AUTOMATYKA • 2006 • Tom 10 • Zeszyt 3

Tomasz Serafiński*

Repozytorium konfiguracji oprogramowania jako podstawa zaawansowanych rozwiązań zarządzania procesem wytwórczym oprogramowania

1. Wprowadzenie

Podstawowym elementem opisywanego rozwiązania jest repozytorium konfiguracji oprogramowania. Element ten zazwyczaj, jeśli nie zawsze, stanowi nieodłączny element rozwiązań zarządzania projektami informatycznymi. W praktyce jest bazą danych, w której przechowuje się poszczególne artefakty projektu informatycznego oraz wszelkie dodatkowe informacje z nimi związane takie jak ich status, historia czy relacje, w jakich pozostają z innymi pozycjami konfiguracji. Zazwyczaj struktura repozytorium oraz typy przechowywanych w nim pozycji konfiguracji różnią się i stanowią odzwierciedlenie specyfiki danego projektu lub częściej, stosowanego narzędzia. Repozytorium jest sprawdzonym i nieodzownym elementem zapewniającym realizacje podstawowych założeń zarządzania konfiguracją oprogramowania.

Z założenia repozytorium powinno się nadawać do użycia w dowolnym scenariuszu zarządzania konfiguracją. Dzięki temu uzyskamy uniezależnienie i ujednolicenie struktury repozytorium dla rozmaitych zadań konfiguracji oprogramowania. Zbytnie uogólnienie grozi jednak zatraceniem zdolności rejestrowania szczegółów specyficznych dla danego projektu, które mogą się okazać kluczowe dla osiągnięcia stawianych celów. Dlatego projekt repozytorium powinien uwzględniać możliwości dostosowywania struktury do poszczególnych przypadków przy jednoczesnym zachowaniu stałych, wspólnych elementów, na których można oprzeć rozwiązanie problemu szacowania zakresu zmiany.

2. Pozycje konfiguracji i ich rodzaje

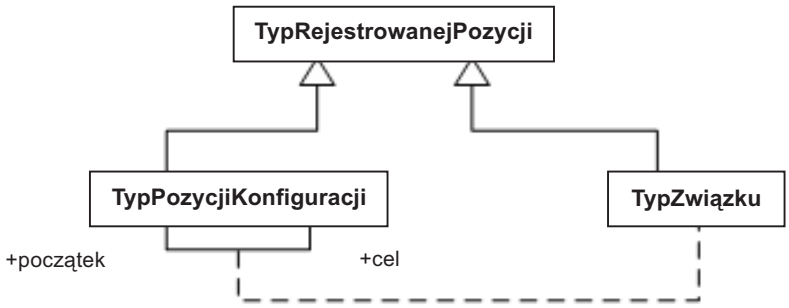
Wychodzimy z założenia, że w repozytorium konfiguracji oprogramowania przechowywane będą wybrane artefakty projektu informatycznego. Gdy znajdą się w repozyto-

* Katedra Informatyki Stosowanej, Politechnika Łódzka

rium stają się pozycjami konfiguracji danego projektu. Z punktu widzenia zbierania danych o projekcie i procesie jego wytwarzania, celem analizy, kluczowa jest znajomość typów poszczególnych pozycji konfiguracji. Dlatego repozytorium powinno przechowywać listę typów pozycji konfiguracji możliwych lub dozwolonych w przypadku danego projektu. Lista ta musi być każdorazowo definiowana przed rozpoczęciem projektu informatycznego, który ma podlegać zarządzaniu konfiguracją oraz proponowanemu procesowi szacowania zmiany.

Dla szacowania zakresu rozprzestrzeniania się zmiany w projekcie kluczowe są zależności i relacje, w jakich pozostają względem siebie poszczególne pozycje konfiguracji. Dlatego zostały one uwzględnione w strukturze repozytorium jako osobne pozycje. Ponieważ są one pojęciowo ściśle związane z pozycjami konfiguracji w tradycyjnym znaczeniu [1] konieczne stało się ich wyraźne rozróżnienie. Pozycje powstałe z artefaktów projektu oraz związki między pozycjami są traktowane jako specjalizowane postaci typu rejestrowanej pozycji. Specjalizacji owa przejawia się w tym, że pozycja będąca rejestrowanym typem relacji stanowi zdefiniowany związek z pomiędzy rejestrowanymi typami pozycji konfiguracji pochodzącymi ze zbioru artefaktów projektu informatycznego.

Na rysunku 1 zapisanym w notacji UML klasy *TypZwiązku* i *TypPozycjiKonfiguracji* dziedziczą ze wspólnego przodka, klasy *TypRejestrowanejPozycji*. Klasa ta stanowi element bazowy listy przechowującej typy pozycji konfiguracji rejestrowane w danym rozwiązaniu.



