Akademia Górniczo-Hutnicza

im. Stanisława Staszica w Krakowie

WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI i ELEKTRONIKI

AUTOREFERAT ROZPRAWY DOKTORSKIEJ

mgr inż. Rafał Brzuchacz

ZASTOSOWANIE JĘZYKA LOTOS W PRZYROSTOWYM PROCESIE TWORZENIA OPROGRAMOWANIA SYSTEMÓW WBUDOWANYCH

Promotor: prof. dr hab. inż. Tomasz Szmuc

Kraków 2010

W rozprawie zaprezentowano metodę budowy oprogramowania systemów wbudowanych charakteryzującą się systematyczną analizą poprawności z zastosowaniem metody formalnej. Rosnąca złożoność oprogramowania tego typu, oraz występowanie współbieżności, skłaniają do zastosowania zaawansowanych metod konstrukcji już we wczesnych etapach cyklu życia aplikacji. Przedstawione w pracy podejście opiera się na wykorzystaniu istniejącej metody formalnej, języka LOTOS, w przyrostowym i zorientowanym obiektowo procesie tworzenia systemów wbudowanych. Zastosowanie proponowanej metody specyfikowania zachowania na poziomie analizy wymagań i projektu architektury pozwala zapewnić pożądany stopień spójności opisu systemu. Spójność opisu tworzonego we wczesnych fazach cyklu życia jest warunkiem koniecznym zgodności oprogramowania ze specyfikacją, co decyduje o wysokiej jakości docelowej aplikacji.

1 Cele i teza rozprawy

System wbudowane (*embedded systems*) stanowią klasę systemów komputerowych, które stają się integralną częścią obsługiwanych urządzeń [Sifa09]. Klasa ta obejmuje układy specjalnego przeznaczenia takie, jak mikrokontrolery sterujące liniami produkcyjnymi, komputery pokładowe w samolotach, sprzęt medyczny, sprzęt pomiarowy, etc. Systemy te realizowane są zazwyczaj na dedykowanych układach mikroprocesorowych, często z odrębnym systemem operacyjnym.

Jedną z cech charakterystycznych współczesnych systemów wbudowanych jest struktura umożliwiająca efektywne wykonywanie aplikacji współbieżnej. Jest to możliwe dzięki coraz szybszym zegarom taktującym prace procesora, który jest sercem każdego systemu klasy *embedded*.

Współbieżność [Ben09], realizowana przez osadzane w tych układach systemy operacyjne, pozwala na efektywną obsługę pewnego zbioru równoległych procesów otoczenia.

Występowanie współbieżności powoduje jednak znaczną złożoność, a przez to problemy z analizą zachowania tworzonych aplikacji systemów wbudowanych. Dokładne testowanie, jak również symulacje utworzonego programu, nie dają wymaganego stopnia pewności co do poprawności jego działania, a w przypadku systemów współbieżnych występuje dodatkowo eksplozja kombinatoryczna wariantów testów praktycznie uniemożliwiająca wystarczające pokrycie możliwych zachowań aplikacji. W wielu dziedzinach zastosowań niedopuszczalnym jest jednak tolerowanie niewłaściwego działania systemu. Niewłaściwa sekwencja akcji systemu może być katastrofalna w skutkach. Przykładem mogą być zastosowania w dziedzinach takich, jak lotnictwo czy medycyna. Niepożądane sytuacje muszą być wykryte odpowiednio wcześniej, na etapach analizy i projektu oprogramowania.

Warunkiem koniecznym wysokiej wiarygodności systemu jest spójność jego opisu we wczesnych etapach jego życia. Cecha ta uzyskiwana jest w pracy poprzez systematyczne tworzenie modeli aplikacji i ich weryfikację na etapie analizy wymagań i projektu architektury. Postępowanie takie umożliwia wczesne wykrycie niespójności opisu, jak też zbadanie interesujących własności systemu. Zweryfikowany model projektowy może zostać dalej transformowany przez dedykowane narzędzia do postaci kodu wykonywalnego. Ten końcowy etap nie jest jednak w pracy rozpatrywany.

