ПРОГРАМНО-ТЕХНІЧНІ КОМПЛЕКСИ

С.Л. КРЫВЫЙ, Е.А. ГРИНЕНКО

УДК 51.681.3

ЭКОСИСТЕМЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ

Аннотация. Рассмотрена модель экосистемы, ориентированной на область программной инженерии. Предложены два подхода к исследованию свойств модели — сетевой и автоматный. В частности, рассмотрены способы поиска оптимального плана выполнения работ в экосистеме, идентификации аварийных ситуаций, возможность параллельного выполнения заданий.

Ключевые слова: экосистема, транзиционные системы, конечные автоматы, временные автоматы, сети Петри.

ВВЕДЕНИЕ

Интенсивное развитие индустрии программного обеспечения (ПО) обусловило необходимость переосмысливания ситуации, которая сложилась в этой области. В связи с этим возникло понятие экологической системы, которое позволило бы определять состояние в области ИТ-индустрии, исследовать тенденции развития, методы, влияние на людей и внешнюю среду. Строгого определения экосистемы на данный момент не существует. Отсутствие такого определения объясняется тем, что понятие экосистемы имеет достаточно общий характер и его можно применять как к отдельно взятому семейству объектов, так и к их совокупности в рамках некоторого региона, страны или мира в целом.

В настоящей статье предлагается некоторая модель экосистемы, ориентированной на область программной инженерии.

1. ПОСТАНОВКА ПРОБЛЕМЫ И МОДЕЛЬ ЭКОСИСТЕМЫ

Объектами исследования в экосистеме (ЭС), которая относится к индустрии ПО, являются методы, модели и средства, дающие возможность исследовать взаимодействие ПО с окружающей средой, а также взаимодействие компонентов ПО внутри самой экосистемы. При этом главным аспектом такого взаимодействия является устойчивое развитие экосистемы в целом [1].

В контексте экологического подхода программное обеспечение исследуется как технический объект, который взаимодействует с внешней средой и организовывает взаимодействие внутренних объектов экосистемы между собой. Цель таких исследований — сохранение материальных и природных ресурсов, а также защита окружающей среды.

Предметом исследования являются процессы, которые происходят в ПО и его окружении (принципы, методы, разработчики, заказчики, контролирующие организации). Таким образом, понятие экосистемы включает ряд новых задач, решение которых должно обеспечивать ее развитие.

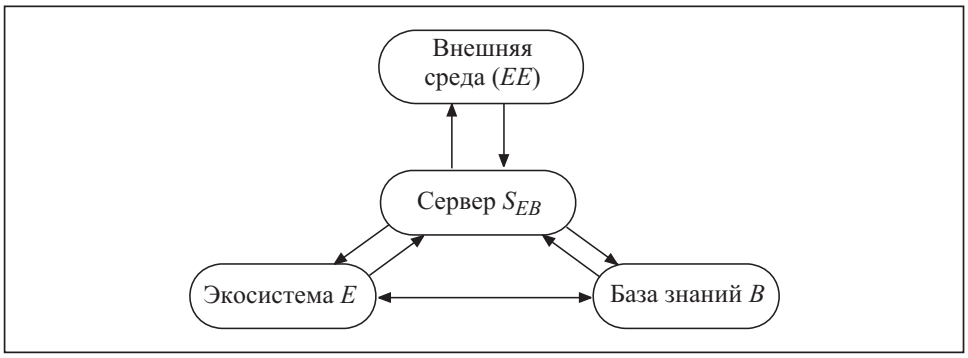

Рис. 1. Модель экосистемы верхнего уровня

Далее исследуем свойства и структуру экосистемы.

Определение 1. Экосистемой называется четверка $ES = (EE, E, B, S_{EB})$, где EE — внешняя среда; E — автономное семейство объектов, которые взаимодействуют между собой и с внешней средой (ядро ЭС); B — база знаний (БЗ), где хранится эволюция развития семейства E как популяции (макрознания ЭС); S_{EB} — сервер, обеспечивающий взаимодействие между E, B и внешней средой (рис. 1).

Внешняя среда EE — это внешнее окружение, в котором функционирует ЭС и которое оказывает внешнее влияние на это функционирование.

Семейство объектов E в ЭС можно интерпретировать по-разному. Например, это может быть множество всех ИТ-компаний, существующих в Украине; это может быть совокупность подразделений отдельной ИТ-компании; это может быть бригада программистов, работающих над реализацией некоторого проекта; это может быть множество агентов, действующих в системе E; это может быть грид-система; компьютерная сеть и т.п.

База знаний B в ЭС представляет хранилище знаний об этапах развития ЭС, о выполненных проектах, о новом ПО, сведения о допущенных ошибках при реализации заданий, об ошибках менеджмента, о позитивном опыте и решениях, о разработанном ПО; об опыте и стаже работы сотрудников ЭС, которые работают в данный момент, и о тех, которые вышли из системы; о структуре ЭС, структуре ее подразделений, о заданиях, которые выполняются подразделениями, и текущем состоянии выполнения заданий.

Сервер S_{EB} служит для обеспечения оперативного взаимодействия между E, B и окружающей средой EE. Система E может непосредственно работать с БЗ в случае, когда нет необходимости использовть сервер. В рассматриваемой модели структура сервера S_{EB} в зависимости от интерпретации может быть достаточно сложной, но здесь она не конкретизируется, а учитывается лишь свойство сервера формировать задания для описываемой ниже подсистемы E.

2. СОСТАВЛЯЮЩИЕ ЭС И ИХ ХАРАКТЕРИСТИКА

Семейство *E*. Ядро экосистемы E представляет четверку, которую назовем серверной или активной частью ЭС:

$$SYS_E = (S, \Omega, o, f), \tag{1}$$

где $S = \{S_1, \ldots, S_n\}$ — конечное множество, элементы которого называются серверами; $\Omega = \{w_1, \ldots, w_m\}$ — конечное множество операций, которые могут быть выполнены на разных серверах; $o : S \to B(\Omega)$ — множество операций, которые может выполнять отдельно взятый сервер, где $B(\Omega)$ — булеан множест-

ва Ω; $f: S \times \Omega \to N^+$ — функция, значениями которой есть время выполнения отдельно взятой операции на отдельно взятом сервере.

Таким образом, SYS_E можно представлять как мультиагентную систему, а ЭС — как систему, включающую мультиагентную подсистему, агентами которой, например, являются разработчики ПО.

База знаний. Создание БЗ основано на использовании языков дескриптивных логик. Это создание предусматривает определение множества атомарных концептов и атомарных ролей, создание терминологии (*TBox*), создание множества фактов (*ABox*) и выбор логического языка и алгоритмов для этого языка, с помощью которых происходит генерация ответов на запросы к БЗ. В терминах *TBox* и *ABox* описываются эволюция ЭС во времени, этапы развития, банкротство, причины банкротства и т.д.

