Моделирование семейств программных систем[1]

[1,2] *Е.М. Лаврищева <lavr@ispras.ru>*
[1,3,4] *А.К.Петренко <petrenko@ispras.ru>*
[1] *Институт системного программирования РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25
[2] *Московский физико-технический институт (гос. университет),*
141701, Московская область, г. Долгопрудный, Институтский переулок, д.9
[3] *Московский государственный университет имени М.В. Ломоносова,*
119991, Россия, Москва, Ленинские горы, д. 1
[4] *Национальный исследовательский университет Высшая школа экономики,*
101000, Россия, г. Москва, ул. Мясницкая, д. 20

Аннотация. Большие программные системы всегда создаются достаточно долго, в несколько этапов, это приводит к необходимости появления версий (релизов) систем. Кроме того, у большой системы всегда есть несколько, а иногда много, конфигураций установки, что обуславливается либо разным набором программно-аппаратного окружения, либо требованиями пользователя системы. Тем самым результатом разработки можно рассматривать не отдельную систему, а цепочку и семейство программных систем или программных продуктов (Product Lines/Product Families). Появление этого понятия можно рассматривать как развитие методов повышения доли повторно-используемого программного обеспечения (re-use). Однако, в отличие от ранних работ по повторно-используемому ПО исследования по семействам программ рассматривают весь спектр работ и задач создания ПО, то есть не только собственно проектирование и программирование, но и документирование, верификацию, поддержку эксплуатации, в частности, инсталляцию и так далее. Одной из работ, выполняемых в ходе создания семейства программ, является моделирование. Статья рассматривает современные подходы к моделированию семейств программных систем, а также, подробнее, одно из направлений этого исследования — моделирование семейств операционных систем (описывая задачи исследования, поддержанного грантом РФФИ).

Ключевые слова: проверка моделей; уточнение моделей; вариабельность; верификация; тестирование; семейство программных систем; управление моделями; конфигурационная сборка, UML, MDA, MDD, MDE, PIM, PSM.

[1] Работа поддержана грантом РФФИ 16-01-00352.

K.M. Lavrischeva, A.K. Petrenko. Software Product Lines Modeling. *Trudy ISP RAN/Proc. ISP RAS*, vol. 28, issue 6, 2016, pp. 49-64

DOI: 10.15514/ISPRAS-2016-28(6)-4

Для цитирования: Лаврищева Е.М., Петренко А.К. Моделирование семейств программных систем. Труды ИСП РАН, том 28, вып. 6, 2016, стр. 49-64. DOI: 10.15514/ISPRAS-2016-28(6)-4

1. Введение

Теме моделирования программных систем уделяется достаточно много внимания, начиная, по крайней мере, с 70-х годов XX века (в качестве примера можно указать на SDL). Моделирование аппаратных и программно-аппаратных устройств зародилось и активно развивалось практически сразу с возникновением ЭВМ. Новое направление работ по моделированию программных систем возникло в начале 90-х годов, в основном, оно связано с появлением UML (Unified Modeling Language, 1994) [7].

Моделирование в этом языке состоит в построении графовой объектной модели, в вершинах которой находятся отдельные объекты системы, а на дугах отношения (связи) между ними. Элементы модели трансформируются в программный код, который тестируется и документируется. И то, и другое не было принципиально новым. К тому времени было несколько графических или текстово-графических языков моделирования, некоторые языки (например, тот же SDL) позволяли генерировать программный код готовый к выполнению или требующий лишь небольшой доделки. Принципиально новым в UML было рассмотрение широкого спектра задач жизненного цикла программного продукта и, соответственно, многих аспектов моделирования, например, моделирования архитектуры в целях обеспечения гибкости, удобства адаптации системы к различных платформам и к ожидаемым требованиям по расширению функциональности, моделирования сценариев работы пользователя, моделирования поведения собственно системы, ее отдельных компонентов, моделирования протоколов взаимодействия подсистем и так далее [6].

