Rozdział

Rezultaty zastosowania technologii C+ BIT do testowania komponentów programowych[1]

Mariusz Momotko, Lilianna Zalewska
Rodan Systems S.A.
{Mariusz.Momotko, Lilianna Zalewska}@rodan.pl

Streszczenie

Jak testować interakcję pomiędzy komponentami? Czy istnieje możliwość testowania w standardowy sposób wybranych cech komponentu w trakcie rzeczywistego działania systemu? Próbą systematycznej odpowiedzi na te pytania jest technologia testowania wbudowanego C+ BIT opracowana w ramach europejskiego projektu badawczo-rozwojowego Component+. Niniejszy artykuł prezentuje podstawowe informacje o tej technologii skupiając się na jakościowych i ilościowych rezultatach eksperymentu, w którym Rodan Systems wraz z czterema innymi parterami z krajów stowarzyszonych oceniał jej skuteczność i przydatność.

1. Wprowadzenie

Głównym celem projektu Component+ (IST-1999-20162, www.component-plus.org) było wypracowanie technologii testowania komponentów programowych. Zgodnie z przyjętymi założeniami technologia ta ma pomóc w testowaniu zarówno interakcji pomiędzy komponentami jak i jakości usług świadczonych przez komponenty w trakcie ich rzeczywistej pracy. W pierwszej fazie projektu pięciu partnerów z krajów Unii Europejskiej pod przewodnictwem IVF Industrial Research and Development Corporation (Szwecja) zdefiniowało podstawy technologii oraz wypracowało metodykę przeprowadzania testów. W drugiej fazie projektu (NAS extension) pięciu partnerów z

[1] Praca sponsorowana przez Komisję Europejską w ramach projektu Component+, project no. IST-1999-20162

krajów stowarzyszonych, w tym także Rodan Systems, oceniało jej skuteczność i możliwość praktycznego zastosowania.

2. Technologia C+ 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 błędu, 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*), pozwalających dostać się do wnętrza komponentu w ściśle określony sposób [Edler2002]. Technologia ta zakłada, że testy wbudowane są wykonywane podczas integracji systemu oraz w czasie produkcyjnego działania systemu.

Technologia C+ BIT proponuje dwie techniki testowania komponentów: **testowanie kontraktów** (interakcji pomiędzy komponentami zintegrowanymi w danym systemie, *contract testing*) oraz **testowanie jakości usług** (*Quality of Service testing – QoS*) świadczonych przez komponenty w trakcie rzeczywistego działania systemu. Obie techniki zakładają automatyczne wykonanie testów (Testery).

Testowanie kontraktów skupia się na znalezieniu błędów na poziomie współpracy pomiędzy dwoma komponentami [Gross2003]. 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 oczekuje, aby klient najpierw podał kryteria odczytu listy zadań, następnie odczytał wyszukane zadania (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 komponentów oraz zmianie konfiguracji oprogramowania (np. instalacja nowej wersji danego komponentu).

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 [Vincent2003b]. Testowanie jakości usług jest wykonywane ciągle w trakcie normalnego użytkowania systemu. Ten rodzaj testowania umożliwia wykrywanie, raportowanie i obsługę błędów komponentów dotyczących integralności kodu i danych, monitorowania zasobów oraz uwarunkowań czasowych, a także błędów rezydualnych.