Базовым ядром дескриптивных логик является логика АЛС, для алгоритмов которой имеются реализации [2].

Сервер обмена S_{EB}. Основными функциями этого сервера являются управление внутренними объектами и взаимодействие с внешней средой ЭС. В частности, к этим функциям относится модернизация софтвера и хардвера SYS_{EB} и БЗ (встраивание (embending) нового ПО, модификация (updata) имеющегося ПО, реконфигурация (reconfiguration) серверной части или ее реструктуризация (restructurisation)). Кроме того, на этом сервере находятся запросы к БЗ, которые еще не были обработаны ввиду большой загрузки, ответы на запросы, которые уже обработаны, но не были востребованы, а также текущая информация о состоянии БЗ и системы E. При взаимодействии с внешней средой получают информацию об изменениях, которые произошли во внешней среде, состоянии и потребности рынка, достижениях конкурентов, новых тенденциях в разработке ПО, покупке нового оборудования, технических новинках, их стоимости и др.

Серверная часть ЭС обновляется, модифицируется, реструктуризуется и изменяется под управлением сервера S_{EB}, от которого поступают команды встраивания нового ПО в имеющиеся компоненты, оновления ПО, замены устаревшей техники новыми техническими средствами, изменения структуры (увеличение или уменьшение мощностей ЭС). В модель серверной части вкладываются, например, команды исполнителей, которые задействованы в E. В таком случае S — это множество исполнителей (отдельные исполнители, бригады, фирмы), Ω — это работы, которые выполняются сотрудниками, $o(S_i) = \Omega_i$ — это совокупность заданий, которые может выполнять сервер S_i, $i = 1, \ldots, |\Omega|$, $f(S_i, w_j) = n_i$ (где $S_i \in S$, $w_j \in \Omega$), $n_i \in N^+$ — время выполнения сервером S_i операции w_j.

Значения функции f играют важную роль в ЭС. Они представляют временные оценки сложности выполнения операций в наихудшем случае или берутся из некоторых эмпирических наблюдений. При этом учитывается только время выполнения действия, а не время трансмиссии данных, поскольку предполагается, что данные находятся на соответствующих серверах. Этим и определяется выбор серверов. Располагая значениями функции f, можно прогнозировать риски в ЭС с учетом состояния выполнения заданий. Прогнозирование задержек в работе или возможность срыва выполнения плановых работ в заданные сроки позволяет оперативно реагировать на вызовы в ЭС. Этого можно достичь, усиливая команду, которая вовремя не справляется с заданием, членами команды, которая уже справилась со своим заданием.

Кроме того, значения функции f служат основанием для поиска оптимального пути выполнения заданий в ЭС, поскольку в модели (1) предполагается выполнимость одного и того же задания разными серверами. А это позволяет ис-

пользовать оптимизационные алгоритмы для поиска таких путей.

Таким образом, для представления моделей конкретных систем интерпретации модели (1) могут быть разными — от макроуровня до микроуровня. Далее рассматриваются макроуровень модели (1), который в минимальной степени зависит от конкретизации, и подходы к анализу свойств полученной модели. Цель такого анализа — исследование свойств модели, которые будут выполняться при любой ее конкретизации.

3. ПОДХОДЫ К ИССЛЕДОВАНИЮ СВОЙСТВ МОДЕЛИ SYS_E

Рассмотрим некоторые подходы к исследованию свойств модели (1) и иллюстрацию введенного формализма на простом примере системы SYS_E. В общем случае такая система может иметь большое число объектов, и тогда возникает необходимость в оперативном оптимальном планировании и управлении выполнением заданий, а также доступом к общим ресурсам. Свойства предлагаемой модели ЭС можно исследовать на разных уровнях путем абстрагирования некоторых деталей, повышающих ее уровень, или, наоборот, понижая уровень путем детализации ее составляющих. Существуют различные подходы к анализу свойств, в частности сетевые, автоматные, логические, вероятностные, агентные, архитектурные и др. Если основное внимание уделяется задачам планирования, то существует множество методов их решения. Сложность решения таких задач заключается в том, что в общем случае они имеют противоречия объективного характера. Этим объясняется большое количество методов, поскольку для успешного решения такого типа задач выбирается определенный уровень абстракции, формулируется оптимизационная задача и разрабатывается соответствующий метод ее решения [3–5].

В настоящей статье рассматриваются подходы к решению задач планирования распределения базовых ресурсов в системе SYS_E, которые базируются на сетях Петри (СП) и транзиционных системах. Эти подходы назовем соответственно сетевым и автоматным.

Пусть в системе SYS_E (1)

$$\Omega_i = \{C_1^i, C_2^i, \dots, C_{|\Omega_i|}^i\} \tag{2}$$

означает множество операций, которые могут выполняться на i-м сервере, $i = 1, 2, \dots, |S|$, $C_1^i, C_2^i, \dots, C_{|\Omega_i|}^i \in \Omega$, т.е. $o(S_i) = \Omega_i$. Из равенства (2) следует, что

$$f(S_i, C_k^i) = \tau_k^i,\ i = 1, 2, \dots, |S|,\ k = 1, 2, \dots, |\Omega_i|. \tag{3}$$

Поскольку в системе допускается выполнение одной и той же операции на разных серверах, то время выполнения операции может быть разным на разных серверах. Следовательно, $\Omega_i \cap \Omega_j \neq \varnothing$ $(i \neq j)$. Отсюда вытекает, что в такой постановке задача планирования может иметь несколько способов ее реализации в системе SYS_E. Действительно, любое задание p в SYS_E может быть определено с помощью последовательности операций

$$\overline{O}_p = o_{p_1} o_{p_2} \cdots o_{p_k}, \tag{4}$$

где $o_{p_1}, o_{p_2}, \dots, o_{p_k} \in \Omega$ — операции, определенные системой (1). Поскольку каждое такое задание определяется своей собственной последовательностью типа (4) и существует несколько способов использования серверов для реализации, то существуют и различные варианты реализации одной и той же по-

следовательности $\overline{O}_p$. Поскольку операции могут выполняться на разных серверах, зададим такое ограничение.

Условие 1. Выполнение каждым сервером следующей операции не может быть начато до тех пор, пока на нем ни будет закончено выполнение предыдущей операции.

Для того чтобы реализовать это ограничение, выделим в системе SYS_E входные серверы $X = \{X_1, X_2, \dots, X_I\}$ и выходные серверы $Y = \{Y_1, Y_2, \dots, Y_O\}$, а также рабочие серверы $S_1, \dots, S_W$. Тогда начало выполнения некоторой последовательности операций (процедур выполнения некоторого задания) в системе (1) может специфицироваться в терминах входных серверов, а ее окончание — в терминах выходных серверов. В частности, процедура планирования может быть определена таким образом: задание, доступное на некотором входном сервере, реализуется во времени, требуемом для его выполнения на выбранном множестве серверов, а затем результат его выполнения передается на выходной сервер и становится доступным для дальнейшей работы с ним.

