ИПМ им.М.В.Келдыша РАН

Абрау-2017 • Труды конференции

Е.М. Лаврищева, В.С. Мутилин,
А.Г. Рыжов

Аспекты моделирования вариабельных программных и операционных систем

Рекомендуемая форма библиографической ссылки

Лаврищева Е.М., Мутилин В.С., Рыжов А.Г. Аспекты моделирования вариабельных программных и операционных систем // Научный сервис в сети Интернет: труды XIX Всероссийской научной конференции (18-23 сентября 2017 г., г. Новороссийск). — М.: ИПМ им. М.В.Келдыша, 2017. — С. 327-340. — URL: http://keldysh.ru/abrau/2017/70.pdf doi:10.20948/abrau-2017-70

Размещена также презентация к докладу

Аспекты моделирования вариабельных программных и операционных систем[1]

Е.М. Лаврищева[2], В.С. Мутилин[3], А.Г. Рыжов[4]

[2]Е.М. Лаврищева <lavr@ispras.ru>
Институт системного программирования РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.
Московский физико-технический институт,
141700, Россия, Московская обл., г. Долгопрудный, Институтский пер., д. 9.
[3]В.С. Мутилин<mutilin@ispras.ru>
[4]А.Г. Рыжов <ryzhov@ispras.ru>
Институт системного программирования РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.

Аннотация. Рассматриваются подходы к формированию вариабельности действующих и создаваемых программных систем (ПС), и их семейств (СПС). Предложенная модель характеристик (Feature Model) K. Pohl в проекте SEI Product Line и Product Family, а также модель конвейерной сборки готовых артефактов в пространстве задач и решений K. Czarnecki послужили исходной концепцией для создания модели вариабельности для программных и операционных систем (ОС). Определены новые модели ПС и ОС, включающие готовые ресурсы (ГОР) – функциональные и интерфейсные элементы, а также модель характеристик (МХ) элементов, по которым генерируются варианты ПС из их семейств. Определена концепция управления вариабельностью и конфигурационной сборкой ПС из ГОР в объектно-компонентном методе (ОКМ) для получения вариантов выходного продукта. Предложен подход к тестированию вариантов ПС.

Ключевые слова: программная система, операционная система, семейство программных систем, вариабельность, функциональный, интерфейсный элемент, модель характеристик, управление конфигурацией, верификация, тестирование.

1. Введение

Последние годы в технологии и инженерии сформировались новые методы моделирования ПС и их семейств. Методы ориентированы на обеспечение изменяемости (вариабельности) готовых (legacy) и вновь создаваемых систем. Одна из первых моделей характеристик (МХ), разработана в SEI (sei.cmu.edu) на продуктовой линии Product Line/Product Family (2002) для изготовления программных продуктов (ПП) и их семейств (СПП) из готовых ресурсов (artifacts, reuses, assets, servises и др.). Семейство СПП согласно стандарта ISO/IEC FDIS 24765:2009 – «это группа продуктов или услуг, которые имеют

[1] Работа поддержана грантом Российского фонда фундаментальных исследований №16-01-00352

общее управляемое множество свойств, удовлетворяющих потребностям определенного сегмента рынка или вида деятельности». Сформирована концепция конвейерной сборки систем и семейств ПС, основанная на модели МХ К. Чарнецки (K. Czarnecki) из готовых reuses (Reusability, 1987) [1-5]. Под *системой* понимается совокупность программных артефактов и reuses, из которых осуществляется конфигурационная сборка системы. *Семейство ПС* – это совокупность ПС, которые имеют множество внешних характеристик, свойственных каждому отдельному члену ПС. Разработан логико-математический объектно-компонентный метод (ОКМ) моделирования функциональных и интерфейсных элементов предметной области. Эти элементы являются общими для всех видов ПС семейства и используются при генерации выходных вариантов ПС или СПС [4-9].

Концепция вариабельности для ПС и их семейств начала использоваться и в действующих Legacy-системах (ОС IBM, Intel, Linux и др.). Она обсуждается на международных конференциях (ICTERI 2006–2015, Reusability 1994–2015, Vamos 2005–2016 и др.) и реализуется индустриально (www.sei.cmu.edu/productlines, www.7dragons.ru/ru и др.). Общее, что их объединяет – это линии производства продуктов из ГОР, модели вариабельности и их верификация, управление конфигурационной сборкой систем из ГОР для получения вариантов систем.

В работе предлагаются новые оригинальные модели ПС и МХ, ориентированные на обеспечение изменяемости программных и операционных систем. Рассматривается метод ОКМ [9] проектирования ПС и их семейств с помощью объектов, компонентов и сервисов. Описывается метод управления конфигурацией вариантов ПС из ГОР с использованием вариабельных моделей МХ, ПС и ОС.

2. Базовые основы вариабельности систем и семейств

Модель характеристик для производства ПС впервые предложил К. Похл (K.Pohl) [1-5] с учетом требований и особенностей новых языков и инструментов их реализации (*VSL, ConIPF, CBFM, Koalish, COVAMOF* и др.), с помощью которых задаются понятийные и количественные зависимости между характеристиками МХ и в конфигурационном файле ПС [5-9].

Дается описание основ моделирования вариабельных продуктов и систем, предложенных в работах К. Похла, К. Чарнецки [4, 5], а также логико-математического четырехуровневого объектно-компонентного метода (ОКМ) [9] к проектированию ПС и их семейств. Предложен формальный аппарат определения объектной модели и МХ в ОКМ для проектирования семейств СПС с помощью функциональных и интерфейсных объектов. Определена модель конфигурационной сборки и управляемое изготовление вариантов систем и их семейств на сайте www.7dragons.ru/ru.

