АНАЛИЗ ВЫПОЛНИМОСТИ СОСТАВНЫХ ЗАДАЧ В СИСТЕМАХ РЕАЛЬНОГО ВРЕМЕНИ

Н.В.Гуцалов, В.В.Никифоров

Санкт-Петербургский институт информатики и автоматизации РАН
199178, Санкт-Петербург, 14-я линия ВО, д. 39

<nik@iias.spb.su>

УДК 681.3.06
Н. В. Гуцалов, В.В.Никифоров. **Анализ выполнимости составных задач в системах реального времени** // Труды СПИИРАН, Вып. 2, т. 2. — СПб.: Наука, 2005.
Аннотация. *Рассматриваются вычислительные модели программных приложений реального времени, содержащих составные задачи. Представлены подходы к анализу выполнимости задач в таких приложениях* — Библ. 5 назв.

UDC 681.3.06
N. V. Gutsalov, V. V. Nikiforov. **Feasibility analysis for compound tasks in real-time systems** // SPIIRAS Proceedings. Issue 2, vol. 2. — SPb.: Nauka, 2005.
Abstract. *Computational models are considered for real-time software applications with compound tasks. The opportunity of feasibility analysis of such applications is represented*. — Bibl. 5 items.

Введение

Исследования, посвященные поиску эффективных методов оценки выполнимости задач в системах реального времени, ведутся в течение более трех десятков лет. С начала 1990-х интенсивность исследований по этой тематике существенно возросла в связи с расширением сфер применения компьютерных систем реального времени и ужесточением требований к надежности, эффективности, предсказуемости поведения таких систем.

Первые результаты в этой области были получены для приложений, соответствующих упрощенным вычислительным моделям. Рассматривались системы, включающие множество независимых периодических задач жесткого реального времени [1]. Последующее развитие методов оценки выполнимости задач касалось, в первую очередь, расширения состава вычислительных моделей с целью более полного их соответствия практике использования систем реального времени. За счет этого достигается снижение степени неоправданного пессимизма в оценке выполнимости задач.

В частности, строились модели приложений, состоящих из периодически выполняемых транзакций — комплектов задач со смещенными моментами времени активизации задач [2]. Использование моделей с транзакциями обеспечивает существенное снижение пессимизма в оценке выполнимости задач в сравнении со случаем независимых задач, что позволяет строить системы с более эффективным использованием аппаратных ресурсов.

Разрабатывавшиеся до последнего времени методы оценки выполнимости ориентированы на приложения, включающие только простые задачи. Вместе с тем, построение вычислительных моделей и методов анализа выполнимости для приложений с составными задачами представляет собой еще одно направление повышения эффективности использования аппаратных ресурсов программными системами реального времени [3].

Составные задачи отличаются от простых тем, что в них используются операторы межзадачных взаимодействий — операторы приостановки выполнения текущего задания, порождения нового задания, передачи или приема сообщения и тому подобные. Простые задачи таких операторов не содержат. При выполнении операторов межзадачных взаимодействий активизируется механизм динамического планирования, в силу этого результатом выполнения таких операторов может стать смена текущей задачи.

Различные типы операторов межзадачных взаимодействий входят в состав стандартных интерфейсов прикладных программ. Отказ от их использования сужает возможности применения эффективных архитектурных решений при построении программных приложений. Использование составных задач позволяет строить естественные схемы межзадачных отношений, упрощает конструирование и восприятие структурных особенностей сложных программных систем. Ниже представлены подходы к анализу выполнимости программных приложений реального времени, содержащих составные задачи.

В разделах 1 и 2 рассматриваются методы оценки выполнимости для приложений с составными задачами, не содержащими операторов ожидания. В разделе 3 показано, как схема приложения с операторами ожидания может быть преобразована в схему без операторов ожидания.

1. Модель приложения с составными задачами

Методы анализа выполнимости задач в системах реального времени строятся на базе использования тех или иных вычислительных моделей, представляющих актуальные для анализа выполнимости свойства программных систем и соответствующих вычислительных процессов.

Задачи, секции задач, транзакции. В представляемой ниже вычислительной модели задачи могут наделяться внутренней структурой: тело задачи τ_{ij} может разбиваться на секции ${}_m\beta_{ij}$ операторами, влияющими на выполнение приложения в целом (операторы межзадачных взаимодействий). Одним из типов операторов такого рода являются операторы, активизирующие другие задачи. Наличие операторов активизации приводит к связыванию задач в комплекты, называемые транзакциями. Комплект задач $\Gamma_i = \{\tau_{i1}, \tau_{i2}, ...\}$ связанных операторами активизации, образует транзакцию Γ_i объединяющую задачи, которые обеспечивают реакцию программной системы на возникновение определенного типа $e(\Gamma_i)$ внешних событий. Внешние события возникают в окружающей информационной среде, регистрируются системой и обрабатываются задачами $\tau_{i1}, \tau_{i2}, ...$, составляющими транзакцию Γ_i. Каждое отдельное событие ${}_k e_i$ типа $e(\Gamma_i)$ возникающее в момент времени $t({}_k e_i)$, вызывает активизацию комплекта задач, составляющих транзакцию Γ_i.

Задача τ_{i1} играет роль ключевой задачи транзакции Γ_i: непосредственным результатом регистрации внешнего события ${}_k e_i$ типа $e(\Gamma_i)$ является активизация задачи τ_{i1} (т.е. порождение задания ${}_k\tau_{i1}$). Задачи $\tau_{i2}, \tau_{i3}, ...$ активизируются либо непосредственно задачей τ_{i1}, либо транзитивно, через другие задачи из Γ_i.

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Отдельная задача τ_{ij} представляет собой **статический** объект, характеризующийся следующими атрибутами:

- имя задачи;
- $p(\tau_{ij})$ — стартовый приоритет задачи (положим для упрощения, что значения стартовых приоритетов различных задач различны);
- идентификатор внутреннего ресурса r_x, используемого задачей, если таковой ресурс имеется;
- список $\beta_{ij1}, \beta_{ij2}, ..., \beta_{ijn}$ секций, составляющих задачу.

Примем для упрощения следующее условие.