Zasadniczym celem stawianym przez autora było opracowanie metody zapewniającej spójność tworzonych artefaktów zachowania tworzonego oprogramowania systemu wbudowanego. Przeprowadzone badania polegały na opracowaniu właściwych dla tego celu zestawu czynności i struktur specyfikacji w LOTOS [BB87][Huz07][Brz06] na poziomie analizy i projektu systemu. Spójność została zdefiniowana w terminach systemu formalnego tego języka co umożliwiło jej algorytmiczną weryfikację. Utworzony proces został oparty na modelu iteracyjnym metody ROPES [Dou04] – dedykowanej dla systemów wbudowanych i bazującej na języku UML. Modele formalne w LOTOS traktowane są jako specyfikacje zachowania struktur wyrażonych w UML i poddawane są weryfikacji na poszczególnych etapach procesu. Przeprowadzone badania pokazują prawdziwość następującej tezy głównej pracy:

„Wykorzystanie języka LOTOS w przyrostowym procesie ROPES umożliwia wytwarzanie wysokiej jakości oprogramowania systemów wbudowanych.”

Podana teza została wykazana przez realizację poniższych badań cząstkowych:

1) Utworzenie ogólnej struktury specyfikacji systemu na poziomie analizy, która uwzględnia przyszłą dekompozycję funkcjonalności i realizację przyrostów wymagań.

2) Utworzenie ogólnej struktury specyfikacji systemu na poziomie architektury. Formalizacja wybranych wzorców projektowych w celu wykorzystania gotowych szablonów na tym etapie budowy.

3) Realizacja przyrostów funkcjonalności w istniejących modelach.

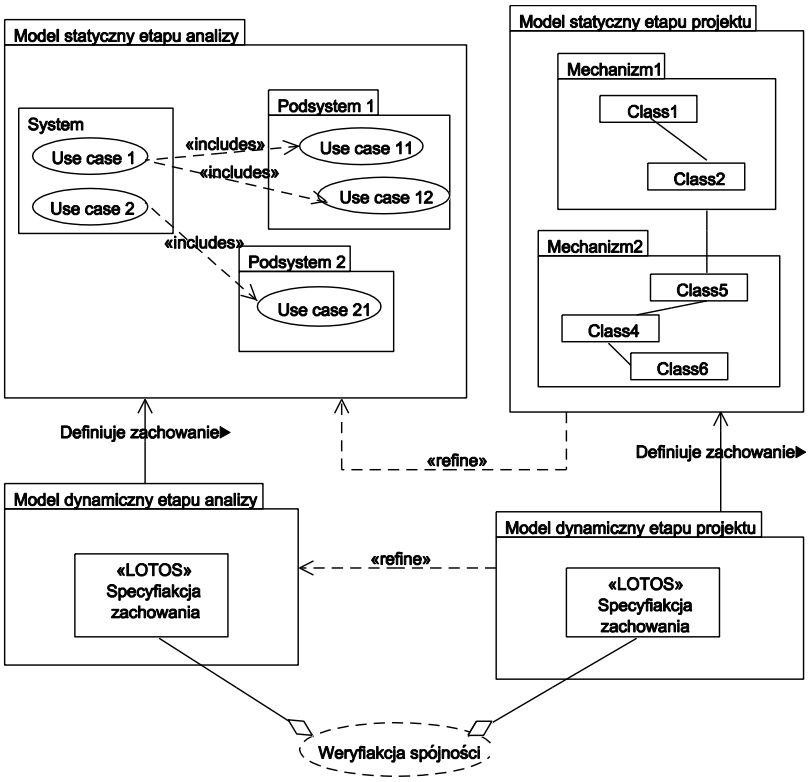
4) Określenie kryteriów spójności modeli zachowania dla iteracji wykonywanych przy niestabilnych i stabilnych wymaganiach.