Собственно UML является лишь нотацией предназначенной для построения моделей. На основе этой нотации (или альтернатив) строятся методологии или даже инструментальные комплексы для построения или анализа программных систем с теми или иными целями. Так известным направлением в конце 90-годов стала методология MDA (Model Driven Architecture). В рамках этой методологии основное внимание уделялось выделению консервативного (re-usable) архитектурного каркаса некоторого класса систем (примерами могут быть: интернет-магазин, система заказа билетов и т.д.). На основе такого каркаса строится платформо-независимая модель (PIM), которая потом транслируется в платформо-зависимую — PSM.

Методология MDA не получила тотального распространения, на что рассчитывали компании, которые их активно продвигали. В качестве основных причин можно указать, с одной стороны, на громоздкость методологии,

сложность композиции моделей-каркасов и, с другой стороны, в игнорировании специфики сложившихся технологий разработки программ.

Однако нужно отметить, что, хотя попытки тотального внедрения методов разработки и верификации программ на основе моделей можно признать неудачными, работы по совершенствованию языков и инструментов моделирования, а также новые средства использования моделей развиваются активно. В первую очередь нужно отметить направления MDD (Model Driven Development) и MDE (Model Driven Engineering), которые широко используются в создании встроенных, критических по безопасности систем.

Специфика задач моделирования программных семейств состоит в том, что цель моделирования состоит не только в выделении консервативного ядра системы, но и в четком описании вариантных частей и механизмов управления вариантностью. Цель анализа модели программного семейства состоит в анализе корректности (или других характеристик системы, например, производительности) при выборе того или иного набора характеристик (features) или задании конфигурации компиляции/сборки/инсталляции системы.

Тема моделирования программных семейств активно обсуждается на научных конференциях и в научных журналах. Основным форумом можно назвать конференцию MODELS (1999-2016), а также эта тема постоянно присутствует на конференциях VAMOS (2004 –2015); ICTERI (2006 –2013); SAM (System Analysis and Modeling), ICSR (International Conference on Software Reuse) и др.

2. Моделирование изменяемых систем

Как уже отмечалось, объект и цели моделирования программных продуктов (ПП) отличаются от сущностей, с которыми приходится работать при моделировании собственно программных систем. Далее дается краткое описание подходов к моделированию программных систем, базирующееся на работах SEI (Software Engineering Institute) и работах одного из авторов статьи [1,3,4,5,16].

2.1 Вариабельность продуктов и систем

В SEI (2005) был предложен метод SPLE (Software Product Line Engineering) [1,3] для изготовления семейств программных систем (СПС), основанный на модели вариабельности характеристик (Feature Model - MF) [8,9,11] и модели конфигурационной сборки готовых ресурсов (которые в англоязычной литературе называются как artifacts, components, reuses, assets, services и др.).

К. Похл [2] определил понятие вариабельности в SPLE, как модели MF для элементов системы, помечаемых вариантными точками.

Вариабельность — это способность продукта (системы) к расширению, изменению, приспособлению или конфигурированию с целью использования в

определенном контексте и обеспечения последующей его эволюции. Модель MF формируется в процессе разработки ПП аналитиками и разработчиками и включает общие функциональные и нефункциональные характеристики элементов системы [15]. К главным аспектам обеспечения вариабельности продукта системы относятся:

- моделирование вариабельности на уровне артефактов и целых подсистем с вариантными характеристиками [11];
- управление (планирование, контроль и регуляция) конфигурацией системы по модели архитектуры системы и модели вариабельности [10,18].

Множество значений одной характеристики (*вариантной характеристики*), образует коллекцию. Каждый член коллекции соответствует требованию к разработанным артефактам в ПП и присоединяется к ядру ПП в точках вариантности.

Точка вариантности — это место в системе, в котором осуществляется выбор варианта reuse-компонента. Вариантная характеристика косвенно определяет количество точек вариантности системы, присоединяемых к ядру системы.

Основу концепции изменяемости/вариабельности ПП составляют внешние характеристики и свойства функций системы, которые могут отличаться в разных вариантах ПП. По модели архитектуры системы и MF создается конфигурационный файл системы для инсталляции, запуска и выполнения системы. Совокупность систем и подсистем с общим множеством характеристик и требований к ним образуют ПП. В принципе, может быть поставлена задача получения и перечисления всех членов семейства или определения наличия определенных свойств (например, совместимости, безопасности и т.д.) для всех членов семейства [12,13].