Условие 1. Значения стартовых приоритетов различных задач различны.

Некоторые из секций задачи могут использовать разделяемые ресурсы. Доступ к разделяемым ресурсам осуществляется согласно протоколу превентивного наследования приоритетов [4]. В соответствии с этим протоколом каждый ресурс r_x характеризуется специальным параметром $p(r_x)$, пороговым приоритетом данного ресурса. Внутренний ресурс, если он указан в спецификации задачи, используется каждой из ее секций. Задача, состоящая из единственной секции, называется простой задачей. Задача, состоящая из двух или более секций, называется составной задачей.

Секция β_{ijm} задачи τ_{ij} характеризуется следующими атрибутами:

- $c(\beta_{ijm})$ — верхняя граница затрат процессорного времени на выполнение части задачи τ_{ij}, которая представляется секцией β_{ijm};
- $D(\beta_{ijm})$ — относительный предельный срок выполнения секции (обязателен только для секции, завершающей выполнение задачи);
- $F(\beta_{ijm})$ — оператор межзадачных взаимодействий, завершающий выполнение секции β_{ijm}.

Верхняя граница $c(\tau_{ij})$ затрат процессорного времени на выполнение задачи τ_{ij} определяется как сумма значений $c(\beta_{ijm})$ для составляющих задачу секций.

Задачи τ_{ij} и их секции β_{ijm} выполняются в рамках транзакции Γ_i, обеспечивающей реакцию системы на внешнее событие ${}_k e_i$; значение $D(\beta_{ijm})$ срока выполнения секции β_{ijm} задается относительно момента $t({}_k e_i)$ возникновения события типа $e(\Gamma_i)$. Относительный срок выполнения последней секции задачи рассматривается как относительный срок $D(\tau_{ij})$ выполнения самой задачи τ_{ij}.

Операторы $F(\beta_{ijm})$ отделяют секции задачи τ_{ij} друг от друга. В качестве таких операторов могут выступать:

- операторы захвата/освобождения ресурса;
- операторы принудительного перепланирования;
- операторы активизации другой задачи.

Операторы захвата/освобождения ресурса r_x обрамляют те критические секции задачи τ_{ij}, которые используют этот ресурс. В рамках критических секций задача τ_{ij} выполняется с приоритетом равным $p(r_x)$. Приоритет остальных сек-

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

ций задачи равен либо приоритету самой задачи, либо приоритету внутреннего ресурса задачи, если таковой имеется. В момент выполнения оператора принудительного перепланирования приоритет задачи понижается до уровня ее стартового приоритета. Последняя секция задачи завершается либо простым оператором окончания задачи, либо оператором активизации другой задачи.

В ходе работы системы реального времени каждая задача τ_{ij} выполняется, как правило, многократно: всякий раз при возникновении необходимости выполнить алгоритм, соответствующий задаче τ_{ij}, порождается **динамический** объект — задание ${}_k\tau_{ij}$ на реализацию этого алгоритма. Задание ${}_k\tau_{ij}$ выполняется процессором вычислительного комплекса (представляемая модель ограничена однопроцессорными вычислительными комплексами) и характеризуется следующими атрибутами:

- имя выполняемой задачи τ_{ij};
- $t_{reg}({}_k\tau_{ij})$ — момент времени порождения (регистрации) данного задания;
- $d({}_k\tau_{ij})$ — абсолютный предельный срок выполнения данного задания;
- указатель выполняемого в текущий момент оператора внутри тела задачи τ_{ij};
- $p({}_k\tau_{ij})$ — действующее в текущий момент времени значение приоритета данного задания;
- код состояния данного задания.

Динамический объект ${}_k\tau_{ij}$ (задание ${}_k\tau_{ij}$) существует в системе в течение временного интервала, начинающегося в момент времени $t_{reg}({}_k\tau_{ij})$, и заканчивающегося в момент $t_{fin}({}_k\tau_{ij})$ завершения выполнения задания.

Абсолютный предельный срок $d({}_k\tau_{ij})$ выполнения задания ${}_k\tau_{ij}$ равен сумме относительного срока выполнения задачи τ_{ij} и момента времени $t({}_k e_i)$ возникновения отдельного события ${}_k e_i$ из класса событий, обрабатываемых транзакцией Γ_i: $d({}_k\tau_{ij}) = D(\tau_{ij}) + t({}_k e_i)$.

Условие 2. Действующее значение приоритета $p({}_k\tau_{ij})$ не может быть меньше, чем стартовый приоритет $p(\tau_{i1})$ ключевой задачи транзакции Γ_i.

Интервалы времени существования заданий, соответствующих различным задачам, могут пересекаться. Это означает, что, в некоторый момент времени t в системе могут одновременно выполняться несколько заданий. Для однопроцессорного вычислительного комплекса в конкретный момент времени t может выполняться может лишь одно задание, которое называется текущим на момент t. Выполнение остальных заданий, существующих в системе в этот момент времени, отложено до предоставления им ресурса процессора. Таким образом, код состояния задания ${}_k\tau_{ij}$ может принимать два значения:

- “текущее” — задание ${}_k\tau_{ij}$ выполняется процессором вычислительного комплекса;

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

- “отложенное” — задание ${}_k\tau_{ij}$ ожидает момента предоставления ресурса процессора.

Перевод заданий из состояния “текущее” в состояние “отложенное” и обратно выполняется в соответствии с принятой дисциплиной обслуживания. Ниже полагается, что управление ресурсом процессора осуществляется вытесняющим планировщиком в соответствии со значениями действующих приоритетов заданий. В рамках критических интервалов действующие приоритеты различных заданий могут совпадать. Для заданий с совпадающими приоритетами принимается FIFO дисциплина обслуживания.

Как отмечалось выше, комплект задач $\{\tau_{i1},\tau_{i2},...\}$, составляющий транзакцию Γ_i ассоциируется со специальным типом $e(\Gamma_i)$ событий, возникающих во внешней среде и регистрируемых системой. Регистрация события ${}_k e_i$ типа $e(\Gamma_i)$ выполняется с некоторой задержкой относительно момента $t({}_k e_i)$ возникновения события ${}_k e_i$. Параметр $J_i = \max\{t_{reg}({}_k\tau_{i1}) - t({}_k e_i)\}$ задает максимальную величину такой задержки.