2.1. Вариабельность продуктов и их семейств

К. Похл ввел понятие вариабельности ПС. ***Вариабельность*** – это свойство системы к расширению, изменению, приспособлению или конфигурированию с целью использования в определенном контексте и обеспечении последующей его эволюции [1-3]. Модель МХ формируется в процессе разработки продукта (Software Product Line Engineering, SPLE) и включает общие функциональные и нефункциональные характеристики входящих элементов, которые могут использоваться членами семейства СПП при создании разных вариантов ПС, сконфигурированных из ГОР по точкам вариантности.

Точка вариантности – это место в системе, по которому осуществляется выбор варианта ПС. Эта точка транслируется в коллекцию вариантов, присоединяемых к ядру изготавливаемой системы. В SPLE определен процесс создания ПС из ГОР (ранее они назывались компонентами повторного использования – КПИ). При производстве ПС из ГОР

создается ПС и семейство СПП. МХ в линии SPLE формировалась на двух процессах создания СПС: инженерия предметной области (ПрО) и инженерия приложений.

Инженерия предметной области (**domain engineering)** состоит в определении и реализации общих функций с обеспечением вариабельности ПО при изготовлении нового *варианта* продукта. В этой инженерии вариабельность относится к артефактам архитектуры, требованиям, компонентам и тестам.

Инженерия приложений **(application engineering)** состоит в определении специфики приложений и задании артефактов, необходимых пользователю. Внесение изменений в СПП может быть выполнено на уровне предметной области и его приложений. В первом случае – это представление изменчивости линии SPLE, и во втором – изменчивости СПП из приложений и готовых артефактов. Главными аспектами вариабельности ПС и СПП являются:

- МХ из ГОР, отмеченных вариантными точками;
- вариабельность архитектуры системы из артефактов, задаваемых точками вариантности;
- управление вариабельностью (планирование, контроль и регулирование) элементов ГОР, ориентированных на изменяемость.

Одновременно с вариабельностью К. Похла развивался метод генерации ПС и СПС К. Чарнецки из ГОР, описываемый ниже.

2.2. Вариабельность в пространстве проблем и решений

К. Чарнецки [3-5] моделирует архитектуру ПС и СПС в пространстве проблем (problem space) и решений (problem solution) аналогично Feature-oriented подхода SPLE. Он проводит анализ характеристик ГОР, которые входят в модель ПС и СПС, и внешние из них отображаются в МХ, если они реализуют требования к ПС. Между характеристиками и требованиями устанавливается $(n \times m)$связей. Для каждой ПС определяются характеристик базовых функций, которые необходимы пользователю и размещаются в областях проблем и решений. Характеристики, которые соответствуют требованиям, описываются в языке DSL (Domain Specific Language) и присоединяются к ядру СПС в точках вариантности. Характеристики отражают интерфейсные, проектные, производственные и реализационные свойства ГОР в ПС и СПС.

2.3. Вариабельность ПС и СПС методом ОКМ из функциональных и интерфейсных элементов

Метод ОКМ [6,9] обеспечивает четырехуровневое проектирование СПС (Рис.1) с помощью функциональных ($Fo = fo_1,\ldots,fo_n$) и интерфейсных элементов ($Io = io_1,\ldots,io_m$) домена или предметной области.

Совокупность функциональных элементов домена образует множество объектов $O=(O_1, O_2, \ldots O_n)$ и их интерфейсов $Io\ (i_{o1}, i_{o2}, \ldots, i_{om})$. Для них определяется множество унарных предикатов $P'=(P_1, P_2, \ldots P_r)$, с помощью которых устанавливаются свойства и принадлежность элементов указанных множеств.

Функциональный объект (f_{o1}) задает формальное описание прикладной функции ПС, которая обеспечивает решение задачи заданной предметной области/домена. Каждый объект принадлежит некоторому множеству объектов $O = (O_1 ,..., O_n)$, где $O_i = O_i\ (Name_i, Den_i, Con_i)$, а $Name_i, Den_i, Con_i$ - соответственно означают – имя, денотат и концепт объекта. Характеристики концепта $Con_i = (P_{i1}, P_{i2} ,..., P_{is})$ определяется на множестве предикатов $P=(P_1, P_2 ,\ldots, P_r.)$.

Принадлежность объекта определяется следующим соотношением:

если $P_t \in P$, $P_t \in Con_i$ и $P_t(O_i)$ = true, то функция определения принадлежности $Conex_i(O_i, P_t)$: $O_i \rightarrow O_t$, где $O_i` = O_i`(Name_i, Den_i, Con_i), Conex_i = Con_{ij} \cup \{P_t\}$.

Интерфейсный объект (i_o) задает формальное описание операций вызова методов и данных функциональных объектов в соответствии с типами данных (TD_1, TD_2,…, TD_n) и это описание является посредником взаимодействующих функциональных объектов f_o:

$i_o(f_o) = (In(f_o), Out(f_o), Inout(f_o)), In, Out, Inout \subseteq \varnothing$, где

In (f_o) – множество входных интерфейсов для передачи данных от f_o другим объектам;

Out (f_o) – множество выходных интерфейсов для передачи выходных данных объекту f_o;

Inout (f_o) – промежуточный интерфейс, преобразующий данные к требуемому виду f_o (туда и обратно).