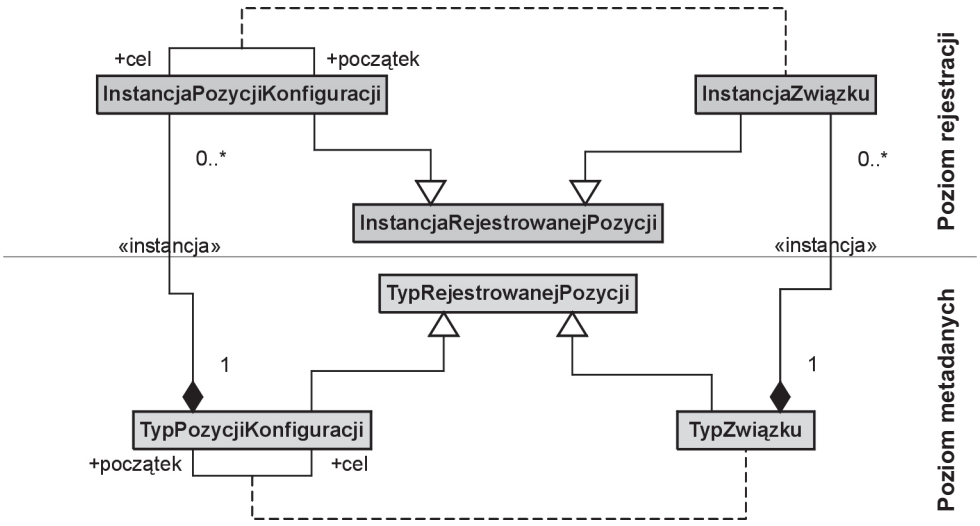
Rys. 1. Struktura typów rejestrowanych pozycji w repozytorium

Klasa *TypZwiązku* jest klasą asocjacji stanowiącą połączenie pomiędzy dwoma typami pozycji konfiguracji pochodzącymi od typu artefaktu objętego w danym przypadku zarządzaniu konfiguracją. W dalszej części pracy pozycje konfiguracji będące artefaktami będziemy nazywać pozycjami konfiguracji pierwszego rzędu, a pozycje konfiguracji będące asocjacjami nazwiemy pozycjami konfiguracji drugiego rzędu. Dokonanie takiego podziału ma ważne implikacje dla konstrukcji repozytorium i jego miejsca w procesie zarządzania konfiguracją. Przede wszystkim wymusza określenie typów pozycji konfiguracji oraz zde-

finiowanie związków, w jakie pozycje pierwszego rzędu o określonych typach mogą wchodzić pomiędzy sobą. Dzięki temu uzyskujemy pewną sztywność konstrukcji, na której możemy później oprzeć mechanizmy analizy zawartości repozytorium. Sztywność ta powoduje, że w momencie dodania do repozytorium pozycji konfiguracji pierwszego rzędu o określonym typie znane są już typy związków, w jakie może ona wejść oraz znane są typy innych pozycji pierwszego rzędu, które mogą być z nią związane. Na poziomie metadanych, dodanie do repozytorium typu pozycji konfiguracji pierwszego rzędu wymaga dodania tylu typów pozycji konfiguracji drugiego rzędu ile wynosi liczba różnych związków, które pozycje danego typu mogą tworzyć. Należy zwrócić uwagę, że takie rozwiązanie pozwala także na definiowane pozycji konfiguracji będących dowolnymi agregatami innych pozycji konfiguracji. Przykładowo moduł programu może być pozycją konfiguracji składającą się z wielu innych pozycji już przechowywanych w repozytorium, takich jak klasy, obiekty, ich dokumentacja oraz oczywiście, związki pomiędzy nimi. Osiągamy to definiując typ związku *agregacja pozycji konfiguracji*, a następnie tworzymy związki tego typu pomiędzy modułem, będącym zagregowaną pozycją konfiguracji, a poszczególnymi składowymi pozycjami konfiguracji.

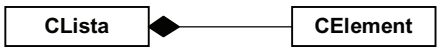
Zauważmy, że do tej pory prowadzone rozważania dotyczą poziomu, na którym nie są jeszcze znane konkretne typy pozycji konfiguracji, gdyż o tych możemy mówić dopiero przez pryzmat konkretnego projektu czy zadania konfiguracji oprogramowania. Osiągnięcie zakładanych efektów wymaga zastosowania podobnej struktury również na poziomie, na którym odbywać się będzie rejestrowanie w systemie zarządzania konfiguracją konkretnych pozycji konfiguracji. W ten sposób w proponowanym modelu analitycznym otrzymujemy symetryczną strukturę, która definiuje dwa poziomy repozytorium. Pierwszy z nich nazywany *poziomem metadanych* zawiera zdefiniowany przez użytkownika szablon repozytorium, które zostanie użyte w danym przypadku. Drugi, *poziom rejestracji*, zawierać będzie dane wprowadzane w trakcie życia projektu informatycznego i co do typów rejestrowanych pozycji konfiguracji odpowiadać będzie poziomowi metadanych. Pozostawienie swobody w definiowaniu typów pozycji konfiguracji i związków między nimi sprawia, że proponowane rozwiązanie jest bardzo elastyczne i może być łatwo dostosowane do określonych, często specyficznych potrzeb zarządzania konfiguracją. Ponadto do określonych celów możemy predefiniować konfigurację repozytorium, otrzymując gotowe szablony konfiguracji oprogramowania, które w istocie są szablonami procesu wytwórczego oprogramowania stosowanego w konkretnym przypadku. Rysunek 2 ukazuje koncepcje dwupoziomowej struktury repozytorium. Jak widać, typy rejestrowanych pozycji zadeklarowane przez użytkownika na etapie definiowania szablonu konfiguracji oprogramowania mogą być potraktowane jako kolekcje instancji pozycji konfiguracji określonych typów na poziomie rejestracji danych.