Частота возникновения событий типа $e(\Gamma_i)$ характеризуется величиной T_i, называемой периодом транзакции Γ_i. Период T_i равен минимально возможному интервалу времени между соседними событиями типа $e(\Gamma_i)$, т.е., $T_i \le t({}_{k+1}e_i) - t({}_k e_i)$.

Каждая задача транзакции Γ_i связана с задачей τ_{1j} цепочкой отношений “активизировать задачу”. Эти отношения формируют структуру типа направленного ациклического графа, включающую в себя все задачи принадлежащие транзакции Γ_i. Задача τ_{1j} является корневым узлом структуры, задающей отношения предшествования задач в рамках транзакции Γ_i.

Полная модель приложения задается спецификацией составляющих ее ресурсов и задач (с описанием последовательности секций) и указанием перечня транзакций Γ_i; для каждой транзакции следует указать значения атрибутов:

- тип $e(\Gamma_i)$ соответствующих Γ_i внешних событий;
- период T_i следования событий типа $e(\Gamma_i)$;
- максимальную задержку J_i регистрации событий типа $e(\Gamma_i)$;
- ключевую задачу τ_{1j}.

Состав транзакции Γ_i (перечень $\tau_{i1},\tau_{i2},...$ образующих ее задач), определяется отношениями предшествования, задаваемыми в описаниях секций задач.

Профиль транзакции. Назначение приоритетов задачам и ресурсам однозначно определяет порядок выполнения секций задач в пределах каждой транзакции. Этот факт позволяет построить для каждой транзакции профиль ее выполнения (рис.1), изображающий ход выполнения секций задач при монопольном использовании ресурса процессора и максимальной продолжительности выполнения каждой секции. Профиль транзакции Γ_i представляется последовательностью элементов $\chi_{j1},\chi_{j2},...$; большинство элементов профиля соответствует секциям задач, остальные представляют начало задачи, использующей внутренний ресурс или оператор принудительного перепланирования. Каждый элемент профиля χ_i (первый индекс опущен) характеризуется парой па-

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

раметров $\chi_i = \langle p_i, c_i \rangle$, где p_i — приоритет элемента, c_i — его длина (соответствует максимальному времени выполнения секции задачи). В левой части рис. 1 представлена внутренняя структура двух задач τ_1 и τ_2, входящих в транзакцию Γ_i (в обозначениях секций индексы *i* опущены). Транзакция Γ_i содержит также задачу τ_3, состоящую из единственной секции β_{31}. Профиль транзакции (справа на рис.1) соответствует следующим значениям приоритетов задач: $p(\tau_1) = 4, p(\tau_2) = 8, p(\tau_3) = 5$.

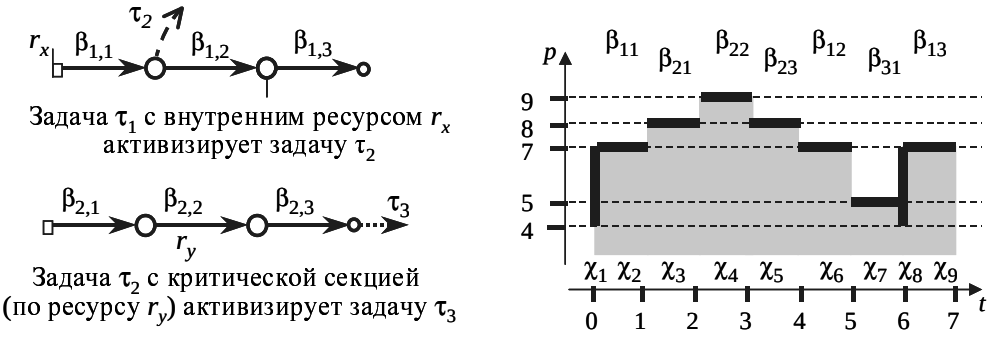

Рис. 1. Внутренняя структура задач и профиль транзакции
(объяснения см. в тексте).

Задача τ_1 использует внутренний ресурс r_x, приоритет которого равен 7; задача τ_2 использует ресурс r_y, приоритет которого равен 9. Первым оператором задачи τ_1 является активизация задачи τ_1. Вторым оператором является принудительное перепланирование. Первым и вторым операторами задачи τ_2 являются захват и освобождение ресурса r_y соответственно. Третьим оператором является активизация задачи τ_3. Профиль этой транзакции состоит из девяти элементов, три из которых имеют нулевую длину; два элемента нулевой длины соответствуют захвату задачей τ_1 внутреннего ресурса. Еще один элемент нулевой длины соответствует принудительному перепланированию.

Построение профиля транзакции позволяет определить порядок выполнения секций задач, входящих в состав транзакции. В частности, определяется, замыкающая задача транзакции, то есть та задача $\tau_{i,fin}$, выполнение которой завершается в последнюю очередь, после завершения всех остальных задач транзакции Γ_i. Момент завершения выполнения задачи $\tau_{i,fin}$ является моментом завершения обработки события типа $e(\Gamma_i)$.

Будем называть длиной профиля транзакции Γ_i сумму $L(\Gamma_i) = \sum_j \chi_{ij}$ длин всех элементов ее профиля. Эта сумма по определению равна продолжительности выполнения транзакции Γ_i при монопольном использовании ресурса процессора этой транзакцией

$$L(\Gamma_i) = \sum_j \sum_k c(\beta_{ijk}),$$

где суммирование ведется по всем секциям всех задач, составляющих Γ_i.

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

2. Определение максимального значения времени отклика

Для анализа выполнимости задач из приложений, отвечающих представленной выше модели, достаточно использовать данные о параметрах профилей всех транзакций Γ_i, образующих приложение, и значениях атрибутов T_i и J_i. В данном разделе приведен подход к анализу выполнимости на основе определения максимального значения R_{ij} времени отклика для задачи τ_{ij}. Метод определения значения R_{ij} представлен на примере задачи $\tau_{i,\text{fin}}$, завершающей выполнение транзакции Γ_i.

