ANDRZEJ SOBCZAK
Szkoła Główna Handlowa w Warszawie

ARCHITEKTUROCENTRYCZNY PROCES OPRACOWYWANIA OPROGRAMOWANIA[1]

Streszczenie

Wraz ze wzrostem złożoności struktury systemów oprogramowania w istotny sposób wzrosła rola architektury tych systemów. Jest to związane z odejściem od monolitycznych lub pracujących w trybie klient–serwer rozwiązań na rzecz systemów pracujących w architekturze wielowarstwowej lub architekturze zorientowanej na usługi, a także upowszechnienia się rozwiązań typu COTS. Z tego też powodu zagadnienia związane z architekturocentrycznym procesem opracowywania tych systemów stało się obszarem licznych prac badawczych jak i aplikacyjnych – zarówno w środowisku akademickim, jak i przemyśle. Celem niniejszego artykułu jest zdefiniowanie cech wyróżniających ten proces oraz określenie jego struktury.

Słowa kluczowe: proces wytwarzania oprogramowania, architektura systemu oprogramowania, proces architekturocentryczny

1. Pojęcie procesu opracowywania systemów oprogramowania

Aby powstał system oprogramowania, należy zastosować pewną strategię jego opracowania, określoną mianem procesu wytwórczego. S. Pressman definiuje proces wytwórczy jako schemat wykonywania pewnych zadań, koniecznych do stworzenia dobrego systemu oprogramowania [14]. A. Kobyliński przytacza dwa ujęcia procesu wytwórczego (określanego mianem procesu programowego). Zgodnie z pierwszym ujęciem jest to ogół czynności, metod, praktyk i przekształceń, które wykonują ludzie, by opracowywać i konserwować oprogramowanie oraz inne pokrewne produkty z nim związane (np. plan przedsięwzięcia, projekt, kod, zestawy testów). W drugim ujęciu jest to zbiór procesów stosowanych w organizacji lub w trakcie realizacji przedsięwzięcia [tutaj: podczas opracowywania systemów oprogramowania – przypis autora], aby planować, zarządzać, realizować, monitorować, kontrolować i doskonalić działania związane z wytwarzaniem oprogramowania [9]. P. Kruchten określa proces wytwórczy jako uporządkowane przydzielanie zadań i zakresów odpowiedzialności w celu doprowadzenia do powstania systemu oprogramowania charakteryzującego się wysoką jakością, spełniającego potrzeby swoich użytkowników końcowych w ramach przewidywanego harmonogramu i budżetu. Proces powinien opisywać „kto" robi „co", „jak" i „kiedy" podczas opracowywania systemów oprogramowania [11].

1 Artykuł został przygotowany w ramach realizacji projektu badawczego Ministerstwa Nauki i Szkolnictwa Wyższego nr N115 010 32/01443.

2. Definicyjne ujęcie architektury systemu oprogramowania

M. Flasiński wskazuje na złożoność jako podstawowe źródło problemów występujących w trakcie konstrukcji systemów oprogramowania. Uważa on, że po przekroczeniu pewnego progu złożoności przestaje się panować nad przygotowanym systemem, ponieważ całościowe jego zrozumienie – w różnych jego aspektach i na dowolnym poziomie szczegółowości – przekracza możliwości człowieka [5]. P. Kruchten zauważa wręcz: „obecnie, gdy wszystkie proste systemy są już zbudowane, złożoność wielkich systemów stała się problemem numer jeden dla przedsiębiorstw tworzących oprogramowanie" [11].

Architektura systemu – w ujęciu ogólnym – definiowana jest jako struktura (lub zbiór struktur) tego systemu składająca się z określonych elementów, widzianych z zewnątrz właściwości tych elementów oraz związków między nimi. Na potrzeby niniejszego artykułu rozpatrywana jest architektura systemów oprogramowania. Pozwala ona zapanować nad złożonością tych systemów [17].

I. Graham stawia tezę, że „nie ma dotychczas jasnej i jednoznacznej definicji tego, czym jest architektura systemu oprogramowania" [6]. Najlepszym na to dowodem jest fakt zgromadzenia na stronie internetowej Software Engineering Institute Carnegie Mellon University blisko 50 definicji – zaczerpniętych z literatury oraz zaproponowanych przez internautów – terminu „architektura systemu oprogramowania".

