INŻYNIERIA OPROGRAMOWANIA W PROCESACH INTEGRACJI SYSTEMÓW INFORMATYCZNYCH
Pod redakcją J. Górskiego, C. Orłowskiego, 2010 PWNT Gdańsk

PĄCZKOWANIE – METODA ROZWOJU INTEROPERACYJNYCH KOMPONENTÓW DLA SYSTEMÓW ROZPROSZONYCH

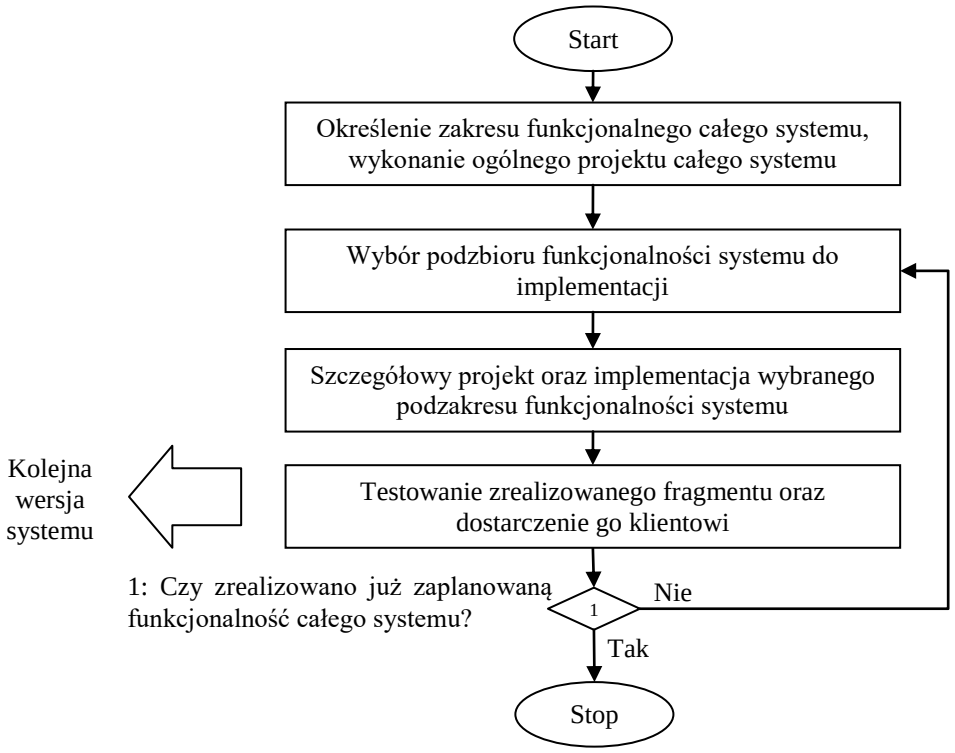
Henryk KRAWCZYK[*], Paweł LUBOMSKI[**]

1. Wprowadzenie

Wraz z rozwojem i popularyzacją interoperacyjnych systemów rozproszonych będących odpowiedzią na szybko zmieniające się wymagania związane z informatyzacją kolejnych sfer życia człowieka, pojawia się nowe wyzwanie dotyczące rozwoju tak specyficznego oprogramowania. W niniejszym artykule zaproponowano oryginalne podejście do szybkiego wytwarzania aplikacji internetowych. Podejście to jest rozwinięciem modelu iteracyjno-przyrostowego z uwzględnieniem technik zwinnych (ang. agile), będące analogią do rozwoju obserwowanej przyrody. Stąd też nazwa: pączkowanie, która bardzo dobrze oddaje ideę metody. Zaproponowano również narzędzia wspomagające i usprawniające tego typu rozwój oprogramowania oraz przedstawiono przykład jej wykorzystania w przypadku bardzo złożonego systemu informatycznego.

Najbardziej popularnym współczesnym modelem rozwoju oprogramowania jest model iteracyjno-przyrostowy [9]. Zakłada on kolejne etapy, w których wytwarzany jest system. Docelowa funkcjonalność systemu podzielona jest na poszczególne części. W ramach każdej iteracji dokładana jest nowa część. Aby wytworzyć dojrzały system potrzeba przeważnie 3 – 4 obiegów pętli przedstawionej na rys. 1. Możliwe jest stosowanie modelu kaskadowego [8] w każdej iteracji, co jest bardzo częstą praktyką w wielu firmach. Model ten jest bardzo popularny w przemyśle, ponieważ skraca czas dostarczenia pierwszych wyników klientowi oraz zebrania informacji zwrotnej o oczekiwaniach klienta. Kolejne nowe funkcjonalności czasami wprowadzane są nawet na zasadzie prototypowania.

