УДК 004(4’22+054)

С. И. ЧАЙНИКОВ, А. С. СОЛОДОВНИКОВ

ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ СИНТЕЗА СТРУКТУРЫ ПРОБЛЕМНО-ОРИЕНТИРОВАННЫХ ПРОГРАММНЫХ КОМПЛЕКСОВ

Предлагается информационная технология синтеза проблемно-ориентированных программных комплексов, которая применяется для эволюционных моделей жизненного цикла программного обеспечения и обеспечивает конфигурирование и комплексирование программных средств с возможностью управления реализацией и восстановлением вычислительных процессов в параллельных и распределенных структурах вычислительных средств. Технология применяется в рамках процессов анализа требований к программным средствам, проектирования архитектуры, конструирования и комплексирования программных средств.

1. Введение

В последнее время широко распространены такие технологии проектирования программного обеспечения (ПО) как SSADM, Meris, TDD, Gherkin [1-3]. Они позволили создать компонентно-ориентированные, сервисно-ориентированные методы, методы генерации, что обусловливается их эффективностью, малыми трудовыми и временными затратами для получения качественного программного продукта.

Однако существующие методы построения ПО имеют недостатки, связанные с отсутствием либо гибкой архитектуры программных средств (ПС), либо возможности динамического конфигурирования функциональности рабочих мест пользователей в условиях изменяющихся требований при использовании эволюционных жизненных циклов (ЖЦ) ПО [1,3]. При этом также не учитывается применение методов упрощения структуры ПК на основе анализа графовой модели его структуры в целях снижения временных затрат на проектирование и конфигурацию.

Одним из новых направлений в проектировании и разработке являются методы автоматизированного синтеза ПО с учетом меняющихся требований заказчика, главное преимущество которых – возможность предварительной оценки архитектурных и функциональных особенностей системы [4,5]. В этой связи является *актуальным* использование методов формализации гибкой архитектуры ПО с возможностью ее динамического изменения на основе графовой модели структуры ПС для решения задач в условиях изменяемых требований. Такие методы позволяют сформировать предварительное представление о системе до ее генерации, обеспечить динамическое конфигурирование и управление реализацией и откатом вычислительных процессов (ВП) для задач пользователей системы. Известные методы синтеза ПС используются, в основном, для формирования системы с компонентно-ориентированной или сервис-ориентированной архитектурой. Однако принципы и модели, которые лежат в основе метода формализации архитектуры ПО, можно адаптировать для решения задач, возникающих в процессе автоматизированного синтеза проблемно-ориентированных ПК. Поэтому *целью исследования* является создание информационной технологии (ИТ) синтеза структуры проблемно-ориентированных ПК с использованием графовых моделей структуры таких систем. *Задачами исследования* являются разработка методов формирования рациональной структуры проблемно-ориентированных комплексов и применение этих методов в предложенной ИТ синтеза структуры ПК.

2. Графовая модель структуры проблемно-ориентированного ПК в ярусно-параллельной форме

Графовая модель представляет собой совокупность вершин, которым сопоставляются элементарные программные модули, выполняемые последовательно или независимо друг от друга. Ориентированные дуги между ними определяют тип связей по данным и описываются парой множеств характеристик программных модулей. Графовая модель формирует-

ся с избыточной для заданной предметной области (ПрО) функциональностью и обеспечивает гибкую конфигурацию системы в условиях эволюционно меняющихся требований.

Для построения модели требуются данные спецификаций требований к конечному программному продукту, информация, получаемая на этапах предпроектных исследований, существующие архитектурные решения, паттерны. Конечный вид графовой модели определяется выражением:

$$G = F(G_{исх}), \quad (1)$$

где $F(G_{исх})$ определяет множество операций над исходной графовой моделью, которые позволяют привести ее к ярусно-параллельной форме с супервершинами, а $G_{исх}$ явдяется исходной графовой моделью вида:

$$G_{исх} = (V_{исх}, X_{исх}), \quad (2)$$

здесь $V_{исх}$ – множество вершин *v*, которым сопоставляются программные модули (или же, в общем понимании, ВП), а $X_{исх}$ – множество ориентированных дуг $x_{ij} = (v_i, v_j)$ графа $G_{исх}$, определяющих зависимость по данным между программными модулями. Формирование множества вершин и дуг графа (2) происходит на основе выделения в ПрО соответствующих функциональных подсистем объекта исследования и установления потоков данных между ними. Для каждой *i*-й вершины модели (1) выделены два вектора данных:

