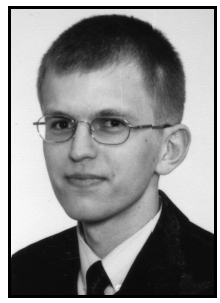
Piotr BUBACZ, Marian ADAMSKI
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI

Platforma .NET i język znaczników XML w systemie projektowania programów dla sterowników logicznych

Mgr inż. Piotr BUBACZ

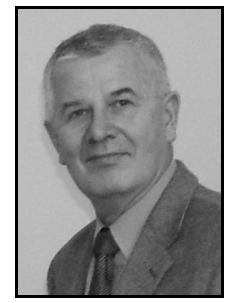
Asystent w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego, absolwent Zintegrowanych Studiów Zagranicznych Uniwersytetu Zielonogórskiego i Fachhochschule Giessen-Friedberg (Niemcy). Opiekun koła UZ.NET. Zainteresowania badawcze obejmują sieci komputerowe, projektowanie systemów cyfrowych oraz formalnych metod oprogramowania sterowników logicznych.



e-mail: P.Bubacz@iie.uz.zgora.pl

Prof. dr hab. inż. Marian ADAMSKI

Dyrektor Instytutu Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Zainteresowania badawcze obejmują projektowanie systemów cyfrowych realizowanych w postaci mikrosystemów cyfrowych oraz formalnych metod programowania sterowników logicznych. Członek IEEE, IEE, ACM, Polskiego Towarzystwa Elektrotechniki Teoretycznej i Stosowanej oraz Polskiego Towarzystwa Informatycznego.



e-mail: M.Adamski@iie.uz.zgora.pl

Streszczenie

Celem artykułu jest przedstawienie ogólnej koncepcji systemu CASE do projektowania programów dla sterowników logicznych PLC (Programmable Logic Controllers) i RLC (Reconfigurable Logic Controllers). Uzasadniono wybór języków SFC, ST i FBD, z normy IEC 61131-3, jako narzędzi do specyfikacji behawioralnej i strukturalnej. Wskazano istotne zalety zastosowania platformy .NET i języka znaczników XML.

Słowa kluczowe: SFC, XML, XSD,. IEC 61131-3, .NET.

Application of .NET platform and XML markup language in Computer Aided Software Engineering for logic controllers

Abstract

The aim of the paper is to introduce a general concept of CASE system for development system for logical controllers PLC (Programmable Logic Controllers) and RLC (Reconfigurable Logic Controllers). The motivation for choosing SFC, ST and FBD languages as tools for behavioral and structural specification of controllers is justified. Important advantages for .NET platform and XML usage are presented.

Keywords: SFC, XML, XSD,. IEC 61131-3, .NET.

1. Wstęp

Konwencjonalne sterowniki logiczne PLC (Programmable Logic Controllers) są specjalizowanymi mikrokomputerami, przeznaczonymi do sterowania dyskretnymi, najczęściej binarnymi procesami przemysłowymi. Alternatywnym sposobem ich realizacji jest implementacja mikroukładowa, w formie sterowników RLC (Reconfigurable Logic Controllers), wbudowanych na przykład do nowoczesnych rekonfigurowanych mikrosystemów cyfrowych, zawierających wewnątrz struktury FPGA [3].

Jednym z ważniejszych problemów związanych z syntezą układową sterowników rekonfigurowanych jest wybór środowiska projektowego i odpowiedniej struktury implementacyjnej. W przeprowadzonych badaniach wykorzystuje się możliwość bezpośredniego opisu diagramów SFC w językach opisu sprzętu, VHDL lub Verilog na poziomie wyrażeń logicznych. Do przygotowania nakładek współpracujących z profesjonalnymi systemami CAD proponuje się wykorzystać nowoczesną platformę .NET. Platforma ta stanowi najlepszy wybór, oferując m.in. wiele gotowych klas i rozwiązań systemowych oraz możliwość współpracy w zakresie różnych języków programowania, co umożliwia szybkie prototypowanie i dołączanie nowych funkcji do aplikacji.

Jednym z języków graficznych, wykorzystywanych do specyfikacji behawioralnej (funkcjonalnej) rekonfigurowanych sterowników logicznych są diagramy sterowania SFC (Sequential Function Chart), zwane też diagramami funkcjonowania sekwencyjnego lub grafem sekwencji [1].

