Tomasz GRATKOWSKI
UNIWERSYTET ZIELONOGÓRSKI WYDZIAŁ ELEKTROTECHNIKI, INFORMATYKI I TELEKOMUNIKACJI, INSTYTUT INFORMATYKI I ELEKTRONIKI,
ul. prof. Z. Szafrana 2, 65-246 Zielona Góra

Formalizacja zadań wielokrotnego użytku

Dr inż. Tomasz GRATKOWSKI

Uzyskał stopień doktora nauk technicznych w zakresie informatyka na Wydziale Informatyki i Zarządzania Politechniki Wrocławskiej. Autor jest adiunktem w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego oraz współpracuje z firmami branży informatycznej. Głównymi tematami zainteresowań naukowych są: projektowanie systemów informatycznych, metodyki projektowania używane w inżynierii oprogramowania oraz systemy wielowarstwowe i rozproszone.

e-mail: T.Gratkowski@iie.uz.zgora.pl

Streszczenie

Proces zarządzania projektem IT, wymusza na menadżerach projektu opracowanie listy zadań, które zawierają opis czynności do wykonania przez członków zespołu. Tworzenie listy zadań można wesprzeć gotowymi zadaniami wielokrotnego użytku. Poniższy artykuł przedstawia mechanizm formalizacji zadań wielokrotnego użytku, który pozwala wyszukiwać zadania według zadanych kryteriów.

Słowa kluczowe: metodyki procesu wytwórczego oprogramowania, wspomaganie dostosowywania procesu wytwórczego, formalizacja definicji zadań, Open Unified Process.

Formalization of reusable tasks

Abstract

The Project Manager (PM) leads a project planning [2]. The plan is described by Work Items List (WIL). WIL captures all scheduled work to be performed within the project. All requirements, defect reports, and change requests have to be mapped to a Work Item (WI). Currently this process is made manually by PM. In this paper there is presented the first step toward building strategy of support PM to create WIL automatically. Before composing WIL automatically it has to be decided which formal methods should be used to describe WIL and WI. The proposed method is defined using a meta-model. In Table 2 there is considered a four-level meta-model based on MOF[6] as the meta-modelling language where: M3 - meta-meta-model (MOF), M2 - meta-model level (SPEM[5] + UML[7]), M1 - model level (classes describe elements of WIL and WI), M0 - model instance level (contains all the objects [real time instances or real world objects] of classifiers [classes] included in the model level). Fig. 2 shows ReusableTasksRepositoryFormalization (RTRF) package. RTRF is designed to reflect structures of WIL and WI. The main concept behind the RTRF package is to allow grouping of work items in sets and describe work items by set of attributes. The attribute definition is very adaptable and allow describing any kind of a work item. Therefore the presented structure allows describing WIL and WI used in different software development methodology (SDM). Fig. 3 shows the definition of attributes required to describe WIL and WI in the OpenUP[2] method. The presented formalization will be used to build Reusable Tasks Repository. The repository will store reusable tasks which can be used for different SDM. The next research will be concentrated on how to automatically find tasks in the repository. Parameters for searching will come from functional requirements stored in the use cases and the use case scenarios.

Keywords: software development methodology, support for adaptation of software development methodology, formalization of work items, Open Unified Process.

1. Wprowadzenie

Wzrost złożoności wytwarzanego oprogramowania komputerowego wpłynął na stosowania procesów wytwórczych oprogramowania (ang. software development process, SDP), dzięki czemu firmy informatyczne zredukowały problem przekraczania ustalonego budżetu oraz precyzyjniej mogą ustalać termin oddania systemu. Łatwiej również zarządza się ryzykiem niepowodzenia inwestycji informatycznej. Zdefiniowany szkielet SDP zawierający określoną strukturę, plan oraz kontrolujący wykonywane zadania nazywany jest metodyką wytwarzania oprogramowania (ang. software development methodology, SDM) [1].

