II Konferencja Naukowa KNWS'05
"Informatyka- sztuka czy rzemiosło"
15-18 czerwca 2005, Złotniki Lubańskie

ROLA I MIEJSCE JĘZYKA UML W PROJEKTOWANIU STEROWNIKÓW CYFROWYCH

Grzegorz Łabiak

Instytut Informatyki i Elektroniki, Uniwersytet Zielonogórski
65-246 Zielona Góra, ul. Podgórna 50

e-mail:G.Labiak@iie.uz.zgora.pl

STRESZCZENIE

W referacie zawarto propozycję wykorzystania technologii *UML* w projektowaniu sterowników cyfrowych. W skrótowej formie opisano najważniejsze diagramy *UML* wskazując ich rolę i miejscie w procesie projektowania sterowników cyfrowych, z jednoczesnym uwzględnieniem reaktywnej natury sterowników. Naważniejsze wnioski dyskusji zostały ujęte w formie tabelarycznej. Ponadto w referacie przedstawiono autorski system *HiCoS* jako praktyczne narzędzie łączące technologię *UML* ze sprzętową implementacją.

1. WPROWADZENIE

Jednym z głównych zadań programowania jest szeroko rozumiane, komputerowe modelowanie świata rzeczywistego, a technologia *UML* ma za zadanie modelować to, co będzie programem reprezentującym wybrany fragment rzeczywistości. Ponieważ możliwości modelowania języka *UML* nie są ograniczone do żadnego partykularnego wycinka świata, to można tę technologię wykorzystać do modelowania tych aspektów, które dotyczą układów cyfrowych, tym bardziej, że obserwuje się silne podobieństwa między procesem projektowania sprzętu i oprogramowania. W kontekście niniejszej pracy należy zwrócić szczególną uwagę na fakt, że zarówno sterownik cyfrowy jak i obiekt programowy (czy też w ogólności komponent) mogą cechować się działaniem reaktywnym. Sterowniki cyfrowe nie są jedynym nowym postulowanym obszarem zastosowań. W pracy [11] autor promuje *UML* jako język projektowania systemów mechatronicznych, gdzie czyni to w kontekscie pracy zespołów interdyscyplinarnych, operujących różnymi technologiami. Co ciekawe, jedną z postulowanych technologii przedstawionych w publikacji [11] są właśnie układy programowalne, doskonale nadające się do implementacji cyfrowych układów sterowania binarnego. Zdaniem autora, cechy *UML* (w szczególności pojęcie stereotypu [1]**Błąd! Nie można odnaleźć źródła odwołania.**[16]) oraz jego wzrastająca popularność (np. nowe zastosowania przy projektownaniu systemów czasu rzeczywistego [4][5]), uzasadniają potrzebę zbadania przydatności tej technologii w projektowaniu sterowników cyfrowych.

2. KRÓTKA CHARAKTERYSTYKA JĘZYKA UML

Język *UML* (ang. *Unified Modelling Language*) jest wynikiem połączonych wysiłków trzech znanych metodologów: Grady Booch'a, Ivara Jacobsona, James'a Rumbaugh'a [1][11]**Błąd! Nie można odnaleźć źródła odwołania.**. Każdy z nich z osobna opracował własne metodyki, które ujmują różne aspekty projektowania obiektowego. Obecnie ich prace są kontynuowane w ramach przedsięwzięcia *IBM Rational Software* **Błąd! Nie można odnaleźć źródła odwołania.**. Pierwsza wersja języka (*UML* 0.8) została opublikowana w roku 1995, a aktualną jest wersja 1.5 z roku 2003 [17].

Podstawowym celem *UML* jest modelowanie różnego rodzaju systemów (nie tylko z dziedziny oprogramowania) z wykorzystaniem pojęć obiektowych. Definicja przytoczona za [15], którą można znaleźć w [17], brzmi następująco:

> *„UML jest językiem do specyfikacji, konstruowania, wizualizacji i dokumentowania wytworów związanych z systemami intensywnie wykorzystującymi oprogramowanie."*

Jak widać, celem tego języka jest stworzenie pomostu między ludzkim rozumieniem struktury i działania systemów, a kodem wynikowym. Ponadto język ten ma służyć do specyfikowania, konstrukcji, wizualizacji i dokumentacji systemów, posiadających cechy programów. W zakresie swym język ten [14]:

- obejmuje specyfikację, konstrukcję, wizualizację i dokumentację,
- łączy w sobie różne dotychczas istniejące metodyki w jedną zunifikowaną,
- posiada wsparcie dla systemów współbieżnych i rozproszonych,
- skupia się na podejściu projektowym, w którym najistotniejszym jest postrzeganie projektowanego systemu z punktu wykorzystania przez przyszłego użytkownika lub innego nadrzędnego systemu.