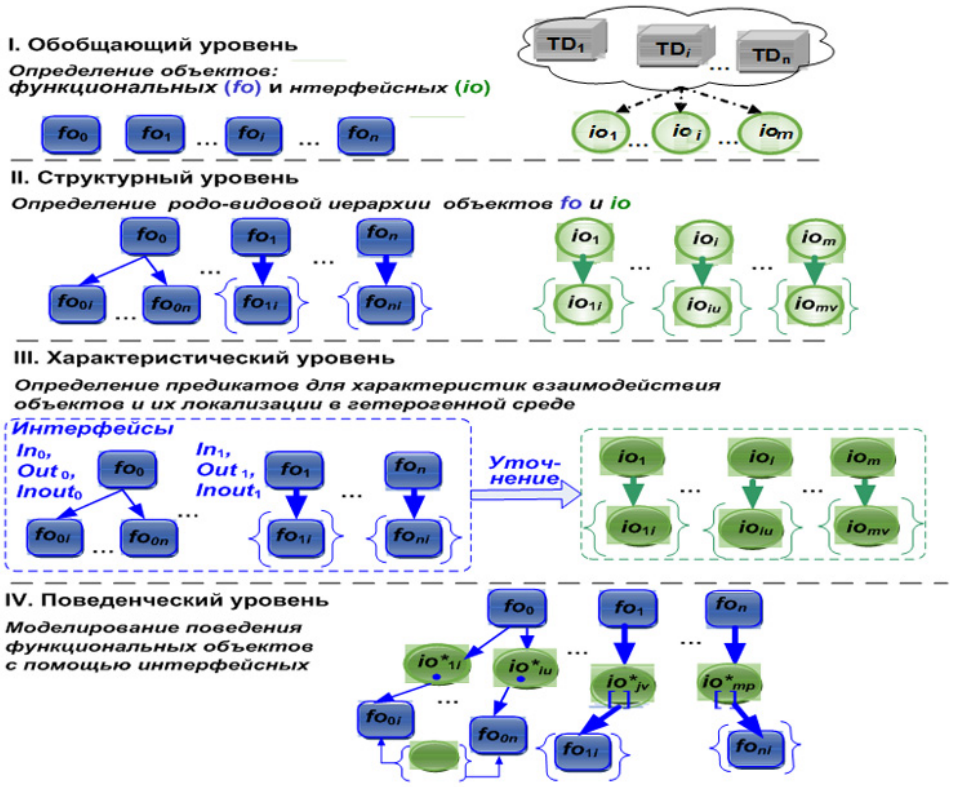

Рис. 1. Схема проектирования функциональных и интерфейсных объектов на уровнях ОКМ

Аксиома. Каждый функциональный объект СПС имеет хотя бы одну характеристику (внешнюю или внутреннюю), которая задает семантику и уникальную идентификацию во множестве объектов *O* и интерфейсов *I*.

Характеристики позволяют установить истинность совпадающих типов с помощью предикатов $P_k(O_i) = true$ на множестве *P* и *O'*. Основное условие определения принадлежности функционального объекта к множеству O – это эквивалентность внутренних свойств объекта и интерфейса с другим элементом *O*. Логико-математическое проектирование СПС на уровнях из функциональных и интерфейсных объектов сводится к построению подграфов уровней и к окончательному построению графа [9]:

$$G = \{O, I, R\}, \qquad (1)$$

где *O* – множество функциональных элементов, *I* – множество интерфейсных объектов, *R* – множество отношений (relations) между объектами (рис. 2). Вершины графа *G* задают

функциональные элементы СПС – f_{O1}, f_{O2}, f_{O3}, f_{O4}, f_{O5}, f_{O6}, f_{O7}, f_{O8} и интерфейсные элементы – i_{O5}, i_{O6}, i_{O7}, i_{O8}.

Элементы графа $f_{O1} - f_{O8}$ описываются в языке программирования, а интерфейсные объекты i_{O5}–i_{O8} в языке интерфейса IDL (Interface Definition Language).

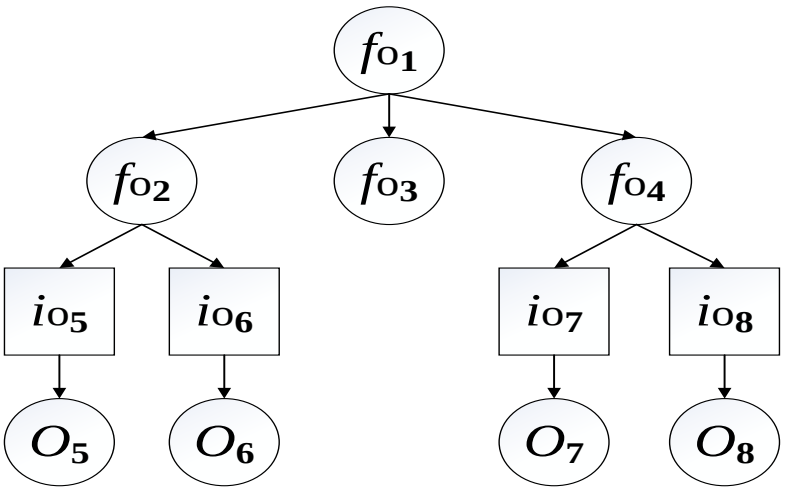

Рис. 2. Граф G на множестве функциональных и интерфейсных объектов

Параметры внешних характеристик интерфейсных объектов передаются между объектами через интерфейсы и помечаются знаками - *In* (входной), *Out* (выходной), *Inout* (входной и выходной).