Standard IEEE 1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems definiuje architekturę systemu oprogramowania jako: fundamentalną organizację systemu zawierającą w sobie: elementy, relacje między tymi elementami oraz środowisko i pryncypia projektowania i ewolucji tej organizacji. Przy czym: „fundamentalna organizacja" oznacza zasadnicze, ujednolicone koncepcje i zasady; określenie „system" obejmuje system oprogramowania, platformę, linię produktową; środowisko rozumiane jest jako projektowy, operacyjny lub programistyczny kontekst systemu.

L. Bass, P. Clements i R. Kazman definiują architekturę systemu oprogramowania jako strukturę (lub zbiór struktur) tego systemu, która składa się z określonych elementów, widzianych z zewnątrz właściwości tych elementów oraz związków między nimi [1].

W. Chmielarz definiuje architekturę systemu jako układ współpracujących ze sobą następujących elementów: struktury oprogramowania, fizycznej struktury danych, mechanizmów gwarantujących bezpieczeństwo danych . Dodatkowo W. Chmielarz zauważa, że: „pełna specyfikacja problematyki architektury systemu [informatycznego – przypis autora] wymaga jeszcze ponadto uwzględnienia w rozważaniach zagadnień sprzętowych oraz związanych z organizacją procesu przetwarzania" [3].

P. Kroll i P. Kruchten rozpatrują natomiast architekturę jako zbiór znaczących decyzji dotyczących struktury systemu, czyli: wyboru elementów strukturalnych i ich interfejsów, z których składa się system, zachowań określających współpracę tych elementów, zorganizowania tych elementów strukturalnych i behawioralnych w większe podsystemy, stylu architektonicznego, który ma wpływ na kształt tej struktury [10]. P. Kruchten pisze także, że „architektura opisana w całości jest czymś wielowymiarowym – architektura nie jest płaska" [11].

Na architekturę systemu oprogramowania (zanim sam system powstanie) można wreszcie patrzeć jak na przedfizyczną formę jego istnienia. Architektura stanowi także odwzorowanie (reprezentację) wiedzy jej projektanta.

3. Projektowanie architektury w architekturocentrycznym procesie opracowywania systemów oprogramowania

L. Bass, P. Clements i R. Kazman zauważają, że każdy system oprogramowania ma określoną architekturę, ponieważ „każdy system może być pokazany jako składający się z elementów i powiązań między nimi" [1]. Niezbędne jest jednak zaprojektowanie tej architektury w taki sposób, aby system zrealizowany na jej podstawie spełniał określone kryteria jakościowe, ilościowe i funkcjonalne. Istotny wpływ na powodzenie w tym obszarze ma wybór procesu wytwórczego. Jak pisze bowiem A. Kobyliński: „istnieje oczywisty związek między procesami a produktami. (...) Bez procesu programowego – niezależnie od tego, jak on jest zorganizowany – chaotycznie i bezwładnie czy też dojrzale i w sposób uporządkowany, nie byłoby produktu (....). Problem tkwi w tym, w jaki sposób zorganizować proces programowy, aby uzyskany produkt spełniał wymagania jakościowe, a ponadto był skończony w terminie i w ramach założonego budżetu" [9]. Dlatego też decyzja o wykorzystaniu określonego procesu jest jednym z istotniejszych elementów przedsięwzięcia i zależy od wielu czynników. Do najważniejszych zaliczyć można rodzaj powstającego rozwiązania, używane w organizacji metody i narzędzia do opracowywania systemów oprogramowania, a także oczekiwania dotyczące produktów roboczych związanych z procesem wytwórczym.

Bazując na tezie L. Bassa, że „w procesie opracowywania systemów oprogramowania architektura jest artefaktem o największym znaczeniu; stanowi ona podstawę do osiągania celów biznesowych i pożądanych atrybutów jakościowych", proponuje się zastosowanie procesu opracowywania systemów oprogramowania określonego mianem architekturocentrycznego [1].

A. Bucchiarone, H. Muccini i P. Pelliccione definiują proces architekturocentryczny jako ten, w którym specyfikacja architektury systemu jest walidowana na bazie odpowiednich wymagań, a następnie zwalidowana architektura stanowi punkt wyjścia do jakichkolwiek innych analiz [2].

L. Bass, P. Clements i R. Kazman wskazują, że w procesie architekturocentrycznym architektura stanowi jego centralny element [1].

Według C. Hofmeister, R. Nord i D. Soni proces architekturocentryczny polega na użyciu architektury oprogramowania do organizowania czynności składających się na opracowanie tego oprogramowania [7].