Dzięki podobieństwu do interpretowanych sieci Petriego sterowania, od których poprzez sieci Grafcet się wywodzą [5], odwzorowanie diagramów SFC w matrycowych strukturach logicznych FPGA lub CPLD nie przedstawia większych trudności teoretycznych. Różnorodne sposoby realizacji układowej sieci Petriego zrelacjonowano między innymi w książce [4]. SFC został ustandaryzowany i stanowi część normy IEC 61131-3.

2. Norma IEC 61131-3

IEC 61131-3 jest to standard określający języki programowania sterowników PLC dla automatyki przemysłowej. Ze względu na ogólnoświatowe wsparcie i uznanie jest niezależny od jakiejkolwiek firmy. Standard ten jest trzecia częścią rodziny standardów IEC 61131. Norma IEC 61131-3 w 2004 roku została zaakceptowana i zaadoptowana jako polska norma PN-EN 61131-3:2004.

Standard 61131-3 można podzielić na dwie części wspólne elementy oraz języki programowania. Wspólne elementy zawierają:

- typy danych – definiują typy wykorzystywanych paramentów,
- zmienne –przydzielane adresom sprzętowym;
- jednostki organizacji programowej (ang. Program Organization Units, POUs)– funkcje (standard owe np. ADD, ABS oraz definiowane przez użytkownika), bloki funkcji (odpowiednik Układów Zintegrowanych IC, zawierają dane jak również algorytmy do ich przetwarzania) i programy (sieć funkcji i bloków funkcyjnych, która może być napisana za pomocą języka zdefiniowanego w standardzie).

Standard zawiera definicję pięciu języków programowania trzech graficznych: Schemat Drabinkowy (ang. *Ladder Diagram* – LD), Diagram Bloków Funkcyjnych (ang. *Function Block Diagram* – FBD) i Sequential Function Chart – SFC i dwóch tekstowych: Lista Instrukcji (ang. *Instruction List* – IL) i Structured Text – ST. Wybór języka programowania zależy od [8]:

- doświadczenia projektanta,
- problemu do rozwiązania,
- struktury sterowania programu,
- interfejsu do innych programistów/wydziałów.

Wszystkie cztery języki są wzajemnie połączone i umożliwiają bezpośrednią wymianę i interakcję pomiędzy nimi.

Sequential Function Chart jest graficznym opisem działania programu sterującego, zbliżonym do interpretowanej sieci Petriego sterowania.

Diagram Bloków Funkcyjnych pozwala w łatwy sposób wprowadzić modularność do projektowanego systemu sterowania binarnego, w sposób podobny jak w programowaniu obiektowym.

Structured Text jest językiem wysokiego poziomu, mającym swoje korzenie w takich językach jak Ada, Pascal i C. Zawiera wszystkie najważniejsze elementy języka programowania, włączając w to instrukcję wyboru oraz pętle. Język ten służy do szczegółowego opisania operacji wykonywanych w poszczególnych krokach (etapach) programu współbieżnego. Może on być również wykorzystany do tekstowego opisu grafu SFC oraz bloku FBD.

W ramach prowadzonych prac przewiduje się łączne wykorzystanie tych trzech języków, z silnym uwypukleniem opisu funkcjonalnego (behawioralnego) w języku graficznym SFC.

3. Diagram sterowania – SFC

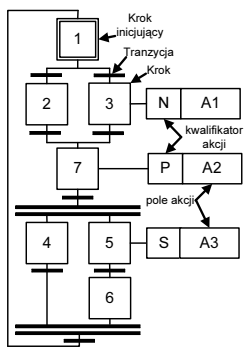
Diagram sterowania – SFC (*Sequential Function Chart*) jest językiem wysokiego poziomu. Graficznie opisuje on operacje w procesie sekwencyjnym. Język graficzny SFC został oparty na języku Grafcet, opisanym we francuskiej normie NF-C-3-190 w roku 1978. W 1988 roku organizacja IEC adaptowała metodę Grafcet jako międzynarodowy standard języka do programowania procesów sekwencyjnych, oznaczono go symbolem IEC 848. Norma ta została następnie zawarta w normie IEC 61131-3 właśnie jako język SFC. Dziś standard i język SFC są zaakceptowane przez przemysł i wykorzystywane przez wiele aplikacji do projektowania sterowników logicznych. SFC może być traktowane, jako przemysłowy wariant sieci Petriego, dzięki temu do analizy sieci SFC można wykorzystać metody formalnej analizy sieci Petriego [5]. Matematyczny model SFC wywodzi się z teorii sieci Petriego. SFC jest skierowanym grafem zdefiniowany, jako czwórka postaci [6]:

