УДК 681.3.06

Д.В. Кознов, И.А. Новицкий, М.Н. Смирнов

ИНСТРУМЕНТЫ ДЛЯ УПРАВЛЕНИЯ ВАРИАТИВНОСТЬЮ: ГОТОВНОСТЬ К ПРОМЫШЛЕННОМУ ПРИМЕНЕНИЮ

Кознов Д.В., Новицкий И.А., Смирнов М.Н. **Инструменты для управления вариативностью: готовность к промышленному применению**
Аннотация. Разработка ПО методом организации семейства программных продуктов – популярный подход к промышленной разработке, основывающийся на повторном использовании различных активов и едином процессе разработки. Управление вариативностью нацелено на управление повторно используемыми активами и является одной из ключевых активностей при разработке семейств. В данной статье делается обзор существующих инструментов управления вариативностью – продуктов TVL, SPLOT, FeatureIDE, XFeature, FMP, FeatureMapper, CVL Tool, PLUM, pure::variants, DocLine, Gears. Основная цель этого обзора – выяснить, на сколько эти средства готовы для промышленного применения и реально используются в разработках.
Ключевые слова:линейка программных продуктов, семейство программных продуктов, инженерия семейства программных продуктов, управление вариативностью, повторно используемые активы.

Koznov D., Novitsky I., Smirnov M. **Variability Management Tools: readiness for industrial use**
Abstract. Software product line engineering is widespread approach for software development. Variability management aims to management for various reusable assets of product line. Variability management is one of the key activity in software product line engineering. In this paper we try to answer the question *Are variability management tools ready for industry use?* For that purpose we review existing variability management tools: TVL, SPLOT, FeatureIDE, XFeature, FMP, FeatureMapper, CVL Tool, PLUM, pure::variants, DocLine, Gears. We estimate current state, features and industrial applications for each tool, and outline the state of the area as a whole.
Keywords: software product lines, software product line approach, variability management, reusable assets.

1. Введение. На сегодняшний день широко распространена разработка ПО методом организации семейства (линейки) программных продуктов (software product lines) [67]. Этот метод подразумевает совместную разработку нескольких сходных продуктов в рамках одной компании. Разработка семейств программных продуктов предоставляет много преимуществ по сравнению с традиционными подходами к разработке ПО: повышение качества выпускаемых продуктов, уменьшение общей стоимости разработки, уменьшение времени вывода продукта на рынок и др. При этом

ключевую роль играет плановое повторное использование различных активов разработки – кода, требований, проектных спецификаций (design specifications), тестов и пр.

В инженерии семейств программных продуктов одно из главных мест занимает управление вариативностью (variability managment) [67]. Семейство удобно проектировать и сопровождать как иерархическую модель свойств, которые: 1) имеются у каждого продукта или 2) могут присутствовать только у некоторого подмножества продуктов или 3) специфичны для одного конкретного продукта и имеются только у него. Модель вариативности должна быть явным образом спроектирована, обладать чёткой структурой и иметь механизм для задания параметров с целью выделения подмножества свойств, описывающих конкретный продукт семейства. Узлы этой модели могут быть связаны с соответствующими активами разработки, и таким образом мы можем автоматически задать набор необходимых программных компонент, тестов и пр. для нового продукта, всего лишь задав выборку из модели вариативности. Эту модель необходимо поддерживать и масштабировать на протяжении всего жизненного цикла семейства.

На сегодняшний день существует значительное количество программных продуктов, позволяющих различными способами наладить управление вариативностью — PLUM [79], pure::variants [55], FeatureIDE [24] и др. Однако отсутствует обзор таких средств и, главное, не ясно, в какой степени эта область в состоянии предоставить индустрии готовые к использованию инструменты. Предлагаемая статья делает попытку ответить на этот вопрос.

2. Инженерия семейств программных продуктов.

2.1 История, основные достижения и текущее состояние. История инженерии семейств программных продуктов берет своё начало с первых идей повторного использования в разработке программных систем. Повторное использование как отдельная область программной инженерии была обозначена в публикации американского инженера Д. Макилроя [72]. Он предлагает идею «массового производства программных компонент» как основу разработки программного обеспечения. Позже, Д. Парнас в [76, 77] разработал принцип «сокрытия информации» (information-hiding principle) и ввёл термин «семейство программных продуктов». Далее Дж. Нейборс предложил системный подход к инженерии и анализу предметной области, известный под аббревиатурой Draco [75],

в рамках которого он впервые ввёл в обиход такие понятия как «предметная область» и «анализ предметной области», активно использованные впоследствии при разработке семейств программных продуктов. В дальнейшем все эти идеи и концепции активно развивались в работах [19,48,50,56] и были объединены институтом программной инженерии (Software Engineering Institute, SEI) в единый метод разработки семейства программных продуктов (software product line approach) [15]. Этот метод, помимо определений и концепций, включает в себя также различные рекомендации по разработке всевозможных активов и созданию конечных продуктов, доходя до уровня организации программного кода [49].

В 1970-х гг. проф. А.Л. Фуксман (Ростовский университет) в монографии «Технологические аспекты создания программных систем» [12] ввел понятие вариантной сети, которое предлагалось использовать для управления вариативностью программных компонент.

Первая промышленная разработка семейства программных продуктов, освещённая в литературе, состоялась в 1977 г. в Японии. Был создан метод под названием «Toshiba Software Factory» [71], который использовался компанией Toshiba Corporation для разработки серии программных систем для генераторов электроэнергии. В дальнейшем разработка семейств программных продуктов широко применялась в самых различных областях — мобильные операционные системы, авиационное или медицинское ПО, информационные системы и пр. Отчёты о наиболее успешных проектах, где применялась инженерия семейств программных продуктов, можно найти здесь [81].

На сегодняшний день данный подход активно развивается. Большое количество исследований ведётся в области формального анализа моделей вариативности, интеграции различных технических аспектов разработки с требованиями бизнеса и организации ИТ–компаний, стандартизации процесса разработки, совершенствования методов прогнозирования и обеспечения качества. Также большое внимание уделяется созданию инструментов автоматизации управления активами.

2.2 Повторно используемые активы. Для организации семейства программных продуктов в компании должны «накопиться» повторно используемые активы разработки (reusable assets) — программные компоненты, процедуры процесса, квалификация пер-

сонала и т. д. Согласно [11] они бывают следующих видов.

- Требования. Существенная часть требований к продуктам семейства является общей, что значительно упрощает разработку и сопровождение требований (requirement engineering) для отдельных систем.

- Архитектура. В рамках семейства продуктов разрабатывается архитектура типовой системы. Архитектура конкретной системы создаётся на её основе и тем самым существенно экономятся ресурсы на проектирование.

- Компоненты. Общая для всех представителей семейства функциональность реализуется в виде повторно используемых программных компонент, из-за чего разработка систем значительно упрощается.

- Активы тестирования. Тестирование особенно важно в разработке семейства программных продуктов, например: модульное тестирование позволяет убедиться в качестве повторно используемых компонент; поскольку готовый продукт семейства, как правило, собирается из некоторого набора компонент, то интеграционное тестирование даёт уверенность в надёжности полученного продукта семейства. Соответственно, повторно используемыми активами могут быть тестовые планы, варианты тестирования, тестовые скрипты и пр.

- Модели — анализа, проектирования, производительности и т. д. — также могут повторно использоваться при создании продуктов семейства.

- Средства разработки. В рамках семейства продуктов формируется общая для разных продуктов технологическая среда — средства разработки (IDEs — Integrated Development Environments), СУБД, средства поддержки планирования, тестирования, конфигурационного управления и т. д. Кроме того, могут создаваться специальные программные средства, «заточенные» под специфику данного семейства (например, кодогенераторы для визуальных моделей).

- Процессы. Здесь речь идёт о повторном использовании приёмов работы с заказчиком, методов управления проектом, способов работы с планами и о других процедурах процесса разработки.

- Квалификация сотрудников. Поскольку системы, разрабатываемые в рамках семейства, похожи, а также существует общая инфраструктура разработки – технологии и программные средства, процедуры процесса, знания в предметной области, на которую ориентировано семейство и т. д., — то в такой ситуации легко «передвигать» работников из одного проекта в другой (например, при необходимости «усилить» какой-то проект), либо после окончания одного проекта подключать разработчиков к новому проекту или вводить в уже существующий.

2.3 Управление вариативностью — это одна из ключевых практик инженерии семейств программных продуктов, отличающая её от традиционных подходов к разработке программного обеспечения. В рамках разработки только одного продукта управление вариативностью превращается в конфигурационное управление [1] — управление изменениями активов разработки, происходящими в течении времени (variation in time). При переходе к инженерии семейств программных продуктов этого оказывается недостаточно — необходимы методы, позволяющие управлять структурными различиями индивидуальных продуктов семейства (variation in space). Итак, управление вариативностью является сочетанием управления изменениями в течении времени и управление структурными различиями продуктов [57, 64, 67].