Ze względu na swoją specyfikę, każdy zespół informatyczny lub projekt mogą wymagać stosowania innej SDM. Jednym z efektów prowadzonych prac badawczych na SDM, jest opracowanie iteracyjnej i przyrostowej metodyki, którą można zaadaptować na potrzeby każdego projektu. Jedną z metodyk spełniających powyższe kryteria jest metodyka Open Unified Process (OpenUP) [2], która zawiera najlepsze praktyki pochodzące zarówno z Rational Unified Process [4], jak i z metodyk zwinnych (ang. agile) [8].

Proces zarządzania procesem wytwórczym wymaga ciągłej adaptacji do zmieniających się wymagań i specyfiki projektu. Prowadzone przez autora badania zmierzają do opracowania metody wspomagania procesu adaptacji metodyki. Adaptacja metodyki jest tutaj rozumiana jako dopasowanie zadań wykonywanych w ramach metodyki, tak aby ich późniejsza realizacja pozwoliła na utworzenie systemu informatycznego zgodnego z wymaganiami funkcjonalnymi. Wprowadzenie wspomagania przyspieszy i uprości proces adaptacji metodyki. W publikacji [3] zaprezentowano ideę bazującą na wykorzystaniu zadań wielokrotnego użytku (szablonów zadań), które mogą być zaadaptowane w ramach różnych projektów IT.

W poniższym artykule zaprezentowano przyjęty sposób formalizacji zadań wielokrotnego użytku (ang. Reusable Tasks, RT) przechowywanych w repozytorium zadań wielokrotnego użytku (ang. Reusable Tasks Repository, RTR). Formalizacja ma na celu, opisanie RT w taki sposób, aby możliwe było ich wielokrotne użycie na potrzeby różnych projektów oraz aby możliwe było ich wyszukiwanie i dopasowywanie do wymagań tworzonego systemu na podstawie niezbędnych kryteriów.

2. Metodyka OpenUP

Metodyka OpenUP oparta jest na czterech głównych zasadach:

1) współpraca, w celu lepszego działania i zrozumienia w ramach zespołu;
2) równoważenie sprzecznych priorytetów, w celu maksymalizacji korzyści dla odbiorcy systemu;
3) jak najszybsza koncentracja na architekturze systemu, w celu minimalizacji ryzyka oraz organizacji fazy wytwarzania;
4) rozwój systemu w ramach małych częstych iteracji na podstawie zbieranych opinii od odbiorcy systemu.

Powyższe zasady odpowiadają regułom zaproponowanym w ramach manifestu Agile (tabela 1).

Tab. 1. Zestawienie zasad OpenUP z regułami manifestu Agile
Tab. 1. Summary of OpenUP and Agile rules

Zasady OpenUP	Reguły manifestu Agile
1	ważniejsi są ludzie i ich wzajemne interakcje (współdziałanie) ponad procedury i narzędzia
2	należy przedkładać współpracę z klientem nad negocjację umów
3	istotniejsze jest działające oprogramowanie nad wyczerpującą dokumentację
4	należy reagować na zmiany niż realizować wcześniej założony planu

OpenUP jest zorganizowana w dwóch różnych skorelowanych obszarach: zawartości metody oraz zawartości procesu. Zawartość metody definiuje elementy metodyki (role, zadania, artefakty oraz

porady), bez wskazywania jak zostaną użyte. Zawartość procesu składa się z szablonów zastosowania poszczególnych elementów metodyki. Wiele różnych projektów może być budowanych na podstawie tego samego zbioru elementów metody.

OpenUP opisuje organizację pracy na trzech poziomach: osobistym, zespołu oraz odbiorcy systemu. Na poziomie osobistym członek zespołu wykonuje mikro przyrosty, które reprezentują uzyskane wyniki kilkugodzinnej lub kilkudniowej pracy. Zespół realizuje kolejne zaplanowane iteracje w ramach, których dostarczane są odbiorcy systemu kolejne elementy. Na poziomie odbiorcy systemu OpenUP została podzielona na cztery fazy:

– rozpoczęcie (ang. inception);
– opracowanie (ang. elaboration);
– wytworzenie (ang. construction);
– przekazanie (ang. transition).