2.2 Модель вариабельности СПС

В SPLE модель вариабельности имеет следующий вид:

$$MF_{var} = (SV; AV),$$

где SV — подмодель вариабельности артефактов системы,
AV — подмодель архитектуры продуктов системы.

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

где G_t — граф артефактов типа t (требования, компоненты, тесты и др.);
TR_t — связи артефактов типа t;
Con, *Dep* — предикаты на декартовом произведении множеств артефактов, которые определяют ограничения и зависимости между функциями и показателями качества СПС.

Подмодель $AV = (G, TR, G_t)$,

где G — граф архитектуры системы СПС из артефактов, переиспользуемых элементов сборки - reuses, а также из набора вариантных характеристик;

TR — связь элементов архитектуры и артефактов G_t.

Модель SV конкретизируется в специальную линию разработки ПП и конфигурируется из артефактов и элементов архитектуры в код системы. Точки вариантности позволяют управлять трансформацией элементов графа и заменой одних артефактов другими, новыми функциональными или более корректными элементами архитектуры.

2.3 Управление вариабельностью

Объектами управления вариабельностью в СПС и ПП являются:

- точка вариантности или вариантный артефакт, т.е., его формальное представление и возможность его реализации для разных СПС;
- вариант — элементарный артефакт СПС одного типа с соответствующим ему вариантным артефактом, который представляет собой описание заданного СПС;
- ограничение — предикат, определенный на декартовом произведении декартовых квадратов множества точек вариантности и вариантов;
- зависимость — предикат, который определяет допустимые взаимосвязи между точкой вариантности и множеством вариантов для СПС.

В работах Е. Дьоминга предложен метод управления организацией разработки СПС, в котором выделено четыре функции управления вариабельностью [18]:

1) *F1* — функция планирования вариабельности в артефактах СПС (на уровнях инженерии домена и приложений);
2) *F2* — функция реализации вариабельности в архитектуре СПС;
3) *F3* — функция системного мониторинга вариабельности СПС;
4) *F4* — функция актуализации вариабельной СПС.

В основе управления вариабельностью СПС и ПП лежат требования *R1-R5* к системе:

R1) обоснованность — наличие объективных оснований принятия решений (assets) для *F1-F4*;

R2) согласованность — одинаковость способа выработки и реализации решений на всех уровнях абстракции и на всех этапах процесса разработки СПС;

R3) масштабируемость — независимость способа выработки и реализации этих решений от объема функциональных возможностей СПС;

R4) трассируемость — возможность отслеживания связей между характеристиками вариабельности на всех процессах разработки ПП;

R5) визуализация свойств вариабельности и связей между характеристиками.

Приведенные требования *R1–R5* для функций *F1–F4* обеспечивают модельную среду их реализации. Необходимым элементом этой среды может быть интегрированная модель вариабельности (*VM*) семейства СПС, которая объединяет модели подсистем семейства и позволяет дать оценку уровню вариабельности системы и степени ее соответствия требованиям.

3. Направление работ по моделированию сложных систем ИСП РАН

В 2016 г. РФФИ поддержал проект «Теория и методы разработки вариабельных программных и операционных систем» (16-01-00352) [32].

Цель данного проекта состоит в разработке

- теории и методов моделирования операционных систем и веб-систем, которые рассматриваются как результат сборки из готовых ресурсов (reuses, модулей, объектов, компонентов, аспектов, сервисов и др.) и их вариантов;
- методов извлечения из готовых СПС (например, ОС Linux или другие унаследованных систем) элементов архитектуры и построения новой модели операционной системы и модели вариабельности для управления вариантами ПП и СПС.

Эти два класса систем моделирования данного проекта РФФИ являются разнородными в плане видов ресурсов, используемых в них, и близкими по показателям сложности.

3.1 Направления исследования по проекту

Исследование систем обоих типов и разработка теории и методов построения этих типов систем проводится в проекте по двум направлениям:

1) Анализ формализмов, используемых для описания моделей вариабельности и конфигурации систем различного типа, включающих изменяемые, вариабельные элементы и характеристики системы, набор правил их конфигурации в системы и верификации моделей, артефактов и систем. Результатом анализа является конкретизация модели вариабельности и формализмов описания конфигурационной сборки современных операционных систем и прикладных Веб-систем из готовых ресурсов [16].
2) Исследование методов моделирования, конфигурации и верификации сформированных конфигураций операционных систем и методов извлечения моделей вариабельности с артефактов и приложений, а также извлечения готовых ресурсов из конфигурации операционной системы для построения вариантов таких систем под конкретные

применения и Веб-систем в общем проведено в [16,27]. В качестве операционной системы используется ОС Linux, а в качестве прикладной системы бизнес-система с Веб-сервисов.