Każda pozycja konfiguracji, która znajdzie się w repozytorium, będzie posiadała swój typ. Możliwe związki pomiędzy nimi są precyzyjnie zdefiniowane. Poniżej zamieszczono przykład utworzenia szablonu celem dokonania rejestracji pozycji konfiguracji.



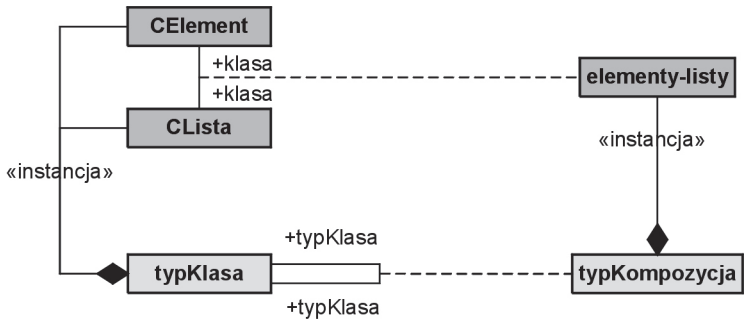
Rys. 2. Dwupoziomowa struktura repozytorium konfiguracji oprogramowania

W przykładzie tym zostanie stworzony szablon repozytorium konfiguracji oraz wykonana przykładowa rejestracja pozycji konfiguracji. Załóżmy, że zamierzamy rejestrować w repozytorium konfiguracji oprogramowania klasy składowe programu pozostające ze sobą w związku całość-część, jak na poniższym przykładzie:



Definiujemy szablon zawierający typ pozycji konfiguracji pierwszego rzędu *klasa* oraz typ związku będący typem pozycji konfiguracji drugiego rzędu *kompozycja*. W przypadku typów pozycji konfiguracji drugiego rzędu musimy również podać typy pozycji konfiguracji pierwszego rzędu, które mogą być wiązane. W naszym przypadku zarówno początek wiązania, jak i jego cel będzie pozycją konfiguracji typu klasa. W ten sposób został zdefiniowany szablon pozwalający rejestrować związki całość-część pomiędzy pozycjami konfiguracji typu klasa. Teraz na poziomie rejestracji możemy dodać dwie pozycje konfiguracji pierwszego rzędu typu klasa o nazwach jak w przykładzie: *CLista* i *CElement*. Następnie rejestrujemy związek pomiędzy nimi w postaci asocjacji *elementy-listy*. Rysunek 3 przedstawia poglądowo strukturę repozytorium oprogramowania w którym zostały zarejestrowane podejmowane w niniejszym przykładzie pozycje konfiguracji oraz ich związek. Dodatkowo w warstwie metadanych zaznaczonej kolorem jasnoszarym, zapisany jest szablon, według którego ma następować rejestracja. Szablon ten pozostaje ważny dla wszystkich innych klas, które będziemy rejestrować w dalszym toku zarządzania konfiguracją oraz wszystkich przyszłych związków typu kompozycja pomiędzy rejestrowanymi klasami. Warto w tym miejscu zauważyć, że ten sposób rejestracji wymaga wcześniejszego ustalenia

semantyki początku i celu związku. W rozważanym przykładzie zarówno początek, jak i koniec są jednym i tym samym typem pozycji konfiguracji. Oznacza to konieczność wcześniejszego uzgodnienia kierunku przebiegu zależności w stosunku do początku i celu.