H. Proper określa, że system jest tworzony w procesie architekturocentrycznym, jeżeli jego architektura jest używana do [15]:

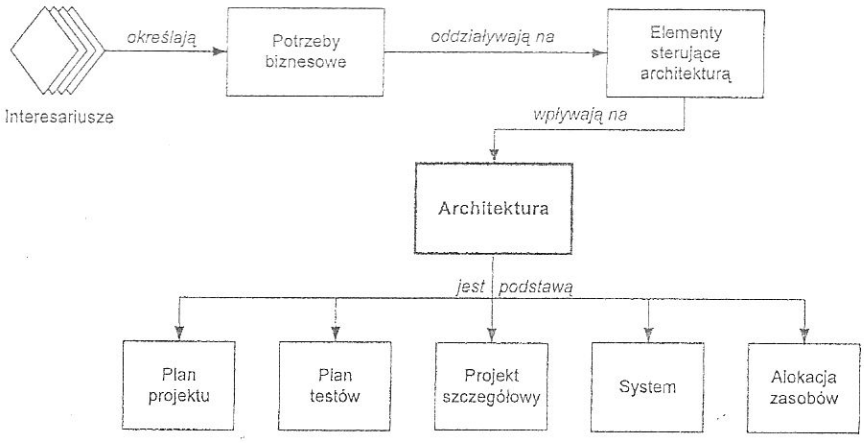
- komunikacji i negocjacji pomiędzy interesariuszami,
- ewaluacji i porównania decyzji architektonicznych,
- planowania i zarządzania rozwojem systemu,
- przeprowadzenia weryfikacji powstałego rozwiązania z decyzjami architektonicznymi.

A. Lattanze zauważa, że w architekturocentrycznym procesie opracowywania systemów oprogramowania architektura stanowi „zewnętrze i centrum" podczas wszystkich faz tego procesu. Na jej podstawie definiowane są wszystkie kolejne działania, artefakty, role [12].

Ponadto A. Lattanze identyfikuje dobre praktyki (ang. best practices) dla procesu architekturocentrycznego [12]:

- Na bazie potrzeb interesariuszy zdefiniuj architekturę tak wcześnie, jak jest to możliwe.
- Utwórz, udoskonalaj i aktualizuj architekturę w iteracyjny sposób podczas całego cyklu życia systemu.

- Waliduj architekturę, czy spełnia ona oczekiwania interesariuszy.
- Zdefiniuj znaczące role dla członków zespołu zajmujących się architekturą.
- Twórz estymacje i harmonogramy, korzystając z architektury.
- Wykorzystuj architekturę do zwiększenia wydajności prac nad systemem.
- Zapewnij, aby przyjęty proces opracowywania systemów był lekki, skalowalny, możliwy do dopasowania i powtarzalny.

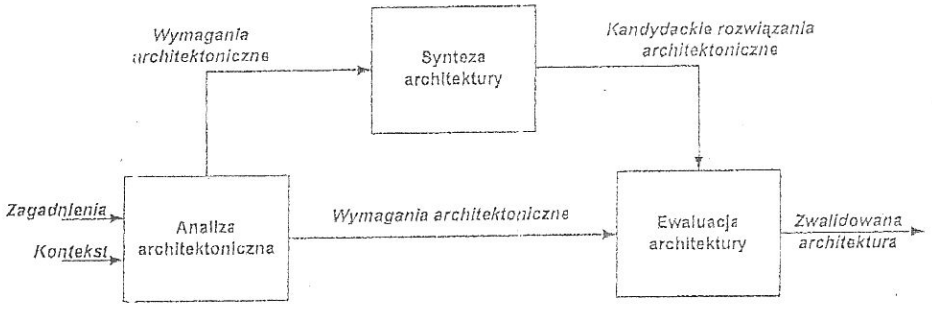


Rys. 1. Umiejscowienie architektury w architekturocentrycznym procesie opracowywania systemów oprogramowania

Źródło: A. Lattanze, The Architecture Centric Development Method, Report no. CMU-ISRI-05-103, School of Computer Science, Carnegie Mellon University, February 2005, s. 56.

