Grzegorz BAZYDŁO
UNIWERSYTET ZIELONOGÓRSKI, WYDZIAŁ ELEKTRONIKI, INFORMATYKI I TELEKOMUNIKACJI

Specyfikacja behawioralna dla rekonfigurowalnych sterowników logicznych z wykorzystaniem diagramów maszyny stanowej z języka UML 2.0

Mgr inż. Grzegorz BAZYDŁO

Jest absolwentem Uniwersytetu Zielonogórskiego (2004). Ukończył studia na specjalności Inżynieria Komputerowa. Zainteresowania naukowe koncentrują się wokół nowoczesnych metod projektowania specjalistycznych układów cyfrowych.

e-mail: G.Bazydlo@weit.uz.zgora.pl

Streszczenie

Język UML to graficzny język do obrazowania, specyfikowania, tworzenia i dokumentowania szeroko pojętych systemów informatycznych. Jego obecna wersja 2.0 wprowadziła wiele zmian, także w diagramach stanów, które teraz nazywają się diagramami maszyny stanowej. W referacie przedstawiono nietypowe wykorzystanie diagramów maszyny stanowej, bo do modelowania programów dla rekonfigurowalnych sterowników logicznych (specyfikacja behawioralna). Na początku referatu krótko zdefiniowano sterownik logiczny, a następnie omówiono opracowaną metodę jego specyfikacji z wykorzystaniem diagramów maszyny stanowej (UML 2.0), która, zdaniem autora, bardzo dobrze nadaje się do modelowania hierarchicznych układów współbieżnych. Zwrócono także uwagę na możliwość używania do specyfikacji programów dla sterowników logicznych, często darmowych, narzędzi UML do modelowania systemów informatycznych. Omawiane zagadnienia poparte zostały stosownymi przykładami.

Słowa kluczowe: UML, specyfikacja behawioralna, statechart.

A behavioral specification for reconfigurable logic controllers using UML 2.0 state machine diagrams

Abstract

The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of the best engineering practices that have proven successful in modeling large and complex systems [4, 5]. The authors of UML are Grady Booch, Ivar Jacobson and James Rumbaugh. The current version of the language is 2.0. The UML language contains thirteen kinds of diagrams (structure and behavior diagrams). One of the behavior diagrams is a state machine diagram that defines a set of concepts that can be used for modeling discrete behavior through finite state transtion systems [11]. The UML language can be used not only for designing software systems, but also for other kinds of them, for example reactive systems [2, 7, 10]. This paper presents a method of using the UML language for behavioral specification for logic controllers such as PLC, RLC and reconfigurable FPGAs. Emphasis is put on diagrams that represent behavioral state machines, because they refer directly to the definition of Finite State Machines [6]. It is worth mentioning that state machine diagrams support various features of the modeling systems such as hierarchy and orthogonality. This support allows for designing the behavior of the complex and orthogonal systems in an intuitive and clear way, on the selected hierarchical level. For example Figure 3 shows a state machine diagram for "Reactor" model on the highest hierarchy level and Figure 4 represent all details of the designed system (lowest hierarchy level). Also a possibility of using UML tools was discussed. As for future research, the use of other diagrams from UML is going to be investigated, e.g., use case diagrams or activity diagrams. The former can be applied to analyze the user's needs and interface of the designed device. The activity diagrams can be used to prepare testbenches for the modeled system. But the main method to model the behavior of a system are state machine diagrams.

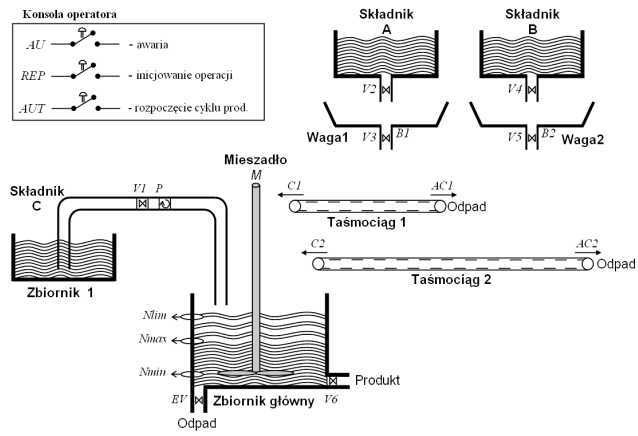
Keywords: UML, behavioral specification, statechart.