Zazwyczaj proces projektowania wymaga od twórcy postrzegania systemu w różnych aspektach. Najczęściej trzeba odpowiedzieć na pytania, dla kogo system jest tworzony, jak się z niego będzie korzystać, co wchodzi w jego skład i jak zostanie zrealizowany. Podstawowym środkiem oferowanym przez *UML* do tego celu są diagramy. Zadaniem diagramów jest spoglądanie na system z różnych perspektyw. Oto niektóre z diagramów:

- *diagramy przypadków użycia* (*ang.* use case diagrams) – rolą ich jest przedstawienie funkcji projektowanego systemu w taki sposób, w jaki będą je widzieć jego przyszli użytkownicy,
- *diagramy klas* (*ang.* class diagrams) – diagramy te, w przypadku modelowania programów w języku *Java*, *C#* czy *C++*, przedstawiają typy danych, charakterystyczne dla tych języków i związki między nimi (np.: klasy, składowe, metody, dziedziczenie, osadzanie obiektów),
- diagramy odwzorowujące dynamiczne zachowanie systemu:

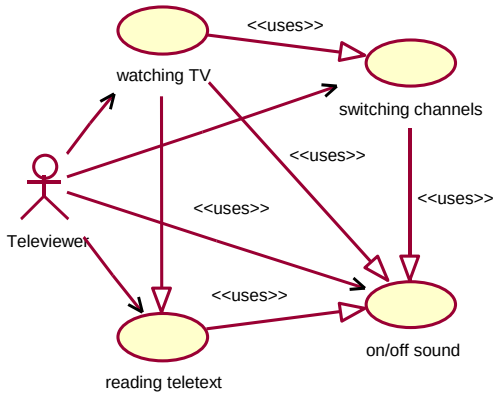
- *diagramy sekwencji* (*ang.* sequential diagrams) – przedstawiają sekwencje komunikatów i zdarzeń, przesyłanych między obiektami dla pewnego przypadku użycia,
- *diagramy współpracy* (*ang.* collaboration diagrams) – podobnie jak diagramy sekwencji, ukazują interakcje między modelowanymi obiektami z uwzględnieniem ich statycznej struktury,
- *diagramy stanów* (*ang.* statechart diagrams) – przedstawiają dynamikę stanów obiektów,
- *diagram aktywności* (*ang.* flowchart diagrams) – diagramy przepływu sterowania,

- diagramy implementacyjne:
  - *diagramy komponentów* (*ang*. component diagrams),
  - *diagramy wdrożeniowe* (*ang*. deployment diagrams).

Dla potrzeb projektowania współbieżnych kontrolerów (sterowników) cyfrowych, najistotniejszymi diagramami wydają się być diagramy przypadków użycia, aktywności, współpracy i diagramy stanów w szczególności, które wprost odwołują się do pojęcia automatu skończonego [3][6][7][8][10][18].

3. ZASTOSOWANIE UML W PROCESIE PROJEKTOWANIA UKŁADÓW CYFROWYCH

Język *UML* zawiera w sobie całe bogactwo środków. Nie wszystkie z nich dobrze nadają się do wykorzystania podczas projektowania sterowników cyfrowych, bo też i nie taka potrzeba przyświecała twórcom tego języka. Z drugiej jednak strony, język ten ma służyć pomocą w projektowaniu jak najszerszej gamy różnych systemów, a zatem wydaje się interesujące zbadanie możliwości języka pod kątem wykorzystania w procesie projektowania kontrolerów cyfrowych implementowanych, za pośrednictwem języka *VHDL*, w strukturach *FPGA*.

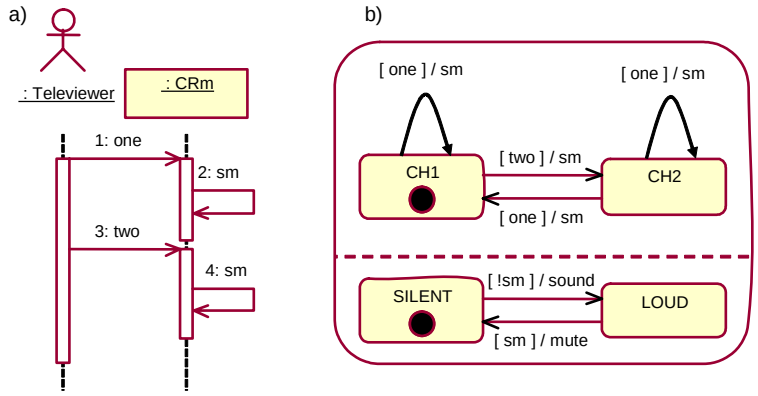


Rys. 1. Diagram przypadków użycia dla pilota telewizyjnego (oprac. na podst.[12])