4. ПРИМЕР РАБОТЫ СИСТЕМЫ SYS_E

Чтобы продемонстрировать подходы к анализу свойств приведенной модели ЭС, рассмотрим пример системы SYS_E с тремя рабочими серверами выполнения операций, двумя входными и двумя выходными серверами. Представим ее в графическом виде (рис. 2). Заметим, что такая конкретизация не ограничивает общности модели. Запись S_1 / S_2 означает, что реализация выполнения операции может состояться или на сервере S_1, или на сервере S_2.

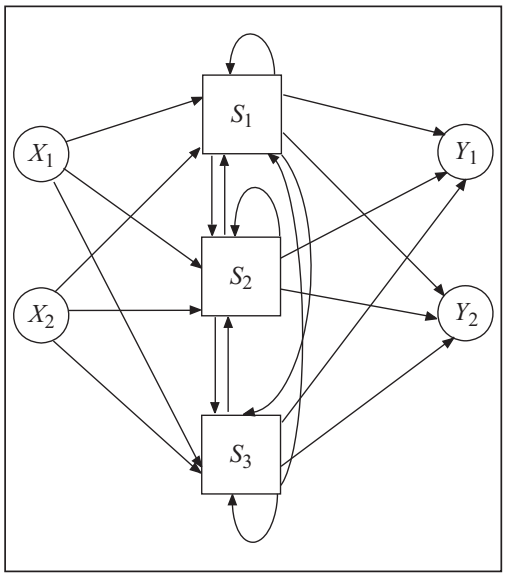

Рис. 2. Графическое изображение рабочего плана выполнения работ

Входные серверы X_1, X_2 получают недерминированным образом последовательность операций, которые необходимо выполнить на рабочих серверах. Результат выполнения пересылается на выходные серверы Y_1, Y_2. Для каждой работы каждому шагу рабочего плана приписывается множество рабочих серверов из множества $S = \{S_1, S_2, S_3\}$ и множества операций из множества $\Omega = \{w_1, w_2, w_3, w_4\}$.

Рассмотрим графическое изображение спецификации плана. Пусть рабочий план выполнения двух работ определяется данными из табл. 1 и реализуется на описанной системе. Тогда, используя обозначения из (2), получаем

$$\begin{aligned} \Omega_1 &= \{C_1^1, C_2^1, C_3^1\} = \{w_1, w_3, w_4\}, \\ \Omega_2 &= \{C_1^2, C_2^2, C_3^2\} = \{w_1, w_2, w_4\}, \\ \Omega_3 &= \{C_1^3, C_2^3, C_3^3\} = \{w_2, w_3, w_4\}. \end{aligned} \tag{5}$$

Таблица 1. Спецификация рабочего плана

Шаг	Работа 1		Шаг	Работа 2	
	операция	выполнима на		операция	выполнима на
1	w_1	S_1 / S_2	1	w_3	S_1 / S_3
2	w_2	S_2 / S_3	2	w_4	$S_1 / S_2 / S_3$

Процедуры реализации плана задаются такими последовательностями:

$$\overline{O}_1 = o_{11}o_{12} = w_1 w_2 = (C_1^1 \vee C_1^2)(C_2^2 \vee C_1^3),$$
$$\overline{O}_2 = o_{21}o_{22} = w_3 w_4 = (C_2^1 \vee C_2^3)(C_3^1 \vee C_3^2 \vee C_3^3), \quad (6)$$

где o_{ij} означает j-ю операцию i-й работы.

На основании рабочего плана, определяемого табл. 1, можно реализовать такие пути выполнения работ 1 и 2:

— операция w_1 первого шага первой работы подается через входной сервер X_1 на сервер S_1 или S_2, поскольку эта операция выполнима на этих серверах;

— операция w_3 первого шага второй работы подается через входной сервер X_2 на сервер S_1 или S_3, поскольку эта операция выполнима на этих серверах;

— операция w_2 второго шага первой работы выполняется на серверах S_2 или S_3 и поэтому результаты выполнения первой операции на серверах S_1 или S_2 передаются на серверы S_2 или S_3;

— операция w_4 второго шага второй работы выполняется на серверах S_1 или S_2, или S_3, поэтому результаты выполнения первой операции на серверах S_1 или S_3 передаются на эти три сервера;

— окончательные результаты выполнения работ передаются на выходные серверы Y_1 и Y_2.

Возможные пути выполнения работ 1 и 2 представлены на рис. 3.

Пометка x, y на дугах графа выполнения работ указывает номер операции и номер задания соответственно, а пара i, j на дугах (S_i, Y_j) указывает на то, что сервер S_i передал результаты выполнения операций работы j на выходной сервер Y_j, где $i = 1, 2, 3$; $j = 1, 2$.

Исходя из временных значений выполнения операций серверами, приведенных в табл. 2, можно выбрать наилучший путь реализации плана выполнения работ.

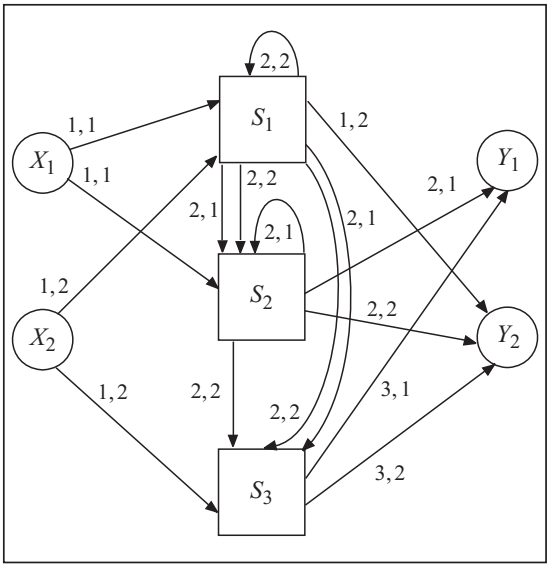

Рис. 3. Возможные пути выполнения работ 1 и 2 согласно рабочему плану

Таблица 2. Временные затраты на выполнение операций

Операция	w_1	w_1	w_2	w_2	w_3	w_3	w_4	w_4	w_4
Сервер	S_1	S_2	S_2	S_3	S_1	S_3	S_1	S_2	S_3
Время	2	3	4	3	4	2	3	2	4

5. СЕТЕВОЕ РЕШЕНИЕ ЗАДАЧИ ПЛАНИРОВАНИЯ

На основании рассмотренных работ представим задачу планирования бинарной безопасной временной сетью Петри (ВСП) [6]. Семантика функционирования ВСП описывается такими правилами.

