

# **Testowanie wbudowane. Europejski eksperyment walidacyjny.<sup>1</sup>**

Mariusz Momotko, Bartosz Nowicki

Rodan Systems S.A.

*{mariusz.momotko, bart}@rodan.pl*

## **Streszczenie**

Artykuł prezentuje podstawowe informacje o technologii testowania wbudowanego BIT opracowanej w ramach projektu Component+ (IST-1999-20162) oraz założenia eksperymentu, w którym Rodan Systems wraz z czterema innymi partnerami z krajów stowarzyszonych, ma ocenić skuteczność i przydatność tej technologii.

## **Wprowadzenie**

Rodan Systems S.A. wraz z czterema innymi partnerami z krajów stowarzyszonych rozszerza toczący się już projekt badawczo-rozwojowy ([www.component-plus.org](http://www.component-plus.org)) w ramach 5. Programu Ramowego, Przyjazne Społeczeństwo Informacyjne. Projekt uzyskano w ramach akcji kluczowej VIII.1.6 Enabling RTD Cooperation with Newly Associated States. Koordynatorem projektu jest IVF Industrial Research and Development Corporation (Szwecja), a pozostałymi partnerami rozszerzenia są Uniwersytet w Tallinie (Estonia), 4D Soft (Węgry), University of Rousse oraz ISoft (oba z Bułgarii). Projekt zaczął się 1 lipca 2002 roku i będzie trwał 9 miesięcy. Zadaniem rozszerzenia jest zbadanie i ocena skuteczności technologii opracowanej w oryginalnym projekcie.

---

<sup>1</sup> Praca zrealizowana ze środków projektu Komisji Europejskiej, projekt IST-1999-20162

# 1. Technologia BIT

Technologia komponentów programowych (CBSE - *component based software engineering*) może znacząco podnieść produktywność i jakość tworzonego oprogramowania. Zamiast rozwijać oprogramowanie całkowicie od początku, CBSE pozwala na złożenie (większości) aplikacji z prefabrykowanych komponentów. Podstawową zaletą komponentów jest zdolność do ukrywania wewnętrznych rozwiązań (enkapsulacja) oraz możliwość wielokrotnego wykorzystania specjalizowanych komponentów, oferujących zestaw usług dotyczący określonego zagadnienia. Przykładem może być komponent do zarządzania użytkownikami, zadaniami do wykonania, czy obsługujący pocztę. Enkapsulacja, dostępność tylko i wyłącznie zewnętrznie obserwowanego zachowania, powoduje jednak następujące problemy:

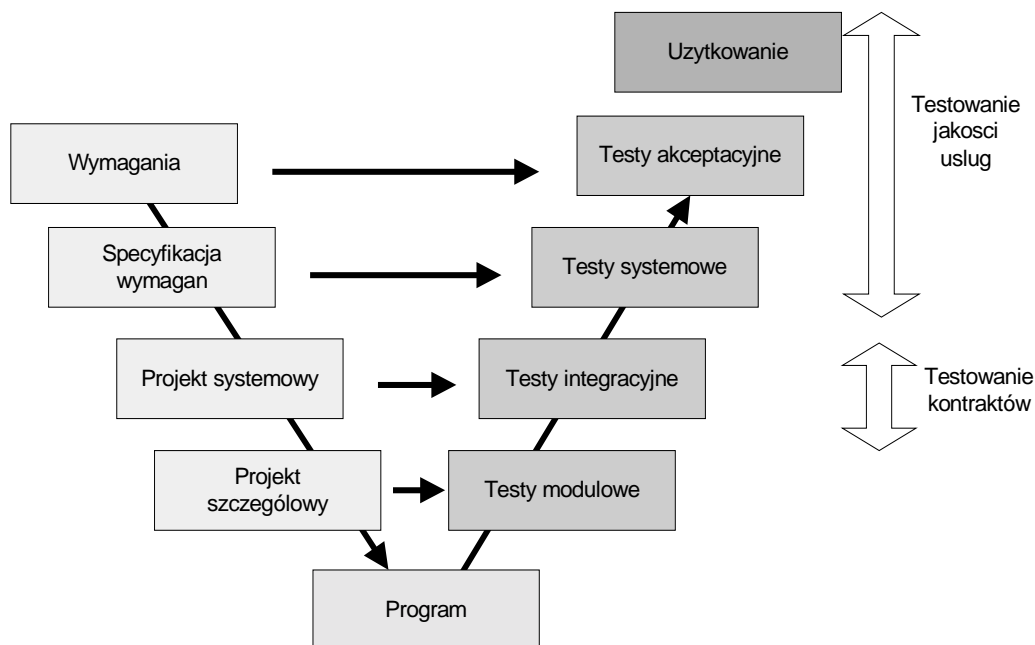
- problemy ze skutecznością testów,
- znaczący koszt konserwacji (lokalizacja błędów, testowanie regresyjne),
- brak wsparcia dla testów wykonywanych podczas produkcyjnego wykorzystywania oprogramowania.