Rys. 3. Przykład szablonu i rejestracji zgodnie z szablonem

Bez takiego uzgodnienia nie możemy powiedzieć, czy w przykładowym związku całość-część-całość znajduje się po stronie początku czy też celu wiązania. Problem ten wystąpi przy wszystkich rodzajach związków skierowanych, tzn. takich, w których istotny jest kierunek przebiegu. Usiłowanie narzucenia jednego sposobu rozwiązania tego problemu mogłoby spowodować istotne ograniczenia generyczności proponowanej struktury. Wymagałoby to bądź zdefiniowania możliwych do rejestracji związków i określenia reguł dla poszczególnych z nich, bądź uszeregowania wszystkich możliwych do pomyślenia rodzajów związków skierowanych w kategorie i przypisania reguł do określonych kategorii. Problem ten ma szczególne znaczenie w czasie analizy propagacji zmiany. Dlatego zdefiniowanie reguły kierunku dla wybranego związku powinno być elementem konfiguracji narzędzia analitycznego nałożonego na opisywane repozytorium. Ponadto dla różnych przypadków celowa może się okazać możliwość dynamicznej zmiany kierunku związku względem początku i celu definiowanego w repozytorium. Dodatkowo kierunek ten może się zmieniać w zależności od typu pozycji konfiguracji, która akurat występuje jako koniec jednego z wiązań. Tak więc problem ten nie jest istotny na poziomie struktury repozytorium. Jednak zwraca to uwagę na istotną cechę proponowanego rozwiązania. Każdy typ związku może posiadać wyłącznie jeden typ pozycji konfiguracji pierwszego rzędu jako wiązanie początkowe i jeden – jako wiązanie docelowe. Oznacza to, że np. asocjacja binarna, która ma sens zarówno jako związek pomiędzy klasami, jak i obiektami, musi zostać zdefiniowana jako dwa osobne typy związków (typy pozycji konfiguracji drugiego rzędu), jeden dla klas i drugi dla obiektów. Zyskujemy dzięki temu możliwość precyzyjnego wyizolowania grupy związków, które w innym przypadku były by trudne do rozróżnienia. Dodatkową zaletą jest możliwość łatwego przeprowadzenia analizy dla związków wybranego typu, ale związanych wyłącznie z pozycjami konfiguracji o określonych typach.

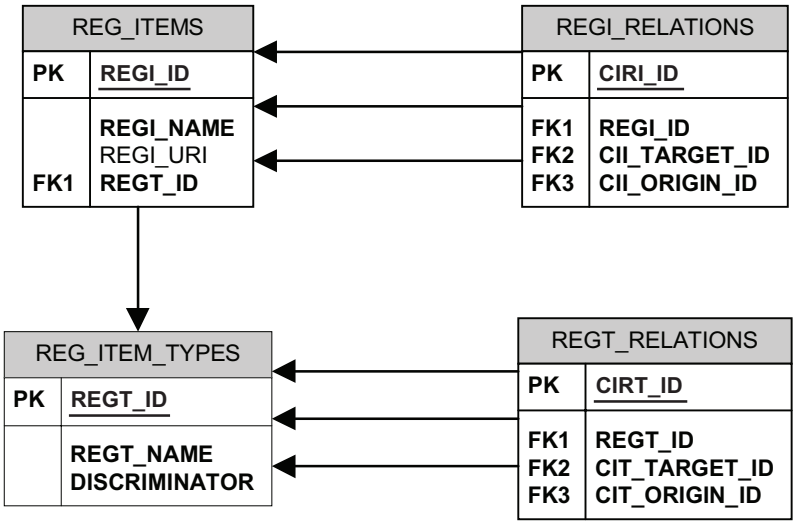
Na potrzeby realizacji prototypu niezbędne okazało się przełożenie przedstawionego modelu obiektowego na model relacyjny. Dzięki temu repozytorium może przyjąć postać bazy danych, co jest uznaną i powszechnie stosowaną praktyką. Jako punkt wyjścia niech

posłuży rysunek 2. Podstawowym obiektem, od którego wyszliśmy, jest *TypRejestrowanejPozycji*, jego odpowiednikiem w modelu relacyjnym niech będzie tabela nazwana *REG_ITEM_TYPES*. Podział na typy pozycji konfiguracji pierwszego i drugiego rzędu zrealizowano przy zastosowaniu dyskryminatora binarnego, którego wartość określa przynależność określonego typu do jednej bądź drugiej grupy. Zawartość szablonu konfiguracji oprogramowania definiuje typy pozycji konfiguracji drugiego rzędu, wraz z typami, do których mogą być dowiązane, zostanie zapisana w tabeli *RGT_RELATIONS*.

Definiujemy trzy relacje pomiędzy wymienionymi tabelami:

FK1 – wskazującą na określony typ związku pomiędzy pozycjami konfiguracji;
FK2 – wskazującą na typ pozycji konfiguracji będący celem wiązania danego typu związku;
FK3 – wskazującą na typ pozycji konfiguracji będący początkiem wiązania danego typu związku.

W ten sposób otrzymujemy relacyjny model poziomu metadanych naszego repozytorium konfiguracji oprogramowania. Podobnie jak w modelu obiektowym, tak i tutaj poziom rejestracji będzie symetryczny do poziomu metadanych. Tak jak zostało to pokazane na rysunku 4, zdefiniowano tabelę reprezentującą instancje pozycji konfiguracji obydwu rzędów *REG_ITEMS* oraz tabelę *REGI_RELATIONS* reprezentującą związki pomiędzy instancjami zarejestrowanych pozycji konfiguracji. W tym przypadku rozróżnienie pomiędzy instancjami pozycji konfiguracji pierwszego i drugiego rzędu osiągamy poprzez dowiązanie określonej instancji do odpowiadającego jej typu w tabeli *REG_ITEM_TYPES*.