Места ВСП ассоциируются с временными интервалами, в которых выполняются операции. Переход фишки из данного места блокируется на время выполнения операции. По окончании времени выполнения операции фишки в сети перемещаются по стандартным правилам функционирования СП. На рис. 4 приведена ВСП, которая моделирует задачу планирования выполнения двух работ тремя серверами из рассмотренного выше примера. Возможность функционирования описанной подсистемы SYS_E в целой ЭС осуществляется посредством управления через сервер S_{EB}. Следовательно, рабочий план выполнения итеративно повторяется, и такой цикл заканчивается, когда обе работы выполнены и их результаты находятся на выходных серверах. Только после этого подсистема снова будет готова к выполнению следующей последовательности операций.

Очевидным ресурсом, который требует минимизации, выступает время повторения цикла работы подсистемы. Решение можно найти, исходя из графа достижимости на путях возможных выполнений и, изменяя в ВСП путь выполнения, достичь оптимального времени выполнения.

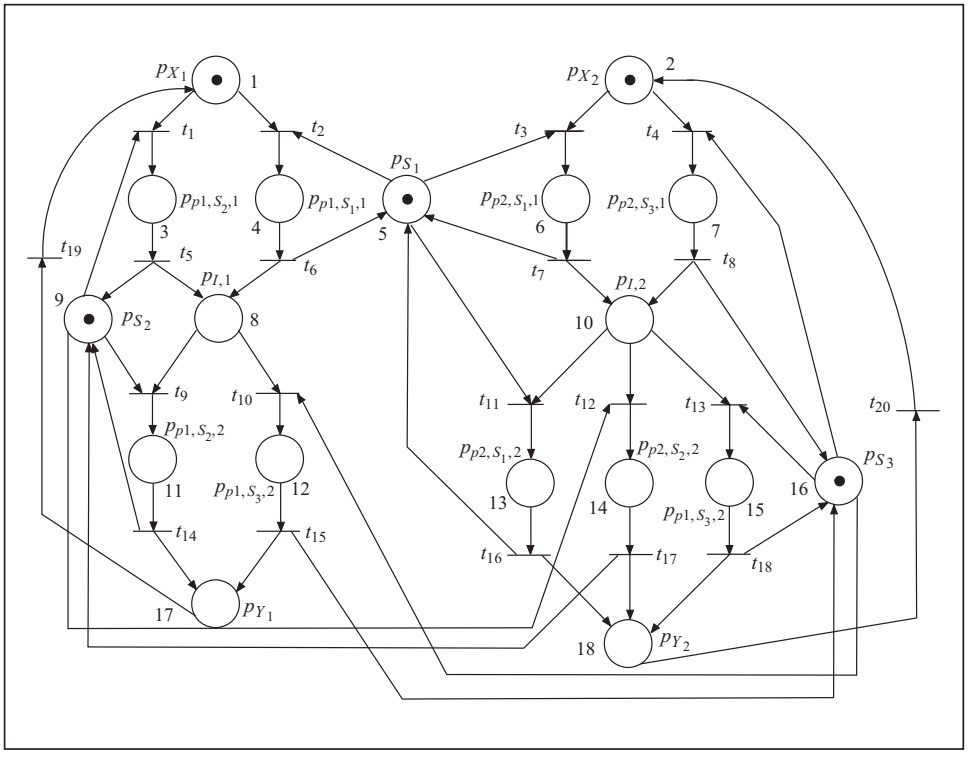

Рис. 4. Временная сеть Петри для задачи планирования выполнения работ

В приведенной ВСП места p_{X_1} и p_{X_2} соответствуют входным серверам, а места p_{Y_1} и p_{Y_2} соответствуют выходным серверам. Наличие фишки в местах p_{X_1} и p_{X_2} означает готовность сервера к работе. Временные места типа $p_{p1,S_1,1}$ соответствуют выполнению первого шага работы $p1$ на сервере S_1, и это относится к другим операциям. Наличие фишки в местах p_{S_i} означает готовность сервера S_i к выполнению операции. Промежуточные места $p_{I,1}$ и $p_{I,2}$ введены во избежание возможных дедлоков.

Исследование свойств ВСП можно свести к исследованию свойств простых СП, если определить все возможные пути выполнения заданий в ВСП, а затем, используя данные табл. 2, выбрать из них те, которые имеют наименьшие временные задержки.

В нашем примере системы SYS_E — это места 3, 4, 6, 7, 11, 12, 13, 14, 15. Переходы t_{19} и t_{20} моделируют действия, которые выполняются на сервере S_{EB}, и осуществляют дальнейшие действия с результатами выполнения работ и формируют новые задания на входных серверах.

Для анализа свойств полученной ВСП рассмотрим ее уравнение состояния $Ax = 0$, где A — матрица инцидентности ВСП, и найдем множество ее t-инвариантов. Матрица A имеет вид

$$A = \begin{bmatrix}
-1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & -1
\end{bmatrix}.$$

Решениями уравнения состояния являются такие векторы:

$$p_1 = (1,\ 0,\ 0,\ 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0),$$

$$p_2 = (0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0),$$

$$p_3 = (0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1),$$

$$p_4 = (0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1),$$

$$p_5 = (0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1),$$

$$p_6 = (0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1),$$

$$p_7 = (1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0),$$

$$p_8 = (0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0),$$

$$p_9 = (0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1),$$

$$p_{10} = (0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1).$$

Поскольку все решения покрывают позитивными значениями все переходы сети, то это значит, что сеть живая. Такие решения представляют пути выполнения операций обеих работ минимальной длины. Для каждого решения вычислим временную задержку, используя данные табл. 2. По найденным значениям временных задержек выбираем оптимальные пути выполнения заданий в системе. В общем случае по таким путям строится граф достижимых разметок ВСП и в нем отыскиваются оптимальные пути. Заметим, что такой граф может иметь большие размеры, и в этом случае применяются определенные эвристики выбора оптимального пути выполнения последовательности операций $\overline{O}_p = o_{p_1} o_{p_2} \cdots o_{p_k}$. Если такой путь найден, то рассматривается только часть графа достижимости, что может сильно упрощать задачу.

Покажем, что данная ВСП ограничена. Для этого найдем решения системы уравнений $A^{\mathrm{T}} y = 0$, где A^{T} — матрица, транспонированная к матрице A:

$$s_1 = (0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0),$$

$$s_2 = (0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0),$$

$$s_3 = (0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0),$$

$$s_4 = (1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0),$$

$$s_5 = (0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1).$$

Как видно, все места данной ВСП покрываются позитивными инвариантами, а это значит, что ВСП ограничена. Таким образом, справедлива следующая теорема.

Теорема 1. Временная сеть Петри ограничена и живая.

На основании результатов выполненного анализа можна заключить следующее:

а) из ограниченности ВСП вытекает, что граф достижимых разметок ВСП конечный и в нем можно находить оптимальные пути выполнения, которые имеют минимальную временную задержку;

б) из живучести ВСП вытекает, что работы будут выполнены, поскольку все переходы в ВСП живые;

в) независимые пути выполнения можно реализовать параллельно;

г) реализовать смешанные пути выполнения с минимальными временными задержками.