Component+ stara się rozwiązać powyższe problemy dostarczając technologię tworzenia testów wbudowanych (*Built-in testing – BIT*), które pozwalają dostać się do wnętrza komponentu w ściśle określony sposób [5]. Technologia ta zakłada, że testy wbudowane są wykonywane podczas integracji systemu oraz w czasie produkcyjnego działania systemu.

W BIT zaproponowano dwie techniki testowania komponentów: **testowanie kontraktu** (zestawu usług udostępnianych przez komponent, *contract testing*) oraz **testowanie jakości usług** (*Quality of Service testing – QoS*) oferowanych przez komponenty łączące tworzący dany system. Pierwsza z technik skupia się na znalezieniu błędów na poziomie współpracy pomiędzy dwoma komponentami. W takim przypadku jeden z komponentów świadczy usługi (serwer) na rzecz drugiego (klient). W danym teście klient zgodnie z określonym scenariuszem wysyła szereg żądań do serwera otrzymując odpowiednie odpowiedzi. Przykładowo program wysyła żądanie odczytu listy zadań danego użytkownika do komponentu zarządzania procesami pracy (ZPP). Zgodnie z wymaganiami (kontraktem), komponent ZPP wymaga, aby klient najpierw podał kryteria odczytu listy zadań, następnie odczytał listę zadań (być może wielokrotnie) i ostatecznie zasygnalizował zakończenie odczytywania listy. Testowanie kontraktu jest ukierunkowane na wyszukiwanie błędów semantycznych we współpracy pomiędzy poszczególnymi komponentami. Technika ta jest wykorzystywana głównie przy integracji oraz zmianie konfiguracji oprogramowania (np. wersji danego komponentu).

Druga technika, testowanie jakości usług, pozwala identyfikować bardziej subtelne błędy wynikające z uruchomienia komponentu w specyficznym środowisku, np. związane ze sposobem dostępu (np. zakleszczenia), zasobami (pamięć, procesor, dysk, drukarki, itp.) lub współdziałaniem konkretnych komponentów. Testowanie jakości usług jest wykonywane ciągle w trakcie

normalnego użytkownika systemu. Ten rodzaj testowania umożliwia wykrywanie, raportowanie i obsługę błędów komponentów dotyczących integralności kodu i danych, monitorowania zasobów oraz uwarunkowań czasowych, błędów rezydualnych. Rysunek 1 wskazuje na etapy tradycyjnego cyklu rozwojowego wspierane przez BIT. Na uwagę zasługuje możliwość testowania systemów podczas użytkowania (dla agentów WWW i usług sieciowych także testów kontraktowych).