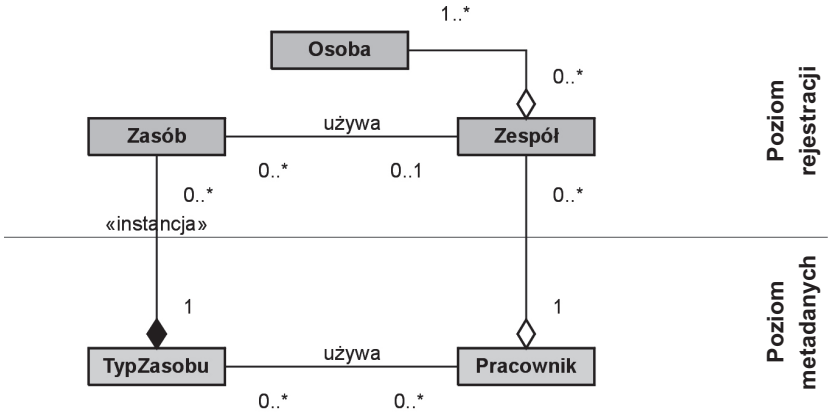
Rys. 4. Model relacyjny struktury repozytorium konfiguracji oprogramowania

W ten sposób skonstruowano model równoległy na bazie modelu obiektowego i relacyjnego. Kontynuując rozważania przy użyciu bardziej intuicyjnego modelu obiektowego, zapewniamy jego implementowalność w środowisku relacyjnej bazy danych.

3. Zasoby i ich rodzaje

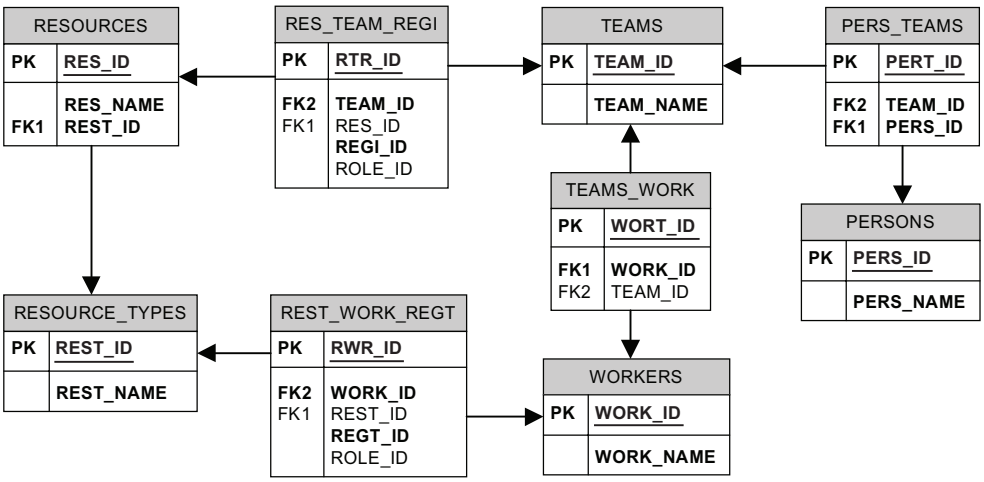
Założeniem jest jednak skonstruowanie takiego repozytorium konfiguracji oprogramowania, że analiza zgromadzonych w nim danych umożliwi szacowanie nie tylko zakresu rozprzestrzeniania się zmiany, ale również potencjalnych kosztów jej wprowadzenia. Dlatego repozytorium musi przechowywać również informacje o zasobach związanych z wytworzeniem poszczególnych pozycji konfiguracji. Kontynuując przyjętą koncepcję modelu dwuwarstwowego, stwierdzamy, że podstawą rejestracji danych o zasobach jest definiowalna lista typów zasobów. W modelu obiektowy reprezentowana jest przez klasę *TypZasobu*. Podobnie na poziomie rejestracji odpowiada jej kolekcja instancji zasobów określonych typów. Charakterystycznym elementem proponowanej struktury jest rozróżnienie pomiędzy zasobami a zasobami ludzkimi. Poprzez zasób rozumiemy składowe elementy środowiska niezbędne do wytworzenia danych pozycji konfiguracji. Zasoby ludzkie rozumiemy natomiast, jako osoby, które wytwarzając określone pozycje konfiguracji, zużywają (lub nie) zasoby. Uwzględnianie zasobów ludzkich jest istotną nowością w konstrukcji repozytorium, gdyż nie jest spotykane w innych rozwiązaniach tego typu. Na poziomie metadanych zasoby ludzkie są symbolizowane przez obiekt *Pracownik*. Rozróżnienie zasobów od personelu realizującego projekt ma istotne znaczenie. Przede wszystkim dostępność zasobów ludzkich jest znacznie mniej przewidywalna. Pracownik może być na urlopie lub odejść z projektu, natomiast zasoby niezbędne do realizacji poszczególnych zadań są najczęściej relatywnie łatwo dostępne, a ich posiadanie (zakup) daje gwarancję dostępności w chwili, gdy będą potrzebne. Dodatkowo w wypadku zasobów ludzkich w grę wchodzi jeszcze element doświadczenia i know-how poszczególnych pracowników, które to nie mogą być łatwo zastąpione w wypadku niedostępności danego pracownika. Z perspektywy szacowania kosztów może to mieć istotne znaczenie. Po pierwsze, w czasie analizy i planowania zadania wprowadzenia zmiany możemy łatwo zlokalizować potencjalne problemy. Przykładowo: musimy wprowadzić modyfikację do modułu oprogramowania odpowiedzialnego za komunikację z zewnętrzną bazą danych. Z dużym prawdopodobieństwem możemy przyjąć, że zasoby użyte do wytworzenia poprzedniej wersji modułu nadal są dostępne. W typowym przypadku będzie to środowisko wytwórcze (kompilatory, narzędzia CASE, biblioteki, itp.) oraz infrastruktura IT zapewniająca działanie środowiska uruchomieniowego. Niestety założenie takie w wypadku pracowników może nie być uzasadnione. Nie tylko w przypadku, gdy zadania realizowane są w trybie jednorazowych zleceń. Określeni pracownicy mogą być przypisani do innych zadań. [2, 3]. Dalej, z punktu widzenia np. efektywności lepiej do realizacji wybranych zadań przypisać pracowników o większym doświadczeniu. Tego typu problemy muszą być brane pod uwagę w przypadku zadania szacowania kosztów realizacji. Również ich efektywna lokalizacja stanowi pożądaną cechę każdego narzędzia wspierającego zarządzanie projektami informatycznymi. Ponieważ na poziomie matadanych ustalany jest jedynie szablon, według którego będą podejmowane