[*] Politechnika Gdańska, Wydział Elektroniki Telekomunikacji i Informatyki, Katedra Architektury Systemów Komputerowych, e-mail: hkrawk@eti.pg.gda.pl.
[**] Politechnika Gdańska, Centrum Usług Informatycznych, e-mail: lubomski@pg.gda.pl.



Rys. 1. Schemat iteracyjno-przyrostowej metody wytwarzania oprogramowania

Wypracowano również techniki zwinne (ang. agile) [14] lub extreme programming [15]. Polegają one nie tyle na przestrzeganiu ścisłych założeń realizacji procesu wytwarzania, co na ograniczeniu się do przyjęcia ustalonych strategii postępowania. Ich mocną stroną jest czas potrzebny na wytworzenie gotowego produktu. Jest on zdecydowanie krótszy niż we wspomnianym wcześniej modelu iteracyjno-przyrostowym. Rozwój oprogramowania opiera się na skupieniu się na samej produkcji oprogramowania przy częstej interakcji z klientem. Interakcja ta umożliwia łatwiejsze wprowadzanie zmian w wymaganiach dotyczących systemu. Metody te jednak zakładają pracę w małych zespołach programistycznych i przeznaczone są dla mniejszych rozwiązań.

Analizując powyższe modele, żaden z nich nie wspiera w pełni rozwoju dużych środowisk rozproszonych charakteryzujących się dużą zmiennością wymagań. Brak jest rozwiązania, które posiadałoby pożądane cechy obu. Lukę tą zapełnia zaproponowana metoda rozwoju oprogramowania przez pączkowanie.

2. Metoda rozwoju oprogramowania przez pączkowanie

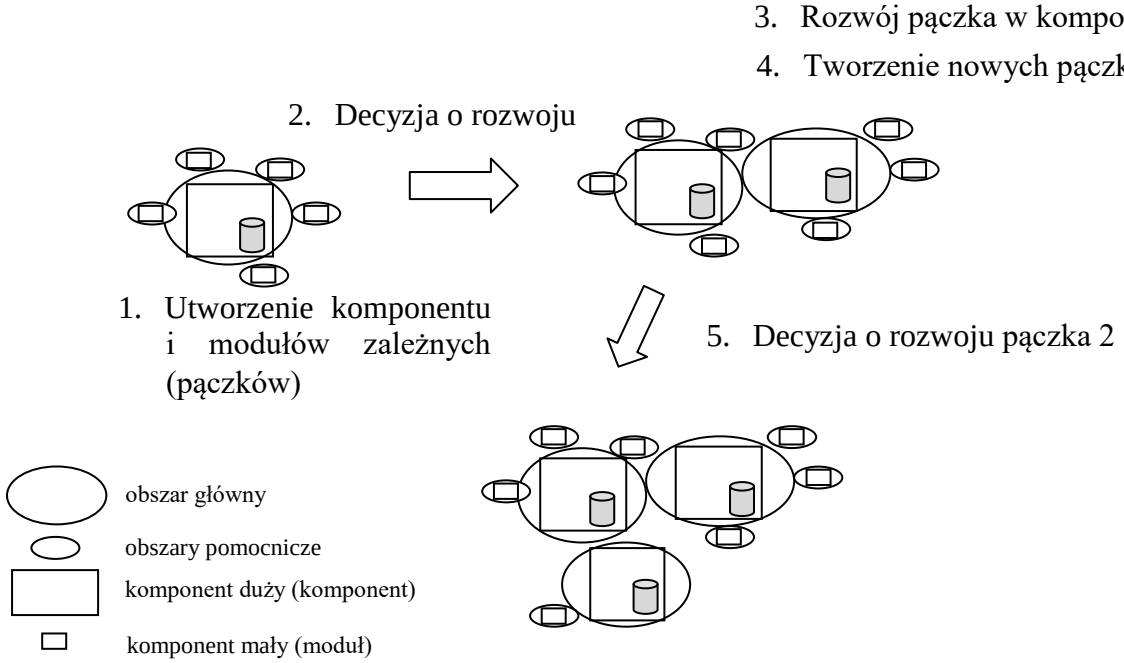
Zaprezentowane rozwiązanie nawiązuje do ewolucjonizmu w przyrodzie. Gatunek, chcąc przetrwać, tak ewoluuje, że w danym momencie rozwija się obszar/cecha najbardziej w danej chwili istotna dla przetrwania populacji biologicznej. W ten sposób organizmy przystosowują się do zmiennych warunków zewnętrznych. Można powiedzieć, że ich rozwój jest sterowany przez czynniki zewnętrzne.