В [64] управление вариативностью разделено на девять подобластей — см. рис. 1. Строкам этой таблицы соответствуют активы, которые обычно рассматриваются в рамках конфигурационного управления (configuration management items) — файлы, компоненты, продукты. Первые два столбца таблицы определяют способы внесения изменения в эти активы — последовательное и параллельное изменение активов. Последний столбец описывает методы поддержки вариативности активов с точки зрения изменений предметной области.

Рассмотрим более детально практики, представленные на рис. 1.

Базовое конфигурационное управление:

	Sequential Time	Parallel Time	Domain Space
Files	version management	branch management	variation point management
Components	baseline management	branched baseline management	customization management
Products	composition management	branched composition management	customization composition management

Рис. 1. Практики для управления вариативностью (рис. взят из работы [64])

- управление версиями (version management) — управление изменением версий файлов;
- управление ветками (branch management) — управление независимыми ветками файловых версий;
- baseline–управление (baseline management) — периодическая фиксация промежуточных результатов в виде текущей официальной версии разрабатываемого актива;
- управление baseline–ветками (branched baseline management) — отвечает за управление независимыми baseline- ветками.

Композиция модулей:

- управление составлением (composition management) — отвечает за фиксацию консистентного результата разработки продукта, состоящего из набора версий компонент;
- управление составлением веток (branched composition management) — отвечает за управление независимыми ветками продукта (за управление отдельными ветками отвечает предыдущая практика).

Массовая персонализация — это широкомасштабное производство высокомодульных продуктов, удовлетворяющих нуждам ин-

дивидуальных заказчиков [36]. В рамках массовой персонализации выделяются следующие виды практик:

- управление точками вариативности (variation point management) — отвечает за поддержку вариативности на уровне файлов и предоставляет конкретный набор вариантов файлов, удовлетворяющих предъявленным требованиям к продукту семейства;
- персонализационное управление (customization management) — отвечает за выборку/построение компонент, удовлетворяющих предъявленным требованиям к продукту семейства;
- композиционное персонализационное управление (customization composition management) — отвечает за управление составлением компонент, получившихся в результате персонализационного управления, в финальный продукт.

Предложенное разделение общей задачи управления вариативностью для разработки линейки программных продуктов даёт преимущества, перечисленные ниже.

- *Вид на семейство как на единую систему,* то есть возможность обеспечить общий взгляд на совокупность общих частей и различий продуктов семейства. При этом данная информация должна быть легко доступной в течение всего процесса разработки семейства.
- *Низкий адаптационный порог.* Инкрементальные стратегии адаптации, повторное использование существующих технологий и активов, а также различные варианты расширения базового метода инженерии линеек продуктов — все это оказывается возможным при развитом управлении вариативностью и в целом позволяет значительно уменьшить время, цену и усилия, затрачиваемые на разработку семейств.
- *Гибкость процесса.* Управление вариативностью легко интегрируется с требованиями бизнеса и позволяет повторно использовать различные активы разработки, применять реинжиниринг, предоставляет возможность для использования проактивного или реактивного управления разработкой, а также позволяет применять рефакторинг [65].

2.4 Модель вариативности. Первая попытка дать формальное определение модели вариативности, включая графическую нотацию, была сделана в 1990 г. в рамках метода FODA [58] группой исследователей из института программной инженерии США во главе с Кио Кангом (Kyo Kang). Эта нотация получила название «диаграмма возможностей» (feature diagram). С течением времени данная нотация претерпела ряд изменений, на сегодняшний день актуальные варианты описания модели вариативности можно найти в работах [33, 40, 58, 59, 80]. Наиболее современный, на наш взгляд, вариант модели вариативности разработан Клаусом Полем (Pohl Klaus) в 2005 г. и детально описан в работе [80]; в данном разделе мы, следуя этому источнику, опишем основные понятия модели вариативности.

Основная задача представления модели вариативности — отразить общие и различные черты систем, входящих в семейство продуктов. Модель вариативности состоит из следующих элементов, перечисленных ниже.

- *Точка вариативности (variation point)* — служит указателем на место в активе, где существуют различия между продуктами семейства, а также несёт конкретную информацию о данных различиях, тем самым ограничивая набор возможных вариантов.

- *Вариант (variant)* — это объект, способный удовлетворить ограничению какой–либо из точек вариативности.

- *Вариативная зависимость (variability dependency)* — определяет связь между вариантом и точкой вариативности. Как правило, такие зависимости делятся на три типа: *обязательная зависимость* — вариант включается в продукт при условии, что его точка вариативности имеет место в контексте создаваемого продукта семейства; *опциональная зависимость* — вариант входит в создаваемые продукты семейства при выполнении дополнительных ограничений; *альтернативный выбор* ограничивает нижнюю и верхнюю допустимую границу количества вариантов, допустимых для включения в продукт.

- *Ограничивающая зависимость (constraint dependency)* — отношение между двумя элементами модели вариативности (как

вариантов, так и точек вариативности). Отношение выражается в форме требований или исключений. Обе формы можно выразить следующим образом: включение первого элемента в продукт требует (исключает) включение второго элемента. Допускаются также более сложные способы задания данной зависимости.

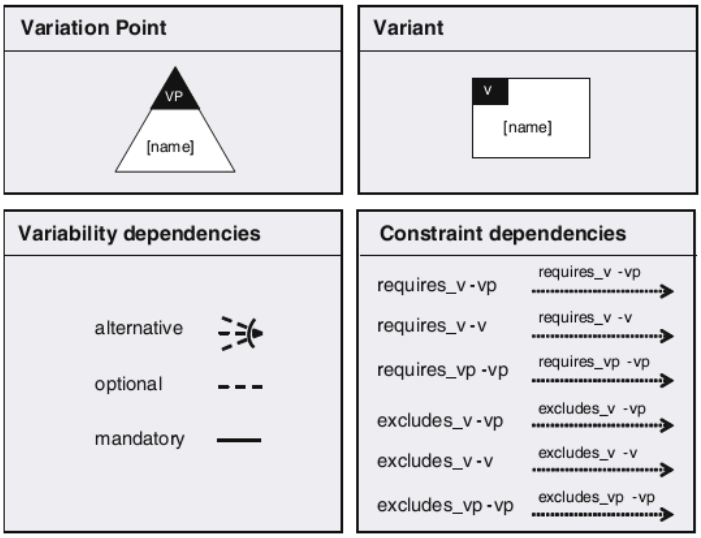

Рис. 2. Графическая нотация модели вариативности (рис. взят [80])

На рис. 2 представлена графическая нотация модели вариативности из [80]. Каждый продукт семейства определяется конкретным набором вариантов. Вариант может входить в конечный продукт семейства только тогда, когда его точка вариативности входит в определение продукта, и данный вариант удовлетворяет условиям вариативной и ограничивающей зависимости (таким образом предотвращает конфликты между выбранными вариантами). Процесс выбора вариантов для каждой из допустимых точек вариативности будем называть *разрешением модели вариативности*. Результатом этого процесса является однозначная спецификация для разрабатываемого продукта семейства — набор активов разработки, используемых в данном продукте (из списка всех воз-

можных) и/или уточнённую спецификацию об этих активов (снятые вариативные неоднозначности).

3. Инструменты для управления вариативностью. Мы проводили выборку инструментов по публикациям в конференции Software Product Line Conference за период с SPLC'08 по SPLC'11. Это ведущая мировая научно-практическая конференция в области инженерии семейства программных продуктов.

В этой работе мы рассматриваем следующие инструменты: TVL [85], SPLOT [84], FeatureIDE [43], XFeature [87], FMP [47], FeatureMapper [44], CVL Tool [35], PLUM [79], pure::variants [82], DocLine [60], Gears [51].

3.1 Text-Based Variability Language (TVL). Этот подход предназначен для создания моделей вариативности с помощью C–подобного языка [27, 31], имеющего целью предоставить простую, легко читаемую текстовую нотацию для описания модели вариативности семейства продуктов. Этот язык включает в себя большинство конструкций из наиболее распространённых языков, используемых для описания вариативности.

Программная реализация языка TVL распространяется в виде JAR–файла и имеет открытый исходный код [27, 85]. Помимо парсера для языка TVL утилита содержит инструменты, предназначенные для анализа и нормализации (декомпозиции «синтаксического сахара») модели вариативности. Также утилита позволяет преобразовать построенную модель вариативности в эквивалентную пропозициональную формулу в конъюнктивной нормальной форме. Разработаны плагины для графических редакторов Notepad++, Smultron/Fraise, TextMate, gedit, реализующие функции автодополнения и подсветки синтаксиса языка TVL.