2 Czynności i artefakty proponowanego procesu

Budowa modeli w proponowanym procesie odbywa się w kliku etapach, w których wyróżniono powtarzalne sekwencje czynności zwane iteracjami lub mikrocyklami. W danym etapie rozwoju iteracja może być wykonywana jeden lub więcej razy. Celem każdej iteracji jest dostarczenie wykonywalnego prototypu realizującego funkcjonalność o coraz mniejszym znaczeniu dla całości systemu. Wykonywalność prototypu gwarantowana jest przez narzędzie interpretujące opis w języku LOTOS. Istotna jest tutaj weryfikacja systemu. W proponowanym procesie jest ona możliwa dzięki wykorzystaniu technik właściwych dla języka LOTOS i ogólniej algebr procesów. Ciężar prac realizowanych w procesie przesuwa się od analizy wymagań w kierunku projektu wraz ze stabilizowaniem się zbioru wymagań, które system musi implementować. Iteracje zostają zgrupowane wg kryterium stabilności (niezmienności) realizowanych wymagań. Dla wymagań niestabilnych wykonywane są iteracje, których celem jest po pierwsze - utworzenie początkowych specyfikacji zachowania na poziomie analizy i projektu, po drugie - utrzymanie wysokiego stopnia spójności tych specyfikacji dla pojawiających się przyrostów funkcjonalności. Stopień spójności jest definiowany w oparciu o siłę relacji, do której należy para utworzonych specyfikacji LOTOS. Przyrost rozumiany jest jako zmiana definicji zachowania i realizowany jest jako rozszerzenie lub modyfikacja istniejących specyfikacji. Stabilizacja wymagań pozwala na wykonywanie drugiej grupy iteracji realizującej zmiany związane z polepszeniem rozwiązania projektowego. Alternatywne rozwiązania projektowe są tutaj porównywane, według kryterium stopnia spójności, z modelem analizy. Konkretne rozwiązanie powstaje przez zastosowanie instancji szablonu LOTOS, który odwzorowuje wybrany wzorzec projektowy lub mechanizm [Dou99]. Formalizacja behawioralnych wzorców projektowych jest kluczowa dla etapu projektu i zostaje wykonana jednokrotnie poza procesem. Formalne modele wzorców mogą tutaj zostać zweryfikowane pod kątem zachodzenia określonych własności, co ułatwia badanie spójności całego modelu zachowania. Poniżej zamieszczono zwięzłe opisy poszczególnych grup iteracji (mikrocykli), które składają się na całość proponowanego procesu.

I Konstrukcja zrębu architektury

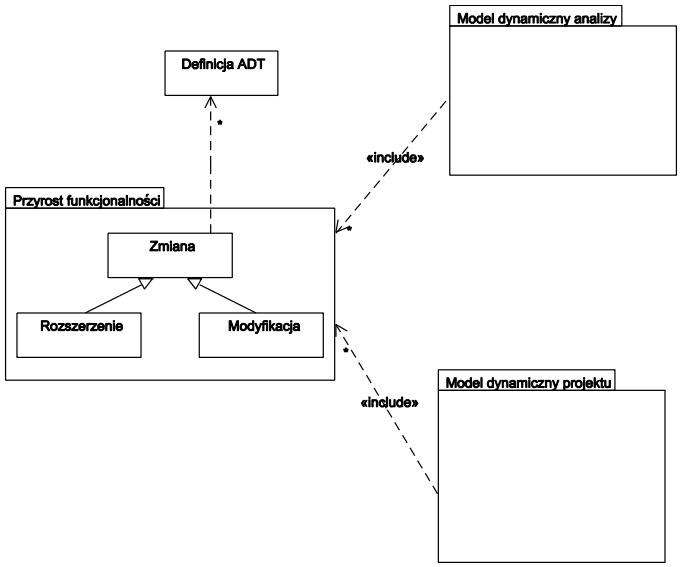
Iteracja jest wykonywana jeden raz i jej celem jest wykonanie początkowego prototypu o wysokim stopniu spójności opisu zachowania. Opis statyczny systemu wykonywany jest zgodnie z metodą ROPES. Proponowany proces definiuje tutaj odwzorowanie podstawowej funkcjonalności w modelu statycznym etapu projektu (Rys. 2.1). Komplementarny opis dynamiki tworzony jest w oparciu o szablon struktury specyfikacji LOTOS. Dla etapu projektu, szablon ten jest inny niż dla analizy. Przejście między modelami LOTOS odbywa się na zasadzie wyodrębnienia niezależnych funkcjonalności i alokowaniu ich w osobnych procesach realizujących role kolaboracji wykryte w architekturze systemu. Postępowanie takie gwarantuje zachowanie funkcjonalności systemu. Synchronizacja wykrytych w projekcie wątków jest realizowana poprzez zastosowanie formalnych modeli wzorców projektowych wprowadzających do opisu dodatkowe akcje. Modele formalne zachowania badane są pod kątem zachodzenia odpowiednich relacji w zbiorze procesów LOTOS.