W rozproszonych interoperacyjnych systemach komputerowych trudno rozwijać w danym momencie wszystkie elementy jednocześnie. Proponowana metoda stanowi pewną modyfikację modelu iteracyjno-przyrostowego z wykorzystaniem zalet programowania zwinnego. Zakłada ona rozwój w danym momencie tylko jednego obszaru funkcjonalności systemu, ale w szerokim kontekście jej wykorzystania. Odpowiada on obszarowi, który w danym

momencie wymaga pilnie informatyzacji. Ponieważ mówimy jednak o interoperacyjnych systemach, nie możemy tworzyć „wysepek". Cały rozwój poszczególnych elementów musi być zgodny z wcześniej określoną architekturą i zaplanowaną modułowością platformy [1].

Często rozwijając jeden obszar funkcjonalności wymagany jest rozwój innych obszarów z nim powiązanych. Czynimy to w stopniu minimalnym. Tak powstają pączki. W danej iteracji koncentrujemy się przede wszystkim na obszarze głównym (najintensywniej rozwijamy odpowiadający mu komponent), a moduły/komponenty odpowiadające obszarom zależnym rozwijamy w stopniu minimalnym, jaki to wymagany jest przez główny komponent. W kolejnej iteracji bardzo możliwe, że kluczowym stanie się jeden z obszarów zależnych i wówczas w pełni rozwijamy odpowiadający mu komponent. Tego typu proces przedstawiono na rys. 2. Przypomina on proces namnażania drożdży, stąd nazwa pączkowanie. Warto zauważyć, że czasami jeden z komponentów odpowiadający obszarowi pobocznemu, który był już w pewnym stopniu rozwinięty, może zostać całkowicie skasowany. Tak więc w danej iteracji rozwijane są zawsze: obszar główny i częściowo pewne obszary zależne od niego.

Takie podejście pozwala skupić się tylko na obszarze najbardziej kluczowym w danym momencie dla klienta. Nie wymaga się z góry znajomości docelowej funkcjonalności całego systemu rozproszonego. Pozwala również na skrócenie czasu dostarczenia klientowi kolejnej wersji (ang. time-to-market).



Rys. 2. Schemat procesu rozwoju systemów przez pączkowanie.

Nawiązując do wcześniej wspomnianej modularnej architektury systemu, możemy powiedzieć, że poszczególne obszary funkcjonalne odpowiadają dobrze określonym komponentom. Otrzymujemy zatem duże i małe, w sensie funkcjonalności, moduły. Małe komponenty (pączki) realizują tylko niezbędne funkcje, duże zaś ukierunkowane są na uniwersalność i kompletność funkcjonalną, obejmującą pełny zakres działania i dodatkowe wymagania niefunkcjonalne. W większości przypadków małe komponenty przeradzają się z czasem w duże, zyskując nową funkcjonalność. Możliwe jest, że duży komponent powstaje przez integrację kilku mniejszych.

Istotną sprawą jest rozmiar komponentu dużego i małego (pączka) oraz ich funkcjonalność. Pączek, zalążek nowego komponentu, pojawia się, gdy chcemy rozwinąć nową funkcjonalność, która nie mieści się w zakresie funkcjonalności komponentu lub spowoduje niepożądany

nadmierny jego rozrost. Przykładem może być ewidencja pracowników – główny komponent, który wymaga dodatkowo mniejszych: słowników dotyczących nazw jednostek organizacyjnych, podziału administracyjnego państwa, stopni i tytułów naukowych, itp. Wymienione mniejsze komponenty (pączki) mogą być współużytkowane przez inne komponenty (zapewnienie interoperacyjności). Z kolei pączek uwzględniający zasady awansowania pracowników może prowadzić do utworzenia nowego dużego komponentu, zawierającego całą historię awansów pracownika.

Warto zauważyć, że w odróżnieniu od monolitycznych systemów, które posiadają jedną logiczną bazę danych, w przypadku systemów rozproszonych każdy komponent posiada swoją, zawierającą wycinek danych wynikający z funkcjonalności, jaką realizuje (na rys. 2 oznaczone symbolami). Bazy te są powiązane ze sobą tylko logicznie, co pozwala na zastosowanie różnych technologii, rozproszenia i wykorzystania innych zalet wynikających z usługowego charakteru rozwiązania.

