Dokumentowanie wymagań w języku XML[1]

Krzysztof Kowalczykiewicz, Jerzy Nawrocki
Instytut Informatyki Politechniki Poznańskiej

Streszczenie

W artykule przedstawiono koncepcję zastosowania języka XML i pochodnych mu technologii do wspomagania procesów inżynierii wymagań w małych firmach informatycznych. Szczególny nacisk został położony na uwzględnienie dobrych praktyk inżynierii wymagań zebranych przez Sommerville'a i Sawyera.

1. Wstęp

Specyfikacja wymagań jako początkowy etap stanowiący podstawę dla wszystkich następnych odgrywa istotną, jeśli nie najistotniejszą, rolę w procesie budowy oprogramowania. Powstałe w tym czasie i przeoczone błędy wpływają na mnożenie późniejszych kosztów realizacji. Dlatego właśnie ogromny wpływ na powodzenie projektu i realność kosztorysu ma poprawne przeprowadzenie specyfikacji wymagań. Tymczasem wg raportu NASA [10] aż 50% błędów w oprogramowaniu powstaje właśnie na tym etapie.

Procesy definiowane przez inżynierię wymagań można wspomóc poprzez użycie specjalizowanych narzędzi. Powstało kilkanaście takich komercyjnych systemów (np. RequisitePro firmy Rational [11], CaliberRM firmy TBI [12], DOORS firmy Telelogic [13] i inne). Pozwalają one na przechowywanie wymagań i zarządzanie nimi przez cały czas realizacji projektu. Wspomagają wersjonowanie wymagań, ułatwiają ich śledzenie oraz automatyzują generowanie raportów pozwalających na ocenę postępów w realizacji projektu.

[1] Praca wykonana w ramach 43-364/01-BW

Inną pożądaną cechą narzędzi wspomagających inżynierię wymagań jest wielodostępność i możliwość korzystania z repozytorium przez wszystkich uczestników danego projektu poprzez Internet. Wiele narzędzi komercyjnych, takich jak np. RequisitePro Web spełnia te wymagania.

Główną wadą narzędzi komercyjnych jest ich cena sięgająca nawet kilku tysięcy dolarów za 1-stanowiskową licencję. Z punktu widzenia małej firmy informatycznej taka cena może być barierą nie do przebycia.

Mimo tych trudności małe firmy informatyczne stoją przed wymogiem poprawy swoich procesów budowy oprogramowania warunkującym dalszy rozwój. Ryzyko inwestycyjne zakupu narzędzi komercyjnych jest wysokie ze względu na duże koszty zakupu i wdrożenia z jednej strony oraz stosunkowo małe rozmiary projektów i małe dochody z drugiej. Może się bowiem okazać, że zakupione narzędzia nie tylko wymagają czasochłonnych szkoleń, ale ich zaawansowanie oraz możliwości są niewspółmierne do zarządzanych nimi projektów i stawianych im wymagań.

Okazję do przełamania tego impasu i udostępnienia narzędzi nie wymagających takich nakładów przyniósł w ostatnich latach rozwój sieci Internet i wielu powiązanych z nią technologii. Szczególnie atrakcyjnym rozwiązaniem wydaje się zastosowanie języka XML i pochodnych. Oparcie inżynierii wymagań na tych właśnie technologiach mogłoby być prostym rozwiązaniem, zapewniającym wymagane cechy użytkowe i utrzymującym niskie koszty wdrożenia. Takie rozwiązanie być może byłoby do przyjęcia dla małych firm informatycznych, przynajmniej jako pierwszy krok na drodze do wdrażania bardziej zaawansowanych narzędzi inżynierii oprogramowania.

W artykule proponuje się język XML i związane z nim standardowe narzędzia (np. przeglądarki internetowe, parsery i procesory) jako bazę do efektywnego zarządzania wymaganiami obejmującą utrzymywanie repozytorium wymagań oraz wspomaganie konstruowania specyfikacji wymagań oprogramowania jako dokumentu.