Исследование по данному проекту состоит в анализе базовых направлений подходов к моделированию ПП и их вариантов (Grid, WebSphere IBM, MS Product Line, Onion Architecture Web-system и др.), а также в усовершенствовании и определении новых моделей моделирования для заданного класса систем. Авторами предложена новая модель системы, требующая доработки функций системы, внесения изменений и проведения верификации и анализа вариабельности операционных систем по новым точкам вариантности.

3.2 Метод конфигурации

Предлагаемый в рамках данного проекта метод конфигурации обеспечивает:

- выполнение операций задания и пополнения требований, проведение их трассирования на соответствие выполняемым функциям и порождение новых вариантов архитектуры СПС из готовых, заданных точками вариантности и с учетом заданных ограничений;
- подготовку эффективных решений (assets) по разработке новых вариантов артефактов и готовых ресурсов путем анализа опыта, накопленного в базе знаний проекта системы.

Сущность управления конфигурацией состоит в**:**

- конфигурационной сборке ресурсов по моделям систем (контроль, верификация ресурсов, тестов);
- систематическом отслеживании внесенных изменений в отдельные составные элементы конфигурации системы, проведение аудита изменений и автоматизированного контроля изменений;
- поддержке целостности конфигурации и ее аудита;
- ревизии конфигурации методом проверки программных или аппаратных элементов в соответствии с версиями конфигурации и требованиями к системе;
- трассировке изменений в схему конфигурации на этапах сопровождения и эксплуатации систем;
- доказательстве наличия изоморфного отображения объектных и компонентных моделей систем в программные структуры методом трансформации моделей (PIM, PSM) и функций объектов, их данных к платформе выполнения операционной или прикладной системы.

3.3 Анализ, верификация и тестирование вариабельных систем

Методы анализа моделей вариабельности делятся на четыре группы [9]:

- На основе пропозициональной логики [19]. В ней ограничения модели характеристик транслируются в логические формулы, которые затем обрабатываются с помощью решателей и инструментов автоматического доказательства различных видов (SADT, BDD и др.) или инструментов с использованием формальных языков типа VDM, Alloy, Event-B, Z и др.
- На основе онтологий. Этот подход основан на трансляции модели вариабельности в модель онтологии. Например, в [20] производится трансляция в OWL DL (Ontology Web Language Description Logic), обладающее достаточной выразительной способностью. После трансляции используются автоматизированные инструменты анализа онтологий, такие как RACER.
- На основе программирования в ограничениях. В этом подходе ограничения модели вариабельности транслируются в описание задачи CSP (Constraint Satisfaction Problem), которая затем проверяется с помощью существующих инструментов программирования с ограничениями (constraint programming) [21,22].
- На основе проверки моделей. Этот анализ сводится к задачам проверки моделей (model checking), с помощью имеющихся инструментов [23,24] и др. Для ряда языков описания конфигураций, например, Kconfig [14], поиск несогласованностей и недопустимости конфигураций выполняются автоматически только для моделей с ограничениями, не включающими характеристики, не имеющие общего родителя. Пользователи должны вручную отслеживать соблюдение такого рода ограничений. Имеется ряд методов на основе анализа моделей программ (software model checking), использующего автоматическую генерацию абстрактных моделей из кода реальных программ, например, на языке Си с последующей верификацией этих абстрактных моделей и, если необходимо, автоматическую детализацию получаемых моделей. Обзор этих методов можно найти в [30].

Методы верификации вариантных СПС включают методы известных специалистов: Hamlet, McGregor, Kishi и др. [26,27,28], использующие тестирование на основе диаграмм характеристик, проверки модели вариабельности и конфигурационной модели создаваемых систем путем проверки корректности протоколов взаимодействия ресурсов, готовые тесты для отдельных ресурсов и функций. Методы верификации СПС на основе программных моделей развиваются несколькими группами исследователей. В

первую очередь нужно отметить работы группы С. Апеля [31] из университета Пассау.