Задача определения значения $R_{i,fin}$ решается тривиально в случае, если для обработки каждого типа внешних событий системой может быть выделен отдельный процессор. Тогда $R_{i,fin}$ вычисляется как сумма $R_{i,fin} = J_i + L(\Gamma_i)$. В однопроцессорной системе на продолжительность обработки внешнего сигнала типа $e(\Gamma_i)$ оказывает влияние отвлечение процессора на обработку внешних сигналов других типов; в этом случае проблема состоит в отыскании способа оценки влияния других транзакций на величину времени отклика $R_{i,fin}$.

В рамках рассматриваемой вычислительной модели существуют две разновидности такого влияния — блокировки и вытеснения. Множество транзакций, входящих в состав приложения, частично упорядочено по значениям стартовых приоритетов их ключевых задач. В соответствии с этим отношением частичного порядка все транзакции, способные влиять на продолжительность выполнения транзакции Γ_i, разделяются на две разновидности: транзакции, блокирующие выполнение Γ_i, и транзакции, вытесняющие Γ_i. К разряду транзакций, блокирующих выполнение Γ_i, относится каждая такая транзакция Γ_j, для которой $p(\tau_{j1}) < p(\tau_{i1})$. Остальные транзакции относятся к разряду транзакций, вытесняющих Γ_i (для них $p(\tau_{j1}) > p(\tau_{i1})$).

Влияние блокирующих транзакций. Условимся для удобства изложения говорить, что задача τ_{ik} вытесняется транзакцией Γ_i, если в момент регистрации ключевая задача τ_{j1} транзакции Γ_i вытесняет задачу τ_{ik}.

Утверждение 1. Транзакция Γ_j, являющаяся блокирующей в отношении транзакции Γ_i, не может вытеснить какую-либо из задач τ_{ik}.

Действующий приоритет $p({}_m\tau_{ik})$ любого из заданий, выполняемых в рамках обработки внешнего события типа $e(\Gamma_i)$, не может быть ниже, чем стартовый приоритет $p(\tau_{i1})$ ключевой задачи транзакции Γ_i. Поскольку Γ_j по отношению к Γ_i — блокирующая транзакция, стартовый приоритет $p(\tau_{j1})$ ее ключевой задачи еще ниже, чем $p(\tau_{i1})$. Поэтому $p({}_m\tau_{ik}) > p(\tau_{j1})$ и ни одна из задач τ_{ik} не может быть вытеснена транзакцией Γ_j.

Следовательно, если Γ_j — блокирующая транзакция по отношению к Γ_i, и в момент регистрации задания ${}_m\tau_{j1}$ (в момент очередной регистрации ключевой задачи транзакции Γ_j) выполняется какая-то из задач транзакции Γ_i, то в

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

ходе регистрации задания ${}_m\tau_{j1}$ его код состояния принимает значение “отложенное”.

Вместе с тем, в ходе выполнения какого-то из заданий ${}_m\tau_{jk}$ его текущий приоритет $p({}_m\tau_{jk})$ может оказаться выше, чем стартовый приоритет $p(\tau_{i1})$ корневой задачи транзакции Γ_i. В этом случае задание ${}_m\tau_{jk}$ продолжает оставаться текущим, а вновь порожденное задание ${}_m\tau_{i1}$ пополняет состав “отложенных”.

Таким образом, транзакция Γ_j с невысоким значением $p(\tau_{j1})$ может некоторое время блокировать выполнение задач более высокоприоритетной транзакции Γ_i. Возникает вопрос — не может ли в этот момент в ряду “отложенных” находиться какое-либо относительно высокоприоритетное задание от другой транзакции, блокирующей Γ_i. Ведь это будет означать составное блокирование — вслед за интервалом блокирования со стороны Γ_j задача ${}_m\tau_{i1}$ будет вынуждена ожидать еще и завершения интервала блокирования со стороны другой блокирующей транзакции. Справедливость следующего утверждения дает возможность убедиться, что такое составное блокирование невозможно.

Утверждение 2. На интервале совместного выполнения двух транзакций Γ_1 и Γ_2 действующий приоритет отложенного задания не может увеличиваться.

Пусть в некоторый момент интервала совместного выполнения задание от транзакций Γ_1 было текущим, а приоритет отложенного задания от Γ_2 был равен p^*. Далее в рамках интервала совместного выполнения этих транзакций Γ_1 и Γ_2 значение приоритета отложенного задания может измениться только в момент, когда в ходе выполнения Γ_1 достигается участок ее профиля с приоритетом $p^{**} < p^*$. Вслед за этим моментом текущим становится задание от Γ_2 с приоритетом p^* (или, возможно, выше) а значение приоритета отложенного задания (теперь это задание от Γ_2) снижено до p^{**}.

Утверждение 3. Составное блокирование невозможно.

Если момент регистрации ключевой задачи транзакции Γ_1 (со стартовым приоритетом $p(\tau_{11})$) оказался внутри интервала существования заданий ${}_m\tau_{2i}$ и ${}_n\tau_{3j}$ от двух блокирующих Γ_1 транзакций Γ_2 и Γ_3, то неравенства $p(\tau_{11}) < p({}_m\tau_{2i})$ и $p({}_n\tau_{3j}) < p(\tau_{11})$ не могут выполняться одновременно. Действительно, в начале интервала совместного выполнения Γ_2 и Γ_3 приоритет отложенного задания от одной из этих транзакций не превышал значения $\max\{p(\tau_{21}), p(\tau_{31})\}$, которое по определению меньше, чем $p(\tau_{11})$. Следовательно, в силу утверждения 2, в момент регистрации ключевой задачи транзакции Γ_1 может выполняться не более одного из неравенств $p(\tau_{11}) < p({}_m\tau_{2i})$ и $p({}_n\tau_{3j}) < p(\tau_{11})$.

Из справедливости утверждения 3 следует, что для определения влияния блокирующих транзакций на величину времени отклика ${}_{fin}R_i$ (для определения величины блокирующего фактора B_i для транзакции Γ_i) достаточно найти в профилях блокирующих транзакций самый широкий участок с приоритетом не ниже чем $p(\tau_{i1})$. Если транзакция Γ_i имеет наивысшее значение приоритета

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

ключевой задачи, то для нее нет вытесняющих транзакций, она использует процессор в монопольном режиме. Для такой транзакции значение $_{fin}R_i$ однозначно определяется величиной блокирующего фактора: $_{fin}R_i = J_i + B_i + L(\Gamma_i)$.