Например, путь $p_8 = t_2 t_6 t_{10} t_{15} t_{19}$ выполнения работы 1 имеет наименьшую временную задержку 5, а путь $p_6 = t_4 t_8 t_{12} t_{17} t_{20}$ выполнения работы 2 имеет наименьшую задержку 4. Эти пути независимы, что означает возможность их параллельного выполнения.

Предлагаемый подход к анализу свойств учитывает специфику сети Петри, и эта специфика позволяет свести анализ ВСП к анализу обычной черно-белой СП. Этот подход является одним из методов анализа, который часто используется.

Другой подход к верификации свойств ВСП базируется на трансляции ВСП во временные автоматы [7–9]. Как правило, ВСП и соответствующий ей временной автомат должны быть бисимуляционно эквивалентными и сохранять логические свойства спецификации. Существует несколько методов трансляции ВСП во временные автоматы, которые используют принцип проверки на модели. В частности, эти методы реализованы в системах NuSMV [10], Rabbit [11].

6. АВТОМАТНЫЙ ПОДХОД

При автоматном подходе выполнение работ 1 и 2 серверами представляется транзиционными системами ТС_1 и ТС_2, затем строится синхронное произведение ТС_1 и ТС_2, которое является моделью SYS_E выполнения работ 1 и 2 на серверах S_1, S_2, S_3.

Существует несколько определений транзиционной системы (ТС). Это также связано с тем, что исследование реальных систем, которые моделируются ТС, можно выполнять на разных уровнях абстракции. Каждому такому уровню соответствует свое определение ТС. Для исследования свойств модели необходимо наиболее полное определение ТС.

Определение 2. Транзиционной системой называется упорядоченная шестерка $\mathcal{A} = (S, Act, \rightarrow, I, AP, L)$, где S — множество состояний ТС; Act — множество действий ТС; $\rightarrow \subseteq S \times Act \times S$ — отношение переходов ТС; $I \subseteq S$ — множество начальных состояний ТС; AP — множество атомарных высказываний, $L : S \rightarrow B(AP)$ — функция разметок состояний ТС, где $B(AP)$ — булеан множества AP.

Переходы ТС записываются в виде $(s \xrightarrow{a} s')$, где $s, s' \in S$, $a \in Act$. Они могут иметь общее начало или общий конец, или общие и начало, и конец. Это значит, что в такой ТС переходы могут выполняться недетерминированно. ТС $\mathcal{A} = (S, Act, \rightarrow, I, AP, L)$ называется конечной, если множества S, Act и AP конечны. Если в множестве состояний выделено единственное начальное состояние $s_0 \in S$, то такая ТС обозначается $\mathcal{A} = (S, Act, \rightarrow, s_0, AP, L)$ и называется инициальной. Инициальную ТС назовем автоматной, если в ней выделено подмножество состояний $F \subseteq S$, элементы которого называются терминальными состояниями. Мы будем использовать только конечные ТС.

Конечная ТС естественным образом изображается размеченным орграфом $G = (V, E)$, множество вершин которого представляет множество состояний S, отмеченных формулами $L(s)$, а множество ребер — это множество переходов ТС согласно отношению переходов $\rightarrow$, которые помечены символами действий из Act.

Поведение ТС теперь формализуется с использованием понятия выполнения в ТС.

Определение 3. Конечным фрагментом выполнения в ТС $\mathcal{A}$ называется конечная последовательность состояний и действий $p = s_0 a_1 s_1 a_2 s_2 \ldots a_n s_n$, которая заканчивается в некотором состоянии s_n и является такой, что $s_i \xrightarrow{a_i} s_{i+1}$ для всех $0 \le i < n$, где $n \ge 0$. Число n называется длиной фрагмента p.

Бесконечным фрагментом выполнения в ТС $\mathcal{A}$ называется бесконечная последовательность состояний и действий $p = s_0 a_1 s_1 a_2 s_2 \ldots$ такая, что $s_i \xrightarrow{a_i} s_{i+1}$ для всех $i \ge 0$.

Максимальным фрагментом выполнения называется либо конечный фрагмент выполнения, который заканчивается в терминальном состоянии, либо бесконечный фрагмент выполнения. Фрагмент называется инициальным, если он начинается в некотором начальном состоянии $s_0 \in I$.

Выполнением в ТС $\mathcal{A}$ называется максимальный инициальный фрагмент выполнения.

Состояние $s \in S$ называется достижимым в ТС $\mathcal{A}$, если существует конечный инициальный фрагмент выполнения, который заканчивается в состоянии s, т.е. $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \ldots s_{n-1} \xrightarrow{a_n} s_n = s$.

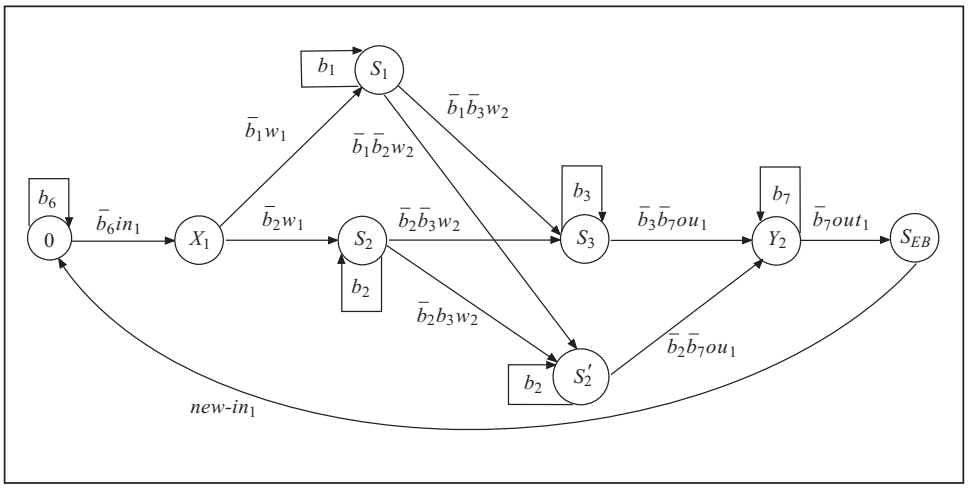

Рис. 5. TC_1 для работы 1

Из определения следует, что с каждым выполнением $p = s_0 a_1 s_1 a_2 s_2 \ldots$ в транзитивной системе $\mathcal{A}$ связывается путь $\pi = s_0 s_1 s_2 \ldots$ в ТС, последовательность действий $act(\pi) = a_1 a_2 \ldots$ и последовательность формул, которая называется трассой, $trace(\pi) = L(s_0) L(s_1) \ldots$

Трассой конечного фрагмента пути $\hat{\pi} = s_0 s_1 \ldots s_n$ называется конечная последовательность $trace(\hat{\pi}) = L(s_0 L(s_1) \ldots L(s_n))$.