$$(S,T,L,I) \tag{1}$$

gdzie: $S=(s_1,...,s_m)$ skończony, niepusty zbiór kroków, $T=(t_1..t_n)$ skończony, niepusty zbiór tranzycji, $L=(l_1..l_k)$ skończony, niepusty zbiór krawędzi łączących kroki z tranzycjami i na odwrót, $I \subset S$ zbiór kroków inicjujących.

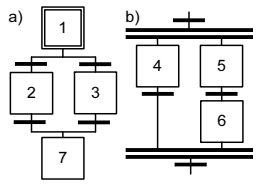
Zbiory S i T reprezentują węzły grafu. Krok inicjujący jest ustawiony na początku procesu i określa stan początkowy. Kroki mogą być aktywne lub nieaktywne. Każdy aktywny krok jest oznaczany żetonem umieszczonym w kroku. Wszystkie aktywne kroki określają stan systemu.

Z każdym krokiem może być powiązany blok akcji, w którym opisany jest rodzaj akcji, jaka będzie wykonana w chwili aktywacji kroku. Blok akcji składa się z kwalifikatora akcji (definiuje on rodzaj wykonywanej akcji) oraz pola akcji (opisanego np. w języku ST) (rys.1).



Rys. 1. Przykład grafu SFC – krok, tranzycja, kwalifikator akcji i pole akcji
Fig. 1. SFC example of step, transition, action qualifiers and actions

Tranzycja może być kontrolowana przez wyrażenie logiczne. Kiedy wszystkie poprzedzające ją kroki są aktywne i warunek przejścia jest spełniony tranzycja staje się aktywna i nastąpi jej realizacja. Realizacja następuje natychmiast. Kiedy tranzycja zostanie zrealizowana kroki poprzedzające stają się nieaktywne, a jej kroki następujące aktywne. Graf SFC może zawierać ścieżki alternatywne (rys. 2a) oraz równoległe (rys. 2b).



Rys. 2. Przykład ścieżki alternatywnej i równoległej
Fig. 2. Example of alternative and parallel path

4. Język znaczników XML dla IEC 61131-3

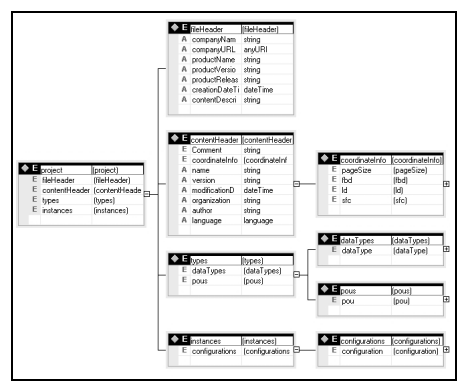
Ze względu na fakt, że aktualny standard nie definiuje uniwersalnego formatu danych do reprezentacji języków zdefiniowanych w standardzie każdy producent oprogramowania zdefiniował własny, z reguły zamknięty format danych. Spowodowało to brak możliwości wymiany programów, bibliotek i projektów pomiędzy różnymi środowiskami do projektowania układów.

W odpowiedzi na ten brak organizacja PLCopen powołała grupę mającą określić interfejs do wymiany danych pomiędzy różnymi narzędziami. Grupa ta (nazwana TC6 for XML) zdefiniowała otwarty interfejs pomiędzy różnego rodzaju programami. Zyskał on aprobatę przemysłu, ze względu na swoją uniwersalność, jak również uczestnictwo w tej organizacji, największych producentów sprzętu i oprogramowania. Dodatkowym atutem jest darmowe jego udostępnienie na stronach organizacji. Dzięki temu można już dziś wykorzystywać ten format w nowo projektowanych aplikacjach [9].