$$D_i^{in} = \left\{d_1^{in}, d_2^{in}, ...d_k^{in}\right\} \quad (3)$$

– вектор входных данных, неизменяемых в процессе работы, и

$$D_i^{out} = \left\{d_1^{out}, d_2^{out}, ...d_k^{out}\right\}, \quad (4)$$

– вектор выходных данных, которые были модифицированы в процессе работы программного модуля. В случае изменяемых требований, определяющих модификацию архитектуры и расширение функциональности, графовая модель (1) дополняется подмножеством новых программных модулей $V^{H} = \{v_i^{н}\}$ и подмножеством модифицируемых модулей $V^{M} = \{v_i^{м}\}$, позволяющих сконфигурировать структуру ПК для новой версии требований. Для получения новой версии структуры программы из графа (1) исключается подмножество вершин $V^{З} \subseteq V$, которые заменяются подмножеством V^{M}, т.е. убираются программные модули устаревшей версии. Такая же операция проделывается с дугами $X^{З} \subseteq X$. Это позволяет определить постоянную часть графа G: $G^{П} = \left(V^{П}, X^{П}\right), G^{П} \subseteq G$, где $V^{П} = V \setminus V^{З}$, а $X^{П} = X \setminus X^{З}$. Полученная графовая модель обладает следующими основными свойствами. На ней отсутствуют висячие вершины, для которых выполняется условие:

$$\forall v \in V, \{V' \subseteq V : |V'| > 1 : \exists v_i \in V' : \deg^-(v_i) = 0\}. \quad (5)$$

Для модели отсутствуют изолированные вершины, для которых выполняется условие:

$$\forall v \in V, \{\exists v_i \in V \,|\, \deg^+(v_i) = \deg^-(v_i) = 0\}. \quad (6)$$

Для нее отсутствуют парные дуги:

$$\forall i, j \in N, \{v_i, v_j \in V \mid \exists! x_{ij} \in X\} \quad (7)$$

и вершины с петлями:

$$\forall i \in N, \{v_i \in V \mid x_{ii} \notin X\} \quad (8)$$

Кроме этого, для ациклической графовой модели все вершины, для которых задан номер с учетом топологической сортировки и выполняется условие:

$$\begin{aligned} &\forall i, j \in N, i < j, \{\exists v_i, v_j \in V : \mu[v_i, v_j] \mid \deg^-(v_i) = \deg^+(v_i) = \\ &= \deg^-(v_{i+1}) = \deg^+(v_{i+1}) = ... = \deg^-(v_j) = \deg^+(v_j) = 1\}, \end{aligned} \quad (9)$$

где $\mu[v_i, v_j]$ – маршрут от вершины v_i к вершине v_j, объединяются в одну супервершину. Назначением предложенной графовой модели является описание структуры формируемого ПК, которая позволяет оценить архитектуру ПС до непосредственного комплексирования и

обеспечить динамическое конфигурирование рабочих мест пользователей, что является преимуществом предложенной модели. Приведение графовой модели к виду (2) для обеспечения выполнимости свойств (5)-(9) опирается на предложенный метод объединения вершин графовой модели структуры програмного обеспечения информационной системы на основе оценки сложности и связности программных модулей [6].

3. Метод объединения вершин графовой модели структуры проблемно-ориентированного ПК

Данный метод предназначен для сокращения временных затрат на разработку системы путем объединения вершин в супервершины на базе модифицированного алгоритма Косарайо, дополнительно обрабатывающего графовую модель на основе комплексного критерия оценки эффективности структуры ПК [6]:

$$K_C = \frac{K_{общ}}{K'_{общ}} \cdot \frac{K'_{слож}}{K_{слож}}, \qquad (10)$$

где $K'_{слож}$ и $K'_{общ}$ – значения среднего коэффициента сложности и обобщенного критерия, полученные после оптимизации графовой структуры, а $K_{слож}$ и $K_{общ}$ – значения критериев до оптимизации.

Метод объединения вершин графовой модели структуры проблемно-ориентированных ПК включает следующие этапы:

Этап 1. Устранение топологических некорректностей графовой модели в соответствии с формулами (5)-(6).