Поскольку ТС работает недетерминированно, то выполнений, которые начинаются в некотором состоянии $s \in S$, может быть много. Обозначим $Paths(s)$ множество путей выполнений в ТС $\mathcal{A}$, которые начинаются в состоянии s. Тогда

$$Paths(\mathcal{A}) = \bigcup_{s \in I} Paths(s),\ Traces(s) = traces(Paths(s)),\ Traces(\mathcal{A}) = \bigcup_{s \in I} Traces(s).$$

Исследуем свойства SYS_E**.** Поскольку операции при выполнении работ в системе SYS_E могут выполняться на разных серверах, то реализация произведения $\mathrm{TC}_1 \times \mathrm{TC}_2$ требует синхронизации действий. Это значит, что если нужно выполнять операцию, например w_1, а серверы, ее реализующие, заняты другими действиями, выполнение операции w_1 согласно условию 1 должно быть отложено до освобождения сервера S_1 или сервера S_2. В целях синхронизации выполнения действий серверами вводятся общие булевы переменные b_1, b_2, b_3, b_4, b_5, b_6, b_7, значения которых может читать каждый сервер или некоторые из них. (Например, значения общих переменных выходных серверов не используются входными серверами.) Число всех серверов в системе SYS_E составляет семь, включая входные и выходные. Тогда значения $b_1 = 0$, $b_2 = 1$, $b_3 = 0$ свидетельствуют о том, что сервер S_2 занят, а серверы S_1 и S_3 свободны.

Исходя из табл. 1 спецификации рабочего плана, получаем размеченную TC_1 для работы 1, которая представлена на рис. 5, где S_2' обозначает сервер S_2.

Значения переменной $b_i = 1$ соответствует операции ожидания *wait* до тех пор, пока b_i ни будет иметь значение 0. Переход в ТС становится возможным тогда, когда значение $b_i = 0$ (когда $\bar{b}_i = 1$). Результаты выполнения работы 1 передаются серверу S_{EB}, где они обрабатываются, и для подсистемы SYS_E формируются новые работы. Аналогично строится TC_2 для работы 2, которая представлена на рис. 6, где S_1' и S_3' обозначают серверы S_1 и S_3 соответственно.

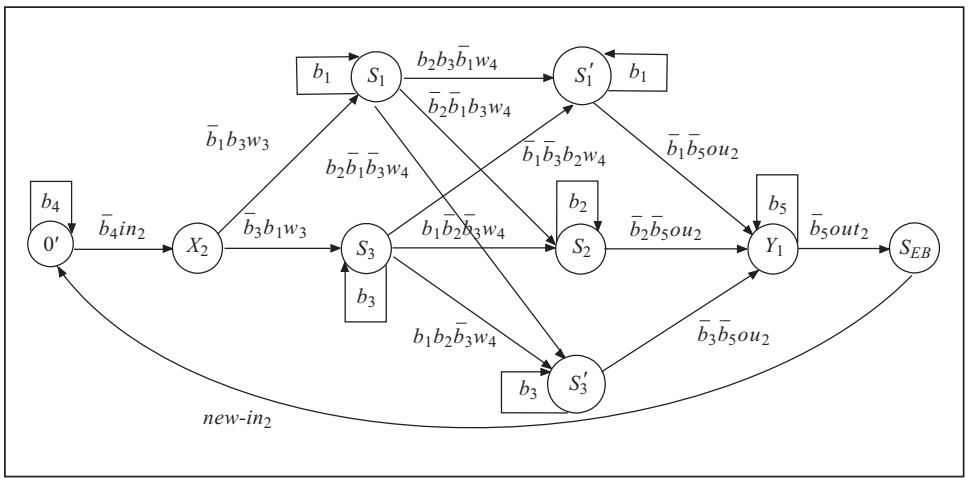

Рис. 6. TC_2 для работы 2

Для совместного выполнения работ 1 и 2 строится ТС, являющаяся синхронным произведением TC_1 и TC_2. Заметим, что если в заданном состоянии TC_i оба перехода возможны, то переход выполняется недетерминированным образом. Например, если в TC_2 из состояния X_2 возможны переходы в состояния S_1 и S_3 ($b_1 = b_3 = 0$), то переход будет в одно из этих состояний.

Дальнейшее планирование выполнения работ осуществляется, исходя из данных табл. 2. Допустим, что оптимальные пути выполнения работ 1 и 2 имеют вид

$$p_1 = 0in_1X_1w_1S_1w_2S_2ou_1Y_2out_1S_{EB},$$

$$p_2 = 0in_2X_2w_3S_3w_4S_1ou_2Y_1out_2S_{EB},$$

где ou_i и out_i — результаты выполнения работы $i = 1, 2$.

Эти пути планируются для выполнения синхронного произведения ТС:

$$T = \{((0,0),(in_1,in_2),(X_1,X_2)),((X_1,X_2),(w_1,w_3),(S_1,S_3)),$$

$$((S_1,S_3),(w_2,w_4),(S_2',S_1')),((S_2',S_1'),(ou_1,ou_2),(Y_2,Y_1)),$$

$$((Y_2,Y_1),(out_1,out_2),(S_{EB},S_{EB}))\}.$$

Таким образом, множество переходов T является множеством глобальных переходов в ТС, которая представляет синхронное произведение TC_1 и TC_2. Это множество может иметь задержки в процессе выполнения. Например, глобальный переход $((S_1,S_2),(w_2,w_4),(S_2,S_1))$ будет задержан до тех пор, пока ни закончит выполнение своего задания сервер S_1 (когда $b_1 = 1$), и только после этого будет возможен переход.

Следовательно, система глобальных переходов требует дополнительной синхронизации путем введения проверок условий готовности серверов к выполнению очередной операции. Такую проверку можно выполнить с использованием значений переменных b_i, $i = 1, \ldots, 7$. Тогда последовательности операций p_1 и p_2 принимают новый вид:

$$p_1' = 0\{b_6\}\bar{b}_6in_1X_1\{b_1\}\bar{b}_1w_1S_1\{b_2\}\bar{b}_2w_2S_2\{b_7\}\bar{b}_7ou_1Y_2out_1S_{EB},$$

$$p_2' = 0\{b_4\}\bar{b}_4in_2X_2\{b_3\}\bar{b}_3w_3S_3\{b_1\}\bar{b}_1w_4S_1\{b_5\}\bar{b}_5ou_2Y_1out_2S_{EB},$$