Метод определения влияния вытесняющих транзакций на величину времени отклика $R_{i,fin}$ опирается на использование преобразованной формы профиля транзакции — гладкого профиля транзакции. Описание гладкого профиля транзакции, содержит не больше (а, как правило, меньше) числовых параметров, чем определенная в разделе 1 исходная форма профиля той же транзакции.

Построение гладкого профиля транзакции. В профиле транзакции информация для анализа выполнимости представлена в более концентрированном виде, чем в исходной модели приложения — в виде, отвлеченном от внутренней структуры задач, входящих в состав транзакции, от отношений предшествования задач.

Оценка максимального времени отклика задачи $\tau_{i,fin}$ начинается с преобразования полного профиля транзакции в форму гладкого профиля. В гладкой форме профиля транзакции Γ_i устранены те детали полного профиля, которые не влияют на оценку значения $R_{i,fin}$.

Гладкий профиль транзакции Γ_a строится в виде последовательности фрагментов $\Phi_{a1},\Phi_{a2},...$. Отдельный фрагмент Φ_{aj} соответствует либо единственному из элементов χ_{ak} (перекрывает единственный элемент χ_{ak} исходного профиля), либо последовательности $\chi_{ak},\chi_{a,k+1},...,\chi_{a,k+m}$ (перекрывает последовательность $\chi_{ak},\chi_{a,k+1},...,\chi_{a,k+m}$ элементов исходного профиля). Если элемент χ_{ak} исходного профиля является последним из перекрываемых фрагментом Φ_{ai}, то элемент $\chi_{a,k+1}$ исходного профиля (если такой существует) является первым из перекрываемых фрагментом $\Phi_{a,i+1}$. В целом последовательность фрагментов $\Phi_{a1},\Phi_{a2},...$ гладкого профиля транзакции Γ_a полностью перекрывает ее исходный профиль.

Аналогично тому, как элемент полного профиля характеризуется парой параметров $\chi_k = \langle p_k, c_k \rangle$, фрагмент Φ_j (первый индекс опущен) характеризуется парой параметров $\Phi_j = \langle P_j, C_j \rangle$; P_j — приоритет фрагмента Φ_j, C_j — его длина. При этом $P_j = p_{k+m}$, $C_j = (c_k + c_{k+1} + ... + c_{k+m})$. В частном случае, при $m = 0$ (т.е., если Φ_j перекрывает единственный элемент χ_k полного профиля) $P_j = p_k$, $C_j = c_k$. Гладкий профиль транзакции должен отвечать следующим условиям гладкости.

Условия гладкости. Если фрагмент Φ_j гладкого профиля транзакции соответствует последовательности $\chi_{ak},\chi_{a,k+1},...,\chi_{a,k+m}$ элементов ее полного профиля, то выполняются следующие условия:

а) для любого элемента χ_k из последовательности $\chi_k,\chi_{k+1},...,\chi_{k+m}$ выполняется неравенство $p_i \geq p_{k+m}$ (или, что то же, $p_i \geq P_j$);

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

б) если элемент χ_k не является первым элементом полного профиля транзакции ($k \neq 1$), то $p_{k-1} < p_k$;

в) если элемент χ_{k+m} не является последним элементом полного профиля транзакции, то $p_{k+m+1} > p_{k+m}$;

Очевидно, что если исходный профиль транзакции (полный профиль транзакции) отвечает условиям гладкости, то параметры ее гладкого профиля отвечают равенствам $P_k = p_k$, $C_k = c_k$. То есть, для такой транзакции все параметры ее полного и гладкого профилей совпадают.

Утверждение 4. Для гладкого профиля транзакции значение параметра P_j монотонно возрастает с ростом значения индекса j.

Пусть χ_k — последний из элементов полного профиля, соответствующих фрагменту Φ_j гладкого профиля транзакции. По определению $P_j = p_k$. Если бы выполнялось неравенство $p_k \geq p_{j+1}$, то элемент χ_k по условию гладкости должен был бы соответствовать фрагменту Φ_j. Следовательно, $P_j < P_{j+1}$ для любых соседних фрагментов Φ_j и Φ_{j+1}.

Утверждение 5. Стартовый приоритет $p(\tau_{i1})$ задачи τ_{i1} равен приоритету P_1 первого фрагмента гладкого профиля.

Это следует из условия гладкости и из принятого выше условия 2.

Утверждение 6. Если в составе анализируемого приложения комплект задач $\{\tau_{a1}, \tau_{a2}, ...\}$, соответствующих транзакции Γ_a, заменить другим комплектом задач $\{\tau_{b1}, \tau_{b2}, ...\}$ так, чтобы гладкие профили транзакций Γ_a и Γ_b совпадали, то совпадут и значения $R_{a,fin}$ и $R_{b,fin}$ времени отклика для этих транзакций.

Стартовые приоритеты транзакции Γ_a и Γ_b совпадают в силу утверждения 5. Следовательно, для них совпадают множество блокирующих транзакций и величина фактора блокирования ($B_a = B_b$).

Совпадают также абсцисса (значение координаты t) и ордината (значение координаты p) правого конца последнего из тех элементов полного профиля транзакций, которые перекрываются фрагментом Φ_1. А это означает, что при совпадении внешних условий (при одинаковых моментах активизации вытесняющих транзакций) как для исходного, так и для измененного варианта транзакции Γ_i секции задач, перекрываемых фрагментом Φ_1, будут вытесняться одинаково — теми же секциями задач тех же вытесняющих транзакций. Отсюда следует, что время отклика секций, перекрываемых фрагментом Φ_1, в обоих случаях будет одинаковым. Аналогичные рассуждения приводят к выводу о совпадении времени отклика секций, перекрываемых и последующими фрагментами гладкого профиля, в том числе и его замыкающим фрагментом. Следовательно, утверждение 6 справедливо, $R_{a,fin} = R_{b,fin}$.