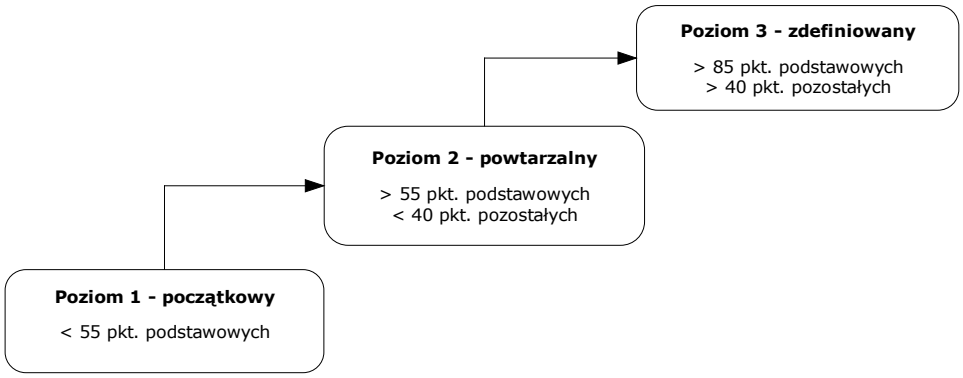
Jako punkt wyjścia przyjęto dobre praktyki inżynierii wymagań zebrane przez Sommerville'a i Sawyera [1]. Pozwalają one na ewaluację poziomu zaawansowania inżynierii wymagań w organizacji specyfikując pożądane cechy procesu oraz wytyczając kierunki rozwoju i poprawy procesów również w odniesieniu do modelu CMM [2].

2. Dobre praktyki dokumentowania wymagań

Ian Sommerville i Pete Sawyer zaproponowali ciekawą metodę oceny i doskonalenia procesów inżynierii wymagań [1]. Opiera się ona na wyodrębnieniu dobrych praktyk, czyli czynności będących pożądanymi elementami wzorcowego procesu inżynierii wymagań. Autorzy wydzielili ponad 60 takich praktyk,

grupując je w 3 kategoriach: podstawowe, średnie i zaawansowane. Przydzielając oceny powszechności stosowania każdej z praktyk w organizacji (0 – nigdy nie używane, 1 – rzadko, 2 – zwykle, 3 – zawsze) można ocenić dojrzałość procesów inżynierii na podstawie sumarycznej liczby uzyskanych punktów (patrz rys. 2.1).

Uzyskane w ten sposób oszacowanie może służyć jako punkt wyjścia do procesu ulepszania inżynierii wymagań. W ten sposób słabo oceniona praktyka staje się wskazówką do doskonalenia procesu. Koncepcja poziomów dojrzałości procesu przedstawiona na rys. 2.1 nawiązuje do modelu dojrzałości CMM [2] obejmującego wszystkie obszary inżynierii oprogramowania lecz nie oferującego tak ścisłych kryteriów ewaluacji w odniesieniu do inżynierii wymagań.



Rysunek 2.1: Poziomy dojrzałości inżynierii wymagań
wg Sommerville'a i Sawyera [1]

Tabela 2.1: Wybrane praktyki Sommerville'a-Sawyera

PODSTAWOWE PRAKTYKI	**P12.** Używaj diagramów w odpowiedni sposób
P1. Zdefiniuj standardową strukturę dokumentu specyfikacji wymagań	**P13.** Sprawdź czy dokument jest zgodny z twoim standardem
P2. Wyjaśnij jak korzystać z tego dokumentu	**P14.** Przypisz wymaganiom identyfikatory
P3. Załącz podsumowanie wymagań	**ŚREDNIOZAAWANSOWANE PRAKTYKI**
P4. Zdefiniuj specjalistyczne terminy	**Ś1.** Przechowuj uzasadnienia wymagań
P5. Zadbaj o czytelną strukturę dokumentacji wymagań	**Ś2.** Stosuj scenariusze użycia
P6. Pomóż czytelnikowi znaleźć informacje	**Ś3.** Używaj słownika danych
P7. Spraw, by dokument wymagań był łatwy do zmian	**Ś4.** Udokumentuj powiązania pomiędzy wymaganiami klienta a modelami systemu
P8. Zidentyfikuj udziałowców systemu i skonsultuj się z nimi	**Ś5.** Zaproponuj testy dla wymagań
P9. Przechowuj źródła wymagań	**ZAAWANSOWANE PRAKTYKI**
P10. Nadaj priorytety wymaganiom	**Z1.** Używaj te same wymagania wielokrotnie
P11. Zdefiniuj szablony do opisywania wymagań	**Z2.** Przypisz ryzyko wymaganiom
	Z3. Zidentyfikuj ulotne wymagania
	Z4. Przechowuj odrzucone wymagania

Spośród wszystkich praktyk Sommerville'a i Sawyera wybraliśmy te, które mają szansę szybkiego wdrożenia w małej firmie informatycznej i nadają się do wspomagania za pomocą języka XML. Wybrane praktyki dotyczą konstruowania dokumentu specyfikacji wymagań, oraz ich zbierania, przechowywania i przetwarzania ich opisu. Podsumowanie wybranych na potrzeby artykułu praktyk zostało przedstawione w tabeli 2.1.