Kluczowym elementem architekturocentrycznego procesu opracowywania systemów jest zaprojektowanie architektury. C. Hofmeister. P. Kruchten, R. Nord, H. Obbink, A. Ran i P. America opracowali generyczny model projektowania architektury systemów (por.: rys. 2).
W modelu tym podczas analizy architektonicznej następuje zdefiniowanie problemów, które muszą zostać uwzględnione w czasie prac nad architekturą, oraz określenie wymagań architektonicznych (czyli nie wszystkich, lecz takich, które mają wpływ na architekturę). Odbywa się to na bazie określonych wcześniej zagadnień (ang. concerns) i kontekstu. Przez zagadnienia rozumie się te okoliczności i aspekty, które są kluczowe dla jednego lub więcej interesariuszy podczas opracowywania architektury. Najczęściej zagadnienia te mają postać wymagań (np. wymagań niefunkcjonalnych dotyczących wydajności, bezpieczeństwa itp.), ale można tutaj uwzględnić obligatoryjne decyzje architektoniczne, przepisy prawne. Kontekst ujmuje środowisko (organizacyjne, polityczne, stan i dostępność technologii), w jakim realizowany jest system. Kryteria rozróżnienia między zagadnieniami a kontekstem mają często rozmytą postać. Zagadnienie dotyczy konkretnego systemu, natomiast kontekst związany jest z ogólną charakterystyką organizacji lub interesariuszy. Synteza architektury jest kluczowym działaniem w ramach projektowania architektury. Następuje tutaj – na bazie wymagań architektonicznych – wygenerowanie kandydackich rozwiązań architektonicznych. Kandydackie rozwiązania

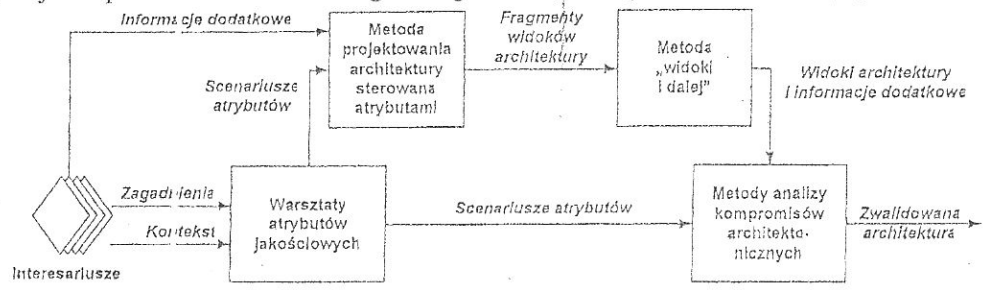
architektoniczne obejmują alternatywne rozwiązania lub częściowe rozwiązania określonych problemów (w formie fragmentów architektury). Decydują one o przyszłej strukturze systemu. Rozwiązania architektoniczne zawierają dodatkowo informacje o podjętych decyzjach, uzasadnienia tych decyzji, i powiązaniach decyzji z wymaganiami. W ramach ewaluacji architektury następuje sprawdzenie, które rozwiązania architektoniczne są spójne z wymaganiami architektonicznymi oraz czy są one wzajemnie spójne. Jedynie taki zbiór rozwiązań może być prezentowany jako zwalidowana architektura. Niezbędne jest jej uzupełnienie o informacje dotyczące uzasadnienia podjęcia określonych decyzji w zakresie oceny walidowanych rozwiązań [8].



Rys. 2. Generyczny model projektowania architektury systemów

Źródło: C. Hofmeister, P. Kruchten, R. Nord, H. Obbink, A. Ran, P. America, A general model of software architecture design derived from five industrial approaches, „The Journal of Systems and Software", vol. 80, issue 1, January 2007, s. 113.

Przykładem realizacji generycznego modelu projektowania architektury systemów jest podejście opracowane w Software Engineering Institute, Carnegie Mellon University (por.: rys. 3).



Rys. 3. Podejście opracowane w Software Engineering Institute w zakresie projektowania architektu systemów

Źródło: Opracowanie własne na podstawie: C. Hofmeister, P. Kruchten, R. Nord, H. Obbink, A. Ran, P. America, A general model of software architecture design derived from five industrial approaches, „The Journal of Systems and Software", vol. 80, issue 1, January 2007, s. 107–18.