а множество глобальных переходов T преобразуется к виду

$$T' = \{((0,0), (\{b_6\}\bar{b}_6 in_1, \{b_4\}\bar{b}_4 in_2), (X_1, X_2)),$$
$$((X_1, X_2), (\{b_1\}\bar{b}_1 w_1, \{b_3\}\bar{b}_3 w_3), (S_1, S_3)),$$
$$((S_1, S_3), (\{b_2\}\bar{b}_2 w_2, \{b_4\}\bar{b}_4 w_4), (S_2', S_1')),$$
$$((S_2', S_1'), (\{b_7\}\bar{b}_7 ou_1, \{b_5\}\bar{b}_5 ou_2), (Y_2, Y_1)),$$
$$((Y_2, Y_1), (out_1, out_2), (S_{EB}, S_{EB}))\},$$

где $\{b_i\}\bar{b}_i w_j$ — регулярное выражение, которое соответствует конечному автомату A_{ij}, представленному на рис. 7. Далее последовательности операций p_1' и p_2' будут иметь вид

$$p_1' = A_{60} X_1 A_{11} S_1 A_{22} S_2 A_{71} Y_2 out_1 S_{EB},$$

$$p_2' = A_{40} X_2 A_{33} S_3 A_{14} S_1 A_{52} Y_1 out_2 S_{EB}.$$

Таким образом, автоматы A_{ij} синхронизируют действия в ТС путем контроля значений общих переменных $b_1, \ldots, b_7$. Включение таких автоматов в систему SYS_E обеспечивает выполнение условия 1, поскольку они контролируют окончание выполнения операций в синхронном произведении $\mathrm{TC}_1 \times \mathrm{TC}_2$. Действительно, имеет место следующая теорема.

Теорема 2. Работа в системе SYS_E, реализующей синхронное произведение $\mathrm{TC}_1 \times \mathrm{TC}_2 \times \ldots \times \mathrm{TC}_m$, выполняет условие 1.

Доказательство. Не ограничивая общности, рассмотрим случай синхронного произведения двух ТС ($m = 2$).

Поскольку справедливость условия 1 гарантируется его выполнением на каждом переходе $\mathrm{TC} = \mathrm{TC}_1 \times \mathrm{TC}_2$, то пусть $((A, B), (\{b_i\}\bar{b}_i w_j, \{b_k\}\bar{b}_k w_l), (C, D))$ является произвольным переходом в ТС. Тогда получаем модель этого перехода, представленную на рис. 8.

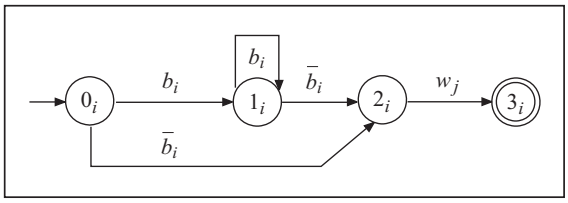

Рис. 7. Автомат A_{ij}

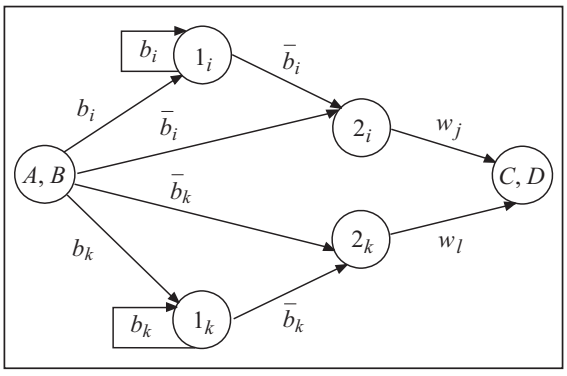

Рис. 8. Графическое изображение перехода в ТС

При переходе в ТС состоянием (C, D) может быть следующее состояние:

а) (A, D), если сервер S_k готов выполнять операцию w_l, а сервер S_j не готов выполнять операцию w_j;

б) (C, B), если сервер S_j готов выполнять операцию w_j, а сервер S_k не готов выполнять операцию w_l;

в) (C, D), если оба сервера готовы выполнять операции w_l и w_j.

В связи с этим возможны такие случаи.

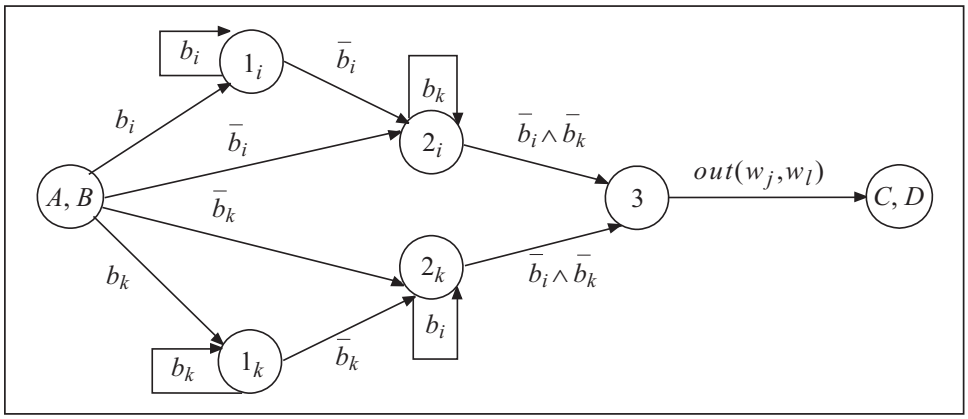

Рис. 9. Графическое изображение перехода в ТС с добавленными циклами

1. Случай $b_i = b_k = 0$ означает готовность обоих серверов выполнять операции. Получаем ситуацию в), которая гарантирует выполнение условия 1, поскольку оба предыдущих задания были выполнены.

2. Случай $b_i = 1$, $b_k = 0$ означает, что операцию w_l сервер готов выполнять, а операцию w_j не готов выполнить. Получаем ситуацию а), когда цикл в состоянии 1_i не позволяет выполнить переход в состояние C, поскольку значение $b_i = 1$ свидетельствует о незаконченности выполнения предыдущей операции.

3. Случай $b_i = 0$, $b_k = 1$ симметричен случаю 2.

4. Случай $b_i = b_k = 1$ означает, что выполнение предыдущих заданий не закончено и циклы в состояниях 1_i и 1_k вынуждают систему ожидать окончания выполнения заданий.

Таким образом, все возможные ситуации гарантируют выполнение условия 1 и ввиду произвольности перехода получаем справедливость теоремы. ■

Теорема 2 обосновывает корректность выполнения ТС в случае синхронного произведения ТС. Синхронизацию в $\mathrm{TC} = \mathrm{TC}_1 \times \mathrm{TC}_2$ можно выполнить с помощью автоматов A_{ij}. Для этого необходимо добавить циклы в состояниях 2_i и 2_k (рис. 9).

Рассмотренные случаи описывают ситуацию, когда система SYS_E функционирует в соответствии с рабочим планом. Однако возможны и аварийные ситуации, например выход из строя одного из серверов. Рассмотрим простой способ идентификации непредвиденных ситуаций в системе. Этот способ очевидным образом вытекает из временных значений, приведенных в табл. 2, и состоит в следующем: если время выполнения операции или другого действия в системе превышает время выполнения этой операции исходя из табл. 2, то это является сигналом об аварии в системе.