1. Sterownik logiczny jako system reaktywny

Sterownik logiczny można rozpatrywać jako przykład systemu reaktywnego [7]. W systemach tych, w odróżnieniu od systemów transformujących (systemów sterowanych danymi), dane wejściowe mogą pojawiać się w dowolnym momencie, co więcej, oczekuje się, aby reakcja systemu na te zdarzenia była natychmiastowa. Ponadto o systemach reaktywnych można powiedzieć, że są sterowane zdarzeniami, prowadzą stałą interakcję z otoczeniem (z użyciem sygnałów i przerwań), zmieniają swój stan w zależności od bieżącego trybu działania i przeszłego zachowania oraz są to często systemy współbieżne, z wyraźnie zarysowaną hierarchią behawioralną [10].

2. Przykład *Mieszalnik*

Przykładem sterownika logicznego może być układ sterowania *Mieszalnikiem*. Jest to zmodyfikowana wersja przykładu *Reaktor* zaczerpnięta z pracy G. Łabiaka [10]. Schemat procesu technologicznego mieszalnika przedstawiono na rys. 1.



Rys. 1. Schemat procesu technologicznego *Mieszalnika*
Fig. 1. The scheme of *Reactor*'s technological process

Działanie systemu mieszalnika można podzielić na kilka etapów:

Etap I – inicjowanie pracy

Po wystąpieniu sygnału *REP* (naciśnięcie przycisku na konsoli operatora) następuje opróżnianie zbiornika głównego (usunięcie odpadu) aż do poziomu *Nmin* poprzez otwarcie zaworu *EV*. W tym czasie taśmociągi 1 i 2 wprawiane są w ruch do tyłu (*AC1*, *AC2*) aby usunąć pozostałości z poprzedniego cyklu technologicznego. Taśmociągi 1 i 2 pracują przez czas *t1*, którego upłynięcie sygnalizowane jest wystąpieniem sygnału *FT1*.

Etap IIa – normalny cykl pracy (Napełnianie)

Jeżeli poziom cieczy w zbiorniku głównym jest mniejszy od poziomu minimalnego *Nmin*, to jeśli aktywny jest sygnał *AUT* (naciśnięcie przycisku na konsoli operatora) otwierają się zawory *V1*, *V2* i *V4* oraz wprawiana jest w ruch pompa *P*. Jeżeli w czasie napełniania zbiornika głównego poziom piany podniesie się powyżej poziomu *Nlim*, zawór *V1* jest zamykany oraz pompa *P* jest zatrzymywana. Po opadnięciu piany poniżej poziomu *Nlim* następuje ponowne otwarcie zaworu *V1* i uruchomienie pompy *P* aż do wypełnienia zbiornika cieczą do poziomu *Nmax*.

W tym samym czasie na wagi 1 i 2 nasypywane są składniki A i B (otwarcie zaworów *V2* i *V4*). Wystąpienie sygnałów *B1* oraz *B2* oznacza, że na wagach znajduje się odpowiednia ilość składników A i B, co oznacza zamknięcie zaworów *V2* i *V4*.

Etap IIb – normalny cykl pracy (Proces)

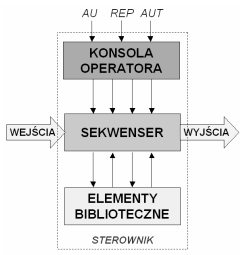
Jeżeli w zbiorniku głównym znajduje się odpowiednia ilość cieczy (poziom *Nmax*) oraz na wagach znajduje się wymierzona ilość składników A i B (czujniki *B1* i *B2*) następuje otwarcie zaworów *V3* i *V5* oraz uruchomienie taśmociągów 1 i 2 (*C1* i *C2*). Składniki dodawane są do cieczy w zbiorniku głównym i uruchamiane jest mieszadło (sygnał *M*). Po upływie czasu *t1* (sygnalizowane wystąpieniem sygnału *FT1*) zamykane są zawory *V3* i *V5* oraz zatrzymywane są taśmociągi 1 i 2. Następuje otwarcie zaworu zbiornika głównego *V6* i odbiór produktu. Po upływie czasu *t2* (sygnał *FT2*) zatrzymywane jest mieszadło. Jeśli poziom cieczy (produktu) w zbiorniku obniży się poniżej poziomu *Nmin* następuje zamknięcie zaworu *V6*.