3. Wykorzystanie istniejących technologii

Rozproszenie środowiska docelowego determinuje podział całej platformy na mniejsze systemy – interoperacyjne usługi [4]. Podążając tym tokiem rozumowania zasadne jest przyjęcie modelu usługowego budowy poszczególnych komponentów.

Analizując model rozwoju oprogramowania przez pączkowanie warto zauważyć, że poszczególne komponenty (usługi) w każdej iteracji różnią się od siebie mniej lub bardziej. Naturalnym więc rozwiązaniem jest przyjęcie wersjonowania poszczególnych komponentów. Zarządzanie tak dużą ilością komponentów oraz zależnościami pomiędzy nimi z uwzględnieniem wielu wersji powoduje dużo problemów. Konieczne jest zastosowanie narzędzi i metodologii wspierającej.

Jeżeli potraktujemy tym razem komponenty jako organizmy należące do populacji, to idealnym środowiskiem rozwoju dla nich są linie produkcyjne (ang. Software Product Line – SPL) [7]. Ideą tej metodologii jest zarządzalny proces wytwarzania serii podobnego oprogramowania w oparciu o wielokrotne użycie (ang. reusing) tych samych komponentów [6] o pewnej, z góry założonej wariantowości. Zastosowanie SPL jako środowiska wytwarzania usług ułatwia nie tylko zarządzanie wersjami komponentów – pozwala ona na wprowadzenie niezbędnych metryk, aby mierzyć jakość powstającego produktu. W ten sposób proces wytwarzania oprogramowania możemy zrównać z innymi procesami inżynieryjnymi, jak np. produkcja samochodów. Jakość końcowego produktu może być mierzalna i porównywalna [7].

Warto zauważyć, że wielokrotne wykorzystanie istniejących już komponentów przyspiesza znacząco czas wytwarzania nowych modułów, a więc i czas ukazania się kolejnej wersji dla klienta (ang. time-to-market) [1], co jest niezwykle istotne w przypadku szybko zmieniającego się środowiska zewnętrznego.

Poszczególne komponenty pozostają w ścisłej zależności między sobą. W celu ich integracji stosowana jest metodologia integracji aplikacji biznesowych (ang. Enterprise Application Integration [3]). To światowy trend w procesie budowania oprogramowania. Jak widać po ostatnich wynikach badań, kryzys finansowy zwiększa popyt na rozwiązania ukierunkowane na minimalizację kosztów operacyjnych i optymalizację procesów gospodarczych. Takim rozwiązaniem jest EAI – określająca plany, metody i narzędzia pozwalające na modernizację, konsolidację i koordynację aplikacji komputerowych. Połączenie istniejących usług i programów z nowopowstającymi produktami w jeden spójny system pozwala na zwiększenie oferty i konkurencyjności organizacji. Integracja może zostać wykonana na wielu poziomach: danych, aplikacji, logiki biznesowej i prezentacji oraz z wykorzystaniem rozmaitych technologii [4].