Format wymiany został zdefiniowany przy pomocy języka definicji schematów XML (XSD), który zapewnia system typów dla środowisk przetwarzania XML. Schemat XML umożliwia opisanie typów, stosowanych w dokumencie XML. Dokument XML, zgodny z typem zdefiniowanym w schemacie XML, jest często nazywany dokumentem instancji — podobnie jak w przypadku tradycyjnej, zorientowanej obiektowo relacji pomiędzy klasą i obiektem. Jest to koncepcyjne odejście od sposobu działania definicji typów dokumentów (Document Type Definitions — DTD). Nowe podejście jest o wiele bardziej elastyczne przy łączeniu z systemami typów tradycyjnych języków programowania lub baz danych. W takich środowiskach schematy XML wypierają z użycia DTD.

Dodatkowa zaleta formatu wymiany jest związana z językiem XML. Dzięki możliwością tego języka istnieje możliwość, niemalże nieograniczonego, rozszerzania dokumentu o elementy zależne od producenta i elementy te będą rozpoznawane i wykorzystywane przez programy potrafiące je interpretować, inne aplikacje nie będą brały ich pod uwagę.

Definicja schematu została tak zaprojektowana, że umożliwia przeniesienie informacji, która jest na ekranie na inne platformy. Format ten zawiera to nie tylko informację tekstową, ale również graficzną, zawiera szczegółowe dane dotyczące rozmieszczenia i połączenia elementów graficznych. W schemacie zawarto definicję wszystkich języków określonych w specyfikacji IEC 61131-3. Języki SFC, FBD, LD są zapisywane wraz z informacjami dotyczącymi elementów graficznych. Języki ST i IL oraz pola dokumentacji są reprezentowane, jako sformatowany tekst w języku XHTML. Na rys. 3 przedstawiono trzy najważniejsze poziomy definicji dokumentu zawartego w przedstawianej specyfikacji.



Rys. 3. Trzy najważniejsze poziomy definicji dokumentu zgodnie z [9]
Fig. 3. Three most important levels of document definition according to [9]

5. Możliwości platformy .NET 2.0

W roku 2000 firma Microsoft przedstawiła pomysł standaryzacji w tworzeniu oprogramowania na platformę Windows. Udostępniła ona programistom bibliotekę oraz narzędzia do tworzenia oprogramowania, które będzie charakteryzowało się pewnym

standardem. Platforma .NET stawia największy naciska na Internet jako źródło komunikacji między poszczególnymi komputerami czy ogólnie urządzeniami (telefony komórkowe, laptopy). technologia .NET nie jest systemem operacyjnym, ale raczej należy ją postrzegać jako pewną ideę i strategię.

Podstawowym elementem platformy jest *.NET Framework*, który stanowi infrastrukturę tworzenia aplikacji. Jest to szereg niezbędnych bibliotek umożliwiających tworzenie aplikacji na tę platform. Jego zadaniem jest zarządzanie elementami systemu, których obsługa do tej pory zajmowała programistom najwięcej czasu. Należy zaznaczyć, że programowanie pod .NET różni się znacząco w stosunku do dotychczasowego programowania pod Windows. Program w .NET tworzony jest na wyższym poziomie abstrakcji, przez co programista nie musi zajmować się kwestiami wykorzystania funkcji systemowych, ale jest skupiony funkcjonalnością programu..

.NET Framework składa się z dwóch głównych elementów: Wirtualnej Maszyny (ang. *Common Language Runtime* – CLR) oraz Biblioteki Klas Podstawowych (ang *Base Class Library* – BCL). Kompilatory .NET nie generują kodu maszynowego, a jedynie kod w Języku Pośrednim (ang. *Intermediate Language* – IL). Jest on podobny do asemblera, ale nie jest zależny od procesora i systemu operacyjnego. Drugi etap kompilacji, polega na skompilowaniu kodu z języka IL na instrukcje języka maszynowego procesora oraz systemu operacyjnego, pozwala to na uruchomienie napisanej aplikacji w wybranym środowisku.