3.4 Методы извлечения объектов из операционных и наследованных (legacy) систем

Современные операционные системы (ОС), такие как Linux, имеют большой набор разнообразных конфигурационных опций, позволяющих получать варианты системы для различных видов оборудования и в различной комплектации ресурсов. Например, в ядре Linux имеется несколько тысяч конфигурационных опций. В версии 4.2 это число составило 7872 опции (архитектура x86_64). В другой операционной системе FreeBSD, для версии v.8.0.0 определено 1 203 опции (все архитектуры).

Сложность управления многообразием конфигураций ОС довольно высока, что часто приводит к чисто конфигурационным ошибкам[2] [25,29], связанным, например, с включением несовместимых опций или исключением необходимой для заданной опции соответствующего окружения. Для более эффективного решения задач управления конфигурациями существуют специализированные формализмы и языки, позволяющие описывать многообразие конфигураций, а также правила корректного построения конфигураций. Вместе с тем, имеющиеся механизмы управления конфигурациями не дают непосредственной возможности представить и анализировать модель вариабельности в целом. Во многом проблема управления конфигурациями типична не только для операционных систем, но для многих наследованных систем.

В рамках исследований, поддержанных грантом РФФИ, планируется в первую очередь решить три следующие задачи:

- собрать информацию о точках вариабельности ОС Linux и провести их классификацию;
- провести исследование характерных ошибок, которые можно отнести к типу «конфигурационных ошибок»;
- разработать методы и соответствующие инструменты для обнаружения конфигурационных ошибок.

Кроме того, ставится общая задача построения общей модели вариабельности ОС Linux, объединяющей модель архитектуры системы и модель ее характеристик.

[2] «конфигурационными ошибками» называют программные конструкции, которые являются корректными в одних конфигурациях и некорректными в других.

4. Заключение

В статье рассмотрены подходы к моделированию семейств программных систем. Рассмотрены основные задачи фундаментального проекта «Теория и методы разработки вариабельных программных и операционных систем» (16-01-00352). Приведен анализ и дана характеристика основных подходов к моделированию семейств программных систем. Сформулированы задачи моделирования, верификации и анализа современных СПС с учетом их вариабельности и конфигурирования из готовых ресурсов. Поставлена задача извлечения модели вариабельности ядра операционной системы Linux и рассмотрены варианты постановки задачи верификации СПС Linux.

Авторы выражают глубокую признательность В.В. Кулямину за обсуждения и помощь в подготовке данной статьи.

Список литературы

[1]. Clements P., Northrop L. Software Product Lines: Practices and Patterns. SEI Series in Software Engineering, Addison-Wesley, 2001. ISBN-13: 978-0201703320.

[2]. 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.

[3]. Bachmann F., Clements P. Variability in software product lines. CMU/SEI Technical Report CMU/SEI-2005-TR-012, 2005.

[4]. Лаврищева Е. М., Коваль Г.И., Слабоспицькая О.О., Колесник А.Л. Особенности процессов управления при создании семейств программных систем. Проблемы программирования, (3):40-49, 2009.

[5]. Лаврищева Е.М., Слабоспицькая О.А., Коваль Г.И., Колесник А.А. Теоретические аспекты управления вариабельностью в семействах программных систем. Весник КНУ, серия физ.–мат. наук, (1):151-158, 2011.

[6]. Jacobson I., Griss M., Jonsson P. Software Reuse, Architecture, Process and Organization for Business Success. Addison-Wesley, 1997. ISBN-13: 978-0201924763.

[7]. Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя: Пер. с англ. М.: ДМК, 2000, 432 с.

[8]. 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.

[9]. Benavides D., Segura S., Ruiz-Cortés A. Automated analysis of feature models 20 years later: a literature review. Information Systems, 35(6):615–636, 2010. DOI: 10.1016/j.is.2010.01.001.

[10]. Chen L., Babar M.A. A systematic review of evaluation of variability management approaches in software product lines. Information and Software Technology, 53(4):344–362, 2011. DOI: 10.1016/j.infsof.2010.12.006.

[11]. 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.