Из справедливости утверждения 6 следует, что для определения значения ${}_{fin}R_i$ достаточно найти это значение для такого варианта транзакции Γ_i, у которого форма ее исходного профиля совпадает с формой ее гладкого профиля (то есть, сама исходная форма профиля отвечает условиям гладкости). Таким образом, задача определения значения $R_{i,fin}$ сводится к замене в анали-

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

зируемом приложении транзакции Γ_i такой транзакцией Γ_a, у которой исходный профиль отвечает условиям гладкости и совпадает с гладким профилем транзакции Γ_i.

Влияние вытесняющих транзакций. Для оценки наибольшего вклада вытесняющих транзакций (фактора интерференции) в величину времени отклика $R_{i,fin}$ следует рассмотреть наиболее неблагоприятный с этой точки зрения сценарий выполнения задач приложения, наихудший сценарий внешних событий, следующих за критическим моментом. В случае рассматриваемой модели, как и в случае аналогичной модели с простыми задачами [5], критическим является такой момент $t_{critical}$ регистрации Γ_a, для которого выполняются следующие условия:

- в тот же момент $t_{critical}$ регистрируются и все транзакции Γ_i, являющиеся вытесняющими по отношению к Γ_a;
- для всех транзакций, регистрируемых в момент $t_{critical}$, задержка регистрации является максимальной (для Γ_a она равна J_a, для каждой из Γ_i она равна соответственно J_i).

Следующий за критическим моментом $t_{critical}$ сценарий наиболее неблагоприятного хода событий предполагает, что в дальнейшем на протяжении временного интервала ($t_{critical}$, $t_{critical} + R_{i,fin}$) каждая вытесняющая транзакция Γ_i регистрируются с нулевой задержкой и последующие внешние события e_i возникают с интервалами, в точности равными T_i. Для оценки фактора интерференции важен следующий факт.

Утверждение 7. Если транзакция Γ_i вытеснила транзакцию Γ_a, то возобновление выполнения Γ_a не может начаться раньше полного завершения выполнения Γ_i.

Утверждение 7 справедливо в силу условия 2.

В качестве первого шага в решении задачи определения значения $R_{i,fin}$ построим соотношение для определения времени отклика $R_{i,1}$ для первого фрагмента $\Phi_{a,1}$ транзакции Γ_a. Согласно утверждению 7 вытесняющие транзакции действуют на первый фрагмент $\Phi_{a,1}$ транзакции Γ_a так же, как на простую задачу действует вытеснение другими простыми задачами. Это означает, что для ${}_1R_a$ справедливо рекуррентное соотношение, аналогичное тому, которое используется для определения времени отклика в случае модели приложения с простыми задачами [5]:

$$R_{a,1} = J_a + B_a + L(\Phi_{a,1}) + \sum_j \left\lceil (R_{a,1} - J_a + J_i)/T_j \right\rceil \cdot L(\Gamma_j),$$

где $L(\Phi_{a,1})$ — длина первого фрагмента $\Phi_{a,1}$ профиля транзакции Γ_a, суммирование ведется по всем транзакциям Γ_j, вытесняющим Γ_a. Решение этого рекуррентного соотношения дает значение $R_{a,1}$. Имея значение $R_{a,1}$, можно построить аналогичное соотношение для $R_{a,2}$ (примем обозначение $N_{j,k} = \left\lceil (R_{a,1} - J_a + J_i)/T_j \right\rceil$):

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

$$R_{a,2} = J_a + R_{a,1} + L(\Phi_{a,2}) + \sum_j \left(\left\lceil (R_{a,2} - J_a + J_i)/T_j \right\rceil - N_{j,1} \right) \cdot L(\Gamma_j),$$

где суммирование ведется по транзакциям Γ_j, вытесняющим Γ_a во время выполнения фрагмента ${}_a\Phi_2$, то есть по тем транзакциям, у которых стартовый приоритет $p(\tau_{j1})$ ключевой задачи выше, чем приоритет $P_{a,1}$ фрагмента $\Phi_{a,2}$. И для каждого из последующих фрагментов $\Phi_{a,k}$ транзакции Γ_a имеет место подобное рекуррентное соотношение:

$$R_{a,k} = J_a + R_{a,k-1} + L(\Phi_{a,k}) + \sum_j \left(\left\lceil (R_{a,k} - J_a + J_i)/T_j \right\rceil - N_{j,k-1} \right) \cdot L(\Gamma_j).$$

Это соотношение имеет место, в частности, для замыкающего фрагмента $\Phi_{a,fin}$ транзакции Γ_a. Таким образом, задача определения значения $R_{a,fin}$ времени отклика для транзакции Γ_a полностью решена. Попутно определены значения времени отклика для каждого из фрагментов, составляющих Γ_a.

Предлагаемый метод можно применить для определения времени отклика для любой из задач, входящих в состав любой из транзакций Γ_i приложения, соответствующего рассмотренной вычислительной модели. Для этого достаточно заменить транзакцию Γ_i транзакцией с профилем, укороченным соответствующим образом. Метод не сложно обобщить на случай приложений, для которых не выполняются условия 1 и 2.

3. Анализ выполнимости задач с состояниями ожидания

Целесообразность использования задач с операторами ожидания может быть вызвана необходимостью заблаговременной подготовки локального контекста с целью быстрой реакции на поступление ожидаемого события, нецелесообразностью разрушения сложного локального контекста задачи между периодами ее активизации, требованиями наглядности текста программы и другими обстоятельствами.

При наличии в приложении задач с операторами ожидания появляются два дополнительные типа операторов, влияющих на работу динамического планировщика (рис. 2а): оператор установки условия и оператор ожидания условия. Использование операторов установки условия приводит к появлению дополнительного уровня структурирования схем задач: операторы этого типа разбивают тело задачи с операторами ожидания на подзадачи.