Взаимодействие функциональных объектов, например, O_k, O_l обеспечивается интерфейсным объектом из множества входных интерфейсов *In*:

$fo_k = (In(fo_k), Out(fo_k))$; $fo_l = (In(fo_l), Out(fo_l))$; $fo_k \cdot fo_l = (Out(fo_l), In(fo_k))$

Теорема. Взаимодействие двух функциональных объектов является корректным, если первый объект полностью обеспечивает функции и передачу данных, необходимых другому объекту: $In(fo_k) \subseteq Out(fo_l)$.

По графу G можно собрать отдельные программы P_0 - P_5 с использованием математической операции $\cup$, соответствующей *link*:

1) $P_0 = (P_1 \cup P_2 \cup P_3 \cup P_4 \cup P_5)$;

2) $P_1 = f_{O2} \cup f_{O5}$, *link* $P_1 = In\ i_{O5}\ (f_{O2} \cup f_{O5})$;

3) $P_2 = f_{O2} \cup f_{O6}$, *link* $P_2 = In\ i_{O6}\ (f_{O2} \cup f_{O8})$;

4) P_3 :

5) $P_4 = f_{O4} \cup f_{O7}$, *link* $P_4 = In\ i_{O7}\ (f_{O4} \cup f_{O7})$;

6) $P_5 = f_{O4} \cup f_{O8}$, *link* $P_4 = In\ i_{O8}\ (f_{O4} \cup f_{O8})$.

Эти программы входят в состав ПС и могут отмечаться вариантными точками для проведения изменений в отдельные функциональные элементы модели ПС, соответствующей графовой модели G.

Далее приводится описание моделей систем и их вариабельности с использованием рассмотренных видов объектов.

2.4. Определение модели ПС, СПС, ОС и их модели вариабельности

Далее определяются модели ПС, СПС и ОС и их вариабельные модели.

2.4.1. Модели ПС и СПС, вариабельность моделей

Модель ПС [9] – это $M_{ПС} = (CL, M_f, M_s, M_i, M_d)$,

где CL – языки $L = L_1, L_2, ..., L_N$;

$M_f = (O_1, O_2, ..., O_r)$ – множество функциональных элементов;

$M_s = (Ms_{in}, Ms_{out}, Ms_{inout})$ – множество интерфейсных сервисов (s) – входного Ms_{in}, выходного Ms_{out} и серверного Ms_{inout};

M_i – множество интерфейсов в языке IDL;

M_d – множество данных и метаданных ПС.

Вариабельность ПС, СПС реализуется путем конфигурирования ГОР или взятых из библиотек готовых reuses.

Модель вариабельности ПС [9] – $MF_{var} = (SV,AV)$, где

SV – подмодель вариабельности артефактов в структуре ПС;

AV – подмодель вариабельности готовых артефактов продукта ПС.

Модель MF_{var} ориентирована на обеспечение изменений артефактов ПС, снижение затрат и уменьшение стоимости разработки системы.

Подмодель $SV = ((G_t, TR_t), Con, Dep)$,

где $G_t = (F_t, LF_t)$ – граф артефактов F_t и языка описания LF_t на уровне t; TR_t – связь артефактов на уровне t;

Con и Dep – предикаты на декартовом произведении множества артефактов, которые задают ограничения и зависимости между функциональными элементами F_t и их показателями качества.

Подмодель AV определяет ГОР, которые хранятся в репозитории. Эта подмодель отображает характеристики ГОР и системы, а также аспекты интерфейсов между ними на разных уровнях модели. Точки вариантности обрабатываются конфигуратором и заменяют некоторые ГОР другими, более корректными или новыми.

Семейство СПС. СПС – это совокупность систем (программных или операционных) с общим множеством понятий, специфических данных, функциональных и интерфейсных характеристик, которые присущи каждому члену семейства [10,11].

Модель СПС имеет вид:

$M_{СПС} = ((M_{ПС1}, M_{ПС2}, \ldots, M_{ПСk}), (M_{gf}, M_{sf}(M_{sf1}, M_{sf2}, \ldots, M_{sfk)}), M_i, M_d)))$,

где $M_{ПС1}, M_{ПС2}, \ldots, M_{ПСk}$ – множество членов семейства ПС;

M_{gf} – множество внешних функций ПрО;

$M_{sf} = \langle M_{sf1}, M_{sf2}, \ldots, M_{sfk}\rangle$ – множество внутренних функций;

M_i – множество интерфейсов объектов;

M_d – множество данных и метаданных ПрО.

Модель СПС включает совокупность функциональных элементов и их интерфейсов (или контрактов) отдельных ПС, взаимодействующих между собой. Эти интерфейсы могут группироваться в разные сочетания с учетом семантики внешних и внутренних характеристик функций, входящих в модель СПС.

Модель вариабельности СПС SVспс – это кортеж:

SVспс = ((CF; (DR,TC); (CM, FR, TS, TA); (ER,TF)); Con; Dep), где (2)

CF – характеристики функций системы для заказчика продукта, Con – ограничения и Dep зависимости между ними;

DR – характеристики функций, связанные с требованиями к ПС;

TC – связи между требованиями к ПС и их свойствами для потребителя;

FR – множество функций;

CM – множество формально описанных элементов функций FR:

TS – множество формально описанных тестов для элементов системы;

TA – интерфейс между программными элементами СПС;

ER и TF – данные БД для обработки ПС на множестве элементов CM.