Rys. 2.1. Artefakty pierwszej grupy iteracji

II Realizacja zmian

Wraz z postępem prac nad systemem, mogą pojawić się nowe wymagania, a istniejące ulec modyfikacji. Celem drugiej grupy iteracji jest zapewnienie spójności opisu dla niestabilnego zbioru wymagań. Przyrost funkcjonalności, składający się z pewnego zbioru nowych wymagań, jest włączany do opisu zachowania (Rys. 2.2). Czynności metody przewidują tutaj dodanie nowych bloków funkcjonalności w strukturze specyfikacji analizy wymagań oraz odpowiednie modyfikacje wzorców i ról kolaboracji w modelu etapu projektu. Koniecznym może być także wykorzystanie nowego typu ADT. Sytuacja taka pokazana została w przykładach prezentowanych w pracy, i dotyczyła realizacji kolejkowania na etapie analizy wymagań. Następnie, tak jak uprzednio, modele dynamiczne poddawane są weryfikacji spójności. Przedstawiony proces gwarantuje zachodzenie co najmniej równoważności bezpieczeństwa utworzonych opisów LOTOS.



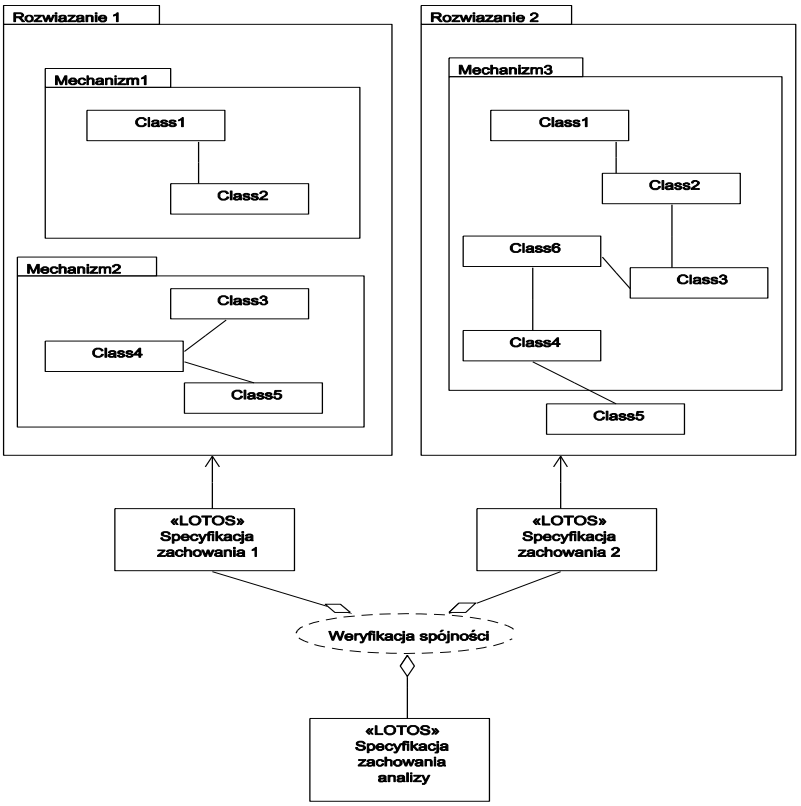
Rys. 2.2. Artefakty drugiej grupy iteracji

III Zmiany w projekcie

Trzecia grupa iteracji realizowana jest dla stabilnego zbioru wymagań. Następuje tutaj powtórzenie czynności wprowadzania zmian do modelu etapu projektu. Ich celem jest optymalizacja stosowanych rozwiązań w zakresie stosownych wzorców i ról kolaboracji. Wyznaczenie stopnia spójności względem opisu etapu analizy odbywa się według innego kryterium niż dla poprzednich grup iteracji. Rozważane są tym razem relacje typu preorder. Podejście takie wynika z następującego faktu. System na etapie analizy wymagań jest opisywany w LOTOS jako jeden proces składających się z wyróżnionych bloków funkcjonalności. Realizacja kolejkowania jest tu mocno ograniczona i sprowadza się do pojedynczego bufora w postaci zmiennej procesowej. W modelu projektu natomiast, występują równoległe kolejki procesów odwzorujących role kolaboracji, synchronizowane na wprowadzonych w tym celu dodatkowych bramach. Graf LTS, reprezentujący możliwe sekwencje akcji w systemie na etapie projektu, będzie więc zawierał tranzycje, które nie są możliwe w grafie z analizy. Sytuacja ta przekreśla możliwość zachodzenia równoważności odpowiednich opisów LOTOS, i skłania do zastosowania słabszego kryterium spójności.