działania w trakcie rozwoju oprogramowania – informacje o wyzyskanych zasobach i personelu, w naturalny sposób muszą być zapisywane na poziomie rejestracji repozytorium konfiguracji oprogramowania. Obiekt *Pracownik* reprezentuje jedynie potrzeby funkcjonalne, konieczne do prowadzenia prac nad określonymi typami pozycji konfiguracji. Na poziomie rejestracji danych odpowiada mu *Zespół*, który może się składać z co najmniej jednej osoby. Pracownikowi przewidzianemu w szablonie może na poziomie rejestracji danych odpowiadać kilka zespołów. W skład różnych zespołów mogą wchodzić te same osoby. Jest to możliwe dzięki temu, że z *pracownikiem* związany jest zespół, a nie bezpośrednio konkretna osoba. Trzymając się przyjętej reguły, na etapie projektowania szablonu, zgodnie z którym będzie przebiegał rozwój oprogramowania, pracownicy są powiązani z typami zasobów, które są im niezbędne do realizowania postawionych zadań. Na poziomie rejestracji, w ten sam sposób zespoły powiązane są z konkretnymi zasobami używanymi w danej sytuacji. Struktura ta przedstawiona została na rysunku 5.



Rys. 5. Struktura zasobów

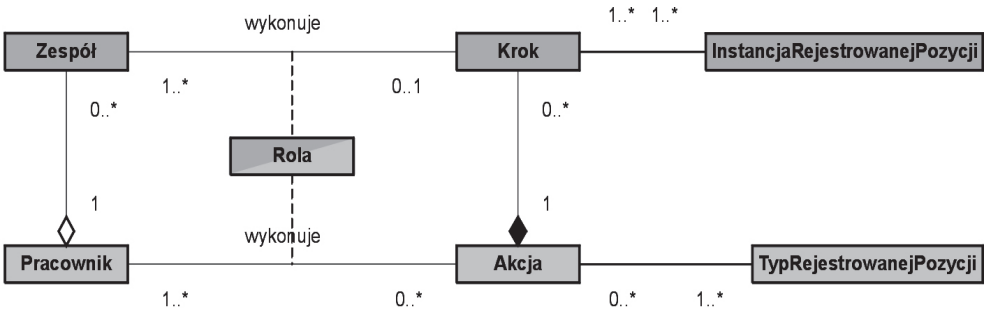
Informacje o tym, czy dany zasób zostaje zużyty przez zespół czy też można go wykorzystać ponownie, są atrybutami typu zasobu. Ma to znaczenie, gdy planujemy połączyć proponowane rozwiązanie zarządzania konfiguracją oprogramowania z systemem zarządzania zasobami magazynowymi. W niniejszej pracy nie będziemy zajmować się zagadnieniami planowania dostaw i gospodarki magazynowej. Zaznaczyć należy jedynie, że istnieją potencjalne możliwości integracji, a co za tym idzie, umożliwienie nie tylko precyzyjnego szacowania kosztów, ale również planowania dostaw, co wydatnie wpływa na podniesienie efektywności przedsiębiorstwa i redukcję kosztów działania. Konsekwentnie, konstruując model równoległy repozytorium, można stwierdzić, że struktura służąca przechowywaniu danych zostaje zapisana również za pomocą relacji. Przedstawiono to na rysunku 6. Model struktury zasobów jest powiązany z resztą modelu za pomocą asocjacji na obydwu poziomach. Na poziomie metadanych *TypZasobu* musi zostać związany z typem pozycji konfiguracji pierwszego rzędu. Zakładamy tu, że wytwarzanie pozycji konfiguracji rzędu drugiego nie angażuje dodatkowych zasobów ponad te, które potrzebne są do wytworzenia wiązanych pozycji konfiguracji rzędu pierwszego.