Jeśli aktywny jest sygnał *AUT* (konsola operatora) następuje ponowne wykonanie normalnego cyklu pracy systemu (etap II).

Etap III – awaryjne zatrzymanie systemu

Ze względów bezpieczeństwa przewiduje się możliwość nagłego zatrzymania pracy całego systemu jeżeli wystąpi awaria – sygnał *AU* (konsola operatora). Jeżeli sygnał *AU* pojawi się w czasie etapu IIa (Proces), produkt reakcji jest niezdatny do wykorzystania. Następuje wtedy zatrzymanie pracy całego układu, a następnie zbiornik jest opróżniany i usuwane są składniki z taśmociągów (etap I – Inicjowanie pracy). Jeżeli natomiast sygnał awarii *AU* wystąpi podczas etapu IIb (Napełnianie), to następuje zatrzymanie pracy całego układu, a po usunięciu awarii (brak sygnału *AU* oraz aktywny sygnał *REP*) system wznawia swoją pracę – etap IIb (Napełnianie).

Schemat blokowy sterownika logicznego *Mieszalnika* przedstawia rysunek 2.



Rys. 2. Schemat blokowy sterownika *Mieszalnika*
Fig. 2. The block diagram of *Reactor*'s controller

3. Język UML

Język UML (ang. *Unified Modeling Language*) jest znormalizowanym językiem modelowania wizualnego, służącym do zapisywania projektu systemu [4]. Może być on, zdaniem autorów (Grady Booch, Ivar Jacobson, James Rumbaugh), używany do obrazowania, specyfikowania, tworzenia i dokumentowania szeroko pojętych systemów informatycznych [4, 5]. Pierwsza wersja języka UML została opublikowana w roku 1995 (ver. 0.8), obecnie UML nosi numer 2.0 [11], choć już mówi się o wersji 2.1.1.

Podstawowym celem UML jest modelowanie różnego rodzaju systemów (nie tylko informatycznych) z wykorzystaniem pojęć obiektowych. Język UML [10]:

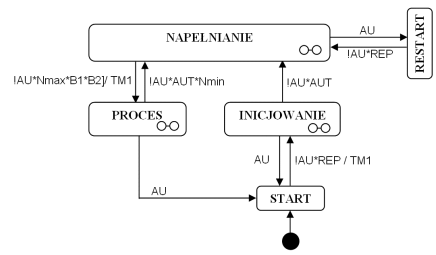
- łączy różne istniejące metodyki w jedną zunifikowaną,
- obejmuje specyfikację, konstrukcję, wizualizację i dokumentację projektowanego systemu,
- pozwala spojrzeć na system z punktu widzenia przyszłego użytkownika (funkcje systemu).
- posiada wsparcie dla systemów współbieżnych, hierarchicznych i rozproszonych,

Język UML ten jest na tyle wyrazisty i uniwersalny, że można za jego pomocą modelować systemy nie związane bezpośrednio z oprogramowaniem [2, 9]. Podstawowym środkiem oferowanym przez język UML są diagramy [11], które można traktować jako swego rodzaju rzut systemu. W wersji 2.0 wyróżnia się trzynaście rodzajów diagramów, które można podzielić na dwie grupy: struktury oraz dynamiki. Do grupy diagramów struktury należą diagramy: klas, obiektów, pakietów, struktur połączonych, komponentów, rozlokowania. Pozostałe siedem diagramów to diagramy dynamiki: diagramy przypadków użycia, czynności, maszyny stanowej, sekwencji, komunikacji, harmonogramowania, sterowania interakcją.

