. техн. наук. – Томск, 2008. – 19 с.
4. Скотт К. UML. Основные концепции. Пер. с англ. – М.: Издательский дом «Вильямс», 2002. – 144 с.

*Поступила 31.03.2009 г.*