[12]. Thüm T., Apel S., Kästner C., Kuhlemann M., Schaefer I., Saake G. A classification and survey of analysis strategies for software product lines. ACM Computing Surveys, 47(1):article 6, 2014. DOI: 10.1145/2580950.

[13]. Liebig J., von Rhein A., Kästner C., Apel S., Dörre J., Lengauer C. Scalable analysis of variable software. Proceedings of the 2013 9-th Joint Meeting on Foundations of Software Engineering, pp. 81-91. ACM, 2013. DOI: 10.1145/2491411.2491437.

[14]. She S., Berger T. Formal semantics of the Kconfig language. Technical note, University of Waterloo, 2010. https://www.kernel.org/doc/Documentation/kbuild/kconfig-language.txt.

[15]. Эммерих В. Конструирование распределенных объектов. Методы и средства программирования интероперабельных объектов в архитектурах OMG/CORBA, Microsoft COM и JAVA RMI. М.: Мир, 2002. 510с.

[16]. Лаврищева Е.М. Теория объектно-компонентного моделирования изменяемых программных систем. ИСП РАН, 2015. http://www.ispras.ru/preprints/docs/prep_29_2015.pdf.

[17]. Berger T., Rublack R., Nair D., Atlee J.M., Becker M., Czarnecki K., Wąsowski A. A survey of variability modeling in industrial practice. Proc. of the 7-th Intl. Workshop on Variability Modelling of Software-intensive Systems (VaMoS'2013), article No. 7, ACM 2013. DOI: 10.1145/2430502.2430513.

[18]. Лаврищева Е.М. Программная инженерия. Парадигмы, технологии, CASE-средства. Изд. Юрайт, М.: 2015, 280 с.

[19]. Batory D. Feature models, grammars, and propositional formulas. Proc. of the 9-th Intl. Conf. on Software Product Lines (SPLC'05), LNCS 3714, pp. 7-20, 2005. DOI: 10.1007/11554844_3.

[20]. Wang H., Li Y., Sun J., Zhang H., Pan J. A semantic web approach to feature modeling and verification. Proc. of Workshop on Semantic Web Enabled Software Engineering (SWESE'05), p. 44, 2005.

[21]. Benavides D., Segura S., Trinidad P., Ruiz-Cortés A. Using Java CSP solvers in the automated analyses of feature models. Generative and Transformational Techniques in Software Engineering, LNCS 4143:399-408. Springer, 2006. DOI: 10.1007/11877028_16.

[22]. White J., Doughtery B., Schmidt D., Benavides D. Automated reasoning for multi-step software product-line configuration problems. Proc. of the 13-th Sofware Product Line Conference, pp. 11-20, 2009.

[23]. Zhang W., Mei H., Zhao H. Feature-driven requirement dependency analysis and high-level software design. Requirements Engineering, 11(3):205-220, 2006. DOI: 10.1007/s00766-006-0033-x.

[24]. Dietrich C., Tartler R., Schröder-Preikshat W., Lohmann D. Understanding Linux feature distribution. Proceedings of the 2012 Workshop on Modularity in Systems Software, pp. 15-20. ACM, 2012. DOI: 10.1145/2162024.2162030.

[25]. 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.

[26]. 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.

[27]. Nguyen T., Colman A., Han J. A web services variability description language (WSVL) for business users oriented service customization. Proc. Int. Workshop on User–Focused Service Engineering, Consumption and Aggregation (USECA 2011).

[28]. Meinicke J., Thüm T., Schröter R., Benduhn F., Saake G. An overview on analysis tools for software product lines. Proc. of the 18-th Intl. Software Product Line Conf.: Companion Vol. for Workshops, Demonstrations and Tools, vol. 2, pp. 94-101. ACM, 2014. DOI: 10.1145/2647908.2655972.

[29]. Hemakumar A. Finding Contradictions in Feature Models. Proc. of 12-th Intl. Conf. on Software Product Lines (SPLC'2008), v. 2, pp. 183-190, 2008.

[30]. Бейер Д., Петренко А.К. Верификация драйверов операционной системы Linux. Труды ИСП РАН, Том 23, 2012 г., стр. 405-412. DOI: 10.15514/ISPRAS-2012-23-23.