Najciekawsze z punktu widzenia wykorzystania do specyfikacji behawioralnej dla sterowników logicznych wydają się być diagramy maszyny stanowej, ponieważ wprost odwołują się do pojęcia automatu skończonego [6].

4. Diagramy maszyny stanowej

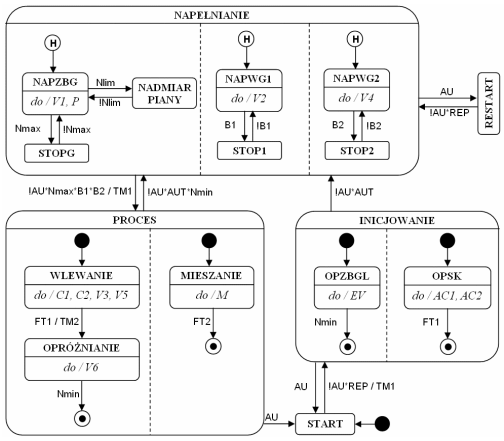
Diagramy maszyny stanowej służą do modelowania dynamicznych aspektów systemu. Opisują one stany, w których może znaleźć się projektowany system, a także jak zmienia się stan systemu pod wpływem zdarzeń, które do niego docierają. Zdarzenie to najczęściej wystąpienie sygnału (nadawczego lub odbiorczego) [12], który może wyzwolić przejście między stanami. Przejście to związek między dwoma stanami, wskazujący, że system znajdujący się w pierwszym stanie wykona pewne akcje i przejdzie do drugiego stanu, ilekroć zajdzie określone zdarzenie i będą spełnione odpowiednie warunki. Z przejściem może też być związana określona czynność, wykonywana podczas realizacji danego przejścia [4]. Można więc powiedzieć, że diagramy maszyny stanowej to graficzne odzwierciedlenie dyskretnego, skokowego zachowania skończonych systemów typu stan-przejście [12]. Na rys. 3 znajduje się diagram maszyny stanowej zachowania (ang. *behavioral state machines*) dla prezentowanego przykładu *Mieszalnika*.



Rys. 3. Diagram maszyny stanowej dla Mieszalnika (najwyższy poziom hier.)
Fig. 3. State machine diagram for Reactor - the highest hierarchy level

Stany reprezentowane są w sposób graficzny jako prostokąty z zaokrąglonymi rogami (tzw. krągłokąty [8]). Przejścia pomiędzy stanami oznaczane są strzałkami. Na prezentowanym diagramie z każdym przejściem związane jest zdarzenie uruchamiające to przejście, oznaczające wystąpienie danego sygnału (np. *AU*) lub kombinacji sygnałów (np. *!AU*Nmax*B1*B2*). Wykrzyknik przed nazwą sygnału oznacza jego negację, a symbol gwiazdki (*) oznacza iloczyn logiczny sygnałów, czyli jednoczesne wystąpienie danej kombinacji sygnałów. Z przejściem może być także skojarzona pewna czynność np. *!AU*REP/TM1*, gdzie *TM1* oznacza uaktywnienie sygnału *TM1* uruchamiającego zegar (*timer*).

Diagramy maszyny stanowej pozwalają w sposób intuicyjny i czytelny specyfikować zachowanie złożonych systemów współbieżnych na wybranym poziomie uszczegółowienia, ponieważ wspierają takie cechy modelowanego układu, jak hierarchiczność i współbieżność. Jeśli nie ma potrzeby prezentacji wszystkich informacji modelowanego systemu, można przyjąć wyższy poziom hierarchii i ukryć zbędne (na danym etapie projektowania) szczegóły. W prezentowanym przykładzie stany *Napełnianie*, *Proces* i *Inicjowanie* to w rzeczywistości stany złożone (oznaczone symbolem dwóch małych, połączonych okręgów w prawym dolnym rogu stanu). Stany złożone mogą zawierać podmaszyny stanowe oraz obszary współbieżne. Na rys. 4 zaprezentowano diagram dla prezentowanego przykładu ze wszystkimi podstanami w stanach złożonych (najniższy poziom hierarchii). Występujący na rys. 4 symbol pseudostanu historii płytkiej (litera *H* w okręgu) oznacza, że po przekazaniu sterowania (uaktywnienie stanu *Napełnianie*) do takiej podmaszyny stanowej pierwszym aktywnym stanem będzie ten, który był aktywny w poprzednim okresie aktywności podmaszyny stanowej. Jeżeli podstan staje się aktywnym po raz pierwszy, sterowanie zostanie przekazane do tego stanu, na który wskazuje pseudostan historii.