Модель вариабельности СПС - FA **в артефактах** – это тройка

FA = (CAS, REP, DBF), $CAS = \cup_{t=1,\ldots,4} Aft$, (3)

REP = (RPR∪RPA∪RPE); RPE = CM∪TS, (4)

где *Aft* = (f*(aft*)) – формальное представление артефактов СПС на уровнях t=1 – *aft*;

RPR, RPA, RPE – репозитории ресурсов СПС уровня t: документов требований к ПС (t=1*)*, типичных архитектур ПС *(*t=2) и готовых ресурсов (*t=3*);

DBF – БД для обработки членов ПС и СПС;

CM, TS – множества функциональных элементов и тестов для них.

К оценке вариабельности СПС сложилось два подхода. В первом подходе интегрируется вариабельность модели, а в другом – строятся специальные модели вариабельности МХ. Дополняется ортогональная модель вариабельности [11,15], которая согласовывает состав и взаимосвязи элементов СПС, артефактов процессов сборки ПС и СПС. Эта модель входит в состав интегрированной модели вариабельности OVM, которая позволяет оценить уровень вариабельности продукта с учетом требований и артефактов.

Модель OVM имеет вид:

OVM = *(EVM, VP. VAR)*; *EVM* = (*VL, VR)*, где (5)

VP = (VpR, VpA, VpE, VpB) – множества точек вариантности в структуре СПС, которые задают характеристики ПС, включая поля таблиц БД (*VpB*), ограничения *Con* и зависимости *Dep*;

VL – модель оценки уровня вариабельности с учетом требований *vrl к* компонентам архитектуры *val*, артефактам *vel* и данных *vbl*;

VR – модель оценки уровня вариабельности СПС с учетом требований *vrr*, архитектуры *var*, артефактов *ver* и данных *vbr*.

Модель OVM определяет два вида оценок вариабельности СПС – уровень и соответствие потребностям. Оценка уровня вариабельности и степени ее соответствия потребностям выполняется с помощью дерева ценности и параметров *VL и VR,* которые учитывают трудозатраты и частоту передачи вариантов ПС заказчику. Прогнозирование вариабельности можно выполнить также с помощью Байесовской сети [23].

2.4.2. Операционная система Linux и описание ее моделей

Операционная система – это совокупность отдельных программных фрагментов функции и специальных функций ядра ОС Linux, управляющих задачами, процессами и прикладными системами. Такие функции могут формировать самостоятельные компоненты с заданным поведением и способствовать общему функционированию варианта ядра ОС [13-18].

С общей точки зрения модель ОС Linux – это множество разных операционных артефактов (функций) и интерфейсов между ними [17-19]:

$M_{oc} = (C_k, M_f, M_s, M_o, M_i)$, где (5)

C_k – множество отдельных фрагментов, артефактов ОС;

M_s – множество характеристик ядра Linux (более 10000 для версии 4.11);

M_o – множество ограничений характеристик функций на глубине дерева зависимостей (глубина 8 для 22 ограничений [20]);

M_i – множество интерфейсных характеристик (menuconfig) в ядре Linux.

Модель вариабельности ОС согласно проведенных исследований и обсуждений на конференциях VAMOS имеет следующий вид:

$MFvar = (SV_{inoc}; AV_{inoc})$, где

SV_{inoc} – подмодель вариабельности артефактов структуры ядра ОС Linux;

AV_{inoc} – подмодель вариабельности продукта ОС Linux.

Модель MF_{var} задает изменяемость отмеченных артефактов, определяет затраты и уменьшает стоимость варианта системы.

Подмодель $SV_{inoc} = ((G_{inoc}, TR_{inoc}), Con_{inoc}, Dep_{inoc})$,

где $G_{inoc} = (F_{inoc}, LF_{inoc})$ – граф артефактов ядра ОС Linux;

TR_{inoc} – набор связей между зафиксированными артефактами ОС Linux;

Con_{inoc} и Dep_{inoc} – предикаты ограничений и зависимостей между отдельными функциями на множестве артефактов ОС Linux.

AV_{inoc} определяет изменяемость структуры ОС из готовых ресурсов, которые хранятся в БД системы. Эта подмодель отображает характеристики артефактов и отношений между ними на дереве зависимостей. Точки вариантности, отмечающие изменяемые артефакты, обрабатываются конфигуратором и способствуют получению варианта ОС для конкретного применения.

Практическими экспериментами в системе ОС Linux определен набор функций и их характеристик. Для генерации из них некоторого варианта ОС требуется извлечь из ядра системы необходимые готовые артефакты или функциональные элементы с их интерфейсами. Для извлечения изменчивости (Mining variability) используется инструмент LEADT [19]. В нем содержатся средства поиска и анализа готовых ГОР в унаследованном коде ОС. Данный инструмент извлекает отдельные характеристики функций в Legacy code системы, восстанавливает архитектуру для практического использования и оформляет их в формате для представления в репозитории системы. Из выявленных функций формируется новый вариант системы, который удовлетворяет требованиям системы.

3. Управление вариабельностью систем

Вариабельность систем зависит от заданных требований к ПС и СПС, задается МХ, архитектурой (моделью) системы из ГОР и набором тестов для проверки правильности. В целом, вариабельность может быть реализована как в ПС, так и в СПС. В случае ПС вариабельность включает набор функций, базовых элементов любой природы и их документацию. В случае СПС вариабельность реализуется совокупностью продуктов семейства СПС [21, 27].

Управление вариабельностью СПС проводится по точкам вариантности, вариантным артефактам ПС, ограничениям и зависимостями через предикаты, определенные на множестве точек вариантов ПС. Для управления вариабельностью используется метод Е. Деминга [22], основанный на функциях *F1- F4* (Рис.4), которые задают действия по планированию и контролю проведения изменений в СПС.