Этап 2. Получение для каждого *i*-го программного модуля количества вызываемых модулей, количества элементов и структур данных, обновляемых *i*-м модулем, и значения коэффициента системной сложности.

Этап 3. Расчет среднего коэффициета сложности $K_{сложн}$ (10) .

Этап 4. Расчет значения критериев $K_{общ}$ и $K_{сложн}$ и получение значения комплексного критерия оценки эффективности структуры программного средства K_C (10) для заданной версии требований к конфигурации системы.

Этап 5. Поиск компоненты сильной связности для заданного графа и конденсация найденной компоненты.

Этап 6. Поиск пары вершин по условию (7) после того, как на графе не осталось ни одной компоненты сильной связности (для любых двух вершин v_i и v_j невозможно найти двух одновременно существующих ориентированных путей $\mu[v_i, v_j]$ и $\mu[v_j, v_i]$).

Этап 7. Вычисление значение комплексного критерия оценки эффективности структуры программного средства (10) для найденной пары вершин.

Этап 8. Объединение пары вершин в одну супервершину, если значение критерия строго меньше единицы, и повторение с шага 3 до тех пор, пока значение комплексного критерия (14) не будет удовлетворять конфигурационным требованиям.

Для предложенного метода проведена оценка сложности, которая совпадает с алгоритмом Косарайо и соответствует величине $O(n)$. Преимуществом предложенного метода является то, что он позволяет уменьшить временные затраты на конфигурирование путем упрощения структуры графовой модели, опираясь на ее топологические особенности и конфигурационные требования к системе.

Графовая модель структуры проблемно-ориентированного ПК, обработанная на основе метода объединения вершин, подзволяет описать статический аспект формируемой ПС. Для оценки поведенческих свойств системы проверки нефункциональных к ней требований необходимо построение автоматной модели комплекса на основе автоматного метода.

4. Автоматный метод оценивания последовательности взаимодействия модулей проблемно-ориентированного ПК

Автоматный метод используется для проектирования исполнителя ВП, моделируя сценарии взаимодействия программных модулей и сравнивая результаты моделирования с нефункциональными требованиями к ПК. Он применяет графовую модель структуры проблемно-ориентированного ПК, для которой определены вершины, отвечающие за формирование пользовательского диалога и за формирование соответствующих им подграфов модулей, реализующих связанные задачи системы. Метод основан на модели конечного автомата (КА), которая задается стандартным набором элементов:

$$A = \{S, X, Y, s_0, \delta, \lambda\},$$

где S – конечное непустое множество состояний; X – конечное непустое множество входных сигналов (входной алфавит); Y – конечное непустое множество выходных сигналов (выходной алфавит); $s_0 \in S$ – начальное состояние; $\delta : S \times X \to S$ – функция переходов; $\lambda : S \times X \to Y$ – функция выходов. При этом множества состояний, переходов, входных и выходных сигналов задаются графически с помощью диаграммы переходов, а структура конечных автоматов, взаимосвязи между вложенными автоматами – с помощью диаграммы классов. Для реализации задач прямого (автомат AForwardSM) и обратного (восстановление результатов – автомат AbackwardSM) управления ходом ВП предложено использовать основной управляющий КА, который обеспечивает взаимодействие двух вложенных КА. На основании определенной структуры сформулированы основные задачи, реализуемые предложенными КА. Для управляющего КА это:

1) формирование подграфа задачи;
2) проверка состояния задачи;
3) проверка результата на корректность;
4) изменение состояния задачи.

Для автомата AForwardSM это:

1) запуск ВП;
2) архивация данных ВП.

Для автомата ABackwardSM это восстановление архивных данных для каждой вершины соответствующих ярусов.

Предложенный метод включает в себя следующие этапы:

Этап 1. Выделение базовых состояний ПК и условий переходов на основе входных и выходных данных вершин графовой модели.

Этап 2. Формирование структуры управляющего КА и его вложенных подавтоматов, реализующих поведение системы, с учетом данных спецификаций вершин графовой модели.

Этап 3. Определение принципов взаимодействия КА с программными модулями на основе архитектурных шаблонов объектно-ориентированного программирования для организации распределенного или параллельного выполнения программы.