Проект разрабатывается членами исследовательского центра PReCISE при кафедре информатики Намюрского университета (University of Namur), Бельгия. Ключевыми разработчиками являются Quentin Boucher, Andreas Classen, Anthony Cleve, Paul Faber, Patrick Heymans, Arnaud Hubaux, Raphael Miche. Единственный выпуск программной утилиты состоялся в августе 2010 году. Основные публикации датированы 2010 и 2011 годами.

Информации об использовании языка TVL и соответствующей утилиты в коммерческих проектах нет.

В целом инструментальная поддержка языка ограничивается разбором и анализом создаваемых моделей. Инструмент не содер-

жит процедур автоматизации создания готовых продуктов и не включает средств для их определения на модельном уровне. Тем не менее, формальный язык спецификации архитектуры является одним из важнейших элементов разработки семейства программных продуктов, и TVL может послужить основой для стандарта диаграмм вариативности.

3.2 Software Product Lines Online Tool (SPLOT). Данный инструмент предназначен для разработки архитектуры семейства продуктов [73]. Инструмент является Web-системой и состоит из набора Web-сервисов, позволяющих пользователю создавать, редактировать, анализировать и разрешать модели вариативности. SPLOT имеет общедоступный репозиторий для хранения пользовательских моделей, модель семейства задаётся на языке SXFM (Simple XML Feature Model) [73]. Язык SFXM позволяет задать модель вариативности как древовидную структуру с набором ограничений, где ограничения — это условия, записанные в виде логических формул в конъюнктивной нормальной форме. Также к модели вариативности можно присоединять набор метаданных — сведения об авторе модели, дата создания, краткое описание и пр. Для создания моделей на языке SXFM система SPLOT предоставляет графический Web-редактор.

Одна из важных особенностей инструмента заключается в том, что пользователь SPLOT может разделить процесс разрешения модели вариативности на несколько этапов, причём каждый этап содержит лишь частичное представление общей архитектуры семейства продуктов[1]. Частичное представление может быть полезным, если над разработкой конечного продукта работает группа людей с различными уровнями ответственности. Рабочий процесс в SPLOT описывается на языке YAWL. Благодаря интеграции с приложением YAWL Process Editor[2] можно следить за ходом процесса разрешения диаграммы вариативности через Web-интерфейс приложения SPLOT.

[1]Этот подход носит название Multi-View Feature Model и описан в работе [13].

[2]YAWL Process Editor [89] — это свободно распространяемая программная система для управления бизнес-процессами, описанными на языке YAWL. Функциональность инструмента позволяет решать самый широкий спектр прикладных задач — например, определять и исполнять процессы преобразования данных, отображать элементы рабочего процесса на вызов методов Java-классов, создавать прототипы Web-сервисов для Tomcat и пр.

SPLOT является некоммерческим продуктом с открытым исходным кодом и разработан на основе технологии Java Servlet. На сайте проекта [84] также есть исходный код парсера языка SXFM.

SPLOT разрабатывается исследовательской группой Generative Software Development Lab при университете Уотерлу (University of Waterloo, UW), Канада. Активными участниками проекта являются Marcilio Mendonca, Moises Branco, Donald Cowan, Ebrahim Abbasi, Arnaud Hubaux, Patrick Heymans. Первый выпуск SPLOT состоялся в мае 2009 года, в декабре 2010 года была выпущена версия, интегрированная с приложением YAWL Process Editor. Последняя версия приложения выпущена в ноябре 2011 года. Информации о коммерческом использовании продукта SPLOT нам не удалось найти. Но стоит отметить, что на текущий момент в репозитории приложения SPLOT содержится более 150 пользовательских моделей.

Приложение SPLOT включает большой набор инструментов для проектирования масштабной архитектуры семейства программных продуктов. Однако SPLOT не обладает средствами для создания конечного продукта, позволяя лишь указывать те требования, которым они должны соответствовать.

3.3 FeatureIDE. Данный продукт является интегрированной средой, предназначенной для создания семейства продуктов через внедрение вариативности в программный код [55, 66]. FeatureIDE предоставляет возможности для разработки семейства продуктов в соответствии с заданной моделью вариативности и определёнными повторно используемыми компонентами. В качестве повторно используемых компонент FeatureIDE рассматривает структурированные фрагменты программного кода, каждый фрагмент реализует какую-либо функциональность продукта. Такой подход к разработке семейств программных продуктов получил название Feature-Oriented Programming (FOP) [21].

FeatureIDE является некоммерческим продуктом с открытым исходным кодом, создан на основе платформы Eclipse и поддерживает следующие языки программирования: Java, JavaCC, C, C++, C #, Haskell, а также XML [43]. Кроме основного направления — разработка семейства продуктов — инструмент успешно применяется в аспектно-ориентированном программировании [6] и интегрирован с продуктом AspectJ [17], а также в контексте дельта-ориентированного программирования (интегрирован с продуктом

DeltaJ [37]) и метапрограммирования (поддерживает препроцессоры Antenna [14] и Munge [74]).

FeatureIDE разработан в Магдебургском университете имени Отто фон Герике (Otto-von-Guericke University Magdeburg, OVGU), Германия. Активными участниками проекта FeatureIDE являются Thomas Thum, Christian Kastner, Thomas Leich, Don Batory, Jens Meinicke, Fabian Benduhn, Sebastian Henneberg, Sven Apel.

Проект стартовал в 2004 году как среда разработки для языка JavaCC (диалект Java 1.4) и назывался AHEAD Tool Suite. Впоследствии было выделено ядро системы, и в 2006 году проект стал называться FeatureIDE. Последний выпуск продукта состоялся в июне 2011 года. На сегодняшний день проект активно развивается.

В дополнение к вышеуказанным средствам FeatureIDE интегрирован также с проектами AHEAD Tool Suite [16] (инструментальная поддержка FOP для языка JavaCC), а также продуктами FeatureC++ [45] (инструментальная поддержка FOP языка C++) и FeatureHouse [42] (реализация общего подхода FOP для языков C #, Java, Haskell и т.д.).

Примером коммерческого использования FeatureIDE является проект по разработке военных симуляторов FSATS (Fire Support Simulators) [22, 23] по заказу минобороны США. Для реализации FSATS применялся инструмент AHEAD Tool Suite. Ядро системы FSATS определяет модель вариативности, которая содержит 21 компоненту. Модель вариативности задаётся 4500 строками кода на языке JavaCC. Авторы проекта FSATS использовали метод FOP для того чтобы, во-первых, поддерживать высокую модульность приложения, во-вторых, эффективно добавлять в приложение эмуляцию различных вариантов боевых действий.

Инструмент FeatureIDE имеет достаточное количество возможностей, которые в совокупности полностью автоматизируют процесс создания семейства программных продуктов. Более того, FeatureIDE предлагает несколько подходов к созданию семейства продуктов.

3.4 XFeature. Данный инструмент является программной реализацией метода моделирования вариативности на основе XML (XML-Based Feature Modeling) [28]. Этот подход позволяет описывать схемы XML-документов, основываясь на модели вариативности. Суть метода заключается в описании семейства с помощью

трехуровневой архитектуры:

- уровень, не зависимый от предметной области (Family-Independent Layer) — включает средства для определения вариативности семейства продуктов, как правило, это XSD-схема, описывающая один из вариантов представления модели вариативности;
- уровень, не зависимый от решения в предметной области (Application Independent Layer) — содержит описание модели семейства продуктов, и эта модель служит общим определением любого продукта семейства;
- уровень модели решения (Application Specific Layer) — включает информацию, которая используется при разработке продуктов семейства.

XFeature позволяет графическое отображение и редактирования XML-документов, проверяет корректность построенного XML-документа в соответствии правилами определёнными в модели семейства, использует различные способы определения модели вариативности и настройки генерации соответствующей XSD-схемы.

XFeature является некоммерческим продуктом с открытым исходным кодом и реализован как плагин для среды Eclipse. Графические редакторы для работы с XML-документами созданы на основе технологии Graphical Editing Framework (GEF) [9]. Для поддержки сложных ограничений, накладываемых на метамодели конечных продуктов, XFeature использует диалект языка XML Schema под названием «Schematron» [28].

XFeature разработан в научно-исследовательской лаборатории Automatic Control Laboratory Швейцарской высшей технической школе Цюриха (Swiss Federal Institute of Technology, ETH-Zurich) совместно с компанией P&P Software. Лидерами проекта XFeature являются Ondrej Rohlik, Alessandro Pasetti. Первая версия инструмента вышла в 2005 году, последняя — в 2007 году.