Рис. 4. Функции действий в цикле Деминга

Каждая функция *F1- F4* выполняет следующее:

F1 – операции для подготовки артефактов СПС (**Act**);

F2 – планирование системы СПС из артефактов (**Plan**) в инженерии предметной области и инженерии приложений);

F3 – системный контроль и проверка состояния изменений СПС (**Check**);

F4 – выполнить актуализацию систем СПС (**Do**).

Управление вариабельностью СПС с учетом требований *R* (Requriment) состоит в:

1) обосновании решений для функции *F1* (*R1*);
2) согласовании способа реализации артефактов на процессах СПС (*R2*);
3) реализации проверки правильности создания СПС (*R3*);
4) отслеживании связей между характеристиками ПС и СПС (*R4*).
5) соответствии требований *R1 – R4* функциям *F1 – F4* модельной среды процесса обработки модели вариабельности СПС.

4. Верификация модели вариабельности

Объектами верификации является модель МХ и требования к разработке ПС. Для верификации используются инструменты, основанные на model checking. В них свойства объектов верификации в модели характеристик описываются средствами линейной темпоральной логики (linear temporal logic (LTL)) или логики деревьев вычислений CTL (Computational Eree Logic). Основным подходом к формальной верификации моделей и объектов является *дедуктивный анализ* (на основе логического вывода) и верификация по модели (model checking). Верификация на модели model checking применима только к моделям объектов с конечным числом состояний. Для проведения верификации модели МХ используются инструменты, описываемые ниже [24,26]:

FeatureModelPlugin или FeaturePlugin (FMP) – реализован как плагин Eclipse. **FMP** поддерживает моделирование, построение модели характеристик и конфигурирования СПС (http://gsd.uwaterloo.ca/featureModelingAndModelTemplates).

CaptainFeature – использует нотацию FODA для построения МХ и содержит конфигуратор для специализации созданных моделей

(https://sourceforge.net/projects/captainfeature/).

Requiline – это инструмент инженерии требований для эффективного управления в SPLE. Из описаний характеристик модели и требований «выводится» конфигурация продукта. Включает функцию контроля (checker) согласованности и не выполняет другие операции анализа. Проводится в таких средах функционирования как Microsoft.NET, Oracle DBMS.

Pure::Variants – инструмент поддержки моделирования характеристик, конфигурирования с помощью средств Prolog, решателя ограничений (constraintsolver) (https://www.pure-systems.com/products/pure-variants-9.html).

AHEAD ToolSuite (ATS) – это набор инструментов SPLE для разработки и поддержки характеристик и их компоновки. Можно выполнять определенные операции анализа модели с помощью SAT-решателей и сохранять ее в виде правил грамматики (http://www.cs.utexas.edu/~schwartz/ATS.html).

5. Конфигурационная сборка ГОР в СПС

К операциям конфигурационной сборки отнесены операции выполнения функций управления вариабельностью F1–F4 с использованием ГОР, которые размещаются в репозитории с учетом требований и предикатов выбора необходимых элементов ГОР [17, 18, 22, 26].

Операциями управления конфигурацией ПС в СПС являются:

- идентификация конфигурации, ее элементов и данных;
- управление процессом изменений ГОР;
- изменение модели вариабельности под новые требования;
- оценка уровня вариабельности ПС и СПС.

Для управления артефактами ПС и СПС и их вариабельностью создается, так называемая *модельная среда* конфигуратора, в которую входят:

- процесс сборки ГОР и артефактов системы;
- схемы формального описания артефактов;
- определение модели МХ с отмеченными вариантными точками;
- БД, включающая описание требований, архитектуру, набор базовых ГОР, тесты;
- Конфигуратор – инструмент для объединения артефактов в ПС и СПС.

При управлении конфигурацией собираются данные для проведения таких стандартных операций, как *отчетность и аудит конфигурации* на установление запланированной функциональности СПС. Конфигуратор собирает требуемые артефакты и ГОР в структуру СПС. Одним из шагов сборщика конфигуратора является процесс компиляции исходного кода, где файлы превращаются в промежуточный код или в код выполнения. Процесс связи представляет собой замену адресов функций внешними характеристиками библиотек и реальными адресами, используемыми в выполняемой программе. Сборщик готовых элементов – это компонент конфигуратора, который объединяет функциональные элементы и их интерфейсы МХ. Организация разработки вариантов ПС и СПС базируется на следующих формальных положениях.

Аксиома 1. Технология задает циклическую последовательность процессов разработки ПС и актуализации СПС.

Аксиома 2. Каждую терминальную характеристику модели ПрО реализует один и только один ГОР, названный базовым.

В [21,26] представлена схема конфигуратора, принцип работы которого приведен на Рис.5. В нем стрелка 1 задает задание на доработку или создание нового ГОР и его отправку в репозиторий. Каждый ГОР состоит из двух файлов: ***.cs**, которые задают бизнес логику процесса и ***.xoml представляют собой атомарные абстрактные объекты домена и алгоритм выполнения логики. Конфигуратор устанавливает связь с репозиторием (стрелка 3-5) и получает список всех доступных ГОР. Они отображаются в правом верхнем углу схемы конфигуратора.

После построения новой модели, она размещается на сервере (стрелка 4) или в репозитории (стрелка 2, 4) ГОР для последующего использования. Каждый элемент конфигуратора является отдельным самостоятельным ГОР.