Zadaniem pierwszego etapu projektowania, nazywanego specyfikacją [1], jest analiza potrzeb użytkownika. Do tego celu można wykorzystać diagramy przypadków użycia, gdzie podaje się jak system będzie współpracował z użytkownikiem lub innym systemem. Rys. 1 przedstawia przykładowy diagram przypadków użycia dla układu, który realizuje funkcje pilota telewizyjnego. Przykład ten jest zaczerpnięty z [12], a diagramy zostały zrealizowane przy użyciu oprogramowania *Rational Rose* **Błąd! Nie można odnaleźć źródła odwołania.**. Piktogram z Rys. 1, przedstawiający człowieka nazywanego w *UML aktorem*, symbolizuje system nadrzędny, tzn. taki system (w tym przypadku człowieka), który z układu będzie korzystać. Owale wraz z podpisami oznaczają realizowane przypadki użycia. Poszczególne przypadki użycia można poddać bardziej szczegółowej analizie.

Szczególnie istotnym jest przedstawienie zachowania układu pod wpływem przychodzących sygnałów. Rys. 2a zawiera diagram sekwencji dla przypadku użycia *switching channels*. Przełączanie kanałów polega na naciskaniu przycisków *one* lub *two* (w przypadku telewizora posiadającego dwa kanały), w odpowiedzi na które jest ustawiany sygnału *sm*. Zadaniem tego sygnału jest krótkotrwałe wyłączenie niechcianego szumu, związanego ze zmianą programów.

Diagramy współpracy pokazują działanie układu (lub jak w tym przypadku jego fragmentu) z uwzględnieniem chronologii zdarzeń i przy ich pomocy nie jest możliwe zamodelowanie w sposób kompletny i jednoznaczny funkcjonowania całego układu. Do tego celu można wykorzystać diagramy stanów (diagramy statechart) [3]. Diagramy te pozwalają na kompletne i jednoznaczne specyfikowanie zachowanie układu. Ten rodzaj opisu spełnia wymogi formalne stawiane przez syntezę strukturalną [1]. Rys. 2b przedstawia diagram stanów dla omawianego wcześniej przypadku użycia. Krągłokąty oznaczają stany, strzałki przejścia między stanami, a linia przerywana oznacza, że dwa przedzielone automaty sekwencyjne znajdują się w relacji współbieżności.



Rys. 2. Przypadek użycia switching channels: a) diagram sekwencji, b) diagram stanów

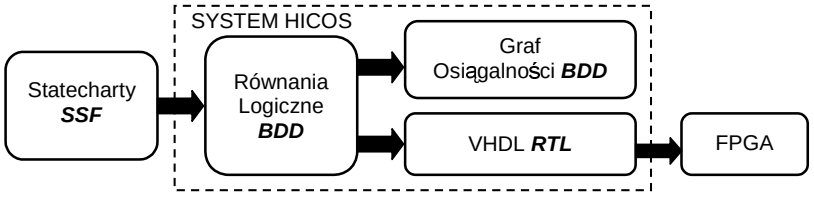
4. SYSTEM HiCoS CZYLI UML W PRAKTYCE

Notacje języka *UML* mogą dobrze wspierać proces projektowania układów cyfrowych, a ich rola jest szczególnie istotna na etapie specyfikacji oraz w procesie dokumentowania projektu. Postępując zgodnie za przedstawionymi w pracy [1] zasadami projektowania układów cyfrowych systematyczną metodą strukturalną, autor pokusił się o zaproponowanie metodologii projektowej wykorzystującej technologię *UML*, gdzie autorski system *HiCoS* [8][10][18] wykonuje automatyczną syntezę. Tabela 1 wymienia fazy procesu projektowania, podając kolejno wykorzystywane diagramy *UML* oraz rolę systemu *HiCoS*. Kursywą zaznaczono elementy nie wspierane przez *HiCoS* lub realizowane poza środowiskiem wykorzystując dane z programu.