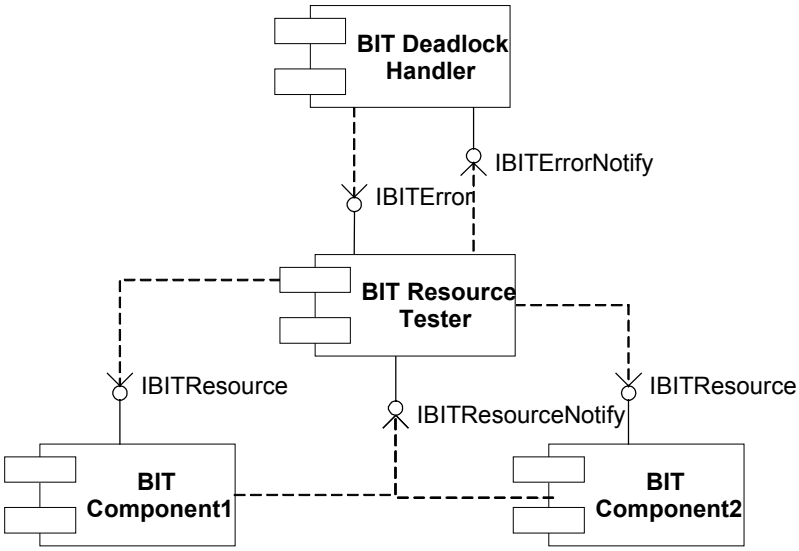
BIT kładzie silny nacisk na badanie stanu komponentu. Stan może być wyrażony jawnie (w klasach odpowiadających bytom o wyraźnym cyklu życia np. czujniki w systemach wbudowanych) lub w oparciu o wartości atrybutów (w klasach odpowiedzialnych za przechowywanie danych np. stos). Aby umożliwić dostęp do stanu, BIT proponuje utworzenie, obok standardowych interfejsów funkcjonalnych komponentu, dodatkowe specjalizowane interfejsy testujące. Za pomocą takich interfejsów możliwe jest ustawianie i odczytywanie stanu komponentu, a także uzyskiwanie szczegółowej informacji o wystąpieniu określonych zdarzeń.

Zgodnie z architekturą BIT, w każdym systemie istnieją trzy podstawowe rodzaje komponentów: komponenty BIT, Testery (*testers*) i komponenty obsługi sytuacji wyjątkowych (*handlers*) [Vincent2003a]. 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 i scenariusze testowe dotyczące wspieranych kontraktów są ogólnie dostępne (są przecież poza testowanym komponentem), mogą być modyfikowane, uzupełniane i wymieniane (rozszerzenie kontraktów). Dodatkowo, w technologii BIT są wykorzystywane komponenty obsługi sytuacji wyjątkowych (*handlers*) wykrytych przez komponenty testujące.

Architektura BIT definiuje także zestaw interfejsów programowych, które muszą lub mogą być implementowane przez poszczególne komponenty. Oprócz standardowych interfejsów definiujących sposób odczytywania konfiguracji komponentów oraz zgłaszania błędów, zdefiniowane zostały też interfejsy specyficzne dla wybranych testerów jakości usług. W aktualnej wersji dokumentu opisującego architekturę BIT są to interfejsy testowania zakleszczeń na użytkowaniu zasobów (*resource interfaces*), monitorowania czasu wykonywania usług wspieranych przez komponenty (*timing interfaces*) oraz śledzenia wykorzystania pamięci operacyjnej (*memory interfaces*).

Sposób wykorzystania tych interfejsów został omówiony na przykładzie testowania zakleszczeń na zasobach dla dwóch komponentów BIT (zob. Rysunek 1). Komponenty BIT sygnalizują gotowość wspierania testów zakleszczeń poprzez interfejs IBITResource. Przydział i zwalnianie zasobów poprzez poszczególne komponenty jest sygnalizowany komponentowi testującemu poprzez interfejs IBITResourceNotify. Na podstawie zgłoszeń o

przydziale/zwalnianiu 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.



Rys. 1. Testowanie zakleszczeń na zasobach poprzez technikę QoS

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" [Jones1997].

3. Założenia i przebieg eksperymentu

Celem eksperymentu przeprowadzonego przez Rodan Systems była ocena technologii Component+ BIT polegająca na porównaniu jakości oraz kosztów wytworzenia i konserwacji systemów bez (system odniesienia) oraz z użyciem technologii BIT (system nowy) [Momotko2002].