Каким образом это можно идентифицировать? В этом случае задействуют автоматы A_{ij}, которые трансформируются во временные автоматы [7]. Действительно, если переходы в автоматах A_{ij} дополнить временными ограничениями, то все автоматы, включая $\mathrm{TC} = \mathrm{TC}_1 \times \mathrm{TC}_2$, преобразуются во временной автомат. Тогда, добавляя аварийное состояние *Alarm* и переходы в это состояние на циклах, когда происходит превышение временных значений, получаем временной автомат, который идентифицирует аварийные ситуации. Введем время x для ТС и получим переходы с временными ограничениями, которые связываются с каждым циклом в ТС (рис. 10).

В момент превышения времени выполнения задания каким-либо сервером состоянием *Alarm* система SYS_E сигнализирует о проблемах.

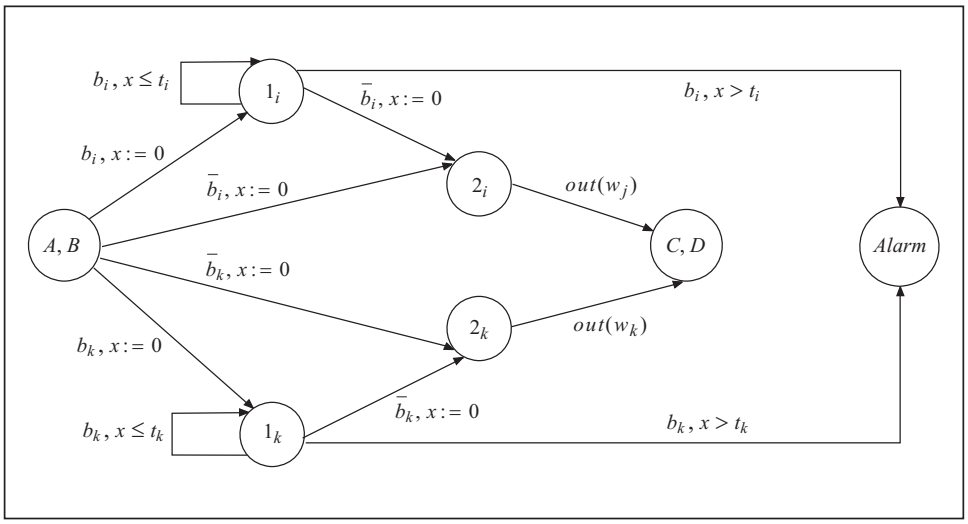

Рис. 10. Временной автомат для идентификации аварийных ситуаций в ТС

ЗАКЛЮЧЕНИЕ

Предложена формальная модель экосистемы, ориентированной на область программной инженерии. Описаны два подхода к исследованию свойств этой модели — сетевой и автоматный. Автоматный подход с использованием временных ограничений дает возможность контролировать аварийные ситуации в подсистеме SYS_E экосистемы, как наиболее активной ее части.

СПИСОК ЛИТЕРАТУРЫ

1. Сидоров Н.А. Экология программного обеспечения. *Інженерія програмного забезпечення*. 2010. № 1. С. 53–61.
2. Baader F., Calvanese D., McGuinness D.L., Nardi D., Patel-Schneider P.F. The description logic handbook. Cambridge: University Press, 2007. 601 p.
3. Kryvyi S.L., Pogorilyy S.D., Boyko Y.V. Network model of IT-infrastructure resource management. *2019 IEEE International Conference on Advanced Trends in Information Theory* (*ATIT*), Kyiv, Ukraine, 2019. P. 149–153.
4. Telenyk S., Rolik A., Bukasov M., Halushko D. Models and methods of resource management VPS hosting. *Technical Transaction*. *Automatic Control.* Politechnika Krakowska. 2013. Vol. 4-AC. P. 41–52.
5. Крывый С.Л., Погорелый С.Д., Глибовец Н.Н., Бойко Ю.В., Сидорова Н.Н. Проектирование ИТ-инфраструктуры. *Кибернетика и системный анализ.* 2018. Т. 54, № 6. С. 141–158.
6. Lee D.Y., DiCesare F. Petri net-based heuristic sheduling for flexible manufacturing. In: *Petri Nets in Flexible and Agile Automation. The Springer International Series in Engineering and Computer Scienc*e. Boston: Springer, 1995. Vol. 310. P. 149–187.
7. Alur R., Dill D.L. A theory of timed automata. *Theoretical Computer Science.* 1994. Vol. 126. P. 183–235.
8. Penczek W., Polrola A. Advanced in verification of timed Petri nets and timed automata. A temporal logic approach. Berlin; Heidelberg: Springer-Verlag, 2006. 257 p.
9. Alur R., Henzinger T., Ho P. Automata symbolic verification of embedded systems. *IEEE Trans. on Software Eng.* 1996. Vol. 22, N 3. P. 74–88.

10. Gimatti A., Clarce E., Giuchiglia E., Giuchiglia F., Pistore M., Rovere M., Sebastiani R., Tacchella A. NuSMV2: An open-source toll for symbolic model checking. *Proc. of the 14th Intern. Conf. on Computer Aided Verification (CAV'02). LNCS*. 2002. Vol. 2404. P. 359–364.

11. Beyer D. Rabbit: Verification of real-time systems. *Proc. of the Workshop on Real-Time Tolls (RT-TOOLS'01)*. 2003. P. 13–21.

Надійшла до редакції 09.01.2020

С.Л. Кривий, О.О. Гріненко

ЕКОСИСТЕМИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Анотація. Розглянуто модель екосистеми, яка орієнтована на галузь програмної інженерії. Запропоновано два підходи до дослідження властивостей моделі — мережевий і автоматний. Зокрема, розглянуто способи пошуку оптимального плану виконання робіт в екосистемі, ідентифікації аварійних ситуацій, можливість паралельного виконання завдань.

Ключові слова: екосистема, транзиційні системи, скінченні автомати, часові автомати, мережі Петрі.

S. Kryvyi, E. Grinenko

ECOSYSTEMS OF SOFTWARE ENGINEERING

Abstract. A formal model of an ecosystem, focused on software engineering, is proposed. Two approaches to the analysis of properties of this model, which are called network and automata approaches, are described. In particular case, methods for searching of optimal plan for task performances in ecosystem, identification of risk situations, possibility of paralell executions of tasks are considered.

Keywords: ecosystem, transition system, timed automata, Petri Net, scheduling.

Крывый Сергей Лукьянович,
доктор физ.-мат. наук, профессор, профессор кафедры Киевского национального университета имени Тараса Шевченко, email: sl.krivoi@gmail.com.

Гриненко Елена Александровна,
старший преподаватель Национального авиационного университета, Киев, email: gsa_ck@ukr.net.