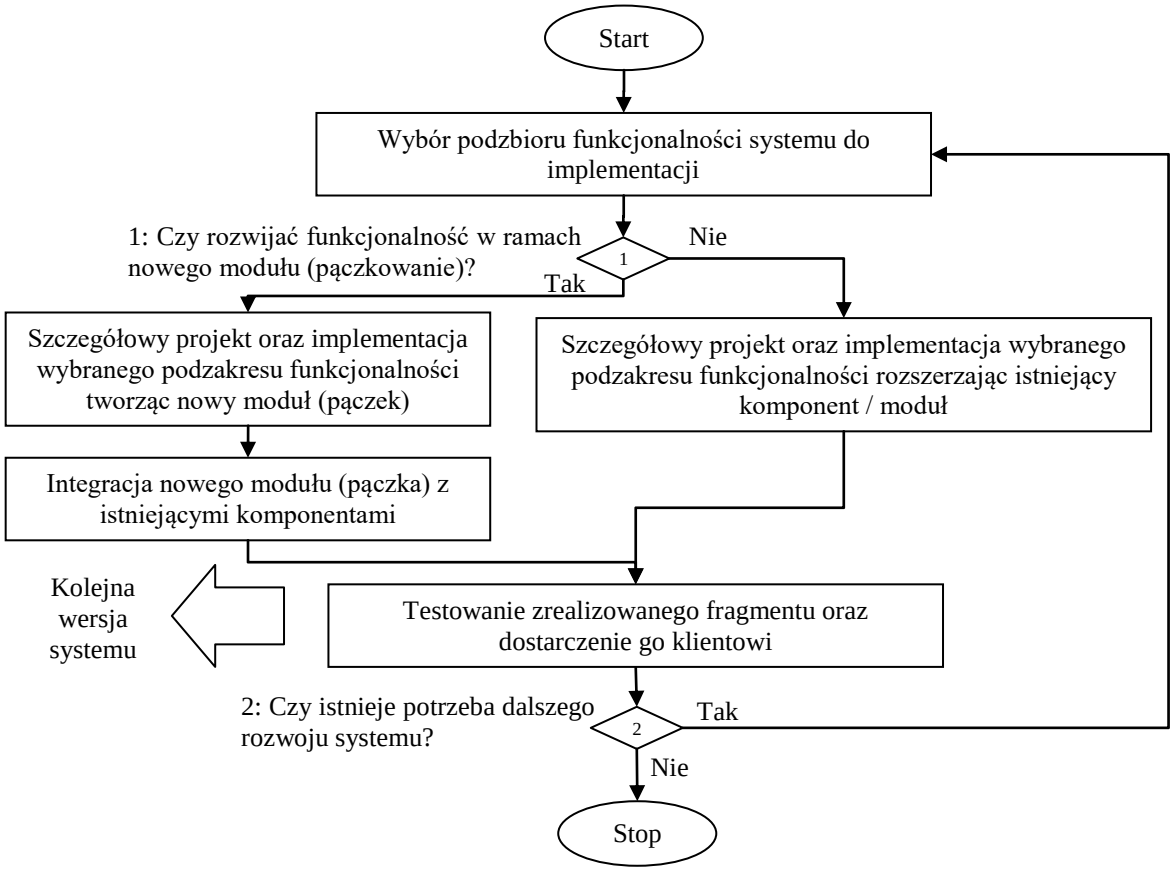
Komponenty i moduły mogą być ogólnodostępne, a prace nad ich rozwojem mogą być prowadzone przez społeczność open-source [2, 5]. Wybór idei wolnego oprogramowania wiąże się z dużym sukcesem tego typu rozwiązań. Produkty open-source stopniowo podbijają rynek na

całym świecie. Ich przewagą są niskie koszty użytkowania oraz możliwości rozwijania programów według ważnych potrzeb. Analitycy Gartnera szacują, że obecnie aplikacje na licencji open-source są wykorzystywane w 85% firm z całego świata. Niskie koszty korzystania z open-source zaczęły doceniać administracje rządowe w wielu krajach. W chwili obecnej z oprogramowania open-source korzysta Departament Obrony Stanów Zjednoczonych oraz administracja wielu stanów USA. Wprowadzenie wolnego oprogramowania planuje rząd brytyjski, który chce dzięki temu zaoszczędzić 600 mln funtów rocznie.

Te technologie są również zalecane przy wykorzystaniu metody rozwoju oprogramowania przez pączkowanie.

4. Procedura wytwarzania oprogramowania przez pączkowanie

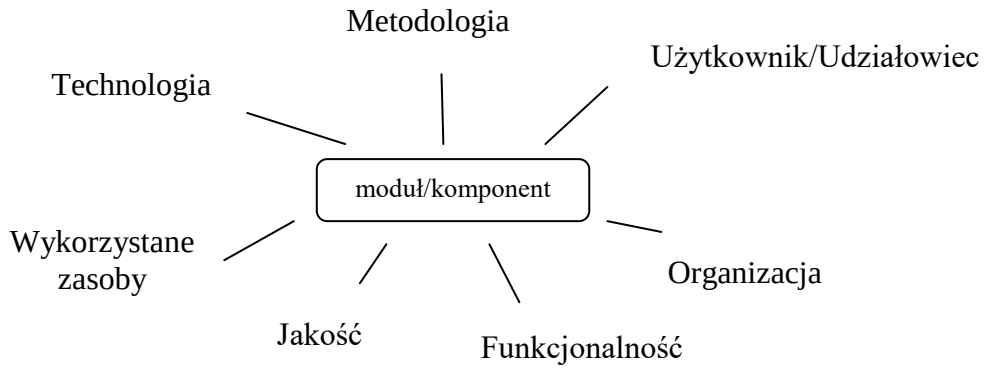
Rys. 2 przedstawia ogólny schemat procesu wytwarzania systemów komponentowo-modułowych przez pączkowanie. Szczegółowa procedura jest opisana na rys. 3.