Aby otrzymane rezultaty były wiarygodne, na początku projektu ściśle określono warunki przeprowadzenia tego eksperymentu. Zasadniczo warunki te dotyczyły czterech obszarów: ogólny proces, technika wytwarzania, zespół wytwarzający i specyfika tworzonych komponentów. W projekcie założono, że wszystkie obszary w trakcie przeprowadzania projektu powinny być stabilne (z wyjątkiem samego BIT). Trzech partnerów projektu wykonywało dwie wersje tych samych komponentów (bez i z użyciem technologii C+ BIT). Podejście takie rozwiązało problem specyfiki komponentów (są to w końcu te same komponenty), jednak wprowadziło duże ryzyko rzetelności eksperymentu w zakresie obszaru zespół (komunikacja pomiędzy dwoma zespołami, poziom kompetencji). Dwóch pozostałych partnerów, w tym także Rodan,

zdecydowało się na inną taktykę i przyjęło jako punkt odniesienia poprzednie wersje komponentów, wytworzone jeszcze przed rozpoczęciem projektu (posiadające odpowiednią bazę pomiarową), a w ramach projektu Component+ rozwijać ich kolejne wersje wyposażone w technologie C+ BIT. Podejście takie zakładało, że złożoność funkcjonalna poprzednich i nowych wersji jest podobna. Przy takim założeniu porównanie wspomnianych wersji można było wykonać poprzez porównanie rozmiaru komponentów wyrażonych w liniach kodu źródłowego (znormalizowana wartość)

Rodan wykonał eksperyment na dwóch różnych komponentach platformy OfficeObjects® Portal: OfficeObjects®WorkFlow Manager i OfficeObjects®Role Manager. Pierwszy komponent wspiera zarządzanie procesami pracy, podczas gdy drugi zarządza użytkownikami, strukturą organizacyjną, uprawnieniami i rolami.

Eksperyment przebiegał następująco. W ramach projektu architektury zdefiniowano interfejsy funkcjonalne i testowe poszczególnych komponentów, określono zakres kontraktów wspieranych przez komponenty i ustalono, jakie ich cechy będą monitorowane w trakcie rzeczywistej pracy. Interfejsy testowe były zgodne ze standardowymi interfejsami testowymi zdefiniowanymi w architekturze BIT. W ramach zastosowania technologii BIT Rodan zaimplementował dwa testery kontraktów i jeden tester jakości usług. Testery kontraktów symulowały hipotetyczne, najpełniejsze wykorzystanie komponentów OOWM i OORM przez ich klientów. Tester jakości usług dotyczył działania komponentu OOWM i sprawdzania, czy istnieją zadania nie przypisane lub błędnie przypisane do wykonawców. Ponadto, aby zmniejszyć nakład pracochłonności Rodan wykorzystał tester monitorowania czasu wykonywania usług oraz komponent logowania sytuacji wyjątkowych zaimplementowane przez Phillips i IVF. Podczas projektowania szczegółowego podano wewnętrzna architekturę komponentów (klasy i interfejsy programowe, baza danych), opracowano wymagane algorytmy i doprecyzowano inne zagadnienia bezpośrednio związane z wytwarzana funkcjonalnością. Ponadto, zdefiniowano listę przypadków i scenariuszy testowych wspieranych przez właściwe dane testowe i oczekiwane rezultaty. Dla OOWM zdefiniowano 18 scenariuszy testowych i 70 przypadków testowych. Dla OORM zdefiniowano 7 scenariuszy testowych i 30 przypadków testowych. W następnym kroku określono wewnętrzna architekturę testerów kontraktów i testerów jakości usług. Implementacja obejmowała wytworzenie zarówno samych komponentów, jak i testerów oraz komponentów obsługujących sytuacje wyjątkowe.

4. Rezultaty eksperymentu

Na zakończenie pierwszej fazy projektu podsumowano rezultaty jakościowe wynikające ze wstępnego zastosowania technologii C+ BIT przez oryginalnych partnerów. Rezultaty te zostały uzupełnione i uaktualnione po zakończeniu drugiej fazy projektu. Ponadto, po zakończeniu trwania eksperymentu, każdy z uczestników drugiej fazy (w tym także Rodan) dostarczył do koordynatora projektu podstawowe miary ilościowe dotyczące procesu wytwórczego (z i bez zastosowania technologii C+ BIT). Dane te zostały wykorzystane do opracowania wstępnej odpowiedzi na dwa podstawowe pytania: „Czy

technologia C+ BIT może być praktycznie zastosowana?" i „Czy opłaca się stosować tą technologię?". Najważniejsze jakościowe i ilościowe rezultaty eksperymentu uzyskane przez Rodan zostały przedstawione poniżej.