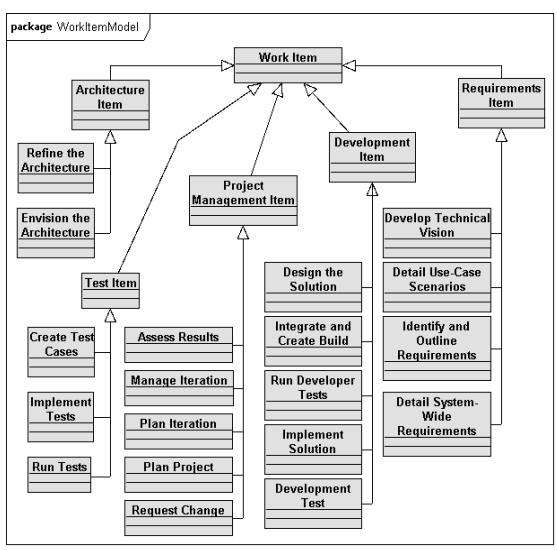
Tak zdefiniowany proces wytwórczy pozwala odbiorcy systemu na prowadzenie nadzoru nad finansami, zakresem oraz ryzykiem projektu.

Metodyka OpenUP posiada również zdefiniowane role członków zespołu, którzy wykonują składające się z kroków zadania, produkując artefakty. Zadania są pogrupowane w dyscypliny, natomiast artefakty w dziedziny.

W ramach fazy „rozpoczęcie" menadżer projektu definiuje plan projektu, natomiast w trakcie trwania projektu odpowiedzialny jest za tworzenie, zarządzanie oraz kontrolowanie wykonania iteracji. Rola menadżera projektu jest kluczowa z punku widzenia wspomagania adaptacji metodyki, ponieważ omawiany mechanizm ma wesprzeć pracę zdefiniowaną w OpenUP w ramach tej roli.

3. Wybór mechanizmu formalizacji

Proces wytwórczy zgodny z OpenUP rozpoczyna się od wstępnego rozpoznania oczekiwań klienta. Najczęściej stosowaną techniką zdobywania wymagań od klienta jest modelowanie przypadków użycia (ang. use-case modeling). Uzyskane w ten sposób informacje, przypadki użycia wraz ze scenariuszami przypadków użycia, jak również ogólne wymagania systemu, zmiany oraz propozycje zmian systemu, wady i kierunki rozwoju, służą do opracowania listy zadań. Lista zawiera zbiór usystematyzowanych zadań niezbędnych do wykonania tworzonego projektu informatycznego. Przyjęto, iż proces tworzenie listy zadań będzie procesem iteracyjnym. W pierwszej iteracji zadania będą tworzone na podstawie informacji zawartych w scenariuszach przypadków użycia (ang. use-case specifications) oraz dostępnych szablonach zadań przechowywanych w repozytorium zadań wielokrotnego użytku. W przypadku klasycznego procesu tworzenia listy oraz zadań, proces ten wykonywany jest ręcznie. Prowadzone badania koncentrują się na próbie automatyzacji tego procesu.



Rys. 1. Zadania w ramach metodyki OpenUP, na podstawie [2]
Fig. 1. Work items from OpenUP, based on [2]

Metodyka OpenUP definiuje listę zadań możliwych do wykonania w ramach procesu wytwórczego. Na rysunku 1 przedstawiono diagram klas grupujący zadania. Podział zadań jest powiązany z dyscyplinami [2]. Praktyka pokazuje jednak, iż zbiór zadań nie jest ograniczony tylko do zaprezentowanych zadań w [2]. Szereg narzędzi wspomagających praktyczne wykorzystanie OpenUP pozwala menadżerom projektu na definiowanie własnych zadań. Dlatego przyjęty mechanizm formalizacji umożliwia opisanie dowolnego zadania.

Do formalizacji listy zadań oraz zadań wykorzystano standard MOF 2.0 [6], służący do definiowania gramatyki języków graficznych. Dzięki zastosowaniu MOF możliwe było zdefiniowanie elementów niezbędnych do opisania zadania oraz powiązań pomiędzy elementami.

W tabeli 1 przedstawiono zastosowaną czterowarstwową hierarchię meta-modeli. Warstwą M2 (meta-meta-modelem) jest UML oraz SPEM. Mata-modelem na warstwie M1 jest diagram klas definiujący poszczególne elementy potrzebne do opisu listy zadań oraz zadań. Warstwą M0 (implementacją) będzie konkretny przykład zadań wykorzystanych w ramach określonego projektu.

