3. Башарин А. В., Новиков В. А., Соколовский Г. Г. Управление электроприводами: Учебное пособие для вузов. Ленинград, Издательство «Энергоиздат», Ленинградское отделение, 1982.

4. Бажинов А. В., Двадненко В. Я. Электропривод для конверсионного гибридного автомобиля. Журнал «Автомобильный транспорт» (Харьков ХНАДУ), 2012.

5. Тяговый электропривод в гибридных транспортных средствах. Станислав Флоренцев, ген. директор «Русэлпром-электропривод»; Дмитрий Изосимов, зам. ген. директора по науке, «Русэлпром-электропривод», www.russianelectronics.ru/developer-r/review/40498/doc/47905/

Плахотній О.В.

магістр,

Черкаський державний технологічний університет

ДОСЛІДЖЕННЯ МЕТОДУ РОЗРОБКИ МОДЕЛІ ІНФОРМАЦІЙНОЇ АНАЛІТИЧНОЇ СИСТЕМИ

Останні п’ятдесят років дослідники й розробники програмного забезпечення створюють абстракції, що допомагають їм програмувати в термінах сутностей свого проекту, а не комп’ютерного середовища, що використовується ними. Створювані абстракції захищають розробників від складнощів середовища розробки. Спочатку ці абстракції включали технології мов програмування й операційні системи. Ранні мови програмування захищали розробників від складнощів програмування в машинних кодах, а ранні операційні системи – від складнощів програмування на рівні апаратури. Але, хоча вони й підвищували рівень абстракції, вони явно були «орієнтованими на обчислення», забезпечуючи абстракції простору рішень (тобто абстракції комп’ютерних технологій), а не абстракції, що дозволяють вести розробку в термінах предметної області.

В останні роки досягнення в області мов програмування й платформ призвели до підвищення рівня абстракцій, доступних для розробників, частково вирішивши один із недоліків підходу CASE. Сьогодні розробники зазвичай використовують виразніші об’єктно-орієнтовані мови (зокрема, C++, Java і C#), а не Фортран або Сі. Повторно використовувані бібліотеки класів і платформи підтримки програм мінімізують потребу у винаході загальних та спеціалізованих сервісів – транзакцій, сповіщення про події, безпеки, розподіленого управління ресурсами і так далі. Проте проблеми залишаються. У центрі цих проблем – складність платформ, яка росте швидше за здатність мов загального призначення її маскувати. Популярні платформи проміжного рівня J2EE, Dot Net і CORBA містять тисячі класів і методів з багатьма складними взаємозв’язками і підступними побічними ефектами, що вимагає значних зусиль при програмуванні і ретельної настройки. Більш того, оскільки ці платформи швидко розвиваються, розробники витрачають багато сил на перенесення коду програм. Крім того, код більшості програм як і раніше пишеться на мовах програмування третього покоління; значних зусиль вимагає виконання інтеграційних дій (зокрема, розгортання, конфігурації й підтримки якості системи). Так, на Java або C# важко написати коректний програмний код розподіленої системи із сотнями тисяч взаємозалежних компонентів. Ситуацію не рятує навіть використання описів розгортання програмних систем на мові XML із-за семантичного розриву між метою розробки, і втіленням цієї мети в тисячах рядків вручну написаного XML-коду,

синтаксис якого не має відношення, ні до семантики предметної області, ні до мети розробки.

Із-за цих проблем програмна індустрія наближається до межі допустимої складності. У той же час платформні технології нового покоління стають настільки складними, що програмісти, роками опановуючи API платформи, шаблони використання, часто виявляються знайомими тільки із частиною можливостей платформи. Фрагментарність проекту ускладнює розуміння того, які частини програми є чутливими до побічних ефектів, що виникають при зміні вимог замовників або середовища розробки. Часто це змушує розробників приймати неоптимальні рішення, дублюючи частини коду, порушуючи ключові архітектурні принципи, ускладнюючи розвиток системи й забезпечення необхідної якості.

Багатообіцяючим підходом, направленим на рішення цих проблем, є інженерія, керована моделями (Model-Driven Engineering, MDE).

Найбільш відомою організацією яка розробляє і супроводжує стандарти в сфері модельно-орієнтованого підходу є Object Management Group (OMG). Запропонований OMG підхід має назву Model-Driven Architecture (MDA). Одним з головних намірів MDA є відділення логіки предметної області від конкретної технології реалізації, дозволяючи цим двом частинам змінюватись незалежно одна від одної.

Для позначення моделей що вміщують логіку предметної області (незалежних від платформи моделей) в MDA використовується термін PIM (Platform Independent Model).

Модель незалежна від платформи (PIM) – це модель побудована на високому рівні абстракції, незалежна від будь-якої технології реалізації.

PIM моделюється з точки зору предметної області.

Не дивлячись на те що PIM є достатньо детальною, її ще не використовують як програмний продукт. Другим кроком є перетворення PIM в платформно-залежну модель – PSM (platform-specific model). Це перетворення відбувається за допомогою стандартних інструментів трансформацій. PSM додатково ще містить елементи характерні для конкретної технології реалізації.

Одна PIM трансформується, зазвичай, не в одну PSM, а в декілька. Так, з одної PIM можна одночасно взяти інформацію для побудови PSM БД та PSM програми яка б працювала з БД.

Наступним кроком є генерація коду програми з платформно-залежної моделі. Після цього, з кодом проводиться оптимізація, доопрацювання де це потрібно, звична компіляція, зборка та налагодження системи.

Отже, можна визначити, механізми роботи з моделями в MDA є досить зручними для побудови прототипу різних систем та їх реалізації в предметній області.

Список використаних джерел:

1 http://www-adele.imag.fr/ ~jmfavre – Jean-Marie Favre Megamodelling and Etymology. A story of Words: from MED to MDE via MODEL in five millenniums ADELE Team, LSR-IMAGUniversity of Grenoble, France.

2 Грибачев К Г Delphi и Model Driven Architecture. Разработка приложений баз данных. – СПб. Питер, 2004. – 348 с. ил.

3 Douglas C. Schmidt Model-Driven Engineering Vanderbilt University.

4 E. Seidewitz, “What Models Mean”, IEEE Software, September 2003.

5 Anneke Kleppe, Jos Warmer, Wim Bast, MDA Explained, The Model Driven Architecture: Practice and Promise, Addison-Wesley, 2003, ISBN 0–321–19442-X

6 ftp://ftp.research.microsoft.com/pub/tr/tr-2000–53. pdf – Bernstein, P.A., Levy, A.L., Pottinger, R.A. A Vision for Management of Complex Systems, MSR-TR-2000–53.