Technologia C+ BIT pozwala firmom trzecim na przygotowywanie własnych testów kontraktowych

Producenci oprogramowania wykorzystujący komponenty innych firm mogą sami przygotowywać scenariusze testowe i implementować je przy pomocy własnych Testerów. Dodatkowo, o ile twórcy komponentów dostarczają standardowe Testery (pokrycie możliwie wielu hipotetycznych kontraktów) w formie kodu źródłowego, istnieje możliwość dopisywania do nich własnych scenariuszy testowych (bez tworzenia nowego Testera).

Skrócenie przypadków testowych poprzez zastosowanie operacji odczytu stanów

W trakcie przygotowywania testów kontraktowych zauważyliśmy, że odczytywanie uzyskanych danych testowych przy pomocy interfejsów funkcjonalnych może być żmudne i wprowadzać dodatkowe błędy do Testerów. Przykładowo, weryfikacja stanu instancji procesu (trzech jego atrybutów), wymagała odczytu pełnej informacji o procesie zapisanej w formacie XML. Operacja ta była wykonywana prawie w każdym scenariuszu testowym (rozmiar scenariuszy). Utworzenie specjalizowanej funkcji do mapowania atrybutów na stan procesu pozwoliło znacząco zmniejszyć rozmiar scenariuszy testowych i pośrednio zmniejszyć szansę wprowadzenia do nich błędów (których wykrycie było pracochłonne).

Oddzielenie interfejsów testowych od funkcjonalnych

Jedna z dobrych reguł tworzenia komponentu mówi, że nie powinien on udostępniać zbyt wielu usług. Zbyt wiele oferowanych funkcji (interfejsy funkcjonalne) oznacza problemy z prawidłowym zastosowaniem komponentu (złożona semantyka). Dodatkowo, umieszczenie w tych interfejsach funkcji odczytujących i zapisujących wewnętrzne dane komponentu (funkcje testujące) może spowodować obniżenie poziomu bezpieczeństwa. Dzięki technologii C+ BIT mogliśmy wydzielić interfejsy testowe od funkcjonalnych zwiększając ich czytelność i użyteczność.

Możliwość testowania ograniczeń czasowych przy pomocy operacji ustawiania stanu

Testy komponentu OOWM, dotyczyły także sprawdzania opóźnienia zadań (czynności wykonywanych w ramach procesów). Aby zasymulować opóźnienie zadania potrzebna była albo zmiana czasu zegara systemowego, albo ustawienie sztucznej wartości czasu utworzenia/zakończenia zadania. W przypadku Rodanu pierwsze rozwiązanie nie było możliwe do zastosowania; serwer testowy był wykorzystywany przez większą liczbę projektów. Drugie rozwiązanie mogło zostać zastosowane po dodaniu funkcji do sztucznej zmiany wspomnianych atrybutów zadania. Funkcja ta, ponieważ ingerowała w wewnętrzny stan komponentu (jednego z procesów przez niego obsługiwanych), została dodana do interfejsu testującego a nie funkcjonalnego.

Automatyczne wykonywanie testów kontraktowych może znacząco zmniejszyć czas wykonywania testów

W przypadku Rodanu zastosowanie automatycznych testerów znacząco zredukowało czas wykonywania testów (z ok.16rbg w wersji non-BIT, w której testy były wykonywane ręcznie, do ok. 0,5rbg w wersji BIT). Zysk jest tym większy, im więcej razy komponent podlegał testom (nowe wersje, poprawki, itp.). Należy jednak nadmienić, że powyższa cecha jest tylko pośrednią korzyścią wynikająca z tej technologii.

Wczesne podjęcie zagadnień testowania – mniejszy koszt wykrywania i naprawy błędów

Dzięki projektowaniu scenariuszy i przypadków testowych wspieranemu przez technologie C+ BIT, możliwe było wczesne wykrywanie poważnych błędów komponentów. W przypadku OOWM, opracowywanie scenariuszy testowych pozwoliło na wykrycie dwóch poważnych błędów, istniejących także w poprzedniej wersji oprogramowania. Spojrzenie na projektowany system pod kątem testów umożliwiło głęboką analizę jego funkcjonalności. W opinii Rodanu cecha ta jest jedną z największych korzyści tej technologii, aczkolwiek jest ona tylko pośrednio związana z technologią C+ BIT.