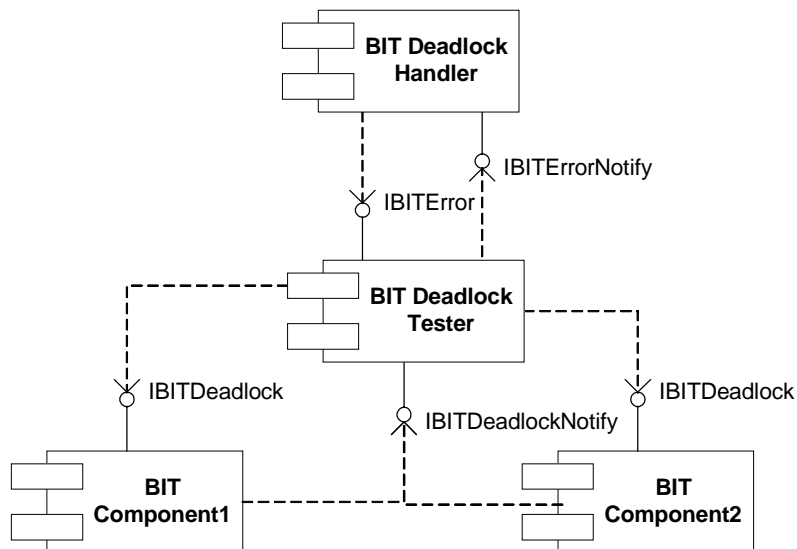


Rysunek 1. Etapy tradycyjnego cyklu rozwojowego wspierane przez BIT

BIT kładzie silny nacisk na badanie stanu komponentu. Stan może być wyrażony jawnie (w klasach odpowiadających bytom o wyraźnym cyklu życia np. czujniki w systemach wbudowanych) lub w oparciu o wartości atrybutów (w klasach odpowiedzialnych za przechowywanie danych np. stos). Aby umożliwić dostęp do stanu, BIT proponuje utworzenie, obok standardowych interfejsów funkcjonalnych komponentu, dodatkowego specjalizowanego interfejsu testującego. Za pomocą tego interfejsu możliwe jest ustawianie i odczytywanie stanu komponentu.

Każdy komponent podlegający testowaniu jest nazywany komponentem BIT. Komponent BIT musi co najmniej wspierać interfejs testujący IBITQuery. Interfejs ten udostępnia pojedynczą funkcję QueryInterface, która dostarcza informacji, czy wyspecyfikowany interfejs testujący i związane z nim testy są wspierane przez komponent. Aby zapewnić elastyczność testowania, BIT stosuje zewnętrzne komponenty testujące (*testers*). Dzięki takiemu podejściu przypadki testowe danego komponentu są ogólnie dostępne (są przecież poza testowanym komponentem), mogą być łatwo modyfikowane, uzupełniane i wymieniane.

Dodatkowo, w technologii BIT są wykorzystywane komponenty obsługi sytuacji wyjątkowych (*handlers*) wykrytych przez komponenty testujące.



Rysunek 2. Testowanie zakleszczeń na zasobach poprzez technikę QoS

Przykładowo, w systemie wykorzystana została technika testowania jakości usług do sprawdzania zakleszczeń na zasobach dla dwóch komponentów BIT (Rysunek 2). Zgodnie z architekturą BIT, w systemie istnieje także komponent testujący oraz komponent obsługi sytuacji wyjątkowych. Komponenty BIT sygnalizują gotowość wspierania testów zakleszczeń poprzez interfejs IBITDeadlock. Przydział i zwalnianie zasobów poprzez poszczególne komponenty jest sygnalizowany komponentowi testującemu poprzez interfejs IBITDeadlockNotify. Na podstawie zgłoszeń o przydziale/zwalnieniu zasobów, komponent testujący wykrywa sytuację zakleszczeń na zasobach i sygnalizuje ją do komponentu obsługi zakleszczeń poprzez interfejs IBITErrorNotify. Komponent ten wykonuje działania niwelujące wystąpienie zakleszczenia. Dodatkowe informacje o zakleszczeniu komponent obsługi sytuacji wyjątkowych uzyskuje od komponentu testującego poprzez interfejs IBITError.