Rys. 3. Schemat metody rozwoju systemów modułowych (usług) przez pączkowanie.

Tworząc zarówno komponent, jak i jego zalążki (pączki), wykorzystuje się na ogół metodologię open-source. Integracja zaś modułu (pączka) z komponentem powinna być realizowana w oparciu o framework EAI. Decyzja o pączkowaniu wynika z potrzeby zainicjowania kolejnego obszaru funkcjonalności, stycznego z aktualnie tworzonym obszarem odpowiadającym komponentowi. Z uwarunkowań organizacyjnych, aktualnych priorytetów informatyzowania poszczególnych obszarów oraz zmiany wymagań wynika natomiast decyzja o dalszym rozwoju systemu. Praktycznie przez cały okres eksploatacji systemu będzie ona twierdząca, ponieważ obszar ten ulega ciągłej ewaluacji. Każdy kolejny element wzbogaca repozytorium uniwersalnych komponentów (technologia SPL), które mogą być wykorzystane podczas

tworzenia nowych. Cała procedura wytwarzania oprogramowania przez pączkowanie realizuje się więc w przestrzeni wielowymiarowej zaprezentowanej na rys. 4.

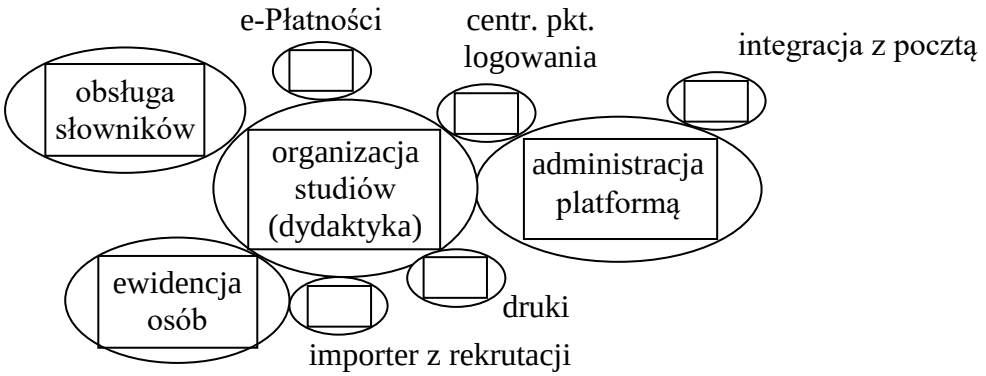


Rys. 4. Wielowymiarowa przestrzeń rozwoju systemów przez pączkowanie.

Z rys. 4 wynika, że rozwój modułów i komponentów nie ma tylko charakteru funkcjonalnego. Może wynikać również z potrzeby dostosowania się do zmian struktury jednostki lub organizacji, zmian uwarunkowań zewnętrznych, upodobań i wygody użytkowników lub zmian dostępnych zasobów. Jednocześnie możliwa jest łagodna, ewolucyjna zmiana technologii środowiska lub metodologii stosowanej w procesie wytwarzania usług, jak również prace związane z poprawą jakości tychże usług. Warto zauważyć, że poszczególne wymiary tej przestrzeni posiadają różne priorytety zmienne w czasie, co pączkowanie wspiera w sposób bardzo elastyczny.

5. Przykład rozwoju systemu przez pączkowanie

Zaproponowany model wytwarzania oprogramowania przez pączkowanie z wykorzystaniem Software Product Lines został opracowany na potrzebę budowy Internetowej Platformy Integracji Politechniki Gdańskiej (e-Politechniki). Najbardziej czasochłonne okazało się przygotowanie całego środowiska linii produkcyjnych. Jednak korzyści płynące z zastosowania obu połączonych metodologii przewyższyły już koszt czasu poświęconego na uruchomienie linii.



Rys. 5. Ilustracja systemu obsługi dziekanatu e-Dziekanat.

