УДК 614.2+574/578+004.38

А.А. Литвинов, Д.Л. Грузин, А.С. Вякилов

ОПТИМИЗАЦИЯ ПРОЦЕССА РАЗРАБОТКИ МНОГОСЛОЙНЫХ ПРОГРАМНЫХ ПРИЛОЖЕНИЙ

В работе рассмотрен вариант модельно-ориентированного подхода, упрощающий создание и сопровождение информационной системы. На базе фреймовой модели производится генерация ряда компонентов, производится интерпретация модели в ходе работы системы, решаются вопросы, связанные с управлением конфигурации системы.
Ключевые слова: онтология, генерация кода, интерпретация, MDA.

Актуальность темы. На текущий момент, одновременно с усложнением программных систем, перед разработчиками стоит задача создания качественного многоуровневого программного обеспечения (ПО) в кратчайшие сроки с одновременным уменьшением затрат, обеспечением высокого коэффициента ROI (return of investments), быстрого внедрения, адаптации ПО под широкий класс задач из смежных областей. Несмотря на многообразие различных подходов, можно выделить следующие основные черты решения данных задач: стандартизация процессов и компонентов, на базе выделения эффективных решений и обеспечение их повторного использования; управление на базе тщательного планирования, мониторинга, предсказания и обработки рисков, учитывая опыт предыдущих разработок; создание новых метафор и повышение уровня абстракции описания задачи с исключением деталей низкоуровневой реализации; использование технологий, позволяющих обеспечить построение эффективных решений. При этом исключительную роль играют активности по выявлению, обобщению, документированию, внедрению, мониторингу и оптимизации лучших практик организации, что соответствует введению управления процессами и отличают предприятия третьего уровня зрелости (defined level) согласно модели CMMI (capability maturity model integration). Стандарты в данном случае являются ценными активами (asset) предприятия, формирующими основу стратегии достижения его устойчивой работы [1], которая заключается в эффектив-

ном планировании активностей и ресурсов; прогнозировании и управлении рисками; повторному использованию уже отработанных, качественных процессов и компонентов. Вопрос формы описания и представления информации о компоненте, ее полезность для разработки подобных компонентов и адаптации компонента под другие условия являются актуальным вопросом.

Анализ последних публикаций. Одним из вариантов решения данных проблем является представление высокоуровневого, человекоориентированного описания компонента с возможностью его дальнейшей трансформации в вид приемлемый для выполнения вычислительной системой, либо существенно облегчающий процесс разработки.

В качестве эффективных подходов по оптимизации разработки можно выделить следующие: полная или частичная генерация кода путем трансформации некоторого более абстрактного описания задачи (заданного в декларативной форме) в формализованную форму ЯП (язык программирования) [2-4]; интерпретация некоторой модели, описывающей систему [5-9]. Оба эти подхода базируются на повышении абстракции описания задачи и создании механизмов ее трансформации. Также важную роль играют проекты, направленные на создание ценных активов, выявление и обеспечение быстрого поиска необходимых решений, что соответствует основным задачам уровня 3 подхода CMMI. Для этих целей используется механизм баз знаний онтологий (семантических сетей) [6], концепты которой проходят путь строгой формализации. Самым высоким уровнем абстракции описания задачи следует считать формализованные функциональные требования или прецеденты (use-cases) системы, основу которых составляют сценарии взаимодействия «актер-система» (базовый и альтернативный), при этом выделяется три уровня описания прецедентов (бизнес, система, компонент), а формализация производится на базе шаблонов и глоссария [10]. Следует отметить, что прецеденты являются основой не только разработки, но также и выделения требований, заданий и планирования, различных видов тестирования и документации. В литературе очень хорошо освещены вопросы описания прецедентов уровня системы, но по поводу описания прецедентов уровня компонентов работ очень немного [11]. В конечном итоге прецеденты компонентного уровня выражаются в требованиях (features

and requirements) и ряде диаграмм UML, теряя при этом последовательность, полноту и гибкость описания. Важно отметить, что при переводе прецедента с уровня системы на уровень компонентов важную роль играет архитектура системы.