BIT powoduje wczesne podjęcie zagadnień testowania w cyklu życia (co jest ogólnie zalecaną praktyką). Pozwala to na:

- uzyskanie krytycznego i uzupełniającego, w stosunku do normalnej analizy wymagań, widoku na system,
- projektowanie systemu pod kątem testowania,
- projektowanie testów na odpowiednim poziomie abstrakcji.

Posiadanie przez komponent wbudowanej kontroli własnego zachowania oraz kontroli zachowania zbioru komponentów w środowisku wykonawczym pozwala na realizację podejścia CBSE w wydaniu „plug, test and play” [6].

## 2. Cel i zadania projektu

Podstawowym celem projektu jest zastosowanie technologii BIT wypracowanej w oryginalnym projekcie Component+ (IST-1999-20162) oraz jej sprawdzenie i weryfikacja. W projekcie rozszerzenia partnerzy mają zweryfikować nowo opracowaną technologię w pilotowych, rzeczywistych projektach. Jednocześnie, zgodnie ze zdefiniowanymi zaleceniami, będą zbierane pomiary dotyczące procesu wytwarzania. Uzyskane rezultaty (np. gęstość błędów) zostaną porównane z odpowiadającymi im danymi pochodzącymi z tradycyjnego procesu wytwórczego. Component+ oferuje predefiniowany zestaw miar służących ocenie produktywności i jakości, który zostanie dopasowany do specyfiki konkretnego projektu pilotowego.

Rozszerzenie projektu Component+ ma następujące cele:

- walidacja miar i procedur oceny (niezbędne w obiektywnej ocenie),
- weryfikacja użyteczności technologii BIT w różnych dziedzinach,
- uzyskanie dodatkowych doświadczeń w rzeczywistych aplikacjach,
- poszukiwanie nowych dziedzin zastosowania,
- rozpowszechnienie wiedzy wśród partnerów z krajów stowarzyszonych.

## 3. Przedmiot eksperymentu

Dotychczas technologię BIT zastosowano w następujących „case study”: system zdalnego dostępu do informacji przez obywateli w samorządach lokalnych, przyrost systemu bankowego ukierunkowany na polepszenie wygody obsługi kont oraz konwersji walut, system „inteligentnego domu” (sterowanie temperaturą i wentylacją), system personalizacji telewizji (preferencje widza są przechowywane na inteligentnej karcie). W omawianym rozszerzeniu projektu Component+ planuje się wykonanie „case study” w następujących dziedzinach: agent automatycznego wyszukiwania informacji w Internecie (Estonia), analizator kodu programów napisanych w języku Java (Węgry), komponenty obsługi struktury organizacyjnej (Bułgarzy), oraz system zarządzania wiedzą i procesami pracy (Rodan). W ramach projektu Rodan będzie rozwijał dwa komponenty opisane poniżej.

**OfficeObjects<sup>®</sup>Workflow Manger (OOWM)** jest komponentem realizującym zarządzanie procesami pracy [2]. Komponent ten jest zgodny ze standardami koalicji WorkFlow Management Coalition (WfMC). Rozwinięty przez Rodan model procesu pracy jest rozszerzeniem meta-modelu koalicji WfMC o możliwość elastycznego przypisywania uczestników procesu do wykonywanych czynności zgodnie z zaproponowanym językiem WPAL [3]. W celu wyznaczania uczestników procesu wykorzystywane są usługi świadczone przez OORM (zob. poniżej). OOWM może być wbudowywany w praktycznie dowolny system

zarządzania informacją oparty o koncepcję obiektów informacyjnych (grupy danych stanowiących logiczną całość). Skuteczność OOWM potwierdziło udane wdrożenia w znaczących instytucjach.

**OfficeObject<sup>®</sup> Role Manager (OORM)** jest komponentem realizującym zarządzanie użytkownikami i ich uprawnieniami. OORM umożliwia także definiowanie grup użytkowników oraz modelowanie struktur organizacyjnych zgodnie ze standardem X.500. Egzekucja uprawnień jest transparentna dla kodu aplikacyjnego. OORM wspiera także zapis informacji o podstawowych działaniach poszczególnych użytkowników (dane audytu).