Skuteczniejsze wykrywanie błędów związanych z niewłaściwą konfiguracją systemu

Przy użyciu technologii C+ BIT instalacja komponentów była znacząco krótsza. Dzięki Testerom kontraktów możliwa była szybka weryfikacja wymagań komponentu na konfigurację środowiska (komponenty, wymagane interfejsy, zmienne środowiskowe, itp.) oraz wymagania innych komponentów wobec nowo instalowanego komponentu. Przykładowo, instalacja i uruchomienie pierwszej wersji komponentu OOWM (non-BIT) u jednego z naszych klientów wyniosło 40rbg. Instalacja drugiej wersji OOWM w nowym środowisku (Tamino Server) była znacząco krótsza i wyniosła ok. 18rbg. Wydaje się, że zastosowanie technologii C+ BIT może zredukować ten czas o ok. 50%.

Wielokrotne wykorzystanie Testerów jakości usług

Zgodnie z technologią C+ BIT testery jakości usług mogą być wykorzystane wielokrotnie, obsługiwać różne komponenty i być uruchamiane w różnych systemach. W przypadku Rodanu, Tester czasu wykonania usług został zastosowany w dwóch komponentach: OOWM i OORM oraz w trzech instalacjach systemu OO Portal/ OO WorkFlow.

Niezależny tester integralności danych

Technika testowania jakości usług daje możliwość utworzenia testera integralności danych niezależnego od systemów zarządzania bazą danych jak i łatwego do utrzymania i modyfikacji. Przed zastosowaniem tej techniki, Rodan tworzył testery integralności danych wykorzystujące głównie funkcjonalności dostępne w relacyjnych bazach danych takich jak ograniczenia, funkcje wyzwalające i procedury składowane. Ponieważ wspomniane mechanizmy są specyficzne dla poszczególnych rodzajów baz danych, a nasze systemy pracują na co najmniej trzech rodzajach baz, konieczne było tworzenie i utrzymywanie kilku takich samych testerów dla poszczególnych rodzajów baz. Koszty takich testerów były znaczące. Po zastosowaniu technologii C+ BIT liczba utrzymywanych testerów zmniejszyła się ponad trzykrotnie.

Tester czasu wykonywania usług – sposób na sprawdzanie, że coś jest nie tak z naszym systemem

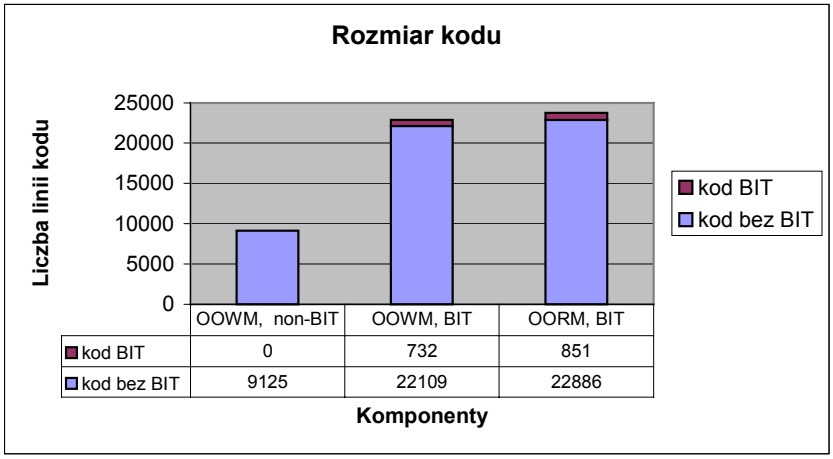
Rodan wykorzystał Tester czasu wykonywania usług, nie do pomiaru wspomnianego czasu, a raczej do sprawdzenia czy dana usługa wysłała odpowiedź. Nastawiając odpowiednio długi maksymalny czas wykonywania usług, można było sprawdzić, czy przypadkiem któraś z usług systemu 'zawiesza się' i nie daje żadnej odpowiedzi. Dla Rodanu zastosowanie takiego testera było także sposobem na wczesne ostrzeganie o możliwych problemach działającego systemu. Było to szczególnie ważne w przypadku rygorystycznych czasów reakcji na wystąpienie błędu po stronie klienta (umowy gwarancyjne).