[31]. Apel S., Beyer D., et al. Strategies for product-line verification: case studies and experiments. Proceeding ICSE '13 Proceedings of the 2013 International Conference on Software Engineering, pp. 482-491.

[32]. Кулямин В.В., Лаврищева Е.М., Мутилин В.С., Петренко А.К. Верификация и анализ вариабельных операционных систем. Труды ИСП РАН, 28(3):189-208, 2016. DOI: 10.15514/ISPRAS-2016-28(3)-12.

Software Product Lines Modeling

[1,2] K.M. Lavrischeva, <lavr@ispras.ru>
[1,3,4] A.K. Petrenko <petrenko@ispras.ru>
[1] Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, Russia, 109004.
[2] Moscow Institute of Physics and Technology,
9, Institutsky per., Dolgoprudny, Moscow Region, Russia, 141701.
[3] Lomonosov Moscow State University.
GSP-1, Leninskie Gory, Moscow, 119991, Russia.
[4] NIU HSE, Miasnitskaya, 20, Moscow, Russia.

Absract. Complex software systems always exist for a long time, sometimes changing, and this leads to a variety of versions of such a system. In additional complex software systems usually have different (sometimes a lot) configurations due to different hardware and software environments, where they are intended to operate, or due to different user types with specific requirements. So, a complex software system can be regarded more correctly as a software system family or a software product line. Taking software families in consideration helps to increase reuse of their components and other software development artifacts. In difference with earlier works on software reuse, mostly focused on code or design reuse, software system family development tries to expand reuse on all kinds of development artifacts and activities, including documentation, verification, operation support, deployment, etc. One of the software system family development activities is modeling of family variability. This paper considers modern methods and approaches to such modeling, especially focusing on modeling of operating systems families variability. The research, which results are presented in this paper, is supported by RFBR.

Keywords: model checking; model refinement; variability; verification; testing; software system family; model management; configuration building, UML, MDA, MDD, MDE, PIM, PSM.

DOI: 10.15514/ISPRAS-2016-28(6)-4

For citation: Lavrischeva K.M., Petrenko A.K. Software Product Lines Modeling. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 6, 2016, pp. 49-64 (in Russian). DOI: 10.15514/ISPRAS-2016-28(6)-4

References

[1]. Clements P., Northrop L. Software Product Lines: Practices and Patterns. SEI Series in Software Engineering, Addison-Wesley, 2001. ISBN-13: 978-0201703320.
[2]. 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.
[3]. Bachmann F., Clements P. Variability in software product lines. CMU/SEI Technical Report CMU/SEI-2005-TR-012, 2005.