Zastosowane podejście pozwoliło na zrealizowanie pierwszego podsystemu e-Politechniki jakim jest system obsługi dziekanatu: e-Dziekanat. System ten wymagał opracowania 4

komponentów głównych (centralna ewidencja osób, słowniki centralne, organizacja studiów – dydaktyka, administracja platformą) oraz kilku modułów zależnych, takich jak m.in. system ewidencji płatności, system generowania druków, centralny punkt logowania oraz modułów – konektorów z istniejącymi na Politechnice Gdańskiej systemami (rekrutacja, poczta, ewidencja pracowników, system kwestury, itp.). Przygotowano również kilka modułów narzędziowych usprawniających pracę interfejsu użytkownika oraz wspomagających walidację poprawności semantycznej i składniowej identyfikatorów osobowych takich jak PESEL, NIP, numer dowodu osobistego, itp. Poglądową ilustrację rozwoju systemu przez pączkowanie przedstawiono na rys. 5. Ponadto zostały przygotowane narzędzia wspierające rozwój z wykorzystaniem metodologii SPL i EAI takich jak maven [12], hudson [10], trac [13]. Jako framework EAI został wykorzystany standard JEE [11].

Niezwykle wartościowa okazała się elastyczność oraz łatwość zmian wymagań i priorytetów, którymi charakteryzuje się wprowadzona metodologia, a jakim przyszło stawić czoła uruchamiając proces intensywnej informatyzacji uczelni. Zauważmy, że moduł e-Płatności może doprowadzić do rozwoju (lub zakupu) e-Finanse, zaś moduł centralnego punktu logowania może przerodzić się w e-Centrum Bezpieczeństwa. Dzięki temu podejściu zachowujemy własność interoperacyjności całego systemu. Co więcej wykorzystanie technologii SOA oraz rozwój usług alternatywnych czyni tę metodologię jeszcze bardziej przydatną.

6. Podsumowanie

W tabeli 1 przedstawiono poglądowe porównanie metodologii iteracyjno-przyrostowej z metodologią pączkowania.

Tab. 1. Porównanie dwóch metodologii wytwarzania interoperacyjnych komponentów oprogramowania.

kryterium	metoda iteracyjno-przyrostowa	metoda pączkowania	uwagi
liczba iteracji	mniejsza	większa	iteracje mniejsze, dostosowywane do aktualnych priorytetów i wymagań
czas realizacji	dłuższy	krótszy	więcej zespołów realizujących, wytwarzanie przyspiesza istniejące repozytorium komponentów
złożoność przedsięwzięcia	większa	mniejsza	zespoły mogą koncentrować się na dobrze określonych obszarach zastosowań i funkcjonalnościach
kompletność rozwiązania	całościowe komponenty, elementy jednej całości	stopniowe, dobrze planowane i zintegrowane z potrzebami	strategia informatyzacji przydatna w dłuższej perspektywie przy dynamicznie zmieniających się wymaganiach i priorytetach
wiarygodność	mniejsza	większa	lepiej zaplanowany rozwój, koncentracja na całości, ale dobrze zdekomponowanej i lepiej przetestowanej
jakość powstałego	mniejsza, trudna do zmierzenia	większa, mierzalna	zastosowanie metryk jakościowych dzięki wdrożeniu SPL

produktu			
kontekst przyrostu	przyrost funkcjonalności w ramach jednej z góry wyspecyfikowanej całości	przyrost funkcjonalności jednego mniejszego elementu lub/i utworzenie niezależnego pączka	rozbudowa funkcjonalności możliwa poprzez tworzenie scenariuszy użycia komponentów oraz pączków / modułów

Sukces wdrożenia na Politechnice Gdańskiej unikatowego systemu e-Dziekanat potwierdza, że opracowana metodologia wytwarzania oprogramowania przez pączkowanie jest niezwykle skuteczną metodą rozwoju interoperacyjnych środowisk rozproszonych. Znalezione analogie do ewolucji (metodologia pączkowania) oraz linii produkcyjnych z innych procesów inżynieryjnych (Software Product Lines) okazują się bardzo przydatne w dziedzinie inżynierii oprogramowania. Łącząc te dwa aspekty można zaobserwować wyraźny efekt synergii.

Zaprezentowana metodologia wymaga dalszych badań w celu zwiększenia dojrzałości procesu wytwarzania oraz rozwoju dodatkowych narzędzi wspomagających. Z uwagi na innowacyjne podejście oraz niezwykły potencjał. Metodologię tę można wykorzystać nie tylko podczas procesu informatyzacji uczelni, ale również w komercyjnych rozwiązaniach dedykowanych, np. przy rozwoju przestrzeni inteligentnych wspomaganych systemami wszechobecnymi czy kontekstowymi. W dalszych pracach istotne będzie określenie szczegółowych kryteriów rozwoju komponentu i modułu, a także definicji kontekstu funkcjonowania różnego typu systemów.