IV Rozwiązania projektowe

Kolejna grupa iteracji dotyczy także optymalizacji modelu projektowego, jednak ma charakter bardziej globalny w stosunku do poprzedniej. Modyfikacji ulega tutaj architektura systemu poprzez wybór innego wzorca projektowego (Rys. 2.3). O wyborze konkretnego rozwiązania decyduje stopień spójności względem modelu analizy. Innym kryterium może być także przejrzystość i naturalność rozwiązania.



Rys. 2.3. Artefakty czwartej grupy iteracji

2.1 Analiza stosowalności metody

Ograniczenia stosowalności metody wynikają w pierwszej kolejności z zastosowania formalizmu języka LOTOS. Złożoność czasowa i pamięciowa istniejących algorytmów może ograniczać rozmiar tworzonej specyfikacji. Język LOTOS nie uwzględnia też wymagań niefunkcjonalnych związanych z czasem. Kolejne ograniczenie może wynikać ze sposobu definiowania wymagań i przyjętej struktury specyfikacji analizy.

2.1.1 **Rozmiar modeli LOTOS**

Sprawdzenie, czy dwa modele są równoważne modulo pewna relacja równoważności, rodzi pytanie o efektywność stosowanego algorytmu i maksymalny rozmiar porównywanych modeli. Zastosowany w pracy pakiet CADP stosuje algorytm autorstwa R.Paige'a i R.Tarjan'a [PT87], o złożoności czasowej O(m log n) i pamięciowej O(m), gdzie: $m = |T_1| + |T_2|$, $n=|Q_1|+|Q_2|$ dla LTS_i , i=1,2. Algorytm ten znajduje zastosowanie (z pewnymi modyfikacjami niezmieniającymi złożoności obliczeniowej) dla wszystkich typów relacji użytych w niniejszej pracy. Dla przedstawionych w [Fer90] wyników, można oszacować czasy wykonania dla największego modelu utworzonego w przykładzie w rozdz. 12 niniejszej pracy. Otrzymany czas sprawdzenia bisymulacji obserwacyjnej wynosi 5.038 min, dla długości słowa m=226761+293, n=55340+131 (sumy ilości tranzycji i stanów modeli z projektu i analizy).

Czas weryfikacji nie stanowi więc tutaj dużego ograniczenia. Dla $m = 40*10^6$ tranzycji i n rzędu 10^6 stanów, weryfikacja trwa, wg danych z [Fer90], około 24 godziny. Przy obecnej szybkości procesorów, czas ten jest rzędu kilku godzin. Z uwagi na fakt, że czynność weryfikacji jest wykonywana co 4-6 tygodni (po każdym mikrocyklu ROPES), czas ten jest w pełni akceptowalny.

Większym problemem jest złożoność pamięciowa stosowanego algorytmu. Dla modelu z $m = 40*10^6$, wynosi 2GB, przy koszcie reprezentacji tranzycji LTS wynoszącym 20 bajtów [Fer90]. Liniowa ze względu na ilość tranzycji złożoność algorytmu w CADP, pozwala w prosty sposób oszacować maksymalny rozmiar porównywanych modeli tak, aby obliczenie było wykonywalne na danej maszynie. Przykładowo, dla modelu z $m=10^6$ tranzycji, ich reprezentacja zajmie 50MB pamięci. Nie bez znaczenia jest także następujący fakt. Algorytm Paige&Tarjan'a stosuje podejście globalne, tzn. buduje pełną reprezentacje grafów LTS w pamięci, i na nich wykonuje obliczenie. Przy niezachodzeniu badanej relacji między grafami, bardziej efektywne czasowo i pamięciowo jest wykonanie sprawdzania w locie. Ten typ weryfikacji jest zastosowany w najnowszej wersji pakietu CADP (narzędzie BISIMULATOR 2.0), i polega na badaniu odpowiednich LTS podczas ich stopniowej budowy w pamięci. Obliczenie jest zatrzymywane przy pierwszym napotkaniu „niezgodności" w tworzonych modelach. Technika ta, jest więc średnio bardziej efektywna dla modeli w początkowych etapach budowy, gdzie z pewnym prawdopodobieństwem wiadomo, że pożądana relacja nie zachodzi.