Инструмент использовался университетом ETH-Zurich в контексте проекта ASSERT [18], ориентированного на разработку встраиваемых систем реального времени. В рамках проекта ASSERT успешно реализовано три пилотных решения: Multi-Domain Advanced Available Automated Systems, Highly Reliable Infrastructures, Multi-Platform Cooperation. Для реализации решений был использован

набор, состоящий более чем из тридцати технологий. К сожалению, конкретного описания того, как именно в проекте применялся инструмент XFeature, нам не удалось найти.

3.5 Feature Modeling Plug-in (FMP). Этот инструмент предназначен для построения и анализа диаграмм вариативности. В нем реализованы диаграммы вариативности, основанные на отношении кардинальности (Cardinality-Based Feature Models) [34]. Инструмент обладает визуальным редактором, предназначенным для модели вариативности, а сама модель представляется в виде дерева. Между элементами модели вариативности можно задавать ограничения на языке XPath 2.0 [88].

FMP является некоммерческим продуктом с открытым исходным кодом. Он реализован как плагин для среды Eclipse и основан на технологии Eclipse Modelling Framework (EMF).

FMP разрабатывался исследовательской группой Generative Software Development Lab при университете Уотерлу (University of Waterloo, UW), Канада. Вот список участников проекта: Chang Hwan Peter Kim, Krzysztof Czarnecki, Michal Antkiewicz, Sean Lau, Steven She. Продукт находиться на стадии прототипа. Первая версия вышла в 2004 году, последняя — в 2011 году. На текущий момент проект больше не поддерживается.

Инструмент FPM позволяет лишь проектировать архитектуру семейства программных продуктов, что покрывает только один из этапов процесса разработки семейства продуктов. Благодаря открытому исходному коду и специфики реализации инструмент может служить дополнительным модулем для более масштабной среды разработки.

3.6 FeatureMapper. Этот инструмент предназначен для внедрения вариативности в модели продуктов, которые основаны на технологии EMF/Ecore[3]. Основная идея, реализованная в инструменте, заключается в декомпозиции модели продукта на ряд фрагментов, и отображении составленных фрагментов на элементы диаграммы вариативности [10]. Финальный продукт состоит лишь из тех фрагментов, которые соответствуют элементам, вошедшим в результирующее множество после разрешения диаграммы вариативности.

FeatureMapper — некоммерческий продукт без открытого ис-

[3]Подробнее см. работу [9].

ходного кода, создан на основе платформы Eclipse [44]. В инструменте реализован визуальный язык описания диаграмм вариативности. Инструмент предоставляет ряд средств для создания и визуализации соответствия (mapping) между элементами диаграмм вариативности и пользовательскими объектными моделями. Также FeatureMapper интегрирован с редактором модели вариативности приложения pure::variants (см. раздел 2.9).

FeatureMapper разработан исследовательской группой Software Technology Group при факультете информатики Дрезденского технического университета (Dresden University of Technology, TUD), в Германии. Florian Heidenreich, Christian Wende, Jan Kopcsek являются основными разработчиками подхода. Проект FeatureMapper стартовал в январе 2008 году, первый выпуск состоялся в декабре 2008 году, всего было выпущено пять версий инструмента FeatureMapper, последняя версия — в июне 2011 года.

FeatureMapper позволяет достаточно быстро создать описание линейки программных продуктов. Он также способен интегрироваться со сторонними DSM-инструментами[4] для расширения возможностей моделирования.

3.7 CVL Tool. Данный инструмент является модельно-ориентированной средой для разработки линейки программных продуктов, описание которых задаётся на специальном предметно–ориентированном языке. CVL Tool позволяет автоматически собирать продукты из общих компонент в соответствии с заданной моделью вариативности и правилами её разрешения.

CVL Tool является некоммерческим продуктом без открытого исходного кода и реализован в виде Eclipse-плагина [35]. Для формального описания модели семейства продуктов используется язык Common Variability Language (CVL). Этот язык основан на общей концепции вариативности (см. [58]), но расширен понятием типа. В состав инструмента CVL Tool входят следующие компоненты: редактор языка CVL, средства интеграции со сторонними DSM-инструментами, средства преобразования сторонних моделей.

CVL Tool разработан независимой исследовательской компани-

[4]DSM-подход (Domain-Specific Modeling) предназначен для быстрого создания визуальных языков и их программных реализаций под конкретные прикладные задачи [2, 29]. Этот подход активно применяется в контексте метода разработки семейств программных продуктов.

ей SINTEF, которая основана Норвежским институтом технологий (Norwegian Institute of Technology, NTH), который, в свою очередь, является частью Норвежского университета естественных и точных наук (Norwegian University of Science and Technology, NTNU). Активными участниками проекта CVL Tool являются Franck Fleurey, Øystein Haugen, Birger Møller-Pedersen, Gøran K. Olsen, Andreas Svendsen, Xiaorui Zhang. Проект начал своё развитие в сентябре 2009 года. Первая версия выпущена в июне 2010 года. С тех пор осуществляется активная поддержка продукта.

На официальном сайте проекта [35] размещена документация продукта и несколько примеров его использования. CVL Tool успешно применялся в разработке некоммерческой адаптивной системы Smart-home System [78]. Система Smart-home System предназначена для взаимосвязанного управления электронными устройствами (например, TV, устройства освещения, охранные системы), расположенными в частных домах. Система должна корректно обрабатывать события добавление/удаление устройств из окружения, в котором она работает. Модель окружения, для которого работает Smart-home System, создаётся на DSL-языке PervML [78]. Задача CVL Tool состояла в конфигурировании и динамическом изменении модели окружения на базе информации о допустимых устройствах, при этом каждое устройство имело описание в виде фрагмента модели PervML. Не смотря на то, что CVL Tool показал положительные результаты в решении возложенных на него задач, этот инструмент не был принят в качестве основной составляющей системы Smart-home System. Упоминаний о других промышленных использованиях инструмента нет.

3.8 Product Line Unified Modeller (PLUM). Данный пакет инструментов предназначен для модельно-ориентированной разработки семейств программных продуктов [41,52]. В отличие от большинства других средств он не навязывает разработчикам фиксированного типа общих активов и не обладает предопределённой процедурой сборки активов в конечный продукт. Все это определяют разработчики, использующие продукт. Благодаря такому подходу PLUM позволяет разрабатывать конечные продукты любого вида (например, документацию, программный код, схему базы данных и пр.).

PLUM — некоммерческий продукт без открытого исходного кода, созданный на основе платформы Eclipse [79]. В продукте реали-

зован метод разработки семейств программных продуктов Direct Product Variability (DPV), предложенный Европейским Институтом Программного Обеспечения [69]. В состав PLUM входят инструменты, применяемые на всех этапах всего жизненного цикла проектирования семейства продуктов. Также PLUM предоставляет визуальный предметно-ориентированный язык для спецификации архитектуры семейства продуктов (DecisionModel, DM). Модель вариативности несколько отличается от классических диаграмм вариативности [58], но не слабее их по выразительной силе.

PLUM разработан членами департамента Программной Инженерии при Европейском Институте Программного Обеспечения (European Software Institute, ESI) в Испании. Активными участниками проекта PLUM являются Jabier Martinez, San Sebastian, Manuel Fernandez, Cristina Lopez, Estibaliz Ulacia, Marta del Hierro. Первый выпуск продукта состоялся в 2007 году. Всего было выпущено три версии, последняя — в 2011 году.

PLUM успешно применяется компанией Conmunica Mediatrader в разработке семейства Web-приложений [70]. Основное направление компании — разработка сайтов Web-каталогов продукции, Интернет-магазинов, промо-сайтов, а также тематических сайтов. PLUM использовался для достижения следующих целей: реализация поддержки различных Интернет-браузеров (учитываются также браузеры для мобильных ОС); использование различных методов подбора информации в зависимости от роли пользователя; реализация специальных возможностей (например, конфигурируется размер шрифта для людей с нарушением зрения).

3.9 Pure::Variants. Данная технология объединяет группу инструментов, предназначенных для управления вариативностью для различных сфер программной индустрии. Инструменты pure::variants объединены концепцией общего процесса разработки семейств программных продуктов. Согласно подходу pure::variants линейку программных продуктов задаёт модель вариативности и модель семейства [24, 44]. Модель вариативности в pure::variants позволяет в общепринятых терминах описывать целевую предметную область и не зависит от определений общих активов и отношений между ними. Модель семейства имеет иерархическую структуру, в листьях которой расположены ссылки на общие активы. В модель также включается описание процесса сборки активов в конечный продукт и, если необходимо, задаются процедуры предва-

рительной обработки активов. Тем самым, благодаря слабой связи между описанием предметной области, программными компонентами, реализующими конкретное решение, и гибкой процедуре обработки конечных активов, инструменты pure::variants позволяют эффективно реализовывать масштабные проекты.