Этап 4. Формирование исходной информации, требуемой для организации работы КА, такой как матрицы смежности графовой модели структуры проблемно-ориентированного ПК и транспонированной матрицы смежности для управления прямым и обратным ходом ВП.

Этап 5. Разработка структуры будущего ПК с учетом взаимосвязи графовой модели и управляющих конечных автоматов.

Этап 6. Генерация ПК на основе полученной структуры.

Предложенные автоматные модели обладают следующими обязательными свойствами. Для автомата AForwardSM: 1) ВП $P(v_i)$ для заданной i-й вершины будет запущен только по запросу пользователя; 2) гарантия того, что процесс $P(v_i)$ обязательно когда-нибудь будет запущен, если не будет ошибок вычислений; 3) гарантия того, что процесс $P(v_i)$ будет запущен только тогда, когда вершина v_i находится в состоянии «не запущен» или «выполнен»; 4) вершина v_i когда-нибудь обязательно будет находиться в состоянии «не запущен» или «выполнен»; 5) для каждой необходимой вершины графовой модели

будет выполнен процесс архивации данных; 6) для последовательности ВП на основе существующей ЯПФ графовой модели можно выделить такой ВП, который становится активным только тогда, если все предшествующие процессы завершат свою работу; 7) для всего множества ВП программной системы можно выделить такое подмножество ВП, которые размещены на одном ярусе ЯПФ графовой модели и не имеют информационных зависимостей друг от друга, что позволяет запускать их одновременно.

Для автомата ABackwardSM: 1) восстановление данных ВП (backtracking) будет выполнено только по запросу пользователя; 2) восстановление данных ВП будет выполнено только при наличии архива данных для каждой вершины подграфа ВП; 3) для каждой вершины выделенного подграфа произойдет восстановление данных только в состоянии, если другой пользователь не задействовал эту вершину для себя (статус «не запущен») и если восстановление данных не происходило вовсе (статус «не восстановлен»), или данные уже были восстановлены ранее (статус «восстановлен»).

На основе определенных и обозначенных состояний КА (табл. 1) и событий, по которым КА переходит в эти состояния (табл. 2), формализуются свойства модели на базе темпоральной логики LTL, которые отвечают требованиям к поведению ПС и дополняются новыми свойствами в случае появления очередных версий нефункциональных требований.

Таблица 1

Переходы конечного автомата

Обозначение	Наименование
e1	получено разрешение
e5	ошибка выполнения
e6	номер вершины меньше максимального для текущего яруса
e7	номер текущего яруса меньше количества ярусов
e16	заданы условия и параметры ВП
e43	данные считаны
e81/ e82	архив существует / отсутствует
e101/ e102	данные архива не восстанавливались / восстанавливаются
e103	данные архива восстановлены
e121/e122/e123	ВП не запущен / выполняется / выполнен

Таблица 2

Состояния конечного автомата

Обозначение	Наименование
o3.zf22	инициализация ВП для автомата AForwardSM
o3.zf24	переход к следующему ВП
o3.zf28/ o3.zf31	запуск ВП/ ожидание завершения ВП
o3.zf35/ o3.zf36	архивация данных/получение данных
o2.zf45	восстановление данных для автомата ABackwardSM

Для автомата AForwardSM это:

1) $\neg(\neg e1 \boldsymbol{U} o3.zf28)$;

2) $\neg(e5 \boldsymbol{U} o3.zf28)$;

3) $\neg(\neg(e121 \vee e123) \boldsymbol{U} o3.zf28) = \neg((\neg e121 \wedge \neg e123) \boldsymbol{U} o3.zf28)$;

4) $\boldsymbol{F} o3.zf31$;

5) $e43 \boldsymbol{F} o3.zf35$.

Свойство 6) для автомата AForwardSM формализовано следующим образом:

$$F_{общ}^{AV} = \neg(F_1^{AV} \wedge F_2^{AV} \wedge ... \wedge F_n^{AV}) = \neg F_1^{AV} \vee \neg F_2^{AV} \vee ... \vee \neg F_n^{AV},$$

где $F^{AV} = \neg(\neg el22 \boldsymbol{U} o3.zf31) \wedge \neg(el21 \boldsymbol{U} o3.zf22) \boldsymbol{U} (\underline{\neg(\neg} el6 \boldsymbol{U} o3.zf28))$, поскольку в том случае, когда существует n вершин, таких что $\forall k = \overline{m,n}; \exists x_{ki} = (v_k, v_i)$, где m и n целые числа и полустепень захода $\deg^+(v_i) = n$, общее логическое высказывание заключается в одновременном выполнении этого свойства для всех n дуг.