Bibliografia

[1] Clements P., Northrop L.: *Software Product Lines: Practices and Patterns.* MA: Addison-Wesley, 2002.

[2] Fogel K.: *Producing Open Source Software: How to Run a Successful Free Software Project*, O'Reilly, 2005.

[3] Gleghorn R.: *Enterprise Application Integration: a Managers Perspective*, IEE Computer Society, Nov/Dec. 2005, pp. 17-23.

[4] Krawczyk H.: *Zapewnienie interoperacyjności usług informatycznych podstawą rozwoju e-uczelni*, Zeszyty naukowe WETI PG, seria Technologie Informacyjne, Gdańsk 2009.

[5] Letellier F.: *Open Source Software: the Role of Nonprofits in Federating Business and Innovation Ecosystems*, AFME, 2008.

[6] McIlroy M.: *Software Engineering Concepts And Techniques, Mass produced software components*, NATO Conference on Software Engineering, 1969.

[7] Northrop L. M., Clements P. C.: *A Framework for Software Product Line Practice. Version 5.0.* Software Engineering Institute, 2007.

[8] Royce W.W.: *Managing the Development of Large Software Systems*, 1970.

[9] *System Lifecycle Framework*: *Selecting a development approach*, U.S. Department of Health & Human Services, 2008.

[10] https://hudson.dev.java.net/, 2010.

[11] http://java.sun.com/javaee/, 2010.

[12] http://maven.apache.org/, 2010.

[13] http://trac.edgewall.org/, 2010.

[14] http://www.agilemanifesto.org/, 2010.

[15] http://www.extremeprogramming.org/, 2010.

Pączkowanie – metoda rozwoju interoperacyjnych komponentów dla systemów rozproszonych

H. Krawczyk*, P. Lubomski**

* Politechnika Gdańska, Wydział Elektroniki Telekomunikacji i Informatyki, Katedra Architektury Systemów Komputerowych, e-mail: hkrawk@eti.pg.gda.pl.

** Politechnika Gdańska, Centrum Usług Informatycznych, e-mail: lubomski@pg.gda.pl.

Przedstawiono 2 współczesne metody wytwarzania oprogramowania: iteracyjno-przyrostową oraz techniki zwinne, ich zalety i wady w kontekście budowy interoperacyjnych platform i środowisk rozproszonych. Zaprezentowano metodę rozwoju oprogramowania przez pączkowanie, jej założenia, zalety i wady. Przedstawiono technologie, na bazie których działa metodologia wytwarzania oprogramowania przez pączkowanie: Software Product Line, Enterprise Application Integration oraz Open-Source. Zaprezentowano ich mocne strony oraz przydatność dla omawianej metodologii. Przedstawiono proceduralnie metodę pączkowania oraz omówiono umiejscowienie kroków w przestrzeni wielowymiarowej. Zaprezentowano przykład wykorzystania metodologii podczas tworzenia interoperacyjnej platformy usług Politechniki Gdańskiej. Podano parametry związane z takim rozwojem. Porównano metodologię iteracyjno-przyrostową z metodologią pączkowania. Omówiono rezultaty wdrożenia metodologii na Politechnice Gdańskiej.

Budding – the software development method of interoperable components for distributed systems

H. Krawczyk*, P. Lubomski**

* Gdansk University of Technology, Faculty of Electronics, Telecommunications and Informatics, e-mail: hkrawk@eti.pg.gda.pl.

** Gdansk University of Technology, Information Technology Centre, e-mail: lubomski@pg.gda.pl.

Two modern software development methods are discussed: the iterative-incremental methodology and agile techniques, their advantages and disadvantages in the context of building interoperable platforms and distributed environments. There is introduced the budding software development methodology. Its advantages and disadvantages are mentioned. There are presented some technologies on the basis of which the budding software development method works: Software Product Line, Enterprise Application Integration and Open-Source. Their strong points and usefulness are also mentioned. The procedure of the budding software development method is presented in a formal language. The steps are placed in the multidimensional space. There is described the use case of the budding software development method while building the interoperable service-oriented platform at Gdansk University of Technology. Some parameters connected with such a method are also discussed. The comparison of the iterative-incremental methodology and the budding software development methodology is presented. Finally, there are discussed the results of the introduction of the software development methodology at Gdansk University of Technology.