Все инструменты pure::variants реализованы как плагины для среды Eclipse. Каждый инструмент нацелен на поддержку активов определённого вида. Такие инструменты, как правило, кроме функциональности по управлению общими активами, имеют также дополнительный модуль интеграции, позволяющий создать более тесную связь с приложением, в котором разрабатываются целевые активы. Вот некоторые из инструментов pure::variants:

- pure::variants Evaluation (Community) управляет повторным использованием программного кода (java, C, C++), XML-документов, текстовых файлов;
- pure::variants for AUTOSAR управляет повторным использованием компонент архитектуры AUTOSAR[5] для автомобильного программного обеспечения;
- pure::variants for Enterprise Architect управляет повторным использованием моделей на языках SysML и UML, разработанных в пакете Enterprise Architect[6];
- pure::variants for IBM Rational DOORS управляет повторным использованием требований, разрабатываемых в системе IBM Rational DOORS[7];

[5] AUTOSAR (AUTomotive Open System ARchitecture) — это открытая платформа для проектирования автомобильного ПО [20]. Основная задача, на решение которой направлена AUTOSAR, заключается в стандартизации описания узлов автотранспортных средств. На сегодняшний день модели AUTOSAR служат основным форматом обмена спецификациями автомобильных систем между поставщиками ПО и производителями автотранспорта.

[6] Enterprise Architect [39] — это модельно-ориентированное средство для проектирования приложений на языках UML, BPMN, SysML. Enterprise Architect используется для разработки ПО в самых разных предметных областях, например, в аэрокосмической индустрии. Инструмент включает возможности для автоматического генерирования кода, вычисления проектных метрик, средства циклической разработки (round-trip engineering tools) и т.д.

[7] IBM Rational DOORS [53] — это приложение для управления требованиями, которое позволяет создать граф требований и позволяет собирать, трассировать, анализировать и управлять изменениями требований, поддерживает

- pure::variants for Simulink управляет повторным использованием компонент динамических систем, моделируемых с помощью приложения Simulink[8].

Инструменты pure::variants распространяются по коммерческой лицензии. Исключение составляют продукты pure::variants Community и pure::variants Evaluation — они распространяются по свободной лицензии.

Продукт pure::variants разработан компанией pure-systems GmbH [82]. Эта компания основана в Германии в 2001 году исследователями из Магдебургского университета имени Отто фон Герике (Otto-von-Guericke University Magdeburg, OVGU) и Института компьютерной архитектуры и программных технологий Общества имени Фраунгофера (Fraunhofer Institute for Computer Architecture and Software Technology, FIRST). Активными участниками проекта являются Holger Schmiedefeldt, Danilo Beuche, Hans Peter Jepsen, Jan Gaardsted Dall. Первый выпуск проекта pure::variants состоялся в сентябре 2004 года, последняя версия выпущена в мае 2012 года, в настоящее время проект активно развивается.

Продукты pure::variants имеют несколько значительных примеров использования в коммерческой индустрии. Датская компания Danfoss Drives (один из крупнейших международных концернов по производству тепловой автоматики, холодильной техники, приводной техники и промышленной автоматики) использует инструменты pure::variants с 2007 года для решения следующих задач при разработке одного своего крупного семейства продуктов: планирование и мониторинг новых продуктов, автоматическая генерация конфигурационных файлов для различных активов продуктов на основе модели вариативности [54]. Немецкий автомобилестроительный концерн Daimler AG — крупнейшая по оборо-

автоматизированную проверку их выполнения, а также групповую работу по управлению требованиями в масштабах организации. Этот продукт является одним из самых известных в области управления требованиями.

[8]Simulink [83] — это инструмент для моделирования, имитации и анализа динамических систем. Он позволяет строить графические блок-диаграммы и эмулировать их выполнение, проводить исследование работоспособности систем. Simulink тесно интегрирован с математическим пакетом MATLAB, что позволяет использовать MATLAB алгоритмы в моделях приложения Simulink, а также экспортировать результаты работы Simulink моделей в программную среду MATLAB. Simulink используется в прикладных задачах теории управления и цифровой обработки сигналов.

ту (по результатам 2006 года) компания Германии — создал совместно с командой pure::variants методику разработки семейств программных продуктов для моделей Simulink (pure::variants for Simulink) [38]. Начиная с 2008 г. продукт pure::variants for Simulink вошёл в стек технологий компании Dailmler AG для управления Simulink-активами.

3.10 DocLine. Данный инструмент предназначен для управления повторным использованием документации — преимущественно, пользовательской документации продуктов семейств [30,51,60,62,63]. Он поддерживает вариативное повторное использование — то есть фрагменты текста, используемые в разных контекстах, могут варьироваться в зависимости от значений параметров. Подход является «надстройкой» над известной в мире Linux XML-технологией DocBook, предназначенной для разработки сложной документации ПО. Для управления вариативностью подход содержит визуальный язык, основанный на feature diagrams [58].

DocLine — некоммерческий продукт с открытым исходным кодом, созданный на основе платформы Eclipse. В состав инструмента входят редакторы диаграмм, текстовый редактор, генератор в DocBook. Последний выполняет разрешение вариативности на основе выбора пользователя (технического писателя) и создаёт целевую документацию на «чистом» DocBook. Далее средствами DocBook утилит можно получить документацию в PDF, HTML и других форматах.

DocLine разрабатывается на кафедре системного программирования Санкт-Петербургского государственного университета в России. Активными участниками проекта являются Дмитрий Кознов, Михаил Смирнов, Константин Романовский, Иван Новицкий. Первый выпуск продукта состоялся в 2006 году, последний выпуск в 2012 году.

Существует единственная индустриальная апробация подхода — разработка документации семейства телефонных станций ЗАО «Ланит-Терком» [46]. Продукт использовался для создания более чем двухсотстраничной документации для двух телефонных станций, функциональность одной из которых являлась вырезкой (с модификациями) из функциональности другой.

3.11 BigLever Gears Product Line Engineering Tool and Lifecycle Framework (Gears). Этот фреймворк является одним

из наиболее зрелых решений в области инструментальной поддержки Product Line подхода, ориентирован на поддержку всех основных фаз жизненного цикла разработки семейств программных продуктов и имеет следующие особенности [63]:

- единая модель вариативности, в которой определяются требования, документация, программный код, тестовые планы, архитектура и прочие активы семейства, включая вариации;
- единый механизм управлением вариативностью, т.е. для всех инструментов, используемых при разработке, применяется единый способ управления точками вариативности;
- единый подход к сборке активов — все типы активов конфигурируются и собираются в конечный продукт единственным инструментом.

Gears основан на 3-Tiered PLE Methodology методологии [63]. Программная часть Gears состоит из ядра и дополнительных инструментов по разработке активов. Ядро служит для создания и управления диаграммами вариативности, конфигурирования активов и сборки активов в конечный продукт, а также для предоставления унифицированного программного интерфейса для интеграции. Разработка активов может осуществляться с помощью сторонних программных средств, для интеграции такого инструмента в Gears он должен реализовать «PLE Bridge»-модуль. Этот модуль обеспечивает связь между ядром и сторонним инструментом: с одной стороны, для корректной реализации активов ядро предоставляет информацию о точках вариативности, с другой стороны, инструмент и соответствующие активы включаются в общую инфраструктуру процесса разработки семейства продуктов.

На данный момент Gears поддерживает следующие виды активов.

- Программный код для следующих языков: Java, C, C++, C#, Ada, Perl, XML, HTML. Gears интегрирован с Microsoft Visual Studio и Eclipse.
- Требования. Gears поддерживает управление вариативностью в инструментах для разработки требований IBM Rational DOORS, Serena Dimensions RM, Microsoft Excel, Word.

- Архитектура. Gears поддерживает языки проектирования SysML and UML, интегрирован с IBM Rational Rhapsody, Sparx Systems Enterprise Architect.

- Активы тестирования. Gears интегрирован с инструментом для управления качеством IBM Rational Quality Manager, а также позволяет повторно использовать модульные тесты, разработанные в средствах JUnit, NUnit, AUnit и CppUnit.

Кроме этого Gears интегрирован со следующими средствами конфигурационного: Serena Dimensions CM, Perforce, IBM Rational ClearCase, IBM Rational Synergy, IBM Rational Team Concert, Subversion, CVS. Gers также позволяет настроить процессы сборки исходного кода в следующих build-системах: Make, gmake, nmake, ANT, Maven, MSBuild и Build Forge.

Фреймворк Gears — этот коммерческий продукт, разработанный компанией BigLever Software Inc (США). Первая версия Gears выпущена в 1999 году. Сегодня проект активно развивается.