Model Sommerville'a-Sawyera nie jest doskonały. Jedną z jego wad jest kompensacyjność – może się zdarzyć, że dwie organizacje sklasyfikowane na tym samym poziomie będą stosowały bardzo różne praktyki inżynierii wymagań. Zaletą tego modelu jest oparcie się na dostatecznie precyzyjnie opisanym zbiorze dobrych praktyk inżynierii wymagań.

3. Język XML

Język XML stanowi podzbiór opracowanego i ustandaryzowanego w latach 80-tych języka SGML [7]. Jest on przystosowany do udostępniania i przetwarzania informacji w sieci Internet. Podobnie jak pierwowzór, oparty jest on na wzbogacaniu przechowywanej informacji tekstowej o elementy semantyczne i strukturalne mające postać tzw. znaczników. Powstały w połowie lat 90-tych język XML przeżywa burzliwy rozwój, znajdując ciągle nowe dziedziny zastosowań. Specyfikacja XML [3] pozwala na łatwe definiowanie własnych języków opartych na znacznikach i podaje tylko ogólne wymogi dotyczące podstawowych elementów składowych, poprawności stosowania znaczników oraz sposobu definiowania reguł poprawności przy pomocy dokumentu DTD (*Document Type Definition*), stanowiącego gramatykę projektowanego języka.

W ciągu kilku lat rozwoju język XML obrósł w dodatkowe technologie wzbogacające jego funkcjonalność. Warto wspomnieć chociażby o XSL [4] (*eXtended Stylesheet Language*) pozwalającym na zdefiniowanie opisu warstwy wizualnej dokumentu XML, jego odmianie XSLT [3] pozwalającej na transformację jednego typu dokumentu w drugi za pomocą reguł dopasowania i zamiany znaczników. Innym ciekawym narzędziem związanym z XML-em jest XLink [6] – elastyczny mechanizm łączenia ze sobą dokumentów lub ich fragmentów. Większość z tych technologii w krótkim czasie została zaimplementowana poszerzając wachlarz narzędzi dostępnych dla aplikacji wykorzystujących XML.

Przedstawione w artykule podejście opiera się na zdefiniowaniu przy pomocy XML języka pozwalającego na przechowywanie i przetwarzanie wymagań w oparciu o dobre praktyki inżynierii wymagań Sommerville'a i Sawyera. Język ten ma zatem wspomagać identyfikację, śledzenie, wersjonowanie i zmiany w wymaganiach. Zdefiniowane przy jego pomocy repozytorium

stanowić ma bazę do konstruowania dokumentu specyfikacji wymagań w sposób półautomatyczny, w oparciu o zdefiniowane przy pomocy XSL i XSLT szablony z uwzględnieniem praktyk dotyczących jego pożądanych cech oraz zawartości i układu proponowanego w standardzie ANSI/IEEE 830 [9]. W dalszej części artykułu opisany zostanie sposób wykorzystania istniejących narzędzi języka XML w inżynierii wymagań oraz zaproponowany zostanie zestaw prostych narzędzi wspomagających zarządzanie wymaganiami.

4. Dokumentowanie wymagań

Podstawowym wymogiem zarządzania wymaganiami i ich przetwarzania jest zdefiniowanie sposobu efektywnego i elastycznego przechowywania. Częściowe sformalizowanie definicji wymagań w miejsce opisu słownego umożliwia ostrzejsze wyodrębnienie jego atrybutów i komponentów, pozwalając na lepszą klasyfikację i identyfikację.