Pierwszym krokiem w ramach tego podejścia jest zrealizowanie pierwszej części warsztatów atrybutów jakościowych (ang. Quality Attribute Workshop – QAW). Pozwala to na zrozumienie problemów poprzez wydobycie atrybutów jakościowych wymagań za pomocą tzw. scenariuszy atrybutów. Scenariusze te identyfikowane są za pomocą „burzy mózgów", w której uczestniczą główni interesariusze. W wyniku tej operacji powstaje około 30–40 scenariuszy, które podlegają następnie priorytetyzacji. Scenariusze o najwyższym priorytecie są uściślane w celu pełniejszego uchwycenia kontekstu oraz dookreślenia związanych z nimi szczegółów. Kolejnym etapem jest zastosowanie metody projektowania architektury sterowanej atrybutami (ang. Attribute-Driven Design – ADD). Projektowanie realizowane za pomocą tej metody bazuje na informacjach wejściowych przygotowanych w formie scenariuszy atrybutów oraz na znajomości relacji między realizacją tych atrybutów a architekturą. Główne kroki metod ADD obejmują następujące czynności:

1. Wybór modułu, który będzie podlegał dekompozycji. Przy pierwszej iteracji modułem jest cały tworzony system.
2. Udoskonalanie modułu zgodnie z poniższymi krokami:

a. Wybór kluczowych, dla danej dekompozycji, zagadnień ze zbioru scenariuszy jakościowych wymagań funkcjonalnych.
b. Wybór odpowiednich w danej sytuacji taktyk i wzorców architektonicznych. Rozpoznanie modułów potomnych, wymaganych do realizacji tych taktyk.
c. Uściślenie modułów potomnych za pomocą przypadków użycia i przedstawienie ich za pomocą wybranych perspektyw architektonicznych.
d. Określenie interfejsów modułów potomnych i charakterystyki interakcji między nimi.
e. Zweryfikowanie i udoskonalenie przypadków użycia oraz scenariuszy jakościowych. Sformułowanie ograniczeń na moduły potomne. Podjęcie decyzji o dalszej dekompozycji modułów lub ich skierowaniu do implementacji.

3. Powtarzanie etapu 2 dla wszystkich modułów wymagających dalszej dekompozycji.

Powstała architektura jest dokumentowana na bazie metody „widoki i dalej" (ang. Views and Beyond – VaB). Metoda ta zakłada, że dokumentowanie architektury polega na dokumentowaniu istotnych widoków, a następnie uzupełnianiu ich o opisy, które mają zastosowanie do więcej niż jednego widoku. P. Clements zaproponował podział widoków na następujące grupy [4]: widoki modułów (pokazujące, jak system jest ustrukturalizowany jako zbiór modułów kodu), widoki komponentów i konektorów (przedstawiające, jak system jest ustrukturalizowany jako zbiór komponentów wykonywalnych), widoki alokacji (ilustrujące, jak część programowa systemu jest powiązana z częścią nieprogramową). W podejściu tym każdy widok jest prezentowany w wystandaryzowany sposób (podstawowa prezentacja, katalogi elementów i ich relacji, interfejsów oraz zachowań, diagram kontekstowy, podstawowe informacje o architekturze kluczowe dla danego widoku). Realizacja słowa „dalej" w nazwie metody polega na uzupełnieniu dokumentacji architektury m. in. o ogólny opis systemu, przedstawienie szablonu do prezentowania widoków wraz z uzasadnieniem ich zastosowania, mapowanie widoków, słownik, objaśnienie decyzji architektonicznych dotyczących więcej niż jednego widoku. Wreszcie powstała architektura ewaluowana jest za pomocą metody analizy kompromisów architektonicznych (ang. Architecture Tradeoff Analysis Method – ATAM).

4. Podsumowanie

Przedstawione w niniejszym artykule rozważania wskazują, że wraz ze wzrostem stopnia skomplikowania struktury systemów oprogramowania w istotny sposób wzrosła rola architektury tych systemów. Jest to związane z odejściem od monolitycznych lub pracujących w trybie klient–serwer rozwiązań na rzecz systemów pracujących w architekturze wielowarstwowej lub architekturze zorientowanej na usługi, a także upowszechnienia się rozwiązań typu COTS. Tym samym można sformułować wniosek, że jednym z najważniejszych elementów procesu opracowywania systemów oprogramowania, który w istotnym stopniu wpływa na przydatność dla organizacji budowanych systemów, jest właściwe zaprojektowanie ich architektury.