Tabela 1. UML w projektowaniu układów cyfrowych (propozycja) oraz metodologia systemu HiCoS

Fazy procesu projektowania	Język *UML*	System *HiCoS*
specyfikacja	diagramy stanów	opis graficzny/tekstowy (*SSF*)
uwiarygodnienie specyfikacji	przypadki użycia, diagramy stanów diagramy sekwencji diagramy aktywności	trawers przestrzeni stanów, *metody symboliczne*
synteza		opis równaniami logicznymi, *optymalizacja*, translacja na *VHDL*
uwiarygodnienie realizacji		*symulacje poziomu VHDL*,
dokumentowanie projektu	przypadki użycia, diagramy stanów diagramy sekwencji, diagramy aktywności	opis graficzny/tekstowy (*SSF*)
przygotowanie testów		

System *HiCoS* został pomyślany jako system *CAD* (Rys. 3), dla którego technologia *UML* ma za zadanie, zgodnie ze swym założeniem, specyfikować, wizualizować i dokumentować takie artefakty dotyczące systemów reaktywnych jak interakcje, docelowe wykorzystywanie, przejścia, czy w ogólności zachowanie. Głównym zadaniem systemu jest synteza modelu sterownika binarnego, specyfikowanego diagramami statechart dla potrzeb implementacji w strukturach programowalnych.



Rys. 3. Schemat ideowy sytemu HiCoS

W proponowanym procesie projektowania cyfrowych sterowników binarnych, specyfikacja ogranicza się do formalnego sporządzenia diagramów stanów ilustrujących zachowanie. W systemie *HiCoS* jest to reprezentowane przez równoważną postać tekstową (format *SSF*, [8][18]). Uwiarygodnienie specyfikacji odbywa się poprzez podanie diagramów przypadków użycia, diagramów sekwencji i aktywności. Również pomocne mogą być diagramy stanów. Te zapisy nie wymagają głębokiej wiedzy inżynierskiej i mogą służyć w rozmowach ze zleceniodawcą projektu układu, często nie posiadającego odpowiedniego przygotowania technicznego. Dodatkowo specyfikację można uzasadniać bardziej szczegółowo metodami symbolicznymi, które pozwalają ustalić, czy zadany układ jest żywotny i bezpieczny. Do tego celu wykorzystuje się przestrzeń stanów układu, np. reprezentowaną przez funkcję charakterystyczną [7]. Etap syntezy jest wykonywany automatycznie, a jego wynikiem są równania logiczne opisujące działanie specyfikowanego układu [9]. Jest to również miejsce dla wykonania stosownych optymalizacji. W tej fazie i fazie następnej – jaką jest uwiarygodnienie realizacji – technologia *UML* nie oferuje żadnych propozycji. Uwiarygodnienie projektu może być częściowo dokonane na drodze symulacji specjalnego modelu testowego (*ang*. test benches) w języku *HDL*. Niektóre systemy, jak np. *MAGNUM Statemate* czy *visualState*, do uwiarygodnienia oferują możliwość uruchomienia specyfikowanego modelu w taki sposób, jak by to miało być wykonane przez rzeczywisty układ. Dokumentowanie, jako jeden z głównych celów *UML*, jest zbiorem wszystkich diagramów, tworzonych w kolejnych fazach procesu projektowania.

5. MODELOWANIE SETROWNIKA CYFROWEGO JAKO SYSTEMU REAKTYWNEGO

Głównym środkiem specyfikowania zachowania sterownika cyfrowego posiądającego cechy systemu reaktywnego, w proponowanej metodyce, są diagramy statechart. Dodatkowo diagramy interakcji szczegółowo mogą opisywać zachowanie systemu związane z konkretnym przypadkiem użycia. Najważniejszą czynnością modelowania jest określenie trzech rzeczy [2]: stanów (czyli warunków w których system może się znaleźć przez pewien zauważalny czas), zdarzeń (uruchamiających przejścia) i akcji (głównie generowane zdarzenia związane z realizacją przejść). Ponadto w pracy [2] można znaleźć m.in. takie oto szczegółowe zalecenia dotyczące czynności projektowych:

- ustalenie otoczenia systemu,
- zidentyfikowanie stanu początkowego i końcowego modelowanego systemu,
- ustalenie możliwych stanów systemu, poczynając od stanów najwyższego poziomu,
- określenie częściowego porządku stanów w historii życia systemu,
- wskazanie zdarzeń uruchamiających przejścia między stanami, zapisanych w postaci listy wcześniej opracowanych sekwencji częściowego porządku stanów,
- skojarzenie akcji z przejściami,

- wprowadzenie podstanów, złączeń i rozgałęzień sterowania celem uproszczenia modelu,
- sprawdzenie podstawowych warunków bezpieczeństwa i żywotności,
- symulacja: ręczna lub automatyczna.