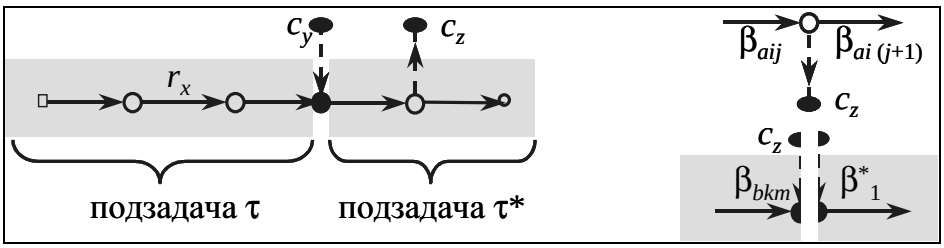

Рис. 2. Внутренняя структура задачи с оператором ожидания. а) Задача с оператором ожидания c_y и оператором установки c_z. б) Разбиение составной задачи по оператору ожидания c_z.

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Если к моменту выполнения оператора ожидания условие c_y не установлено, то происходит перевод текущей задачи в состояние ожидания. В этом случае выполнение оператора ожидания разбивается на две части (рис. 2б): первая часть завершает выполнение подзадачи τ аналогично завершению простой задачи, вторая часть является началом выполнения подзадачи τ^*.

Рис. 3 представляет два типа поведения операторов ожидания, различающиеся состоянием проверяемого условия c_x в момент вызова оператора ожидания. На рис.3а условие c_x устанавливается до выполнения оператора ожидания. В этом случае первая часть оператора ожидания условия работает так же, как оператор завершения задачи с порождением следующей. Вторая часть оператора ожидания эквивалентна началу следующей подзадачи составной задачи, при этом эта подзадача принадлежит той же самой транзакции Γ_c. В этом случае с точки зрения задачи τ_{ab} выполнение оператора установки условия c_x не может привести ни к каким изменениям параметров планирования объектов приложения и, следовательно, может рассматриваться как пустой оператор.

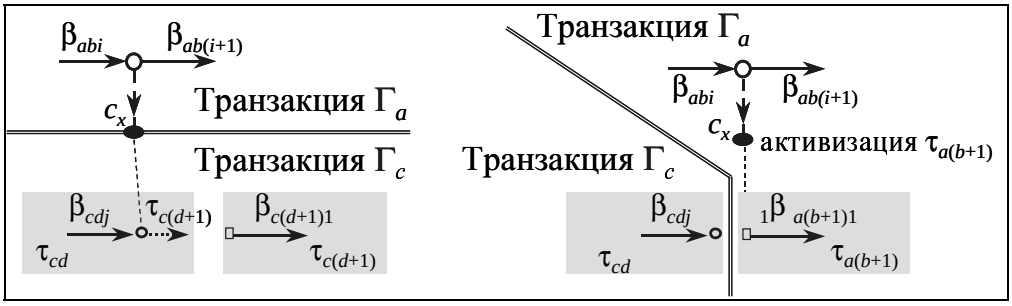

Рис. 3. Два типа поведения операторов ожидания события. а) c_x устанавливается до окончания выполнения секции β_{cdj}. б) c_x устанавливается после окончания выполнения секции β_{cdj}

На рис.3б процесс выполнения составной задачи τ_{cd} достигает оператора ожидания условия до момента его установки. В этом случае первая часть оператора ожидания прерывает выполнение подзадачи τ_{cd}, ее можно рассматривать как оператор завершения задачи. Вторая часть оператора ожидания вновь работает аналогично оператору начала задачи; но теперь следующая за оператором ожидания подзадача не будет выполняться в рамках транзакции Γ_c, а будет выполняться в рамках транзакции Γ_a, к которой принадлежит оператор установки условия c_x. Поэтому вторая подзадача в данном случае принадлежит транзакции Γ_a, а оператор установки условия эквивалентен оператору активизации цепочки подзадач.

Типы Рис. 3а и Рис. 3б исчерпывают возможные динамические варианты поведения операторов ожидания условия. Это дает основание выделить (статически) три типа операторов ожидания:

Тип W1 оператора ожидания. Оператор, для которого условие c_x всегда устанавливается после выполнения оператора ожидания этого условия.

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Тип W2 оператора ожидания. Оператор, для которого условие c_X всегда устанавливается до выполнения оператора ожидания этого условия.

Тип W3. оператора ожидания. Оператор, для которого условие c_X может устанавливаться как до, так и после выполнения оператора ожидания этого условия

Операторы ожидания типа *W*2 всегда выполняются без переведения вызывающей задачи в состояние ожидания. Следовательно, первая часть оператора ожидания типа *W*2 всегда работает как оператор завершения задачи с активизацией следующей, а подзадача, следующая за оператором ожидания типа *W*2, всегда выполняется в рамках той же транзакции, что и подзадача, завершающаяся оператором ожидания. Поэтому оператор ожидания типа *W*2 никогда не приводит к вариации профиля транзакции, содержащей подзадачу, завершающуюся оператором ожидания.

Операторы ожидания типа *W*1 всегда выполняются с переведением вызывающей задачи в состояние ожидания. Следовательно, первая часть оператора ожидания типа *W*1 всегда работает как оператор завершения задачи, а подзадача, следующая за оператором ожидания типа *W*1, всегда выполняется в рамках транзакции отличной от той, к которой принадлежит подзадача, завершающаяся оператором ожидания. Поэтому оператор ожидания типа W_1 тоже никогда не приводит к вариации профиля транзакции, содержащей подзадачу, завершающуюся оператором ожидания.

Напротив, оператор ожидания типа *W*3 выполняется то как оператор ожидания типа *W*1, то как оператор ожидания типа *W*2; следовательно, оператор ожидания типа *W*3 является источником вариации профиля транзакции. В рамках методов, ориентированных на безвариантные профили транзакций, такое недопустимо. Для применения таких методов пригодны приложения, содержащие только операторы ожидания типов *W*1 и *W*2.

В рассматриваемых типах моделей приложения условие c_x, проверяемое оператором ожидания условия, может быть установлено двумя способами:

- посредством выполнения оператора установки условия;
- как результат возникновения внешнего события.

Каждый источник установки условия (оператор или источник внешних событий) соответствует одному или нескольким операторам, проверяющим или ожидающим установку этого условия (в рамках методов, ориентированных на безвариантные профили транзакций это могут быть только операторы ожидания условия типа *W*1 или типа *W*2). Такое соответствие позволяет выделить следующие типы источников установки условий:

*Тип S*0 *источника установки условия.* Источник связан только с операторами ожидания типа W2.