Rys. 4. Diagram maszyny stanowej dla *Mieszalnika* (najniższy poziom hier.)
Fig. 4. State machine diagram for *Reactor* - the lowest hierarchy level

Obecna wersja UML (2.0) wprowadziła wiele zmian do diagramów maszyny stanowej. Przede wszystkim zostały one rozszerzone o nowe elementy, takie jak np.: punkty wejścia (*entry point*), wyjścia (*exit point*) oraz zniszczenia (*terminate*). Ponadto uporządkowano semantykę diagramów, dokonano redefinicji niektórych elementów np. pseudostanów płytkiego i głębokiego wznowienia (*shallow* oraz *deep history*) oraz zmieniono nazwę diagramów [11].

W literaturze można spotkać wiele postaci diagramów stanów (ang. statechart), każda posiada nieco inną semantykę. Diagramy maszyny stanowej zawarte w UML oparte są na mapach stanów (statecharts) zaproponowanych w 1987 roku przez Davida Harela [6,7,8,10,1]. Obserwując kolejne wersje UML można zauważyć jak bardzo diagramy te ewaluowały. Obecna semantyka jest bardziej czytelna, dokonano wyraźnej klasyfikacji stanów i przejść, zdefiniowano lub redefiniowano podmaszyny stanowe oraz pseudostany (np. punkt wejścia, wyjścia, punkt zniszczenia, decyzja, punkt węzłowy, płytkie i głębokie wznowienie), określono rodzaje zdarzeń dodając między innymi zdarzenia zmiany stanu (ang. change event), czy zdarzenia odroczone (ang. deferred event).

Jednak nie wszystkie elementy, które weszły do kolejnych wersji języka UML można wykorzystać do specyfikacji behawioralnej dla sterowników logicznych. W porównaniu do poprzednich wersji UML obecna specyfikacja jest bardziej precyzyjna, ale niestety nie na tyle, by można było wykonać bezpośrednie odwzorowanie do specyfikacji akceptowalnej przez współczesne przemysłowe sterowniki (np. PLC). Specyfikacja w języku UML wymaga więc albo określenia precyzyjnej składni i semantyki diagramów (np. na poziomie metamodelu [3]), bądź zbioru wskazówek i wytycznych dla projektanta. Możliwości implementacyjne diagramów maszyn stanowych z UML 2.0 w strukturach programowalnych są przedmiotem dalszych prac autora.

5. Narzędzia UML

Zaletą proponowanej metody zastosowania języka UML do specyfikacji behawioralnej dla sterowników logicznych jest możliwość wykorzystania narzędzi UML. Profesjonalne, komercyjne narzędzia do UML (np. *Enterprise Architect*) lub darmowe środowiska (np. *Jude*, *ArgoUML*, *Umbrello UML Modeller*) pozwalają na łatwe i intuicyjne tworzenie diagramów UML (w tym także diagramów maszyny stanowej), co może stanowić wsparcie procesu specyfikacji oraz dokumentacji programów dla sterowników logicznych. Zastosowanie wielu różnych rodzajów diagramów ułatwia komunikację pomiędzy grupami projektantów a także pozwala lepiej określić potrzeby i oczekiwania użytkownika, zaprojektować interfejs (diagramy przypadków użycia), programy testujące dla modelowanego systemu (diagramy czynności, diagramy sekwencji) czy w końcu zbudować bardziej czytelną dokumentację projektu. Niestety obecne narzędzia UML nie zastąpią profesjonalnych systemów do behawioralnej specyfikacji dla sterowników logicznych (np. *BetterState*, *Statemate MAGNUM*). Pewnym rozwiązaniem jest stworzenie translatora, którego zadaniem byłoby przetworzenie specyfikacji diagramów np. w języku XML (wiele edytorów UML generuje taką specyfikację) na język opisu sprzętu (np. *Verilog*, *VHDL*). Opisywane rozwiązanie i realizacja takiego narzędzia jest przedmiotem dalszych prac autora.