Постановка задачи. Как правило, для реализации типовой функции-прецедента уровня системы, описанной базовым и альтернативным сценариями, необходимо затронуть большее количество уровней системы. На рис. 1 показана связь прецедента с типовым набором основных и вспомогательных уровней системы.

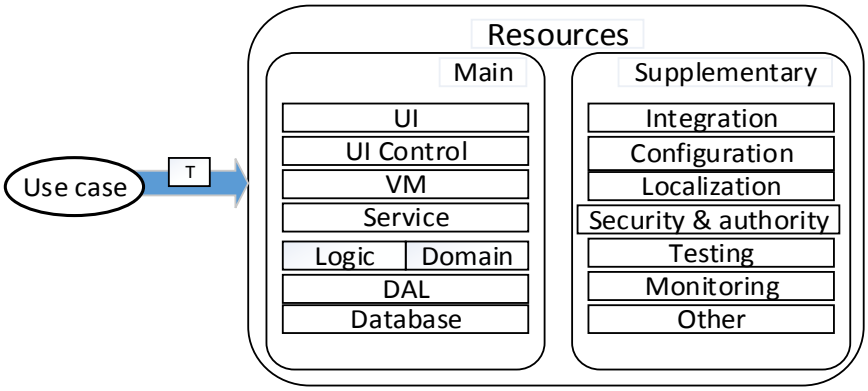

Рисунок 1 – Связь прецедента с ресурсами, необходимыми его реализации

Число уровней, с одной стороны, делает систему более гибкой и устойчивой к изменениям, с другой – усложняет ее сопровождение. Путь перехода на чистые языки высокой абстракции чреват потерей гибкости. Предлагаемые подходы описания моделей не дают ожидаемого эффекта в ключе облегчения построения системы с использованием средств генерации/интерпретации: нет возможности полноценной генерации логики, не учитывается полный спектр ресурсов, необходимых для реализации функции. Это происходит из-за ограничений средств описания предметной области, которые базируются на графической модели представления знаний. Таким образом, основное внимание в работе сфокусировано на задаче поиска упрощенного, высокоуровневого, полного описания системы с возможностью полноценной трансформации этого описания в форму, позволяющих полностью или частично реализовать (путем генерации/интерпретации) ресурсы, необходимые для реализации поставленной задачи.

В качестве основы используется фреймовая парадигма [12] и идея послойного описания системы согласно MDA (Model driven architecture) [2-4]. Подход MDA базируется на определении системы

на трех базовых уровнях: бизнес-процесс (CIM – Computation Independent Model), система (PIM – Platform Independent Model), компонент (PSM – Platform Specific Model). В качестве основного языка описания моделей используется UML (unified modelling language). Для преобразования моделей одного уровня в другой используются трансформации, благодаря которым модели уровня PIM, отвечающих за архитектуру системы, могут быть преобразованы в модели уровня PSM, описывающие структуру и поведение компонентов (база данных, бизнес логика, сервисы). Основными проблемами, которые стоят перед разработчиками являются: формальное описание бизнес-процессов на уровне CIM, трансформация и отслеживание зависимостей «CIM-PIM», представление поведенческой составляющей и полнота генерации кода.

Основная часть. Информационная система может быть представлена в виде множества документов-фреймов, отвечающих за реализацию той или иной функции (напр., визуальное программирование, 1С). Документы-фреймы могут быть не визуальными (например, справочники системы), но связаны с реализацией одного или нескольких прецедентов уровня системы. В качестве аналогии можно привести подход ICONIX [13], согласно которому создание storyboard является первым шагом и объединяющим началом последующей разработки: формирования и детализации функциональных требований, описание домена.

Документы включают в себя множество блоков-слотов, которые являются независимыми повторяющимися функциональными блоками, но отвечают не только за представление данных, реализацию функций, но и за любые ресурсы, которые необходимы для решения задачи, стоящей перед документом-фреймом. Для описания блоков-слотов используется механизм «граней» (facet), отличающий фреймовые системы от объектно-ориентированного подхода. Грани дают возможность полно описать блоки-слоты, что создает базу для полноценной трансформации модели. Важной особенностью является то, что один и тот же блок может входить в состав разных документов, что создает основу для повторного использования. Диаграмма на рис. 2 показывает отношения между документами, блоками и гранями. Важной особенностью является связь документов с типами пользователей, чем обеспечивается гибкость управления конфигурациями приложения. В более сложном ва-

рианте могут быть введены дополнительные связи «тип пользователя – блок документа» и «тип пользователя – грань блока документа».