*Тип S*1 *источника установки условия.* Источник связан с одним оператором ожидания типа W1 и, возможно, с одним или несколькими операторами ожидания типа W2.

*Тип S*2 *источника установки условия.* Источник связан с одним или несколькими операторами ожидания типа W1 и, возможно, с одним или несколькими операторами ожидания типа W2.

*Тип S*3 *источника установки условия.* Источник связан с оператором ожидания типа W3 и, возможно, с другими операторами ожидания.

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Связь источника с операторами ожидания условия типа W_2 не влияет на работу динамического планировщика и, следовательно, на профили транзакций.

Источник типа *S*1 вызывает активизацию подзадачи, следующей за оператором ожидания условия типа *W*1; это не вызывает вариации профиля транзакции: соответствующая задача всегда находится в состоянии ожидания. При этом если условие устанавливается в результате возникновения внешнего события, то всякий раз порождается новая транзакция, корнем которой является подзадача, следующая за оператором ожидания условия типа *W*1. В случае источника типа *S*2 таких подзадач оказывается несколько. Это тоже не приводит к вариации профилей транзакций, так как к моменту установки условия все соответствующие ему задачи всегда находится в состоянии ожидания.

Вариации профилей транзакций могут быть вызваны только операторами установки условия типа *S*3. Если приложение ориентировано на применение анализа выполнимости с безвариантными профилями, то в нем не должно быть источников установки условий типа *S*3.

Если приложение, содержащее задачи с операторами ожидания, не содержит ни операторов ожидания условий типа *W*3, ни источников установки условий типа *S*3, то совокупность схем его задач может быть преобразована в эквивалентную (с точки зрения анализа выполнимости) совокупность схем задач без операторов ожидания. Ниже представлен способ такого преобразования.

Приложение, содержащее задачи с операторами ожидания, может включать в себя задачи трех типов:

- задачи, в которых отсутствуют не только операторы ожидания, но и операторы установки условий;
- задачи без операторов ожидания с операторами установки условий;
- задачи с операторами ожидания.

Анализ выполнимости приложения может быть проведен в следующие четыре шага:

- преобразование каждой схемы задачи с операторами ожидания в набор промежуточных схем;
- преобразование всех схем простых задач и промежуточных схем к стандартным схемам задач без операторов ожидания;
- построение профилей транзакций для задач без операторов ожидания;
- использование методов анализа выполнимости приложений из задач без операторов ожидания.

На первом шаге схема каждой задачи с операторами ожидания преобразуется в набор промежуточных схем: одна схема для каждой подзадачи; каждая секция подзадачи преобразуется в соответствующую секцию промежуточной схемы. Внутренние секции подзадач переносятся в секции промежуточных схем без какого-либо изменения; изменения касаются только секций, разделяемых оператором ожидания условия.

Если секция задачи начинается с оператора ожидания, то в промежуточной схеме этот оператор заменяется началом тела задачи. Если исходная задача использует внутренний ресурс, то промежуточная схема наследует этот приоритет.

Если секция задачи завершается оператором ожидания условия, то используется один из двух вариантов:

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

- если оператор ожидания условия принадлежит типу W_1, то этот оператор заменяется оператором завершения задачи без активизации следующей;
- если оператор ожидания условия принадлежит типу $W2$, то этот оператор заменяется оператором завершения задачи с активизацией следующей за оператором ожидания подзадачи.

На втором шаге модифицируются схемы, содержащие операторы установки условий и внешние источники установки событий. Каждый источник установки условия связан с соответствующим оператором ожидания условия. Тем самым он связан с соответствующей промежуточной схемой. Учитывая эту связь, все источники установки условия заменяются следующим образом:

- если оператор установки условия относится типу $S0$, то он заменяется пустым оператором;
- если оператор установки условия относится типу $S1$, то он заменяется оператором активизации задачи, представляемой соответствующей промежуточной схемой;
- если оператор установки условия принадлежит типу $S2$, то он заменяется оператором активизации нескольких задач, представляемых соответствующими промежуточными схемами.

Аналогичные действия производятся с внешними событиями, устанавливающими условия. Если событие является источником установки условия типа $S2$, то оно игнорируется. Если внешнее событие является источником установки условия типа $S1$ ($S1$), то оно преобразуется в событие, активизирующее задачу, представляемую соответствующей промежуточной схемой. Каждая такая задача становится ключевой задачей транзакции.

После таких преобразований модель приложения полностью переведена в эквивалентную модель приложения, содержащего только задачи без операторов ожидания. Остается применить к преобразованной модели технику определения времени отклика задач, изложенную в разделе 2.

Заключение

Предложенный подход к оценке выполнимости задач в системах реального времени позволяет анализировать приложения, содержащие операторы межзадачных взаимодействий. Рекуррентные соотношения, позволяющие определить время отклика для составных задач без операторов ожидания, являются обобщениями аналогичных соотношений для простых задач. Схемы задач с операторами ожидания при соблюдении определенных ограничений сводятся к схемам задач без операторов ожидания.

Литература

[1] *Liu C. L., Layland J. W.* Scheduling Algorithms for Multiprogramming in a Hard Real-Time environment. JACM, Volume 20, Number 1, p. 46–61, 1973

[2] *Palencia J., Harbor M.* Schedulability Analysis for Task with Static and Dynamic Offsets. University of Cantabria, Spain, 1998

[3] *Гуцалов Н. В., Никифоров В. В., Павлов В. А., Червинский М. П.* Анализ выполнимости приложений реального времени для робототехнических систем // Труды 14-ой конференции по экстремальной робототехнике. СПб, СПбГТУ, 2003

[4] *Sha L., Rajkumar R., Lehoczky J.* Priority Inheritance Protocols: An Approach to Real-Time Synchronisation. IEEE Transactions on Computers 39(9), Sep 1990, pp.1175–1185

[5] *Joseph M., Pandya P.* Finding Response Times in a Real-Time System. The Computer Journal, Volume 29, Number 5, p. 390–395, 1986