## 4. Sposób realizacji eksperymentu

Celem projektu jest eksperymentalna ocena technologii BIT polegająca na porównaniu jakości i kosztów wytworzenia i konserwacji systemów bez (system odniesienia) oraz z użyciem technologii BIT (system nowy). W przebiegu eksperymentu niezwykle ważną będzie baza pomiarowa, która musi dostarczyć pomiarów w zakresie rozmiaru systemów, błędów systemów (ich klasa, metoda / etap identyfikacji) oraz pracochłonności (np. dodatkowa na BIT, identyfikacji i usunięcia, testowania regresyjnego). Na obecnym poziomie rozpoznania BIT wydaje się, że wcześniejszy większy wysiłek stosowania BIT zaowocuje mniejszą pracochłonnością testowania, mniejszą liczbą błędów testowania systemowego i akceptacyjnego (wyższa jakość), oraz mniejszą pracochłonnością lokalizacji błędów oraz testowania regresyjnego (konserwacja). Łączna pracochłonność będzie więc niższa. Rodan zamierza wykorzystać standardowo stosowane mechanizmy rejestrów problemów oraz pomiaru pracochłonności [4].

Aby otrzymane rezultaty były wiarygodne, należy określić a następnie przestrzegać warunki przeprowadzenia tego eksperymentu. Warunki te zostaną dokładnie wyspecyfikowane łącznie z wymaganą bazą pomiarową (będą one uwzględniały doświadczenia zastosowania BIT w oryginalnym projekcie). Zasadniczo warunki te dotyczą czterech obszarów: ogólny **proces** i **techniki** wytwarzania oraz **zespół** wytwarzający i **specyfika** tworzonego systemu. W idealnym przypadku w trakcie przeprowadzania eksperymentu wszystkie obszary powinny być stabilne (z wyjątkiem samego BIT). Zmiany w zakresie procesu i technik mogą potencjalnie mieć ogromny wpływ na system docelowy. Rozważmy np. zmiany w zakresie analizy kodu. Jeżeli zamiast recenzji (*peer review*) zastosujemy bardziej formalną inspekcję i dodatkowo przeznaczymy na nią względnie więcej zasobów, to testowany system będzie miał mniej błędów, testy BIT wykryją mniej błędów niż średnio rygorystyczne testy systemowe i mniej błędów przedostanie się do końcowych użytkowników. Taka mieszanka przeciwstawnych informacji będzie naprawdę trudna do zinterpretowania w odniesieniu do skuteczności BIT. Rodan od lat tworzy i pielęgnuje procesy

wytwarzania oprogramowania [4], dlatego obszary procesu i technik są stabilne i nie spowodują zakłóceń w eksperymencie.

Koordynatorzy projektu wstępnie zakładali **podwójne wykonanie tego samego systemu**. Podejście to całkowicie rozwiązuje problem specyfiki systemu (jest to w końcu ten sam system), jednak wprowadza poważne ryzyko rzetelności eksperymentu w zakresie obszaru zespołu. Jeżeli ten sam zespół będzie wykonywał projekt dwukrotnie, to oczywiście druga edycja będzie lepsza nie ze względu na BIT, lecz ze względu na wzrost doświadczenia zespołu. Jeżeli pokusimy się o dwa zespoły realizujące system równolegle, to problem stanowi jakość zespołów oraz ich izolacja. Najważniejszy jednak wydaje się czynnik motywacyjny – firma wielkości Rodanu musi pracować (i pracuje) efektywnie i wykonywanie podwójnej pracy byłoby dla pracowników niezwykle trudne do zaakceptowania.