Testowanie jakości usług - sposób weryfikacji złożonych algorytmów

Testery jakości usług mogą być także wykorzystywane do sprawdzania złożonych algorytmów, dla których istnieje prosty mechanizm weryfikacji. Przykładowo, rezultat algorytmu 'Wieże Hanoi' o złożoności 2^n może zostać sprawdzony przez prosty algorytm o złożoności liniowej.

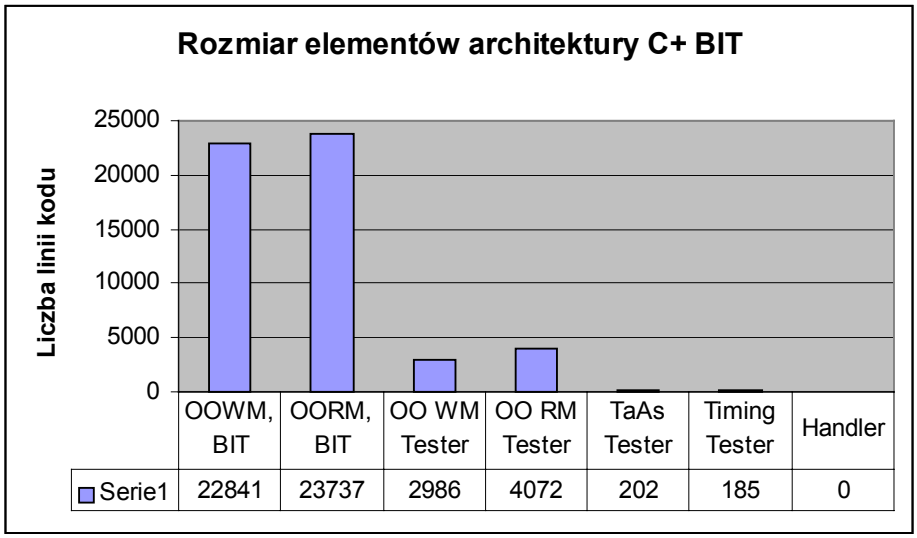
Technologia C+ BIT nie zwiększa rozmiaru kodu komponentów w znaczący sposób

Użycie technologii C+ BIT tylko nieznacznie zwiększyło rozmiar komponentów. Dla obu komponentów rozmiar zwiększył się o ok. 700 linii, co stanowi ok. 6-7% ich rozmiaru.



Rys. 2. Porównanie rozmiaru komponentów (non-BIT, BIT)

Z drugiej strony rozmiar Testerów kontraktów był znaczący i wynosił co najmniej 10-15% rozmiaru komponentu. Spowodowane było to głownie ilością implementowanych scenariuszy testowych i przypadków testowych. W przypadku testerów jakości usług, ich rozmiar był niewielki. W przypadku testera integralności danych (TaAs Tester) spowodowane to było prostym algorytmem spójności przypisanych zadań w ramach procesów pracy. W przypadku testera czasu wykonania usług (*Timing Tester*) oraz komponentu obsługi sytuacji wyjątkowych (*Handler*) ich niewielki rozmiar był efektem użycia istniejących już komponentów (zaimplementowanych przez Phillips i IVF i nieznacznie zmodyfikowanych dla potrzeb Rodanu)



Rys. 3. Porównanie rozmiaru elementów architektury C+ BIT

Technologia BIT - pracochłonność procesu tworzenia oprogramowania

Jak pokazał przypadek Rodanu, technologia C+ BIT nie powinna znacząco zwiększyć całkowitej pracochłonności procesu tworzenia oprogramowania dla dużych komponentów (15-20KLOC). Różnica ta wynosiła ok. 2-3% jakkolwiek naszym zdaniem procent ten mieści się w granicy błędu eksperymentu (np. większa liczba linii komentarza, spokojniejsze warunki pracy, itp.). Z drugiej strony, jak wykazały rezultaty firmy iSoft (Bułgaria), dla małych komponentów (1-2KLOC) pracochłonność po zastosowaniu BIT'a może zwiększyć się znacząco. ISoft raportował ok. 60% wzrost pracochłonności [Hörnstein2003]. Także rezultaty wskazujące na czas poświęcony na uczenie się technologii różniły się. Większość partnerów wskazywała na niski koszt uczenia się technologii. Przykładowo dla Rodanu wyniósł on 94rbg, podczas gdy dla iSoft 288rbg.