Свойство 7) для автомата AForwardSM формализовано таким образом:

$$F_{общ}^{PV} = \neg(F_1^{PV} \wedge F_2^{PV} \wedge ... \wedge F_n^{PV}) = \neg F_1^{PV} \vee \neg F_2^{PV} \vee ... \vee \neg F_n^{PV},$$

где

$$F^{PV} = \neg(\neg(\neg(\neg(e6 \wedge e7) \boldsymbol{U} o3.zf28)) \boldsymbol{U} (\neg(\neg el22 \boldsymbol{U} (o3.zf24 \wedge o3.zf36))))) =$$
$$= \neg((\neg e6 \vee \neg e7) \boldsymbol{U} o3.zf28 \boldsymbol{U} (\neg(\neg el22 \boldsymbol{U} (o3.zf24 \wedge o3.zf36)))),$$

так как, если задать подмножество вершин $v_i \in V_r$, где $\overline{i = m,n}$, принадлежащих одному ярусу и между которыми не существует дуги, то в общем случае логическая формула также будет выглядеть, как конъюнкция формул, соотнесенных с заданными вершинами.

Для автомата ABackwardSM свойства формализованы таким образом:

1) $\neg(\neg el \boldsymbol{U} o2.zf45)$;

2) $\neg(\neg e82 \boldsymbol{U} o2.zf45) \vee \neg(\neg e81 \boldsymbol{U} o2.f45)$;

3) $\neg(\neg(el21 \wedge (el01 \vee el03) \boldsymbol{U} o2.zf45) \vee \neg((el22 \wedge el02) \boldsymbol{U} o2.f45) =$

$= \neg((\neg el21 \vee \neg el01 \wedge \neg el03) \boldsymbol{U} o2.zf45) \vee \neg((el22 \wedge el02) \boldsymbol{U} o2.f45)$.

Сформулированные свойства являются основой для определения правил исполнения ВП, поэтому автоматная модель, разработанная в соответствии с предложенным методом, позволяет сформировать ограничения к структуре ПК с учетом меняющихся требований и повысить надежность его функционирования, опираясь на данные графовой модели структуры проблемно-ориентированного ПК, что отличает метод от технологии BDD.

5. ИТ конфигурирования поблемно-ориентированного ПК в условиях меняющихся требований

ИТ конфигурирования поблемно-ориентированных ПК в условиях меняющихся требований позволяет отразить требования текущей версии на графовой модели и обеспечить переконфигурацию модулей проблемно-ориентированных ПК. Технология включает в себя следующие этапы (рис.1).