Tab. 2. Czterowarstwowa hierarchia meta-modelu dla przyjętej formalizacji listy zadań i zadań
Tab. 2. Four-level meta-model for formalization of work items list and work items

Warstwa	Opis	Użyte elementy
M3	meta-meta-model	MOF
M2	meta-model	UML+SPEM
M1	model	lista zadań i zadanie
M0	implementacja	instancje listy zadań i zadań dla konkretnego projektu

Lista zadań oraz zadanie są przykładem definicji produktu pracy opisanego bardzo ogólnie w specyfikacji SPEM [5]. Dlatego klasą bazową definiującą podstawową funkcjonalność będzie klasa WorkProductDefinition pochodząca z pakietu MethodContent [5]. W celu deklarowania powiązań pomiędzy elementami opisującymi listę zadań oraz zadania wykorzystano klasę WorkProduct-DefinitionRelationship [5] z tego samego pakietu. WorkProduct-DefinitionRelationship przechowuje informację o źródle (source) oraz o obiekcie (target) powiązania. Na rysunku 2 zaprezentowano meta-model, który umożliwia definiowanie listy zadań oraz dowolnych zadań. Poniżej zostały opisane poszczególne klasy z diagramu.

WorkItemList

Reprezentuje listę zadań. WorkItemList zawiera listę zadań workItems, w której każde zadanie jest unikalne. Klasa rozszerza klasę WorkProduct-Definition, dzięki czemu możliwe jest tworzenie powiązań pomiędzy listami zadań i zadaniami.

WorkItem

Reprezentuje zadanie. Klasa rozszerza klasę WorkProduct-Definition, dzięki czemu możliwe jest tworzenie powiązań pomiędzy listami zadań, zadaniami oraz atrybutami opisującymi zadanie.

Atribute

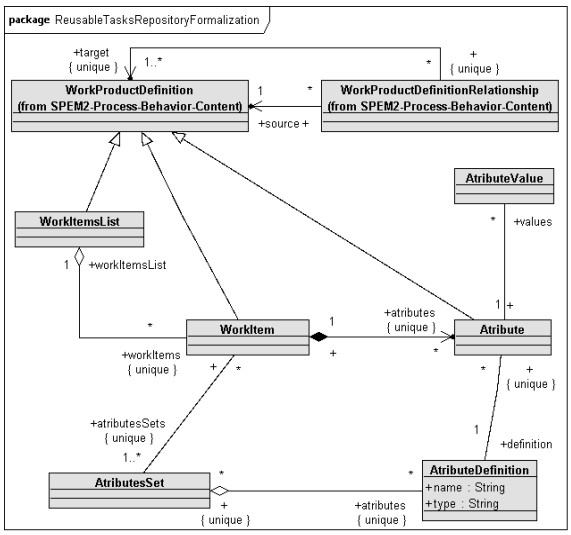
Klasa Atribute opisuje właściwości zadania. Atrybut przechowuje wartość lub wartości w values. Każdy atrybut jest zdefiniowany poprzez AtributeDefinion. Klasa rozszerza klasę WorkProduct-Definition, dzięki czemu możliwe jest tworzenie powiązań pomiędzy zadaniami i atrybutami.

AtributeDefintion

Klasa opisuje Atrybut oraz wprowadza ograniczenia dla jego wartości. AtributeDefintion definiuje nazwę (name) oraz typ (type) używane przez atrybut.

AtributesSet

AtributesSet grupuje atrybuty poprzez ich definicję (Atribute-Defintion). Klasa jest powiązana z WorkItem i definiuje, jaki zbiór atrybutów powinien opisać zadanie. Relacja atributesSets jest traktowana jako szablon deklarujący zbiór oczekiwanych atrybutów.



Rys. 2. Meta-model definiujący elementy listy zadań oraz zadań
Fig. 2. Meta-model for definition of elements of work items list and work items

4. Przykład wykorzystania przyjętej formalizacji

Poniższy rozdział zawiera przykładowe wykorzystanie zaprezentowanej formalizacji zadań wielokrotnego użytku. Meta-model zostanie zastosowany do pisania zadań wykorzystywanych w ramach metodyki OpenUP.