Jak piszą M. Shaw i P. Clements, dla architekturocentrycznego podejścia „nadszedł złoty wiek" [16]. Z jednej strony jest ono wystarczająco dojrzałe (zgodnie z sześciostopniowym modelem Redwine'a i Riddle'a, dotyczącym opracowywania i propagacji koncepcji w inżynierii oprogramowania, architektura oprogramowania znajduje się obecnie na szóstym, najbardziej dojrzałym poziomie), a z drugiej strony upowszechnienie się rozwiązań e-commerce wpłynęło na konieczność tworzenia rozwiązań n-warstwowych, budowania systemów w architekturze zorientowanej na usługi, wykorzystywania komponentów COTS, opracowywania języków i narzędzi pozwalających implementować techniki i style architektoniczne. Jednocześnie w środowisku osób zajmujących się tworzeniem systemów informatycznych wzrosła świadomość roli zagadnień architektonicznych – bezpośrednim tego wyrazem może być powołanie Worldwide Institute of Software Architects i International Association of Software Architect. Można więc sformułować drugi wniosek, że obecnie architekturocentryczny proces opracowywania systemów oprogramowania stał się jednym z dominujących paradygmatów w inżynierii oprogramowania.

5. Literatura

1. Bass L. Clements P., Kazman R.: Architektura oprogramowania w praktyce, Wydawnictwa Naukowo-Techniczne, Warszawa 2006.
2. Bucchiarone A., Muccini H., Pelliccione P.: A Practical Architecture-Centric Analysis Process. W: *Proceedings of the QoSA 2006*, Lecture Notes In Computer Science, vol. 4214, Springer-Verlag, 2006.
3. Chmielarz W.: Systemy informatyczne wspomagające zarządzanie, Aspekt modelowy w budowie systemów, Dom Wydawniczy Elipsa, Warszawa 1996.
4. Clements P., Kazman R., Klein M.: Architektura oprogramowania. Metody oceny oraz analiza przypadków, Helion, Gliwice 2003.
5. Flasiński M.: Wstęp do analitycznych metod projektowania systemów oprogramowania, Wydawnictwa Naukowo-Techniczne, Warszawa 1997.
6. Graham I., O'Callaghan A., Wills A.: Metody obiektowe w teorii i praktyce, Wydawnictwa Naukowo-Techniczne, Warszawa 2004.
7. Hofmeister C., Nord R., Soni D.: Tworzenie architektury oprogramowania, Wydawnictwa Naukowo-Techniczne, Warszawa 2007.
8. Hofmeister C., Kruchten P., Nord R., Obbink H., Ran A., America P.: A general model of software architecture design derived from five industrial approaches. W: *The Journal of Systems and Software*, vol. 80, issue 1, January 2007, s. 106–126.

9. Kobyliński A.: Modele jakości produktów i procesów programowych, Szkoła Główna Handlowa w Warszawie, Warszawa 2005.
10. Kroll P., Kruchten P.: Rational Unified Process od strony praktycznej, Wydawnictwa Naukowo-Techniczne, Warszawa 2007.
11. Kruchten P.: Rational Unified Process od strony teoretycznej, Wydawnictwa Naukowo-Techniczne, Warszawa 2007.
12. Lattanze A.: The Architecture Centric Development Method, Report no. CMU-ISRI-05-103, School of Computer Science, Carnegie Mellon University, February 2005.
13. Paulish D.: Zarządzanie architekturocentrycznym procesem tworzenia oprogramowania, Wydawnictwa Naukowo-Techniczne, Warszawa 2007, s. XI.
14. Pressman S.: Praktyczne podejście do inżynierii oprogramowania, Wydawnictwa Naukowo-Techniczne, Warszawa 2004.
15. Proper E.: Architecture-driven Information System Development – Toward a framework for understanding. W: *Proceedings of the 7th World Multi Conference On Systemics, Cybernetics and Informatics*, 27–30 July 2003, Orlando, Florida, USA 2003.
16. Shaw M., Clements P.: The Golden Age of Software Architecture, Software, IEEE, March–April 2006, vol. 23, issue 2, s. 34–35.
17. Targowski A.: Strategia i architektura systemów informatycznych przedsiębiorstwa w gospodarce rynkowej, Nowe Wydawnictwo Polskie, Warszawa 1992.

ARCHITECTUROCENTRIC PROCESS OF SOFTWARE DEVELOPMENT

Summary

Software architecture has been widely used to describe the design of a software system. This paper has been focus on the process of software system development. The approach is discussed in relation to other existing methods of systems' developments.

Keywords: software system architecture, process of software system development, architecture-centric process

Andrzej Sobczak
Katedra Informatyki Gospodarczej
Szkoła Główna Handlowa w Warszawie
02-554 Warszawa, al. Niepodległości 162
e-mail: sobczak@sgh.waw.pl
http://kig.sgh.waw.pl