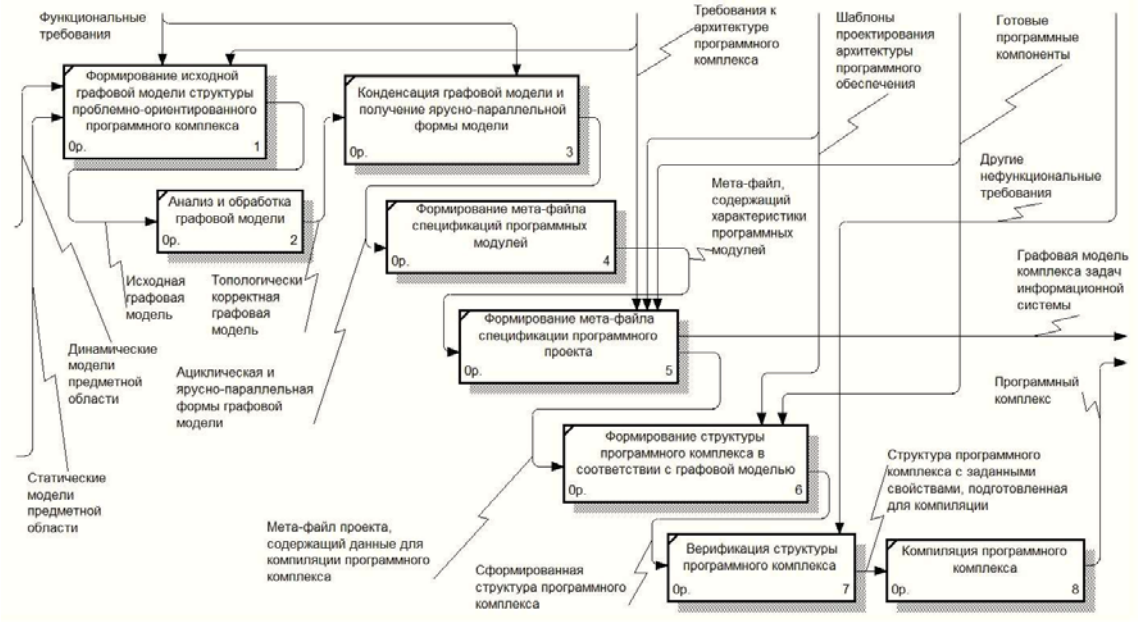

Рис. 1. Этапы информационной технологии синтеза проблемно-ориентированных ПК

Этап 1. Формирование исходной графовой модели структуры проблемно-ориентированного ПК.

Этап 2. Анализ и обработка графовой модели.

Этап 3. Конденсация и получение ЯПФ модели.

Этап 4. Формирование мета-файла спецификаций модулей.

Этап 5. Формирование мета-файла спецификации проекта.

Этап 6. Формирование структуры ПК в соответствии с графовой моделью.

Этап 7. Верификация структуры ПК.

Этап 8. Компиляция ПК.

Первый этап информационной технологии обеспечивается информацией из доступных документов, описывающих требования заказчика: первичные спецификации, диаграммы статического и динамического моделирования ПрО. На основе функциональных и архитектурных требований к ПК для сущностей модели ПрО определяются функции объектов, формирующих графовую модель, которая на втором этапе проверяется на топологическую корректность. На третьем этапе на основе требований заказчика выполняется метод объединения вершин графовой модели и формируется рациональная структура графа. На четвертом этапе на базе полученной структуры происходит описание программных модулей при помощи мета-файлов спецификаций. На пятом этапе с использованием архитектурных шаблонов формируется необходимый для генерации ПК и динамической конфигурации мета-файл программного проекта, который описывает принципы взаимодействия модулей. Данный файл является входным для процесса проверки выполнения нефункциональных требований на основе оценки верификации последовательности взаимодействия модулей на седьмом этапе, после успешного окончания которого выполняется генерация комплекса и отчуждение графовой модели для конечного пользователя сформированной системы. В случае новой версии требований процесс повторяется с третьего этапа.

Информационная технология позволяет осуществить синтез графовой модели структуры проблемно-ориентированного ПК и динамически выполнить конфигурацию его архитектуры и функциональности рабочих мест в условиях меняющихся требований заказчика. При проверке применимости предложенной ИТ для синтеза структуры ПК «Автоматизированная система подготовки производства сложных электронных устройств» по предоставленной информации о ПрО был рассчитан процент ручной настройки системы, необходимой для динамического конфигурирования. Расчет выполнен на основе LOC-оценок программного кода и сравнивался со стандартным методом конфигурирования системы на основе данных отчета по внедрению ERP-систем в производство за 2015 год [7], согласно которому большинство предприятий (в среднем 44% предприятий всех отраслей промышленности и 38% предприятий сферы машиностроения) требуют от 26 до 50% изменения исходного кода в соответствии с настройками и требованиями заказчиков системы (рис. 2). Предложенная технология позволила сократить процент ручной работы персонала до 10% для модификации исходного кода традиционным способом в размере 26 и до 18% для соответствующей границы в 50%, т.е. предложенная информационная технология значительно снижает временные затраты на переконфигурацию ПК – на 36 %.

6. Выводы

ИТ конфигурирования проблемно-ориентированных ПК в условиях изменяющихся требований в отличие от существующих технологий дополнена этапами анализа и обработки модели ПрО в целях синтеза графовой модели проблемно-ориентированного ПК, конфигурирования этой структуры с учетом текущих функциональных требований и проверки выполнения нефункциональных требований на основе оценивания последовательности взаимодействия модулей в рамках процессов анализа требований, проектирования архитектуры, конструирования и комплексирования ПК, что обеспечивает значительное снижение стоимости изменений программного обеспечения. Предложенная технология позволяет снизить временные затраты на конфигурирование программного обеспечения на 36 % и применяется как дополнение к существующим технологиям проектирования и разработки программного обеспечения, подразумевая ее использование для эволюционных ЖЦ таких технологий как BDD, TDD и других в рамках процессов анализа требований к программ-

ным средствам, проектирования архитектуры, конструирования и комплексирования программных средств.