[4]. Lavrischeva E.M., Koval' G.I., Slabospitskaya O.O., Kolesnik A.L. Product Line Development Management Specifics. Problemy programmiovaniya [Problems of Software Developement], (3):40-49, 2009 (in Ukrainian).
[5]. Lavrischeva E.M., Slabospitskaya O.O., Koval' G.I., Kolesnik A.L. Theoretical Aspects of Variability Management in Product Lines. Vesnik KNU seria fiz.-mat. nauk [Notes of KNU, series on maths and physics], (1):151-158, 2011 (in Ukrainian).
[6]. Jacobson I., Griss M., Jonsson P. Software Reuse, Architecture, Process and Organization for Business Success. Addison-Wesley, 1997. ISBN-13: 978-0201924763.
[7]. Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language Reference Guide. 2-nd edition. Addison-Wesley Professional, 2005.
[8]. 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.
[9]. Benavides D., Segura S., Ruiz-Cortés A. Automated analysis of feature models 20 years later: a literature review. Information Systems, 35(6):615–636, 2010. DOI: 10.1016/j.is.2010.01.001.
[10]. Chen L., Babar M.A. A systematic review of evaluation of variability management approaches in software product lines. Information and Software Technology, 53(4):344–362, 2011. DOI: 10.1016/j.infsof.2010.12.006.
[11]. 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.
[12]. Thüm T., Apel S., Kästner C., Kuhlemann M., Schaefer I., Saake G. A classification and survey of analysis strategies for software product lines. ACM Computing Surveys, 47(1):article 6, 2014. DOI: 10.1145/2580950.
[13]. Liebig J., von Rhein A., Kästner C., Apel S., Dörre J., Lengauer C. Scalable analysis of variable software. Proceedings of the 2013 9-th Joint Meeting on Foundations of Software Engineering, pp. 81-91. ACM, 2013. DOI: 10.1145/2491411.2491437.
[14]. She S., Berger T. Formal semantics of the Kconfig language. Technical note, University of Waterloo, 2010. https://www.kernel.org/doc/Documentation/kbuild/kconfig-language.txt.
[15]. Emmerich W. Engineering Distributed Objects. Wiley, 2000.
[16]. Lavrischeva E.M. Theory of object-component modeling of variable software systems. ISPRAS, 2015 (in Russian). www/ispras.ru/preprints/docs/prep_29_2015.pdf.
[17]. Berger T., Rublack R., Nair D., Atlee J.M., Becker M., Czarnecki K., Wąsowski A. A survey of variability modeling in industrial practice. Proc. of the 7-th Intl. Workshop on Variability Modelling of Software-intensive Systems (VaMoS'2013), article No. 7, ACM 2013. DOI: 10.1145/2430502.2430513.
[18]. Lavrischeva E.M. Software engineering. Paradigms, technologies, CASE-tools. Urait, Moscow, 2015 (in Russian).
[19]. Batory D. Feature models, grammars, and propositional formulas. Proc. of the 9-th Intl. Conf. on Software Product Lines (SPLC'05), LNCS 3714, pp. 7-20, 2005. DOI: 10.1007/11554844_3.
[20]. Wang H., Li Y., Sun J., Zhang H., Pan J. A semantic web approach to feature modeling and verification. Proc. of Workshop on Semantic Web Enabled Software Engineering (SWESE'05), p. 44, 2005.
[21]. Benavides D., Segura S., Trinidad P., Ruiz-Cortés A. Using Java CSP solvers in the automated analyses of feature models. Generative and Transformational Techniques in

Software Engineering, LNCS 4143:399-408. Springer, 2006. DOI: 10.1007/11877028_16.

[22]. White J., Doughtery B., Schmidt D., Benavides D. Automated reasoning for multi-step software product-line configuration problems. Proc. of the 13-th Sofware Product Line Conference, pp. 11-20, 2009.

[23]. Zhang W., Mei H., Zhao H. Feature-driven requirement dependency analysis and high-level software design. Requirements Engineering, 11(3):205-220, 2006. DOI: 10.1007/s00766-006-0033-x.

[24]. Dietrich C., Tartler R., Schröder-Preikshat W., Lohmann D. Understanding Linux feature distribution. Proceedings of the 2012 Workshop on Modularity in Systems Software, pp. 15-20. ACM, 2012. DOI: 10.1145/2162024.2162030.

[25]. 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.

[26]. 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.

[27]. Nguyen T., Colman A., Han J. A web services variability description language (WSVL) for business users oriented service customization. Proc. Int. Workshop on User–Focused Service Engineering, Consumption and Aggregation (USECA 2011).

[28]. Meinicke J., Thüm T., Schröter R., Benduhn F., Saake G. An overview on analysis tools for software product lines. Proc. of the 18-th Intl. Software Product Line Conf.: Companion Vol. for Workshops, Demonstrations and Tools, vol. 2, pp. 94-101. ACM, 2014. DOI: 10.1145/2647908.2655972.

[29]. Hemakumar A. Finding Contradictions in Feature Models. Proc. of 12-th Intl. Conf. on Software Product Lines (SPLC'2008), v. 2, pp. 183-190, 2008.

[30]. Beyer D., Petrenko A.K. Verification of Linux operating system drivers. Trudy ISP RAN/Proc. ISP RAS, 23:405-412, 2012 (in Russian). DOI: 10.15514/ISPRAS-2012-23-23.

[31]. Apel S., Beyer D., et al. Strategies for product-line verification: case studies and experiments. Proceeding ICSE '13 Proceedings of the 2013 International Conference on Software Engineering, pp. 482-491.

[32]. Kuliamin V.V., Lavrischeva E.M., Mutilin V.S., Petrenko A.K. Verification and analysis of variable operating systems. Trudy ISP RAN/Proc. ISP RAS, 28(3):189-208, 2016 (in Russian). DOI: 10.15514/ISPRAS-2016-1(2)-12.