Zgodnie z wytycznymi OpenUP, każde z zadań powinno zawierać:

i) obligatoryjnie:
- nazwę i opis (ang. name and description);
- priorytet (ang. priority);
- oszacowanie rozmiar (ang. size estimate);
- stan (ang. state);
- odwołania (ang. references).

ii) opcjonalnie:
- data iteracji lub realizacji (ang. iteration or completion date);
- osoba przypisana (ang. assignee);
- pozostały oszacowany czas (ang. estimated effort remaining);
- godziny przepracowane (ang. hours worked).

Aby możliwe było opisanie przykładowego zadania, utworzono odpowiednie definicje atrybutów, które zostały pogrupowane w zbiory i przedstawione na rysunku 3. Definicja wartości atrybutów może być przygotowana dokładnie tak jak definicja atrybutów. W przykładnie umożliwiła opisanie przykładowej listy zadań z dokumentacji OpenUP zaprezentowanej na rysunku 4. Powszechnie listę zadań przedstawia się w formie arkusza kalkulacyjnego lub odpowiedniej struktury w bazie danych.

Obligatory attributes
- Name attribute definition
  - name = name
  - type = String
- Description attribute definition
  - name = description
  - type = String
- Priority attribute definition
  - name = priority
  - type = Integer
- Size estimate attribute definition
  - name = size estimate
  - type = Integer
- State attribute definition
  - name = state
  - type = String
- Relation definition
  - source = artefact
  - target = workItem

Optionally attributes
- Iteration attribute definition
  - name = iteration
  - type = Integer
- Assignee attribute definition
  - name = assignee
  - type = String
- Estimated effort remaining attribute definition
  - name = estimated effort
  - type = Integer
- Hours worked attribute definition
  - name = hours worked
  - type = Integer

Rys. 3. Tytuł rysunku w języku polskim
Fig. 3. Definition of attributes for OpenUP

Name / Description	Priority	Size estimate (Points)	State	Target iteration	Assig-ned To	Effort estimate left (hours)	Hours worked	Reference material
Produce draft vision	1	2	Closed	1		0	18	
Support basic order entry process	1	8	Closed	1	Lisa	0	56	<UC:Inventory Control>
Prototype UI			Closed	1	Ann	0	5	<User Interface Guidelines.pdf>
Do the design			Closed	1	Lisa, Ann, Johan	0	8	
Implement and test server portion			Closed	1	Ann	0	23	
Implement and test client portion			Closed	1	Johan	0	14	
Update end user documentation			Closed	1	Lisa	0	6	
Finalize the Vision	1	1	Resolved	2	Peter	2	12	<Vision.doc>
Complete the order entry process	2	8	Verified	2	Johan	2	66	<UC: Order Entry>
Confirmation page displays incorrect date	2		Closed	2	Johan	0	23	
Sort orders by name	2		Closed	2	Ann	0	18	
Sort orders by order number	3		Verified	2	Johan	2	25	
Play happy sound when order is complete	5		Closed (out of scope)					
Support simple inventory control	2	8	Assigned	3	Lisa	70	0	<UC:Inventory Control>
Prototype UI			Resolved	2	Ann	10	0	<User Interface Guidelines.pdf>
Do the design			Assigned	3	Lisa, Ann, Johan	12	0	
Implement and test server portion			Assigned	3	Ann	14	0	
Implement and test client portion			Assigned	3	Johan	28	0	
Update end user documentation			New	3	Lisa	6	0	
Allow disconnected use	5	100	New					<Long term vision.doc>
Produce demo for Supply Chain 2007 Conference	3	5	Assigned	3	Ann	18	2	
Edit end user documentation	2	5	Assigned	3	Lisa	65	11	

Rys. 4. Przykładowa lista zadań, źródło [2]
Fig. 4. Example of work items list, source [2]