Фреймворк Gears широко востребован коммерческих проектах компаний международного уровня. На официальном сайте компании [51] представлено семь product line-решений, три из которых вошли в список «Product Line Hall of Fame» [81] . Вот краткий перечень самых крупных проектов, разрабтанных с помощью Gears: семейство программных продуктов для военно-морского флота США (компания Lockheed Martin) [30], семейство военных-симуляторов для армии США (компания General Dynamics) [62], семейство встраиваемых программных систем для автомобильной промышлености (компания General Motors) [46], семейство контроллеров и систем взаимодействия для ветроэлектрических установок (компания Ikerlan/Alstom) [68], семейство интернациональных Web-сайтов аренды недвижимости (компания HomeAway) [61], семейство контроллеров для систем хранения информации на базе RAID-технологии (компания NetApp) [26], семейство программных продуктов управления финансовой деятельностью компаний (компания Salion) [32].

4. Анализ результатов. В табл. 1 представлены сравнительные характеристики рассмотренных нами инструментов. В столбце «Нотации» перечисляются средства спецификации вариативности — как для задания модели вариативности, так и для спецификации ограничений (последнее важно, поскольку одними модельны-

ми средствами крайне затруднительно отразить различные частные ситуации — они обычно плохо «ложатся» в модельные языки, то есть выражаются громоздко или вовсе невыразимы). Почти все инструменты предлагают для этого свои собственные языки, что косвенно свидетельствует о том, что данная область является пока экспериментальной, так как не устоялись основные языковые средства для моделирования вариативности[9].

В столбце «Типы поддерживаемых активов» приводится информация о тех повторно используемых активах, которые поддерживают инструменты. Это преимущественно программный код, многие продукты поддерживают также XML-спецификации. Разнообразных специфических активов поддерживается пока мало — такие продукты как TVL, SPLOT, FMP, например, вообще предназначены только для моделирования и не имеют средств связи с какими бы то ни было реальными активами разработки. Инструмент pure::variants поддерживает визуальные модели для продукта Enterprise Architect и требования для IBM Rational DOORS. Однако в целом рассмотренные нами продукты не поддерживают вариативность тестов, формальных моделей процессов, документов и др. активов, хотя, как следует из ряда отчётов о внедрении инженерии линеек продуктов в конкретные проекты, подобные средства на практике востребованы.

Столбец «Промышленное применение» включает в себя информации о промышленном использовании рассматриваемых инструментов. Из информации, собранной нами, с очевидностью следует, что почти все инструменты являются научными разработками и используются на практике эпизодически. Исключение составляет продукт pure::variants, который имеет серьёзные внедрения. К сожалению, отчёты о реализации серьёзных промышленных семейств программных продуктов — см., например, [58, 67] — не содержат упоминаний об использовании рассмотренных продуктов.

В столбце «Год начала и текущее состояние» указывается год, в котором проект стартовал, а также текущее состояние проекта. В столбце «Компания–разработчик» указывается наименование предприятия, которое разрабатывает продукт.

[9]Большинство инструментов являются университетскими разработками, а для учёных важно испытать и реализовать свои собственные научные идеи, которые концентрируются, главным образом, именно вокруг модельных языков — основного научного артефакта здесь.

Отметим также и тот факт, что рассмотренные инструменты поддерживают только управление структурными различиями индивидуальных продуктов семейства (variation in space), а решение задач управления вариативностью во времени оставляется инструментам конфигурационного управления.

5. Заключение. В данной работе сделан обзор следующих инструментов, реализующих управление вариативностью: TVL, SPLOT, FeatureIDE, XFeature, FMP, FeatureMapper, CVL Tool, PLUM, pure::variants, DocLine, Gears. Проведённое исследование свидетельствует о том, что в последние 10 лет средства управления вариативностью активно разрабатываются. Однако в целом в настоящий момент данная область является в большей степени экспериментальной, а инструменты, как правило, создаются в университетах или околоуниверситетских компаниях. И только два продукта — pure::variants в Германии и Gears в США — являются промышленными разработками и имеют серьёзные промышленные внедрения. С другой стороны, управление вариативностью активно используется в индустрии — см., например, отчёты о промышленных семействах программных продуктов в [58, 67] — и поддерживается либо в рамках дисциплины процесса (то есть практически без средств автоматизации), либо с помощью программных решений, которые создаются непосредственно в рамках инфраструктуры семейства (исключение составляет продут Gears, который единственный из исследуемого нами списка упоминается в этих отчётах). Возможно, что стандартизация языка описания вариативности и использование этих стандартов в промышленности, а также бо́льшая зрелость самих инструментов (в частности, большее внимание поддержке различных типов активов и различных средств ЖЦ разработки ПО) изменят текущее положение дел.

Таблица 1. Инструменты управления вариативностью

Название	Нотации	Поддерживаемые активы	Лицензия	Пром. применение	Год начала и тек.сост.	Компания–разраб.
TVL	С подоб. язык, КНФ-формулы	Не подд.	Неком., открытый код	Нет упомин.	2010, поддерж.	Ун-т Намюра

SPLOT	XML, граф. нот., КНФ-формулы	Не подд.	Неком., открытый код	Нет упомин.	2009, подержж.	Ун-т Уотерлу
FeatureIDE	Граф. нотация, КНФ-формулы	Прогр. код, XML-документы	Неком., открытый код	Неск. небольших проектов.	2004, подержж.	Магдеб. ун-т им. Отто фон Герике
XFeature	XML	XML-документы	Неком., без открытого кода	1 проект	2005, не подд.	ETH Zürich, комп. P&P Software
FMP	Граф. нотация, XPath 2.0	Не подд., только «чистое» моделир.	Неком., открытый код	Нет упоминаний	2004, не подд.	Ун-т Уотерлу
Feature Mapper	Граф. нотация	EMF-модели	Некомм., без открытого кода	Нет упоминаний	2008, подержж.	Дрезд. техни. ун-т
CVL Tool	Граф. нотация	XMI-модели	Некомм., без открытого кода	Нет упоминаний	2009, подержж.	Норвеж. ин-т технологий
PLUM	Граф. нотация, OCL 2.0	Произвольный текст	Некомм., без открытого кода	Исп. в средней по размерам комп.	2007, подержж.	Европ. ин-т програм. обеспеч.

pure:: variants	Граф. нотация, диалект Пролоа	Прогр. код, XML, плоск. текст, UML SysML модели, треб., Simulink- AUTOSAR-модели.	Комм., без открытого кода	Крупные комм. проекты	2004, поддерж.	Комп. pure-systems GmbH (Германия)
DocLine	Граф. нотация, XML	Польз. документация.	Неком., открытый код	1 проект	2006, подд.	СПб государств. ун-т
Gears	Граф. нотация, КНФ-формулы	Прогр. код, XML, UML SysML модели, треб., тесты	Комм., без открытого кода	Крупные комм. проекты	1999, поддерж.	Комп. BigLever Software Inc (США)

Список литературы

[1] Кияев, В.И., "О терминологии и требованиях международного стандарта качества разработки программного обеспечения SW CMM", *Системное программирование*, **1** (2005), 311—334.

[2] Кознов, Д.В., "Разработка и сопровождение DSM-решений на основе MSF", *Системное программирование*, **3**:1 (2008), 80–96.

[3] Кознов, Д.В., Романовский, К.Ю., "Автоматизированный рефакторинг документации семейств программных продуктов", *Системное программирование*, **4**:1 (2009), 128–150.

[4] Романовский, К.Ю., "Метод разработки документации семейств программных продуктов", *Системное программирование*, **2**:1 (2006), 191–218.

[5] Романовский, К.Ю., "Разработка повторно используемой документации семейства телефонных станций средствами технологии DocLine", *Вестник Санкт-Петербургского университета. Серия 10: Прикладная математика. Информатика. Процессы управления*, **2** (2009), 166–180.

[6] Сафонов, В.О., Муханов, Р.С., “Реализация и практическое применение системы Aspect.Net для академической версии .Net”, *Научно-технические ведомости Санкт-Петербургского государственного политехнического университета*, **103** (2010), 174–179.

[7] Смирнов, М.Н., Кознов, Д.В., Дорохов, В.А., Романовский, К.Ю. Программная среда WebMLDoc для автоматизированного отслеживания изменений в пользовательской документации Web-приложений, *Системное программирование*, **5**:1 (2010), 32—51.

[8] Смирнов, М.Н., Соколов, Н.Е., Романовский, К.Ю., “DocLineFM: среда разработки повторно используемой документации семейств программных продуктов на базе пакета Adobe FrameMaker”, *Системное программирование*, **6**:1 (2011), 80–98.

[9] Сорокин, А.Н., Кознов, Д.В., “Обзор проекта Eclipse Modeling Project”, *Системное программирование*, **5**:1 (2010), 6–31.

[10] Павлинов, А.А., Кознов, Д.В., Перегудов, А.Ф., Бугайченко, Д.Ю., Казакова, А.С., Чернятчик, Р.И., Иванов, А.Н., “О средствах разработки проблемно-ориентированных визуальных языков”, *Системное программирование*, **2**:1 (2006), 142–168.