Rys. 6. Struktura relacyjna repozytorium – zasoby projektu

Podobnie, zachowując przyjętą logikę symetrycznej struktury obydwu warstw, reprezentacja zasobu zostaje związana z określoną instancją pozycji konfiguracji. Dzięki temu na etapie projektowania szablonu, które odbywa się na poziomie metadanych określamy, jakie typy pozycji konfiguracji wymagają jakiego rodzaju zasobów. Na poziomie rejestracji, każda pozycja konfiguracji ujęta w repozytorium posiada zapis mówiący, jakie zasoby zostały użyte do jej wytworzenia oraz kto pracował nad jej wytworzeniem. Dane takie okazują się niezmiernie przydatne przy szacowaniu kosztów zmiany metodą porównawczą, czyli opierającą się na kosztach wytworzenia poprzednich wersji danej pozycji konfiguracji.

Na tym etapie rozważań skonstruowano model równoległy repozytorium konfiguracji oprogramowania, składający się z modelu obiektowego i odpowiadającego mu modelu relacyjnego. Zauważmy, że w zaproponowanej strukturze możemy już przechowywać wszelkie informacje opisujące elementy i zasoby biorące udział w realizowanym projekcie informatycznym. Brakuje natomiast sposobu i miejsca zapisu dynamicznych elementów procesu wytwarzania oprogramowania. Jeżeli celem jest określanie nie tylko zakresu pozycji konfiguracji objętych zmianą, ale również kosztu wprowadzenia zmiany, poleganie wyłącznie na danych wiążących w statyczny sposób poszczególne pozycje konfiguracji z zasobami może się okazać niewystarczające. Proponowane powiązanie pozycji konfiguracji z zasobami nie uwzględnia kilku istotnych czynników. Pracownik nie może być sztywno związany z typem zasobu. Podczas pracy nad określoną pozycją konfiguracji może nie zużyć lub zużyć częściowo przypisane mu zasoby. Podobnie czynności, które wykonuje nad jedną i tą samą pozycją konfiguracji mogą mieć różny charakter i być związane z różnymi zasobami lub różnymi ich ilościami. Prowadzi to do wniosku, że w repozytorium powinny być zapisywane poszczególne kroki, które zostają wykonane podczas rozwoju określonych pozycji konfiguracji (rys. 7). Naturalnie, zapis ów musi się odbywać na poziomie rejestracji danych.



Rys. 7. Rejestracja kroków w rozwoju oprogramowania

Wprowadzenie w przedstawianym modelu bytów reprezentujących czynności, które mają miejsce podczas rozwoju oprogramowania na tym poziomie, wymaga uzupełnienia modelu na poziomie metadanych. Zauważmy, że dla zachowania spójności i kompletności modelu na poziomie metadanych zostanie umieszczona lista kroków, które będą konieczne do osiągnięcia określonego celu. Celem tym jest wykonanie zadania na danej pozycji konfiguracji lub ich zestawie. Prowadzi to do dwóch ważnych wniosków. Po pierwsze, etap tworzenia szablonu repozytorium staje się faktycznie planowaniem procesu wytwórczego oprogramowania. Zaprojektowanie szablonu jest przez to zadaniem o pierwszorzędnym znaczeniu, jako że dotyczy on całego procesu wytwarzania oprogramowania. Po drugie, na poziomie metadanych zostają zapisane elementy składowe, z których zbudowany będzie harmonogram prac nad projektem z dokładnością, co do pozycji konfiguracji. Elementy te przechowują informację o tym, jakie kroki będą podejmowane podczas rozwoju pozycji konfiguracji poszczególnych typów. Przechowywanie w repozytorium harmonogramu prac nie jest konieczne, ale struktura repozytorium stwarza taką możliwość. Uzupełniamy, więc model obiektowy o kolejne dwa byty. Na poziomie metadanych będzie to *TypKroku* i odpowiadający mu na poziomie rejestracji danych *Krok*. Powiązanie pomiędzy nimi jest zrealizowane w dokładnie ten sam sposób, co przy poprzednich odpowiadających sobie elementach. Krok jest instancją odpowiadającego mu typu kroku, czyli jest wykonaną (planowaną) czynnością spośród czynności dopuszczalnych dla danego typu pozycji konfiguracji. Ponieważ jak wspomniano we wcześniejszej części tekstu, jeden zespół czy pracownik (na odpowiednim poziomie modelu) może wykonywać różne prace nad pozycją konfiguracji lub jej typem wprowadzono pojęcie roli. Rola występuje zarówno na poziomie metadanych, jak i rejestracji. Podobnie jak w przypadku instancji czy typu związku, rola jest w modelu obiektowym klasą asocjacji. Rola jako jedyny element modelu występuje na obydwu poziomach. Typowe przykłady ról, jakie mogą być odgrywane podczas tworzenia pozycji konfiguracji to autor, właściciel, audytor czy zarządca. Zapisanie w repozytorium harmonogramu prac może się wydać w wielu przypadkach użyteczne. Dzięki temu uzyskujemy scentralizowanie kontroli nad wykonywaniem projektem. Dodatkowo mamy gwarancję, że harmonogram taki będzie zawsze aktualny, ponieważ stan wykonania poszczególnych zadań będzie pobierany bezpośrednio z repozytorium. Zwróćmy uwagę, że dzięki temu na