Rodan będzie wykonywał eksperymenty z **BIT na dwóch różnych komponentach**. To podejście wymaga precyzyjnej organizacji eksperymentu w zakresie zespołu oraz specyfika tworzonego systemu. Systemem odniesienia jest istniejąca i wykorzystywana wersja OOWM, a standardowo stosowany proces jej tworzenia i konserwacji zawiera informacje ilościowe na poziomie, który pozwoli na względną ocenę jakości oraz kosztów przy tworzeniu nowych wersji. Argumentem za stabilnością zespołu jest fakt, że nowe wersje będą realizowane przez zespół, który wykonał system odniesienia. Już podczas tworzenia systemu odniesienia zespół prezentował niezwykle wysoki poziom kompetencji (zarówno proces jaki i umiejętności programistyczne) i nie zakłada się, by od czasu realizacji systemu odniesienia zespół w znaczący sposób ulepszył swój warsztat (co mogłoby zakłócić eksperyment).

Problem specyfiki systemu wymaga również dokładnej analizy. Zarówno system odniesienia jak i system nowy należą do klasy oprogramowania narzędziowego, na bazie którego buduje się docelowe systemy aplikacyjne. Nacisk w obu systemach jest na działanie typu „motor” (engine) – pierwszy jest odpowiedzialny za obsługę procesów pracy a drugi za autoryzację praw dostępu. Udział interfejsu użytkownika jest podobny i niezbyt duży. Dodatkowo użytkownikiem obu systemów jest administrator, a więc te systemy muszą wykazywać ten sam poziom przyjazności. Zakres rozwijanej funkcjonalności komponentów dobrano w taki sposób, by szacowany rozmiar, w liniach kodu (w obu przypadkach jest to JAVA), oraz złożoność (niestety oceniana subiektywnie, a nie za pomocą np. punktów funkcyjnych) były podobne do systemu odniesienia. W przypadku znaczącej rozbieżności rozmiarów i / lub złożoności systemów należałoby uwzględnić wpływ rozmiaru na gęstość błędów (pokazane w zależności od punktów funkcyjnych w [1]).

## Podsumowanie

W artykule zaprezentowano technologię BIT, informacje o projekcie europejskim COMPONENT+ oraz plany działań Rodanu w tym projekcie. Rodan zamierza wykorzystać COMPONENT+ (to 3 badawczo-rozwojowy projekt europejski) do wzmocnienia ogólnofirmowej metodyki wytwarzania oprogramowania [4] w zakresie technologii komponentowych, a w szczególności testowania komponentów.

W chwili obecnej trudno oceniać skuteczność technologii BIT - jej ocena jest przecież przedmiotem prezentowanego projektu. Nasze zainteresowanie budzi formalne wprowadzenie aspektu testowania wcześniej w cyklu rozwoju oprogramowania. Pewien niedosyt pozostawia niewykorzystanie możliwości powtórnego wykorzystania testów wbudowanych na drodze dziedziczenia. Pytanie otwarte stanowi również przydatność technologii BIT w rozwoju oprogramowania Rodanu – narzędziowych systemów zarządzania wiedzą.

## Bibliografia

- [1]. Jones C., Software Quality. Analysis and Guidelines for Success, International Thomson Computer Press, 1997
- [2]. Momotko M., OfficeObjects Workflow - komponent wspierający zarządzanie procesami pracy, III Krajowa Konferencja Inżynierii Oprogramowania KKIO'01, Otwock, 17-20 października 2001
- [3]. Momotko, M., Subieta, K., Dynamic change of Workflow Participant Assignment, Sixth East-European Conference on Advances in Databases and Information Systems ADBIS, September 8-11, 2002, Bratislava, Slovakia
- [4]. Nowicki B., Metodyka Rodan Nowe Technologie, III Krajowa Konferencja Inżynierii Oprogramowania KKIO'01, Otwock, 17-20 października 2001
- [5]. Wang Y., King G., Wickburg H., A Method for Built-in Tests in Component-based Software Maintenance, IEEE International Conference on Software Maintenance and Reengineering (CSMR'99), March 1999, pp.186-189
- [6]. Hörnstein, J., Edler, H., "Test Reuse in CBSE Using Built-in Tests", Workshop on Component-based Software Engineering, Composing systems from components, Lund, 8-11 April 2002.