Kluczowym elementem jest wyróżnienie atrybutów pojedynczego wymagania i podział strukturalny jego części opisowej w celu wydzielenia możliwie małych porcji informacji. Atrybuty wymagań zostały sformułowane na podstawie dobrych praktyk Sommerville'a i Sawyera [1], opisu zarządzania wymaganiami w modelu CMM [2] oraz wskazówek opisu wymagań w standardzie ANSI/IEEE 830 [9]. Lista atrybutów i innych elementów składowych opisu wymagania stanowi bazę do ich specyfikacji w języku XML. Przedstawiono ją w tabeli 4.1 (w nawiasach podano identyfikator praktyki Sommerville'a i Sawyera, która sugeruje istnienie danego atrybutu).

Tabela 4.1: Atrybuty wymagań i ich reprezentacja w języki XML

Atrybuty ogólne – *znacznik* ***requirement***
- identyfikator (P14) - *atrybut* ***id***
- nazwa – *atrybut* ***name***
- priorytet (P10) – *atrybut* ***priority***
- status – *atrybut* ***status***

Informacje o wersji - *znacznik* ***revision***
- numer wersji – *atrybut* ***version***
- etykieta wersji – *atrybut* ***label***
- autor – *atrybut* ***author***
- data – *atrybut* ***date***
- opis zmian – *zawartosc* ***znacznika***

Słowa kluczowe – *znacznik* ***keywords***
Opis – *znacznik* ***description***
Uzasadnienie (Ś1) – *znacznik* ***rationale***
Źródło (P9) – *znacznik* ***source***
Perspektywa – *znacznik* ***viewpoint***

Relacje z innymi - *znacznik* ***relation***
- bazowe (Z1) – *atrybut* ***type=base***
- macierzyste – *atrybut* ***type=parent***
- powiązane (Ś4) – *atrybut* ***type=related***
- wyprowadzone z – ***type=derived-from***
- wyprowadzone – *atrybut* ***type=derived***

Oszacowania ważność – *znacznik* ***estimates***
- koszt – *atrybut* ***cost***
- stabilność – *atrybut* ***stability***
- ryzyko (Z2) – *atrybut* ***risk***
- weryfikowalność – *atrybut* ***verifiability***

Definicja – *znacznik* ***definition***
- wejście – *znacznik* ***input***
- przetwarzanie – *znacznik* ***processing***
- wyjście – *znacznik* ***output***
- przykład – *znacznik* ***sample***
- scenariusz (Ś2) – *znacznik* ***scenario***

Testy (Ś5) – *znacznik* ***test-case***

Przekładając taki opis na format DTD, umieszczając dane w znacznikach lub ich atrybutach oraz uwzględniając reguły zagnieżdżania, uzyskano elastyczny język opisu wymagań. Formułowanie wymagań w tym języku, pomimo jego częściowego sformalizowania, jest ułatwione poprzez automatyczną weryfikację poprawności definicji przy użyciu DTD. Operowanie na tak zdefiniowanym repozytorium wymagań jest możliwe przy pomocy dowolnego narzędzia wspomagającego język XML, jak również przy wykorzystaniu dedykowanych narzędzi wykorzystujących tę strukturę. Przy pomocy XML tworzony jest czytelny dla człowieka, a jednocześnie łatwy do maszynowego przetwarzania opis wymagań. Przykład takiej definicji wymagania przedstawiony jest na rys. 4.1.

```
<requirement
  id="R01"
  name="Autoryzowana personalizacja"
  priority="medium"
  status="draft"
  content="complete">
      <revision
        version="2"
        author="KK"
        date="2001-05-06">
            Poprawione oszacowania
      </revision>
      <keywords>
            <keyword>personalizacja</keyword>
            <keyword>autoryzacja hasłem</keyword>
      </keywords>
      <description>
            Użytkownik powienien mieć możliwość
            personalizacji poprzez autoryzowany
            hasłem dostęp do ustawień
      </description>
      <rationale>
            Poszczególne klasy użytkowników
            korzystają z różnych zestawów opcji
      </rationale>
      <source>E-mail do J.K. z 4 V 2001</source>
      <estimates
        importance="medium"
        cost="medium"
        stability="low"
        risk="medium"
        verifiability="high"/>
      <definition>
            <input>
                  Dane autoryzacyjne użytkownika
            </input>
            <processing>
                  Sprawdzenie poprawności danych
                  autoryzacyjnych i pobranie
                  aktualnych ustawień
            </processing>
            <output>
                  Dostęp do opcji
                  personalizacyjnych
            </output>
      </definition>
</requirement>
```