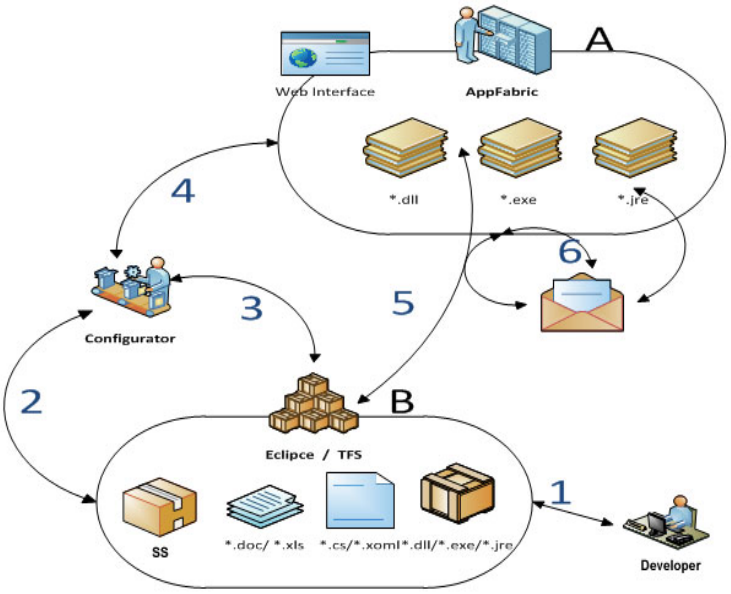

Рис. 5. Общая модель работы конфигуратора в среде .Net

6. Тестирование семейства СПС

В состав СПС входят ГОР и *рабочие продукты* для тестирования (планы, набор тестов, тестовые данные и др.). Тест-код создается для тестирования отдельных элементов ПС и образует набор тестов СПС. Метод тестирования СПС базируется на методе Дж. МакГрегора (McGregor) «от требований» (requirements-based testing) [27]. В нем задаются действия по управлению тестированием «от требований» с помощью тестов, проверяющих функциональные и интерфейсные объекты. Контроль степени тестирования объектов и интерфейсов обеспечивает оценку критерия качества СПС для вариантов ПС. В качестве инструмента тестирования используется *TestManager* фреймворка VisualStudio, содержащий средства проверки правильности тестирования разных ГОР и планирования процесса тестирования при выполнении тестовых сценариев, приведенных в таблице 1

Таблица 1. Основная схема тестирования семейства ПС

Виды тестирования	Объекты тестирования
Тестирование программной архитектуры системы	Все ПС семейства, отдельные ПС и функциональные элементы.
Набор тестов СПС, адаптированных к конкретной ПС	Все ПС семейства, отдельные ПС семейства
Тестирование требований (ScenTED)	Отдельные ПС семейства
Самотестирование	Отдельные объекты и интерфейсы
Применение метаданных	Отдельные объекты и интерфейсы
Оценка тестопригодности компонентов	Отдельные объекты и интерфейсы
Ориентация на сервисное обеспечение и обеспечение надежности	Отдельные объекты и интерфейсы
Автоматическая генерация тестов по спецификациям в булевой форме	Все ПС семейства, отдельные ПС семейства
FCTA (Fault Contribution Tree Analysis)	ПС семейства

По данному методу проводится:

1. Тестирование артефактов, приложений, отдельных ПС и ГОР.

2. Тестирование отдельных характеристик объектов СПС с помощью тестов (автономных и общих).

3. Проверка степени тестирования функциональных и интерфейсных объектов СПС в среде Microsoft VisualStudio на инструментах TeamFoundationBuild и Microsoft Test Manager [27].

В завершении тестирования определяется количественная метрика *КТ* степени тестирования объектов ПС:

КТ = 1, если операции тестирования объекта независимы одна от другой;

КТ = 0, если операции зависят от пути выполнения и взаимосвязей.

КТ принадлежит отрезку [0; 1] и вычисляется по следующей формуле:

$$КТ = \frac{1}{n}\sum_{i=1}^{n} КТi$$

Конечное значение *КТ* СПС задает степень проверки:

КТ = 1, если все объекты проконтролированы;;

КТ = 0, если не все объекты проконтролированы;

0<*КТ*<1 означает, что объекты СПС частино проконтролированы.

При тестировании интерфейсов взаимодействующих объектов проводится оценка метрики контроля интерфейса С*I* по формуле:

$$CI = 1/n\sum_{i=1}^{n} CI_i$$

где СI_i – степень корректности преобразования типов данных в *i*-ом интерфейсном объекте. Если *CI* = 1, интерфейс полностью проконтролирован, *CI* = 0 интерфейс не полностью проконтролирован; $CI \in (0;1)$ – интерфейс частино проконтролирован.

Количественное значение метрики *CI* означает*:*

1 – контроль CI_i интерфейса завершен;

0 – в противном случае - нет.

Этим завершается оценка тестирования функциональных объектов и их интерфейсов с помощью тестов в СПС.

7. Заключение

Рассмотрены базовые концепции моделирования вариабельных ПС и СПС. Приведены оригинальные решения К. Похла по модели вариабельности МХ в SPLE, К. Чарнецки в пространстве проблем и решений в плане генерации ГОР и в методе ОКМ для проектирования графовой модели СПС из функциональных и интерфейсных элементов. Разработан подход к теории определения модели СПС из готовых ресурсов (ГОР) и ОС Linux с использованием модели вариабельности МХ. Предложена модельная среда и управление вариабельностью СПС с учетом заданных требований. Дано расширение модели вариабельности в интегрированной модели для оценки качества вариантов систем с учетом требований к ПС, планирования средств и сроков выполнения оценочной модели вариабельности. Определены методы верификации, тестирования и выполнения конфигураций ПС и СПС.