2.1.2 **Wymagania niefunkcjonalne**

Jednym z założeń dotyczącym zakresu stosowalności metody jest uwzględnienie tylko wymagań funkcjonalnych. Wynika to z samej natury stosowanego formalizmu. Język LOTOS pozwala wyrazić tylko wymagania funkcjonalne w postaci sekwencji akcji systemu. Uwzględnienie czasu reakcji systemu nie jest tutaj możliwe. Proponowana metoda nie nadaje się więc do specyfikowania wymagań niefunkcjonalnych związanych z czasem. Znajduje natomiast zastosowanie dla systemów, w których aspekt czasowy może zostać oddzielony od projektu logicznego tak, jak ma to miejsce w metodyce HRT-HOOD. Proces projektowy dotyczy tutaj w pierwszej kolejności wymagań funkcjonalnych - następnie uwzględniane są inne wymagania. Prezentowany w pracy proces nie realizuje jednak tego drugiego etapu. Rozwiązaniem może być tutaj budowa specyfikacji zachowania w języku RT-LOTOS, który jest rozszerzeniem czasowym LOTOS.

2.1.3 **Proces definiowania wymagań**

Brak struktury w docelowej specyfikacji zachowania w analizie wymagań implikuje małą przejrzystość tworzonego opisu systemu. Tworzona specyfikacja wymaga stosowania komentarzy w celu wyróżnienia fragmentów funkcjonalności odpowiadających poszczególnym maszynom stanowym. Wynika to z przyjętego sposobu synchronizacji tych maszyn, która odbywa się przez wprowadzenie globalnego stanu. Dla systemów realizujących większą, niż prezentowano w pracy, liczbę przypadków użycia, proponowana metoda wymaga jednak wsparcia w postaci dedykowanego narzędzia. Proces korzystający jedynie z narzędzi (w szczególności edytora) zawartych w pakiecie CADP, nie może być jednak stosowany dla dużych systemów.

2.2 Ocena wyników

Głównym czynnikiem powodującym wzrost LTS generowanego z tworzonego opisu LOTOS jest rozmiaru modelu formalnego w etapie projektu. Ilość stanów i tranzycji LTS uzależniona jest tutaj od trzech wielkości: ilości współbieżnych zadań, ich wielkości (ilości akcji) oraz od sposobu ich synchronizacji. Czynniki te warunkują rozmiar końcowego modelu wyznaczając klasę systemów, dla których proponowana metoda może być stosowana. Biorąc pod uwagę dwa pozostałe i omówione w poprzednim punkcie ograniczenia związane z rodzajem i sposobem definiowania wymagań, można przyjąć, że proponowana metoda może być stosowana dla typowych aplikacji automatyki przemysłowej bez twardych wymagań czasowych. W aplikacjach tych, równoległe zadania mogą być synchronizowane w ramach prostych wzorców projektowych, jakie zaprezentowano w pracy. Zadania

nie charakteryzują się też złożonym przetwarzaniem i mają zazwyczaj charakter reaktywny. W przypadku jednak uwarunkowań czasowych, które muszą być uwzględniane we wczesnych etapach cyklu życia, metoda powinna być zmodyfikowana pod kątem możliwości wykonania odpowiednich analiz.

W zakresie wyznaczonej klasy systemów wbudowanych, wykorzystanie formalizmu języka LOTOS przyniosło natomiast szereg korzyści. W poniższych punktach zebrano najważniejsze.

- **Weryfikacja spójnośći modeli zachowania systemu.** Podstawowym kryterium jakości tworzonego opisu zachowania systemu na poziomie analizy i projektu jest jego wewnętrzna spójność. Użycie języka LOTOS umożliwia ścisłe wyrażenie spójności w terminach relacji równoważności i preorder określonych w zbiorze wyrażeń opisujących zachowanie systemu. Zaproponowana metoda pozwala weryfikować tą spójność technikami właściwymi dla algebr procesów. Wynik weryfikacji stanowi tutaj rodzaj sprzężenia zwrotnego pomiędzy projektantem oprogramowania a tworzonym przez niego opisem systemu.