Rysunek 4.1:Specyfikacja pojedynczego wymagania

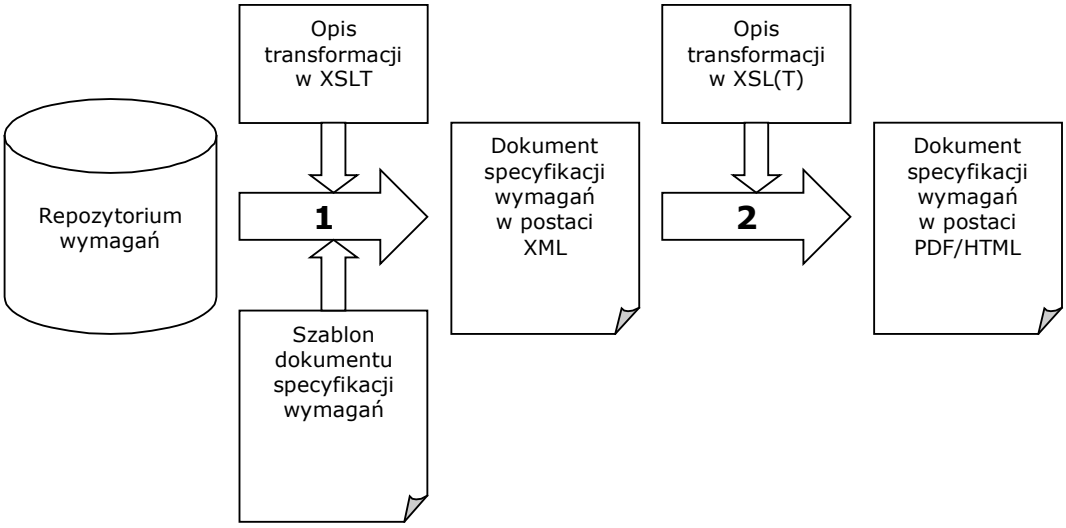
5. Formułowanie standardu dokumentu

Posiadając zdefiniowane przy pomocy XML repozytorium wymagań, następny krok prezentowanego podejścia stanowi wspomaganie konstruowania dokumentu specyfikacji wymagań. Podstawową koncepcją w przyjętym rozwiązaniu jest oddzielenie treści dokumentów od ich reprezentacji wizualnej. Treść dokumentu przechowywana jest w postaci dokumentu XML a warstwa

wizualna zdefiniowana jest przy pomocy opisu w XSL(T). Treść specyfikacji sformułowana jest przy pomocy prostego języka opartego na XML, pozwalającego na wyodrębnienie rozdziałów, akapitów, tytułów i włączanie grafiki. Język ten, jak wspomniano, nic nie mówi o sposobie reprezentacji wizualnej tych elementów.

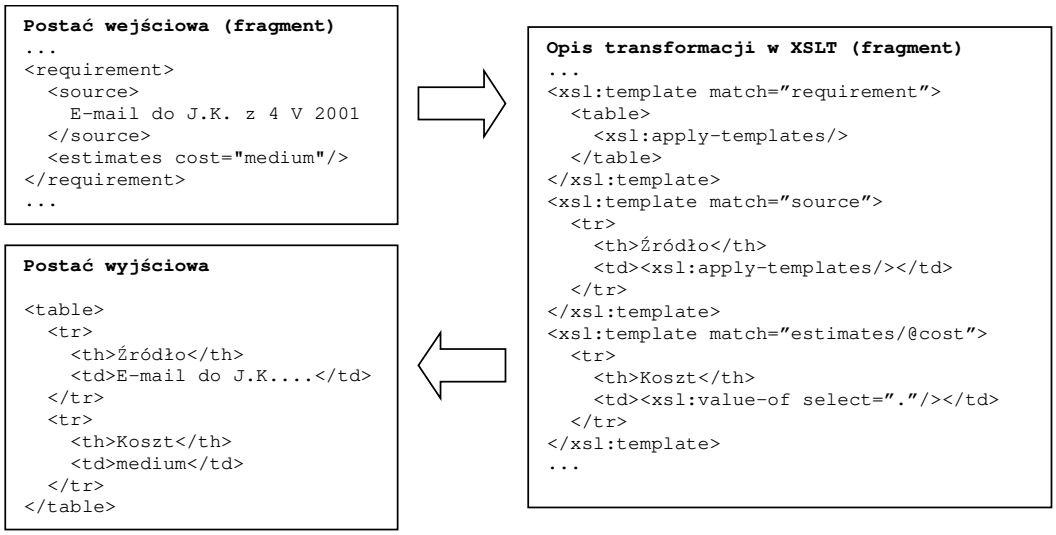
Pierwszym krokiem w konstrukcji specyfikacji wymagań jest zdefiniowanie przy pomocy opisanego powyżej języka szablonu dokumentu specyfikacji wymagań. Można to uczynić modyfikując szablon z wcześniejszego projektu lub definiując całkiem nowy. Szablon ten zawiera sekcje dokumentu, które nie zmieniają się mimo zmian w wymaganiach.