Jakość oprogramowania

Założenie, że BIT podnosi jakość oprogramowania znalazło już częściowo potwierdzenie w uzyskanych rezultatach. Częściowo, ponieważ cały czas trwają wdrożenia opracowanych komponentów i wyniki dotyczące znalezionych błędów nie są ostateczne. Jakkolwiek, w przypadku Rodanu zauważyliśmy, że jak dotychczas podczas użytkowania BIT-owych wersji komponentów nie wystąpiły błędy krytyczne (dla porównania, w pierwszej wersji po stronie klienta znaleziono 3 takie błędy – koszt ich naprawy był znaczący). Ponadto, inni partnerzy, tworzący równolegle obie wersje komponentów raportowali większą liczbę znalezionych błędów podczas testowania wersji BIT w stosunku do non-BIT.

Aby w pełni skorzystać z zalet technologii C+BIT należy też dbać o rozbudowę bazy testów kontraktowych. Praktycznie każdy zgłoszony błąd, niezależnie od sposobu wykrycia, powinien stać się podstawą do budowy testu, wykorzystanego przy produkcji/odbiorze kolejnej wersji komponentu.

Co do zauważonych problemów z technologią C+ BIT trzeba podkreślić, że jak dotychczas **brak jest mechanizmów, które umożliwiłyby testowanie technologii BIT**. Jest to szczególnie ważne w przypadku dużej liczby przypadków testowych i długich scenariuszy testowych.

Wydajność systemu może się obniżyć przy nierozważnym zastosowaniu testerów jakości usług

Szczególnie w przypadku Testera czasu wykonania usług, monitorowanie zbyt wielu funkcji lub włączenie aktywnego monitorowania (tzn. Tester sam sprawdza, czy uruchomione funkcje się zakończyły) z bardzo małą rozdzielczością (np. 1ms) może spowodować widoczne pogorszenie wydajności systemu.

4. Podsumowanie

W artykule zaprezentowano podstawy technologii C+ BIT skupiając się na rezultatach eksperymentu walidacyjnego uzyskanego przez partnerów projektu Component+ w szczególności przez firmę Rodan. Na podstawie uzyskanych rezultatów uważamy, że technologia BIT ma praktyczne zastosowanie i, przy rozważnym jej wykorzystaniu (np. liczba monitorowanych funkcji przez tester czasu wykonania usług), może się opłacać.

W najbliższej przyszłości Rodan zamierza wykorzystać testy kontraktowe do testowania kolejnych komponentów/modułów platformy OfficeObjects® ICONS.

Bibliografia

[Jones1997] Jones C., *Software Quality. Analysis and Guidelines for Success*, International Thomson Computer Press, 1997.

[Edler2002] Edler, H., Hörnstein, J., *Test Reuse in CBSE Using Built-in Tests*, Workshop on Component-based Software Engineering, Composing systems from components, Lund, 8-11 April 2002.

[Gross2003] Gross, H., G., Built-In-Test Vade Mecum – Part II, Built-In Contract Testing: Method and Process., Fraunhofer Institute Experimental Software, Engineering Germany, Component+.

[Hörnstein2003] Hörnstein, J.,Assesment and evaluation addendum, D6.2Ad report, Component+.

[Momotko2002] Momotko, M., Nowicki, B., *Testowanie wbudowane. Europejski eksperyment walidacyjny.*, IV. Krajowa Konferencja Inżynierii Oprogramowania, Poznań, 2002.

[[Vincent2003a] Vincent, J., Built-In-Test Vade Mecum – Part I, A Common BIT Architecture, Bournemouth University, UK, Component+.

[Vincent2003b] Vincent, J., Built-In-Test Vade Mecum – Part II, Interface Specification:Types, Syntax and Semantics, Bournemouth University, UK, Component+.