bazie repozytorium o proponowanej strukturze będzie można skonstruować również narzędzie raportujące.

Raporty te mogą dotyczyć, planów, stanu prac, zużycia zasobów czy, tak jak zostało to wcześniej wspomniane, planowania zasobów. Jeśli harmonogram ma być „zapamiętany" w repozytorium, jego tworzenie musi się odbywać na poziomie rejestracji danych. Oznacza to, że przed przystąpieniem do harmonogramowania musimy już posiadać zdefiniowany proces rozwoju oprogramowania, zapisany pod postacią szablonu konfiguracji oprogramowania. Następnie musimy zdefiniować pozycje konfiguracji określonych typów. Proces ten można zautomatyzować poprzez import żądanych danych z dokumentacji projektowej. Kolejnym krokiem jest sformowanie zespołów i określenie zasobów. Dopiero teraz możemy skonstruować plan prac, będący jednocześnie harmonogramem używając znanego szablonu procesu z poziomu metadanych oraz rzeczywistych pozycji konfiguracji i zasobów. W takim przypadku, wszystkie kroki rozwoju oprogramowania są już zaplanowane, a ich wykonanie oznacza zmianę stanu obiektu klasy *Krok* z *do wykonania* na *wykonany*.

4. Podsumowanie

Podsumowując, można stwierdzić, że w oparciu o przedstawiony model repozytorium konfiguracji oprogramowania można zbudować nie tylko system zarządzania konfiguracją oprogramowania, ale kompletny system zarządzania projektem informatycznym. System ten może uwzględniać nie tylko elementy składowe projektu, jak ma to miejsce w powszechnie spotykanych rozwiązaniach.

Proponowana struktura repozytorium umożliwia również przechowywanie całego spektrum informacji o zasobach, procesie wytwarzania oraz planach i harmonogramach realizacji projektu. Dodatkowo stanowi wygodnie źródło informacji dla narzędzi analitycznych i raportujących. Rozwój projektu informatycznego, czy jego kolejne iteracje, będą się odbywać w trzech następujących po sobie fazach.

1. **Projektowanie procesu** – zostaje zdefiniowany szablon procesu rozwoju oprogramowania. Zawiera on definicje typów przechowywanych pozycji konfiguracji, zasobów wszelkiego typu, w tym ludzkich, kroków, które będą wykonywane w czasie realizacji zdefiniowanego procesu.
2. **Planowanie przebiegu** – zostają zweryfikowane posiadane zasoby oraz zostaje zaplanowane ich użycie. Formowane są zespoły i przypisywane są im zadania. Zapisywany jest plan prac.
3. **Rejestracja procesu** – poszczególne kroki zostają zarejestrowane jako wykonane.

W kolejnych iteracjach, związanych z wytwarzaniem następnych wersji oprogramowania, powtarzane są fazy druga i trzecia. Plany przebiegu w części dotyczącej użycia określonych zasobów, sformowanych zespołów czy stałych elementów przebiegu prac (jak np. projektowanie, weryfikacja, implementacja), często będą niezmienne w kolejnych iteracjach. Szablon zdefiniowany w kroku pierwszym może, a nawet powinien, być przeznaczony do ponownego użycia. Ponowne użycie szablonu nie musi się ograniczać jedynie do wersji rozwojowych tego samego projektu.

Literatura

[1] Hass A.: *Configuration Management Principles and Practice*. Addison-Wesley, January 2003
[2] Larman C.: *Agile and Iterative Development: A Manager's Guide*. Addison Wesley, August 2003
[3] Telelogic A.B.: *Transparent Software Configuration Management.* 2002