[11] Попова, Т.Н., Кознов, Д.В., Тинова, А.А., Романовский, К.Ю, “Эволюция общих активов в семействе средств реинжиниринга программного обеспечения”, *Системное программирование*, **1** (2005), 184–198.

[12] Фуксман, А.Л., *Технологические аспекты создания программных систем*, М.: Статистика, 1979.

[13] Abbasi, E. K., Hubaux, A., Heymans, P., “An interactive multi-perspective toolset for non-linear product configuration processes”, Proceedings of the 15th International Software Product Line Conference, **2** (2011), 50:1–50:1.

[14] *Antenna project site*, http://antenna.sourceforge.net/.

[15] *A Framework for Software Product Line Practice Version 5.0*, http://www.sei.cmu.edu/productlines/frame_report/index.html.

[16] *AHEAD Tool Suite project site*, http://www.cs.utexas.edu/schwartz/ATS.html.

[17] *AspectJ project site*, http://www.eclipse.org/aspectj/.

[18] *ASSERT project site*, http://www.assert-project.net.

[19] Atkinson, C., Bayer, J., Bunse, C., Kamsties, E., Laitenberger, O., Laqua, R., Muthig, D., Paech, B., Wüst, J. and Zettel, J., *Component-based product line engineering with UML publ Addison-Wesley Longman Publishing Co., Inc.*, 2002.

[20] *AUTOSAR project site*, http://www.autosar.org/.

[21] Batory, D., “A tutorial on feature oriented programming and the AHEAD tool suite”, Proceedings of the 2005 international conference on

Generative and Transformational Techniques in Software Engineering, 2006, 3–35.

[22] Batory, D., Sarvela, J. N., Rauschmayer, A., "Scaling step-wise refinement", Proceedings of the 25th International Conference on Software Engineering, 2003, 187–197.

[23] Batory, D., Johnson, C., MacDonald, B., von Heeder, D., "Achieving extensibility through product-lines and domain-specific languages: a case study", *ACM Trans. Softw. Eng. Method.*, **11**:2 (2002), 191–214.

[24] Beuche, D., "Modeling and building software product lines with pure::variants", Proceedings of the 15th International Software Product Line Conference, **2** (2011), 46:1.

[25] Beuche, D., "Modeling and Building Software Product Lines with pure::variants", Proceedings of the 2008 12th International Software Product Line Conference, 2008, P. 358.

[26] *BigLever Software Case Study:Engenio*, Technical Report, 2005.

[27] Boucher, Q., Classen, A., Faber, P., Heymans, P., "Introducing TVL, a Text-based Feature Modelling Language", Proceedings of the Fourth International Workshop on Variability Modelling of Software-intensive Systems (VaMoS'10), Linz, Austria, January 27–29. University of Duisburg-Essen, 2010, 159–62.

[28] Cechticky, V., Pasetti, A., Rohlik, O., Schaufelberger, W., "Xml-based feature modelling", Software Reuse: Methods, Techniques and Tools: 8th International Conference, ICSR 2004, 2004, 5–9.

[29] Cetina, C., Haugen, Zhang, X., Fleurey, F., Pelechano, V., "Strategies for variability transformation at run-time", Proceedings of the 13th International Software Product Line Conference. Carnegie Mellon University, 2009, 61–70.

[30] Cezo, J., Krueger, C. W., "Use product line engineering to reduce the total costs required to create, deploy & maintain systems & software", *EE Times*, 2008, 23–32.

[31] Classen, A., Boucher, Q., Faber, P., Heymans, P., *The TVL Specification*, PReCISE Research Center, University of Namur, 2010.

[32] Clements, P. C., Northrop, L. M., *Salion, Inc. A Software Product Line Case Study.*, Technical Report, 2002.

[33] Czarnecki, K. Kim, C., "Cardinality-Based Feature Modeling and Constraintss: A Progress Report", OOPSLA Workshop on Software Factories, 2005, 1–9.

[34] Czarnecki, K., Helsen, S., Eisenecker, U., "Staged configuration using feature models", Software Product Lines: Third International Conference, SPLC 2004, 2004, 266–283.

[35] *CVL project site*, http://variabilitymodeling.org.

[36] Davis, S. M., *Future Perfect*, Addison-Wesley, 1987.

[37] *DeltaJ project site*, http://deltaj.sourceforge.net/.

[38] Dziobek C., Loew, J., Przystas, W., Weiland, J., "Model Diversity and Variability — Handling of Functional Variants in Simulink-Models", Elektronik automotive, 2008.

[39] *Enterprise Architect project site*, http://www.sparxsystems.com/products/ea/index.html.

[40] Eriksson, M., Burstler, J., Borg, K., "The PLUSS approach: domain modeling with features, use cases and use case realizations", Proceedings of the 9th international conference on Software Product Lines, 2005, 33–44.

[41] Erofeev, S., "Applying SPL Techniques to Support Agile in the Large", FLEXI Project (Flexible Integration in Global Product Development), 2008, P. 9.

[42] *FeatureHouse project site*, http://www.fosd.de/fh.

[43] *FeatureIDE project site*, http://www.fosd.de/featureide.

[44] *FeatureMapper project site*, http://featuremapper.org/.

[45] *FeatureC++ project site*, http://wwwiti.cs.uni-magdeburg.de/iti_db/fcc/.

[46] Flores, R., Krueger, C. W., Clements, P. C., "Mega-scale product line engineering at General Motors", SPLC, **1** (2012), 259–268.

[47] *FMP project site*, http://gp.uwaterloo.ca/fmp.

[48] Frakes, W., Prieto-Diaz, R., and Fox, C., "DARE: Domain analysis and reuse environment", *Ann. Softw. Eng.*, **5** (1998), 125–141.

[49] Gacek, C., Anastasopoules, M., "Implementing product line variabilities", *SIGSOFT Softw. Eng. Notes*, **26**:3 (2001), 109–117.

[50] Gomaa, H., "Designing Software Product Lines with the Unified Modeling Language (UML)", Software Product Line Conference PLC, 2004, 317.

[51] *Gears project site*, http://www.biglever.com.

[52] Hamza, H. S., Martinez, J., and Alonso, C., "Introducing Product Line Architectures in the ERP Industry: Challenges and Lessons Learned", SPLC Workshops, 2010, 263–266.

[53] *IBM Rational DOORS project site*, http://www-01.ibm.com/software/awdtools/doors/.

[54] Jepsen, H. P., Dall, J. G., Beuche, D., "Minimally Invasive Migration to Software Product Lines", Proceedings of the 11th International Software Product Line Conference, 2007, 203–211.

[55] Kastner, C., Thum, T., Saake, G., Feigenspan, J., Leich, T., Wielgorz, F., and Apel, S., "FeatureIDE: A tool framework for feature-oriented software development", Proceedings of the 31st International Conference on Software Engineering, 2009, 611–614.

[56] Kang, K. C., Lee, J. and Donohoe, P., "Feature-oriented product line engineering", *IEEE Software*, **19**:4 (2002), 58–65.

[57] Kang, K. C., Sugumaran, V., Park, S., *Applied Software Product Line Engineering*, Auerbach Publications, 2009.

[58] Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E. and Peterson, A. S., *Feature-Oriented Domain Analysis (FODA) Feasibility Study*, Technical Report. CMU/SEI-90-TR-21. ESD-90-TR-222. Carnegie-Mellon University Software Engineering Institute, 1990.

[59] Kang, K. C., Kim, S., Lee, J., Kim, K., Shin, E. and Huh, M., "FORM: A feature-oriented reuse method with domain-specific reference architectures.", *Ann. Softw. Eng.*, **5** (1998), 143–168.

[60] Koznov D.V., Romanovsky K.Yu., "DocLine: a method for software product line documentation development", *Programming and Computer Software.*, **34**:4 (2008), 216–224.

[61] Krueger, C. W., Churchett, D., Buhrdorf, R., "HomeAway's Transition to Software Product Line Practice: Engineering and Business Results in 60 Days", SPLC, 2008, 297–306.

[62] Krueger, C. W., Clements, P. C., "Second generation systems and software product line engineering", SPLC, **2** (2012), 280.

[63] Krueger, C. W., Clements, P. C., "Systems and software product line engineering with BigLever software gears", SPLC, **2** (2012), 256—259.

[64] Krueger, C. W., "Variation Management for Software Production Lines", Proceedings of the Second International Conference on Software Product Lines, 37–48.

[65] Krueger, C. W., "Easing the Transition to Software Mass Customization", Revised Papers from the 4th International Workshop on Software Product-Family Engineering, 2002, 282–293.