6. Podsumowanie

Język UML zawiera wiele notacji graficznych służących do obrazowania, specyfikowania, tworzenia i dokumentowania modelowanych systemów. Wydaje się, że diagramy maszyny stanowej z języka UML 2.0 mogą dobrze służyć jako narzędzie specyfikacji behawioralnej dla sterowników logicznych. Diagramy te pozwalają na kompletne i jednoznaczne specyfikowanie zachowania projektowanego systemu, a także wspierają takie jego cechy jak współbieżność i hierarchiczność. Specyfikacja behawioralna algorytmu sterowania binarnego, przedstawiona w postaci użytecznego, nienadmiarowego i zwartego podzbioru diagramów maszyny stanowej, może być potraktowana jako abstrakcyjna forma programu dla sterownika logicznego. Obecna wersja języka UML zawiera całe bogactwo środków służących do modelowania systemów nie tylko informatycznych. Jednak nie wszystkie elementy nadają się do specyfikacji behawioralnej. Należy pamiętać, że język UML powstał i jest rozwijany przede wszystkim w kierunku inżynierii oprogramowania. Dlatego wielu projektantów wybiera z UML tylko te elementy (diagramy), które są im na danym etapie projektowania niezbędne [10]. Wykorzystanie innych diagramów (szczególnie diagramów przypadków użycia, czynności, sekwencji czy harmonogramowania) pod kątem specyfikacji behawioralnej dla sterowników logicznych jest przedmiotem dalszych prac autora. Zaletą proponowanej metody jest także możliwość wykorzystania, często darmowych, narzędzi UML w procesie projektowania układów cyfrowych na etapie specyfikacji behawioralnej.

Pracę wykonano w ramach projektu badawczego finansowanego ze środków Zintegrowanego Programu Operacyjnego Rozwoju Regionalnego (Działanie 2.6: Regionalne strategie innowacyjne i transfer wiedzy) z udziałem Europejskiego Funduszu Społecznego.

7. Literatura

[1] Bauer N., Engell S.: A Comparison of Sequential Function Charts and Statecharts and an Approach towards Integration, 58-69
[2] Bazydło G.: Wykorzystanie języka UML do modelowania małych systemów reaktywnych, Materiały VI Międzynarodowych Warsztatów Doktoranckich OWD 2004 – Wisła 2004, vol. 19, 2004
[3] McUmber W. E., Cheng B. H. C..: UML-Based Analysis of Embedded Systems Using a Mapping to VHDL, High-Assurance Systems Engineering, 1999. Proceedings. 4th IEEE International Symposium on, Washington, 1999
[4] Booch G., Rumbaugh J., Jacobson I.: UML przewodnik użytkownika, WNT, Warszawa 2001
[5] Fowler M., Scott K.: UML w kropelce, LTP, Warszawa 2002
[6] Harel D.: Statecharts, A visual formalism for complex Systems, Science of Computer Programming, Vol.8, 1987
[7] Harel D., Politi M.: Modeling Reactive Systems With Statecharts: The Statemate Approach, McGraw Hill Text, 1998
[8] Harel D.: Rzecz o istocie informatyki, Algorytmika, WNT, Warszawa 1992
[9] Jaragh M., Saleh I. A.: Modeling Computer hardware using the Unified Modeling Language, TENCON'02. Proceedings, 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering, 2002, vol. 1, 19-22
[10] Łabiak G.: Wykorzystanie hierarchicznego modelu współbieżnego automatu w projektowaniu sterowników cyfrowych, Oficyna Wydawnicza Uniwersytetu Zielonogórskiego, Zielona Góra 2005
[11] OMG: UML 2.0 Superstructure Specification, 2005 (http://www.omg.org/cgi-bin/doc?formal/05-07-04)
[12] Wrycza S., Marcinkowski B., Wyrzykowski K.: Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005

Artykuł recenzowany