Kolejnym krokiem jest przekształcenie wymagań z repozytorium do postaci fragmentu dokumentu specyfikacji przy pomocy transformacji XSLT. Fragment ten zostaje włączony do szablonu specyfikacji tworząc kompletny dokument w postaci XML (patrz rysunek 5.1). Następnie dokument w XML zostaje, przy pomocy opisu wizualnego w postaci arkusza XSL(T), przetransformowany do postaci PDF (Portable Document Format) lub HTML.



Rysunek 5.1: Schemat generowania dokumentu specyfikacji wymagań

Przykład takiej prostej transformacji z języka opisu wymagania bezpośrednio do reprezentacji w języku HTML przedstawiony jest na rys. 5.2. Postać wejściowa to fragment opisu wymagania w XML. Transformacja opisana jest przy pomocy XSLT definiującego szablony dopasowań i sposób ich przełożenia na postać docelową. Jako postać wyjściową w tym przypadku przyjęto bezpośrednio język HTML. Takiej transformacji dokonać można przy pomocy dowolnego narzędzia zwanego procesorem XSL.



Rysunek 5.2: Przykład transformacji opisu wymagania do postaci HTML

6. Powiązania między wymaganiami

Istotną rolę w efektywnym przetwarzaniu wymagań odgrywa utrzymywanie informacji o powiązaniach pomiędzy nimi. Informacje te pozwalają na śledzenie propagacji zmian (ang. *traceablity*). Właśnie to zagadnienie jest często uważane za jeden z poważniejszych problemów inżynierii wymagań [8]. Oprócz śledzenia powiązań pomiędzy samymi wymaganiami (często ten rodzaj powiązań określa się jako poziome), istotną rolę odgrywa również przechowywanie łączy pomiędzy wymaganiami a fragmentami innych dokumentów projektowych (dokumentacją użytkową, kodem, przypadkami testowymi), które zostały z nich wywiedzione (ten rodzaj określa się jako pionowe).

Zaproponowana koncepcja śledzenia powiązań opiera się na języku XLink (*XML Linking Language*). Pozwala on na definiowanie połączeń pomiędzy zasobami z wykorzystaniem języka XML. Łącza te mogą być bardziej rozbudowane niż klasyczne, znane np. z języka HTML. XLink umożliwia definiowanie powiązań pomiędzy więcej niż dwoma zasobami (łącza takie mogą być dwukierunkowe, a ich spisy mogą się znajdować poza dokumentami, których dotyczą).

Przy pomocy języka XLink możliwe jest zdefiniowanie powiązań pomiędzy wymaganiami (poziomych) poprzez umieszczenie odpowiednich znaczników w opisie wymagania. W ten sposób wyszczególnić można wymagania bazowe (zapożyczone lub zmodyfikowane z innych projektów), macierzyste (nadrzędne w hierarchii), powiązane (semantycznie podobne) oraz wymagania, z których wyprowadzono aktualne, jak i te, które zostały z niego wyprowadzone. Przykład

takiego łącza przedstawiony jest poniżej (łącze do wymagania R01, z którego zostało wyprowadzone aktualne).

```
<relation href="#xpointer(requirement[@id='R01'])" type="derived-from"/>
```

Oprócz powiązań poziomych (pomiędzy wymaganiami) XLink pozwala także na przechowywanie zestawu łączy pionowych odwołujących się do fragmentów innych dokumentów. Łącza takie najlepiej przechowywać w osobnym pliku, aby wyłączyć je z treści dokumentów docelowych. Należy przy tym założyć, że zasoby, do których następować będzie łączenie są zapisane w postaci dokumentów XML. Jeżeli wszystkie dokumenty (specyfikacja wymagań, kod, przypadki testowe) są zapisane w języku XML, to powiązania mogą dotyczyć nie tylko całych dokumentów, ale również ich fragmentów (możliwe jest wtedy precyzyjne adresowanie pojedynczych znaczników w dokumencie).