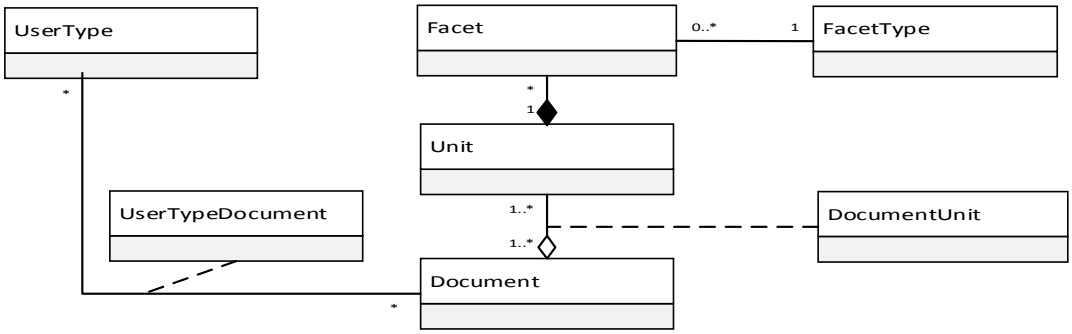

Рисунок 2 – Схема модели представления метаинформации

Описание системы производится на уровнях CIM и PIM, с последующей генерацией/интерпретацией классов уровня PSM. В качестве модели CIM нами предлагается использовать фреймовое описание знаний о решаемой задаче с указанием ресурсов, необходимых для ее решения без указания специфики (сборок, запросов, исключений). Можно также предположить возможность преобразования формализованных сценариев прецедентов в предлагаемое фреймовое представление, что выходит за рамки данной работы.

Например, документ редактирования справочников государств описывается таким образом.

```
document:stateEditor
{
    units:
    stateCollection:itemCollection;
  …
}
unit:stateCollection
{
    facets:
    ref -> domain.state;
}
domain.state
{
    asm:Common.Infrastructure.Domain;
    cn:Common.Infrastructure.Domain.StateBase;
}
```

Суффикс "Editor" подразумевает перечень необходимых функций (CRUD), а основным ресурсом является коллекция государств, которая определяется соответствующим блоком. Блок не содержит специфической информации, а включает лишь ссылку на ресурс domain.state, который в свою очередь определяет объект уровня PIM

(эта ссылка описывает переход CIM2PIM). По содержимому ресурса domain.state определяется объект уровня PIM, содержащий все необходимые детали для последующей генерации объектов уровня PSM.

```
State<entity> (abstract) : VocabularyItem
{
  RegionCollection:Region[], nrmax=*;
    (ns) XMIS.Core.Domain.Address; (asm) XMIS.Core.Domain;
}
Region<entity> (abstract) : VocabularyItem
{   …        }
```

Взаимосвязь моделей с реализацией системы (множеством ресурсов, обеспечивающих работу системы) наглядно представлена на рис.3.

Так, на базе модели PIM производится генерация объектов, связанных с архитектурой приложения. Не только стандартных SQL и доменных классов, но и ряда вспомогательных, продиктованных архитектурой, ресурсов, связанных с пользовательским интерфейсом, доступом к данным, интерфейсно-ориентированным программированием. На основании метаинформации, представленной в виде «документ блок - грань» производится динамическое создание классов, отвечающих за бизнес логику.