8. Литература

1. Pohl K., Böckle G., van der Linden F. J. Software Product Line Engineering: Foundations, Principles and Techniques. Springer-Verlag, 2005. DOI: 10.1007/3-540-28901-1.
2. Bachmann F., Clements P. Variability in software product lines. CMU/SEI Technical Report CMU/SEI-2005-TR-012, 2005.
3. Lotufo R., She S., Berger T., Czarnecki K., Wąsowski A. Evolution of the ux kernel variability model. Proc. of SPLC'10, LNCS 6287:136-150, Springer, 2010. DOI: 10.1007/978-3-642-15579-6_10.

4. Чарнецки К., Азенакер У. Порождающее программирование. Методы, инструменты, применение. – Изд. Дом №Питер» - М.: СПб.-Харьков-Минск.- 2005.-730 с.
5. Berger T., She S., Lotufo R., Wąsowski A., Czarnecki K. A study of variability models and languages in the systems software domain. IEEE Transactions on Software Engineering, 39(12):1611-1640, 2013. DOI: 10.1109/TSE.2013.34.
6. Лаврищева Е.М., Грищенко В.Н. Методы и средства объектно-компонентного программирования.- Кибернетика и системный анализ. - 2003.-№1, с. 39-55.
7. Kang K., Cohen S., Hess J., Novak W., Peterson S. Feature-oriented domain analysis (FODA) feasibility study. CMU/SEI Technical Report CMU/SEI-90-TR-21, 1990.
8. Zippel R. et al. Kconfig language. https://www.kernel.org/doc/Documentation/kbuild/kconfig-language.txt/.
9. Лаврищева Е.М. Теория объектно-компонентного моделирования программных систем // http://www.ispras.ru/preprints/docs/prep_29_2016.pdf.
10. Лаврищева Е. М., Коваль Г.И., Слабоспицькая О.О., Колесник А.Л. Особенности процессов управления созданием семейств программных систем. Проблемы программирования, (3): **с**.40-49, 2009.
11. Лаврищева Е.М., Слабоспицькая О.А., Коваль Г.И., Колесник А.А. Теоретические аспекты управления вариабельностью в семействах ПС. Весник КНУ, серия физ.–мат. наук, 2011, № 1.- с. 151-158.
12. Лаврищева Е.М., Грищенко В.Н. Сборочное программирование. Основы индустрии программных продуктов. – К.: Наук. Думка, 2009 – 371 с.
13. Кулямин В.В. Лаврищева Е.М., Мутилин В.С., Петренко А.К. Верификация и анализ вариабельных операционных систем. Труды ИСП. РАН. Том 28. Вып.3.с.189-209.
14. Лаврищева Е.М. Петренко А.К. Моделирование систем и их семейств. Труды ИСП РАН, 2016, том 28. вып. 6, с. 180 -190.
15. Лаврищева Е.М., Слабоспицкая О.А. Подход к построению объектно-компонентной модели семейства программных продуктов // Проблемы программирования, 2013.–№3 – с.14–26 (укр.).
16. Ekaterina Lavrischeva, Andrey Stenyashin, Andriy Kolesnyk, Object-Component Development of Application and Systems. Theory and Practice, Journal of Software Engineering and Applications, 2014, 7, Published Online August 2014 in SciRes http://www.scirp.org/journal/jsea.
17. Лаврищева Е.М., Колесник А.Л., Стеняшин А.Ю. Объектно-компонентное проектированием программных систем. Теоретические и прикладные вопросы – Весник КНУ, серия физ.-мат. наук. –2013. –№4. – С. 150–164 (укр.).
18. Kästner C., Dreiling A., Ostermann K., Variability Mining with LEADT, In Proc. Int'l Conf. Generative Programming and Component Engineering (GPCE), pp. 157–166, 2009.
19. Melo J., Flesborg E., Brabrand C., Wąsowski A. A quantitative analysis of variability warnings in Linux. Proceedings of the 10-th International Workshop on Variability Modelling of Software-intensive Systems, pp. 3-8. ACM, 2016. DOI: 10.1145/2866614.2866615.
20. Колесник А.Л. Поддержка процесса управления вариабельностью в семействах программных систем // Проблемы программирования. – 2012, № 2-3.- с.182-191.
21. Deming E. New economics for manufactures, governments and education, 1993.
22. Коваль Г.І. Байесовские сети – способ оценивания и прогнозирования качества програмного забезпечення // Проблеми програмування. – 2005. - №2. – С.15-23.
23. Lauenroth K., Toehning S., Pohl K. Model checking of domain artifacts in product line engineering. Proc. Intl. Conf. on Automated Software Engineering (ASE), pp. 269-280. IEEE, 2009.DOI: 10.1109/ASE.2009.16.

24. Лаврищева Е.М. Программная инженерия. Парадигмы, Технологии, CASE-средства программирования.2 изд. – М: Юрайт, 2016. – 280 с.
25. Sayyad A. S., Ingram J., Menzies T., Ammar H. Scalable product line configuration: a straw to break the camel's back. Proc. of IEEE/ACM 28-th International Conference on Automated Software Engineering (ASE 2013), pp. 465-474. IEEE, 2013.DOI: 10.1109/ASE.2013.6693104.
26. Колесник А.Л. Модели и методы разработки семейств вариантных программных систем.- Автореф.- КНУ.- 2013. 22 с.
27. MacGregor S.D., Sykes D.A. Practical Guide to testing Object-oriented Software.-2001, Addison-Wesley Professional.