Przykład fragmentu dokumentu nawiązującego do definicji wymagania przedstawiony jest poniżej (łącze do wymagania R01 z datą sprawdzenia spójności).

```
...<consistency-link
    href="project.xml#xpointer(requirement[@id='R01'])"
    last-validated="2001-05-15 13:20:04">
  ...tekst nawiązujący do definicji wymagania...
</consistency-link >...
```

Należy wspomnieć, że utrzymywanie spójności łączy i danych na jego końcach wymaga zastosowania dodatkowego narzędzia propagującego zmiany wymagania i wyszukującego fragmenty wymagające sprawdzenia spójności.

7. Zakończenie

Podejście do inżynierii wymagań bazujące na języku XML rokuje szanse stworzenia taniego i efektywnego środowiska zarządzania wymaganiami. Otwarta reprezentacja danych umożliwia dostęp do nich z poziomu edytorów tekstowych i przeglądarek internetowych oraz ułatwia przeglądanie jak i modyfikację wymagań. Z drugiej strony pozwala na rozszerzenie zbioru narzędzi o własne, dostosowane do specyficznych zastosowań i potrzeb. Integracja takich narzędzi w jedną aplikację wpłynęłaby na uproszczenie obsługi i podniesienie efektywności zarządzania wymaganiami. Aby w pełni wykorzystać możliwości zaprezentowanego podejścia, należałoby zastosować język XML i związane z nim narzędzia do zarządzania i prezentacji pozostałych artefaktów (projektu, kodu, przypadków testowych).

Omawiane w artykule zagadnienia są szczegółowo opisane w pracy [14]. Zawiera ona pełne przykłady oraz dokładny opis konfiguracji środowiska i użytych narzędzi.

Bibliografia

[1]. I. Sommerville, P. Sawyer, Requirements Engineering: A Good Practice Guide, John Wiley & Sons, Chicheter 1997.

[2]. M.C. Paulk et al., The Capability Maturity Model: Guidelines for Improving the Software Process, Addison-Wesley, Reading Massachusetts 1999.

[3]. Extensible Markup Language (XML) 1.0, W3C Recommendation 10-Feb-98, http://www.w3.org/TR/1998/REC-xml-19980210, 25 V 2001.

[4]. XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16-Nov-1999, http://www.w3.org/TR/1999/REC-xslt-19991116, 25 V 2001.

[5]. Extensible Stylesheet Language (XSL) Version 1.0, W3C Candidate Recommendation 21-Nov-2000, http://www.w3.org/TR/2000/CR-xsl-20001121, 25 V 2001.

[6]. XML Linking Language (XLink), W3C Proposed Recommendation 20-Dec-2000, http://www.w3.org/TR/2000/PR-xlink-20001220, 25 V 2001.

[7]. ISO 8879:1986, Information Processing – Text and Office Systems - Standard Generalized Markup Language (SGML), International Organization for Standardization, Geneva 1986.

[8]. O.C.Z. Gotel, A.C.W. Finkelstein, An Analysis of the Requirements Traceability Problem. http://www.cs.ucl.ac.uk/staff/A.Finkelstein/papers/rtprob.pdf, 25 V 2001

[9]. ANSI/IEEE Std 830-1984, IEEE Guide to Software Requirements Specifications, IEEE, New York 1984.

[10]. Software Engineering Program. Software Process Improvement Guidebook, NASA-GB-001-95, NASA, Washington 1996

[11]. Rational, producent systemu RequisitePro, http://www.rational.com

[12]. StarBase, producent systemu CaliberRM, http://www.tbi.com

[13]. Telelogic AB, producent systemu DOORS, http://www.telelogic.com

[14]. K. Kowalczykiewicz, Dokumentowanie wymagań w języku XML, Praca magisterska, Politechnika Poznańska, Poznań 2001.