Tak określone zalecenia decydowały o funkcjonalności przygotowanego programu *HiCoS* oraz stanowiły bazę dla zdefiniowania semantyki diagramów wykorzystywanej przez system. Nie wszystkie założone funkcje udało się zrealizować w programie, lecz rezultaty już uzyskane stanowią solidną podstawę do dalszych prac.

6. POSUMOWANIE

Metody projektowania układów cyfrowych wykazują silne podobieństwo do metod tworzenia oprogramowania. Nowoczesna technologia *UML*, wywodząca się z inżynierii oprogramowania może być z sukcesem stosowana w projektowaniu cyfrowych układów sterowania binarnego. Jest to tym bardziej zasadne, że układy takie wykazują naturę reaktywną, a do modelowania zachowania takich układów wyjątkowo skutecznie nadają się diagramy statechart.

Prace są finansowane ze środków KBN, jako projekt badawczy nr 4 T11C 006 24.

LITERATURA

[1] M. Adamski: *Projektowanie układów cyfrowych systematyczną metodą strukturalną*, Monografia Nr 49, Wydawnictwo Wyższej Szkoły Inżynierskiej w Zielonej Górze, Zielona Góra, 1990

[2] G. Booch, J. Rumbaugh, I. Jacobson: *UML przewodnik użytkownika*, Wydawnictwa Naukowo-Techniczne, Warszawa 2001

[3] D. Harel: *Statecharts: A visual formalism for complex Systems*, Science of Computer Programming, Vol.8, 1987

[4] T. Licht, W. Fengler: *Timing analysis of UML-RT diagrams using timed automata*, 27th IFAC/IFIP/IEEE Workshop on Real-Time Programming, Oxford, Elsevier, Łagów, Poland 2003, ss. 123-128

[5] S. Lu, W. A. Halang, A. Pöschmann: *Extending UML with PEARL features for the design of embedded real-time systems*, 27th IFAC/IFIP/IEEE Workshop on Real-Time Programming, Oxford, Elsevier, Łagów, Poland 2003, ss. 91-96

[6] G. Łabiak: *Statecharts Diagram as Linked Hierarchical Sequential Automata*, The Fourth International Conference on Computer-Aided Design of Discrete Devices CAD DD'2001, November 14-16, Mińsk, Belarus, 2001, Vol. 1, ss. 38-43

[7] G. Łabiak: *Symbolic States Exploration of Controllers Specified by Means of Statecharts*, Proc. of the Intl. Workshop DESDes'01, Przytok 2001, ss. 209-214

[8] G. Łabiak: *HiCoS – akademicki system do projektowania hierarchicznych współbieżnych cyfrowych układów sterowania*, I Krajowa Konferencja Elektroniki, KKE'02, Kołobrzeg – Dźwirzyno czerwiec 2002, ss. 397-402

[9] G. Łabiak: *From UML statecharts to FPGA - the HiCoS approach*, Forum on Specification & Design Languages - FDL '03, Frankfurt, Niemcy, 2003, ss. 354-363

[10] G. Łabiak, M. Adamski: *Hierarchiczny diagram stanów i jego odwzorowanie w strukturze logicznej FPGA*, Reprogramowalne Układy Cyfrowe - RUC 2004, Materiały VII Krajowej Konferencji Naukowej, Szczecin, Polska, 2004, ss. 103-110

[11] Z. Mrozek: *UML as integration tool for design of the mechatronic system*, Proc. of the Second Interational Workshop on Robot Motion and Control, Bukowy Dworek, Poland, October 18-20, 2001, ss. 189-194

[12] D. Nazareth, F. Regensburger, P. Sholz: Mini-Statecharts, *A Lean Version of Statecharts*, Technical Report TUM–I9610, Technische Universität München 1996

[13] IBM Rational Software, http://www-306.ibm.com/software/rational/

[14] K. Subieta: *Język UML*, V Konferencja PLOUG, Zakopane '99, ss. 235-252

[15] K. Subieta: *Wprowadzenie do obiektowych metodyk projektowania i notacji UML*, Jedenasta Górska Szkoła PTI, Szczyrk '99, http://www.ipipan.waw.pl/~subieta/OOSEMIN/UMLSzczyrk.pdf

[16] K. Subieta: *Słownik terminów z zakresu obiektowości*, Akademicka Oficyna Wydawnicza PLJ, Warszawa 1999

[17] UML 1.5 Specification, Object Management Group, http://www.omg.org/technology/documents/formal/uml.htm

[18] HiCoS, System Homepage, www.uz.zgora.pl/~glabiak