[66] Leich, T., Apel, S., Marnitz, L., Saake, G., "Tool support for feature-oriented software development: featureIDE: an Eclipse-based approach", Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange, 2005, 55–59.

[67] Linden, F. J., Schmid, K., and Rommes, E., *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*, Springer Publishing Company, Inc., 2010.

[68] Lombardi, C., "IBM partners on "smart" wind system", *CNET News*, 2010, 12–17.

[69] Martinez, J., Vicedo, J., "PLUM (Product Line Unified Modeller)", Eclipse Summit Europe, 2008, 16–20.

[70] Martinez, J., "User Stories for Agile Product Line Engineering Tooling", FLEXI Project (Flexible Integration in Global Product Development), 2009, P. 13.

[71] Matsumoto, Y., "A Software Factory, An Overall Approach to Software Production", *Software Reusability*, 1987, 155–178.

[72] McIlroy, M. D., "Mass-produced software components", Proc. NATO Conf. on Software Engineering, Garmisch, Germany, 1968.

[73] Mendonca, M., Branco, M., Cowan, D., "S.P.L.O.T.: software product lines online tools", Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications, 2009, 761–762.

[74] *Munge project site*, http://weblogs.java.net/blog/tball/archive/2006/09/munge_swings_se.html.

[75] Neighbors, J. M., *Software reusability: vol. 1, concepts and models*, ACM Press, 1989.

[76] Parnas, D. L., "Classics in software engineering", 1979, 139 –150.

[77] Parnas, D. L., "On the criteria to be used in decomposing systems into modules", *Commun. ACM*, **15**:12 (1972), 1053–1058.

[78] *PervML project site*, http://www.pros.upv.es/labs/index.php?option=com_content&view=article&id=40&Itemid=77.

[79] *PLUM project site*, http://www.esi.es/plum/index.php.

[80] Pohl, K., Bckle, G. and van der Linden, F. J., *Software Product Line Engineering: Foundations, Principles and Techniques*, Springer Publishing Company, Incorporated, 2010.

[81] *Product line hall of fame*, http://splc.net/fame.html.

[82] *pure::variants project site*, http://www.pure-systems.com.

[83] *Simulink project site*, http://www.mathworks.com/products/simulink/?s_cid=wiki_simulink_8.

[84] *SPLOT project site*, http://www.splot-research.org/.

[85] *Text-based Variability Language project site*, http://www.info.fundp.ac.be/ acs/tvl/.

[86] Wende, C., Heidenreich, F., Dresden, T. U., "A Model-based Product-Line for Scalable Ontology Languages", Proceedings of the 1st International Workshop on Model-Driven Product Line Engineering., 2009, 15–24.

[87] *XFeature project site*, http://www.pnp-software.com/XFeature/.

[88] *XML Path Language (XPath) 2.0 (Second Edition) W3C Recommendation*, http://www.w3.org/TR/xpath20/.

[89] *YAWL Process Editor project site*, http://www.yawlfoundation.org/.

Дмитрий Владимирович Кознов — к.ф.-м.н.; доцент кафедры системного программирования Санкт-Петербургского государственного университета (СПбГУ). Область научных интересов: программная инженерия, модельно-ориентированная разработка ПО, семейства программных продуктов, разработка технической документации. Число научных публикаций — 41.

dkoznov@yandex.ru; СПбГУ, к. 4114, Университетский пр. д. 28 Санкт-Петербург, 198504, РФ; р.т. +7(812)428-71-09, факс +7(812)428-71-09.

Dmitrij V. Koznov — PhD in Software Engineering; Associate Prof of Software Engineering Chair, Mathematics and Mechanics Faculty, Saint-Petersburg State University. Research topics: software engineering, model-based software development, product lines, software documentation. Number of publications — 41.
dkoznov@yandex.ru; SPbSU, room 4114, Universitetsy pr. 28, St.-Petersburg, 198504, RF; phone +7(812)428-71-09, fax+7(812)428-71-09.

Иван Александрович Новицкий — аспирант кафедры системного программирования Санкт-Петербургского государственного университета (СПбГУ). Область научных интересов: линейки программных продуктов, управление вариативностью, Java, Eclipse. Число научных публикаций — 1.
ivannovitskii@gmail.com; СПбГУ, Университетский пр. д. 28 Санкт-Петербург, 198504, РФ; р.т. +7(812)428-71-09, факс +7(812)428-71-09.

Ivan A. Novitskii — PhD student of Software Engineering Chair, Mathematics and Mechanics Faculty, Saint-Petersburg State University. Research topics: product lines, variability management, Java, Eclipse. Number of publications — 1.
ivannovitskii@gmail.com; SPbSU, Universitetsy pr. 28, St.-Petersburg, 198504, RF; phone +7(812)428-71-09, fax+7(812)428-71-09.

Михаил Николаевич Смирнов — старший преподаватель кафедры системного программирования Санкт-Петербургского государственного университета (СПбГУ). Область научных интересов: линейки программных продуктов, управление вариативностью, управление требованиями. Число научных публикаций — 5. smnsmn1979@gmail.com; СПбГУ, Университетский пр. д. 28 Санкт-Петербург, 198504, РФ; р.т. +7(812)428-71-09, факс +7(812)428-71-09.

Michail N. Smirnov — Assistant Professor of Software Engineering Chair, Mathematics and Mechanics Faculty, Saint-Petersburg State University. Research topics: product lines, variability management, requirement engineering. Number of publications — 5. smnsmn1979@gmail.com; SPbSU, Universitetsy pr. 28, St.-Petersburg, 198504, RF; phone +7(812)428-71-09, fax+7(812)428-71-09.

Поддержка исследований. Работа выполнена при частичной финансовой поддержке гранта РФФИ 11-01-00622-а.

Рекомендовано лабораторий и систем программирования, заведующий лабораторией Шкиртиль В.И., к.т.н.

Статья поступила в редакцию 30.01.2013.

РЕФЕРАТ

Кознов Д.Х., Новицкий И.Х., Смирнов М.Х. **Инструменты для управления вариативностью: готовность к промышленному применению**

Разработка ПО методом организации семейства программных продуктов – популярный подход к промышленной разработке, основывающийся на повторном использовании различных активов и едином процессе разработки. Управление вариативностью нацелено на управление повторно используемыми активами и является одной из ключевых активностей при разработке семейств. В данной работе сделан обзор следующих инструментов, реализующих управление вариативностью: TVL, SPLOT, FeatureIDE, XFeature, FMP, FeatureMapper, CVL Tool, PLUM, pure::variants, DocLine, Gears. Основная цель этого обзора – выяснить, насколько эти средства готовы для промышленного применения и реально используются в разработках. Проведённое исследование свидетельствует о том, что в последние 10 лет средства управления вариативностью активно разрабатываются. Однако в целом в настоящий момент данная область является в большей степени экспериментальной, а инструменты, как правило, создаются в университетах или околоуниверситетских компаниях. И только два продукта — pure::variants в Германии и Gears в США — являются промышленными разработками и имеют серьёзные промышленные внедрения. С другой стороны, управление вариативностью активно используется в индустрии, что отражается в открытых отчётах, публикуемых product line сообществом, но поддерживается либо в рамках дисциплины процесса (то есть практически без средств автоматизации), либо с помощью программных решений, которые создаются непосредственно в рамках инфраструктуры семейства (исключение составляет продут Gears, который единственный из исследуемого нами списка упоминается в этих отчётах). Возможно, что стандартизация языка описания вариативности и использование этих стандартов в промышленности, а также большая зрелость самих инструментов (в частности, большее внимание поддержке различных типов активов и различных средств ЖЦ разработки ПО) изменят текущее положение дел.

SUMMARY

Koznov D., Novitsky I., Smirnov M. **Variability Management Tools: readiness for industrial use**

Software line engineering family is a popular approach to industrial development which is based on reusable assets and a single development process. Variability management is aimed to operate with reusable assets and is one of the key activities in family development. This paper reviews the following variability management tools: TVL, SPLOT, FeatureIDE, XFeature, FMP, FeatureMapper, CVL Tool, PLUM, pure::variants, DocLine, Gears. The main goal of the paper is to find out whether these tools are ready for industrial usage and are actually used in the development. The research done shows that over the past 10 years variability management tools have been developed rapidly. However this area is more experimental at the moment, and the tools are, in general, created in universities. And only two products: pure::variants (Germany) and Gears (USA) are industrial products and have significant commercialization. On the other hand, variability management is widely used in industry, which is reflected in the reports, published by product line community. But it is maintained in the process discipline (that means with little or no automation) or with help of program solutions that are created directly in the family infrastructure. The only exception is Gears, which is the only tool mentioned in the records. Perhaps the variability language standardization and the use of these standards in the industry with improvement of the tools (in particular, a greater attention for supporting different asset types and different tools of software development lifecycle) will change the current situation.