5. Podsumowanie

W artykule zaprezentowano mechanizm formalizacji zadań wielokrotnego użytku w oparciu o czterowarstwowy meta-model bazujący na specyfikacji MOF. Zaprezentowano, w jaki sposób należy wykorzystać przyjętą formalizację do opisania zadań w ramach metodyki OpenUP. Przyjęty mechanizm charakteryzuje się dużą elastycznością zastosowań i może być zaadaptowany do opisu zadań z dowolnej metodyki. Dzięki temu możliwe będzie utworzenie repozytorium składującego zadania, które mogą być następnie dostosowywane do potrzeb dowolnej metodyki oraz narzędzi wspomagających prace menadżerów projektów. Idea repozytorium składującego zadania wspomagającego proces adaptacji metodyki jest nowatorskim rozwiązaniem opracowywany w ramach prowadzonych badań.

Zaproponowany mechanizm formalizacji jest początkowym etapem budowania strategii wspomagania procesu adaptacji procesu wytwórczego dla potrzeb projektu IT. Pierwszym wymiernym efektem powinien być system wspomagający wybór zadań wielokrotnego użytku na podstawie wymagań funkcjonalnych systemu.

6. Literatura

[1] The Centers for Medicare & Medicaid Services, Selecting a development approach, 27 Oct 2008.
[2] Introduction to OpenUP (Open Unified Process), Version 4 - openup_1.5.1.1_20101130, http://epf.eclipse.org/wikis/openup/.
[3] Gratkowski T.: Wspomaganie procesu definiowania zadań wykonywanych w ramach projektu informatycznego, Metody Informatyki Stosowanej 2010, nr 3, s. 105--109.
[4] Kruchten P.: The Rational Unified Process: An Introduction, Addison-Wesley, 2003.
[5] OMG, Software & Software Process Engineering Meta-Model (SPEM 2.0), April 2008.
[6] OMG, Meta Object Facility (MOF) Core Specification, Version 2.0, January 2006.
[7] OMG, OMG Unified Modeling Language, Superstructure, Version 2.3, May 2010.
[8] Shore J., The Art of Agile Development, O'Reilly 2007.

otrzymano / received: 15.04.2011
przyjęto do druku / accepted: 06.06.2011 *artykuł recenzowany*

INFORMACJE

Nowa inicjatywa PAK

Na stronie internetowej Wydawnictwa PAK został utworzony dział: **Niepewność wyników pomiarów** w którym są zamieszczane aktualne informacje dotyczące problemów teoretycznych i praktycznych związanych z szacowaniem niepewności wyników pomiarów. W dziale znajdują się:

- aktualne informacje o publikacjach dotyczących niepewności wyników,
- informacje o przedsięwzięciach naukowo–technicznych i edukacyjnych, o tematyce związanej z niepewnością,
- dokumenty dotyczące niepewności,
- pytania do ekspertów (FAQs).

Zapraszamy:

- autorów opublikowanych prac dotyczących niepewności o nadsyłanie tekstów do zamieszczenia w tym dziale,
- organizatorów przedsięwzięć naukowo – technicznych lub edukacyjnych do nadsyłania informacji o imprezach planowanych lub odbytych,
- zainteresowanych zagadnieniami szczegółowymi do nadsyłania pytań do ekspertów.

Materiały mogą mieć formę plików lub linków do źródeł. Warunkiem zamieszczenia w tym dziale strony internetowej PAK materiałów lub linków jest przysłanie do redakcji PAK pocztą zwykłą zgody właściciela praw autorskich na takie rozpowszechnienie. Zamieszczanie i pobieranie materiałów i informacji w tym dziale strony internetowej jest bezpłatne. Redakcja PAK będzie nadzorować zawartość działu, ale za szczegółowe treści merytoryczne odpowiadają autorzy nadsyłanych materiałów.

Tadeusz SKUBIS
Redaktor naczelny Wydawnictwa PAK

Zapraszamy do publikacji artykułów naukowych w czasopiśmie PAK

WYDAWNICTWO PAK
ul. Świętokrzyska 14A, pok. 530, 00-050 Warszawa,
tel./fax: 22 827 25 40

Redakcja czasopisma POMIARY AUTOMATYKA KONTROLA
44-100 Gliwice, ul. Akademicka 10, pok. 30b,
tel./fax: 32 237 19 45, e-mail: wydawnictwo@pak.info.pl