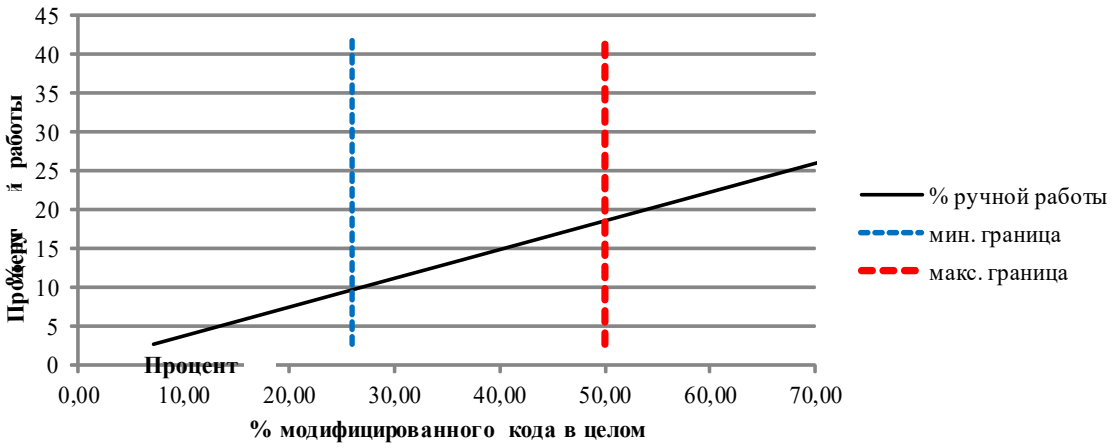

Рис. 2. Доля ручной настройки системы относительно процента изменения исходного программного кода

Определение принципов управления, отката и синхронизации ВП в параллельных вычислительных средах, а также методов формирования гибкой архитектуры программных комплексов с ее адаптацией для сервис-ориентированной и облачной архитектур является предметом дальнейших исследований.

Список литературы: **1**. *Amodeo E.* Learning Behavior-driven Development with javascript + Code / Enrique Amodeo. Birmingham:Packt Publishing. 2015. 392 p. **2**. *Бек К.* Экстремальное программирование: разработка через тестирование. Библиотека программиста. СПб.: Питер, 2003. 224 с. **3**. *Smart J. F.* BDD in Action: Behavior-driven development for the whole software lifecycle / JOHN FERGUSON SMART / Publisher: Manning Publications, Shelter Island, NY. 2015. 384 p. **4**. *Лаврищева Е.М.* Сборочное программирование. Основы индустрии программных продуктов/ Е.М. Лаврищева, В.Н. Грищенко. Киев: Наук. думка. 2009. 372 с. **5**. *Лаврищева Е.М.* Software Engineering компьютерных систем. Парадигмы, технологии и CASE-средства программирования / Е. М. Лаврищева. К.: Наук. думка. 2013. 283 с. **6**. *Солодовников А.С.* К вопросу оценивания эффективности и сложности структуры программного средства / А.С. Солодовников // Проблеми інформаційних технологій. 2014. №2 (16). С.125-129. **7**. The 2015 Manufacturing ERP Report / Panorama Consulting Solutions. Denver, Colorado. 2015. 13 p. https://cdn2.hubspot.net/hubfs/240423/Whitepapers/2015_Pan_Man_ERP_Report.pdf

Поступила в редколлегию 07.12.2015

Чайников Сергей Иванович, канд. техн. наук, профессор кафедры системотехники ХНУРЭ. Адрес: Украина, 61166, Харьков, пр. Науки, 14.

Солодовников Андрей Сергеевич, ассистент кафедры биологической и медицинской физики и медицинской информатики, Харьковский национальный медицинский университет, тел. 066-774-773-1, e-mail: andrey.sldv@rambler.ru