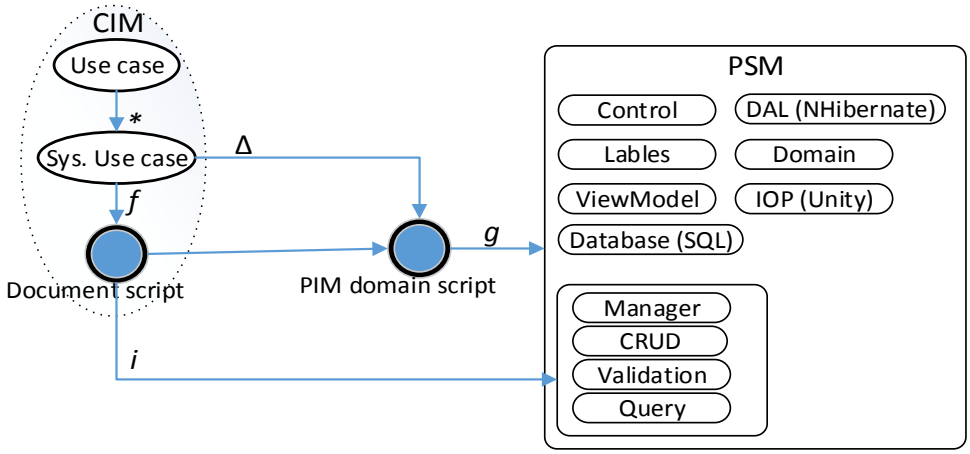

Рисунок 3 – Взаимосвязь моделей с реализацией системы

Вывод. В статье представлена интерпретация модельно-ориентированного подхода, упрощающего создание и сопровождение информационной системы, основой которого является модель, описанная с использованием фреймового подхода представления знаний. На базе описанной модели производится генерация ряда компонентов, что существенно упрощает процесс разработки многоуровневых приложений, а также производится интерпретация модели в ходе работы

 ISSN 1562-9945

системы, что исключает необходимость построения и тестирования дополнительных классов. Рассмотрены также вопросы управления конфигурацией системы. Данный подход был опробован на построении ряда типовых проектов и показал эффективные результаты.

ЛИТЕРАТУРА

1. Mary Beth Chrissis. CMMI® for Development Guidelines for Process Integration and Product Improvement, Addison-Wesley Professional; 3 edition (March 20, 2011). – 688 p.
2. David S. Frankel. Model Driven Architecture: Applying MDA to Enterprise Computing. John Wiley & Sons, Inc. New York, NY, USA. 2003. – 352 p.
3. Kevin Lano. Advanced Systems Design with Java, UML and MDA. Kings College, London, UK. 2005. – 416 p.
4. David S. Frankel, John Parodi. The MDA Journal: Model driven architecture Straight from the Masters. Meghan-Kiffer Press. 2004. – 219p.
5. Yonggang Zhang. An ontology-based program comprehension model. A Thesis in the Department of Computer Science and Software Engineering. Presented in Partial Fulfillment of the Requirements for the Degree of Doctor of philosophy at Concordia University Montreal, Quebec, Canada. 2007. – 207 p.
6. Andrej Bachmann, Wolfgang Hesse etc. OBSE – an approach to Ontology-based Software Engineering in the practice. Philipps-University Marburg University of Klagenfurt, Hans-Meerwein-Strasse UniversitдtsstraЯe 65 - 67 35032 Marburg 9020 Klagenfurt. 2007. – 14 p.
7. T. Gruber, S. Vemuri, and J. Rice. Model-Based Virtual Document Generation. International Journal of Human-Computer Studies. 1997. –20 p.
8. Hesse, W. Ontologies in the software engineering process. Berlin, Germany. GITO–Verlag, 2005. – 13 p.
9. Falbo, Guizzardi, et al. An Ontological Approach to Domain Engineering. Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering, 2002. – 8 p.
10. Cockburn A. Writing Effective Use Cases. Addison-Wesley Professional. 2000. – 304 p.
11. Blake M.B., Cleary K., Ranjan S., Ibбcez L., Gary K. Use case-driven component specification: a medical applications perspective to product line development. Proceedings of the 2005 ACM Symposium on Applied Computing (SAC), ACM, :1470-1477, Jan-2005.
12. Марвин Мински. Фреймы для представления знаний. Издательство: «Мир». 1979. – 152 с.
13. Doug Rosenberg, Matt Stephens. Use Case Driven Object Modeling with UML: Theory and Practice. Apress (January 10, 2007). – 438 p.