- **Implementacja przyrostów funkcjonalności.** Zmodyfikowany styl zorientowany na stany użyty w analizie, pozwala na łatwe włączanie przyrostów wymagań. Styl ten umożliwia bezpośrednie odwołanie się do globalnego stanu procesu i przez zastosowanie dozorów (warunków) na realizację dowolnej sekwencji akcji. Wykorzystanie formalnej postaci wzorców w projekcie implikuje funkcjonalną dekompozycję systemu, co także ułatwia wprowadzanie zmian w powstałej specyfikacji. Odpowiednie modyfikacje często dotyczą tylko jednego składnika struktury specyfikacji projektu. Najprostsze zmiany obejmują inną parametryzację wykorzystanych wzorców.

- **Powtarzalność procesu.** Dla wybranej klasy systemów, proces konstrukcji modeli jest powtarzalny, tzn. zdefiniowane czynności nie ulegają modyfikacjom i następują w ustalonej arbitralnie kolejności. Cecha ta decyduje o możliwości utworzenia narzędzia wspierającego zaproponowany proces. W wyjątkowych przypadkach niektóre modele nie muszą być tworzone a stosowna czynność zostaje pominięta.

- **Ograniczenie ilości notacji.** Język LOTOS stanowi formalizm podstawowy opisu zachowania zastępujący diagramu opisu dynamiki w UML. Model wyjściowy i przyrosty funkcjonalności wynikające z iteracji metody bazowej ROPES są uwzględniane bezpośrednio w strukturze specyfikacji formalnej. Nie stosuje się tutaj notacji pośrednich.

- **Użycie gotowych szablonów.** Translacje wzorców projektowych dokonywane są poza projektem i stanowią bibliotekę gotowych do użycia szablonów LOTOS. Mogą zostać niezależnie dopracowane pod kątem zachodzenia określonych własności.

- **Ocena rozwiązań projektowych.** Zastosowana technika umożliwia porównanie różnych rozwiązań projektowych. Wyższy stopień spójności dla danego modelu analizy może decydować tutaj o wyborze danego rozwiązania. Rozwiązanie definiowane jest przez zastosowanie formalnego modelu wzorca projektowego - odpowiedniego dla utworzonej w procesie statycznej struktury systemu.

Przedstawiony w pracy proces realizuje więc cel postawiony we wstępie pracy. Zastosowanie języka LOTOS w procesie przyrostowym wspiera wytwarzanie wysokiej jakości oprogramowania systemów wbudowanych. Tym samym główna teza pracy zostaje wykazana.

LITERATURA

[BB87] Bolognesi T., Birnksma E. :*Introdution to ISO Specification Language LOTOS*, Computer Networks and ISDN Systems, Vol. 14, No. 1, North-Holland, 1987

[Ben09] Ben-Ari M. *Podstawy programowania współbieżnego i rozproszonego.* WNT Warszawa 2009

[Brz06] Brzuchacz R.: *Zastosowanie języka LOTOS do modelowania wybranych struktur metody HOOD*. Elektrotechnika i Elektronika: półrocznik Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie, t. 25 z. 1 s. 1–9. Kraków, 2006

[Dou04] Douglass B.P.: *Real Time UML Third Edition* Advances in the UML for Real-Time Systems. Person Education, Inc. Boston 2004

[Dou99] Douglass B.P.: *Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems*, Addison-Wesley , Boston, 2002

[Fer90] Fernandez J-C. : *An Implementation of an Efficient Algorithm for Bisimulation Equivalence.* Science of Computer Programming, volume 13, number 2-3, pages 219-236, May 1990

[Huz07] Huzar Z.: *LOTOS – język formalnych specyfikacji systemów informatycznych*. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2007

[PT87] R. Paige and R. Tarjan.: *Three partition refinement algorithms.* SIAM J. Comput., No.6, 16, 1987.

[Sifa09] Sifakis, J.: *Embedded systems design - Scientific challenges and work directions*. Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE '09. 20-24 April 2009 Page(s):2 - 2