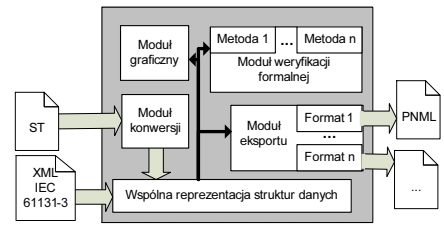
Biblioteka Klas Podstawowych (BCL) to zestaw klas i funkcji umożliwiających współprace z systemem operacyjnym i innymi technologiami, takimi jak: XML, ASP.NET czy ADO.NET. Wykorzystując *.NET Framework* możliwe jest tworzenie programów za pomocą tej biblioteki, nie używając Win32API. Dzięki temu, gdy CLR razem z BCL zostaną przeniesione na inne platformy, takie programy będą działały w innych systemach operacyjnych i na innych procesorach. Dodatkowo wykorzystując tę bibliotekę programista ma możliwość napisania jednego kodu, który po niewielkich modyfikacjach będzie mógł wykorzystywać w aplikacji okienkowej sytemu Windows, aplikacji internetowej ASP.NET oraz aplikacji na urządzenia mobilne Windows CE.

Microsoft ciągle rozwija platformę .NET, ostatnia wersja 2.0 został rozszerzona o takie mechanizmy jak: typy generyczne, iteratory klas, typy częściowe, metody anonimowe oraz typy Nullable. Ważnym rozszerzeniem są wykorzystywane w proponowanym systemie typy generyczne. Typy te rozszerzają język C# o polimorfizm parametryczny. Dodatkowym parametrem klasy lub metody mogą być zmienne określające typy. Aby użyć takiej klasy lub metody należy tę zmienną ukonkretnić. Składniowo typy generyczne przypominają szablony klas z C++, jednak w odróżnieniu od szablonów, które można traktować jak makra, są klasami za wyjątkiem określenia typu. Typy te w .NET są instancjonowane podczas uruchomienia, a nie kompilacji, jak to jest w przypadku szablonów [7]. Dzięki zastosowaniu typów generycznych możliwe jest wykorzystanie projektowanej aplikacji nie tylko do analizy grafów SFC, ale również do sieci Petriego, co ułatwi proces testowania aplikacji, a w przyszłości umożliwi stworzenie jednego wspólnego systemu dla różnych rodzajów sieci.

Dodatkowo platforma .NET umożliwia tworzenia aplikacji równolegle w różnych językach programowania i łączenia tego w jedną aplikację. Językiem dedykowanym dla platformy jest C#. Dodatkowym atutem jest możliwość bezpośredniego importu kodu napisanego w języku Java przy pomocy języka J#, którego składnia tylko w niewielkim stopniu odbiega od standardowego języka Java. Również kod napisany w języku C++ może być praktycznie bezpośrednio importowany projektu. Dzięki takiemu rozwiązaniu mogą zostać wykorzystane prototypowe aplikacje napisane do tej pory w ramach badań własnych, w tym prac dyplomowych i projektów studenckich.

6. Koncepcja systemu CASE

System CASE do projektowania programów dla sterowników logicznych jest budowany na platformie .NET. Formatem zapisu i wymiany danych jest dokument XML zgodny ze specyfikacją PLCOpen. Aplikacja będzie umożliwiała import programów zapisanych w formacie ST i automatyczną konwersję do graficznego formatu SFC (rys. 4).



Rys. 4. Schemat systemu CASE
Fig. 4. Schema of the CASE system

System będzie umożliwiał edycję i formalną weryfikację grafów SFC wykorzystując większość metod opracowanych w zespole zielonogórskim. Program będzie umożliwiał eksport grafu SFC do formatu sieci Petriego, zapisanego w języku PNML, w celu ewentualnej dalszej analizy w zewnętrznych aplikacjach. Przewidywany jest eksport sieci SFC na wyrażenia logiczne, tablice decyzyjne oraz zaimplementowanie szeregu metod kodowania w przypadku realizacji układowej.

7. Podsumowanie

Projektowany system jest ukierunkowany w stronę realizacji układowej sterownika w mikrosystemach cyfrowych. Część systemu związana z edycją, analizą oraz weryfikacją, włączając w to symulację w środowisku VHDL lub Verilog ma bardziej ogólne zastosowanie. Może być ona wykorzystana również do uwiarygodnienia (walidacji) programów dla konwencjonalnych PLC metodą intensywnej symulacji i testowania.

Platforma .NET doskonale nadaje się do implementacji środowiska projektowania programów dla sterowników logicznych. Została ona wybrana do implementacji środowiska ze względu na:

- rozbudowane wsparcie dla tworzenia i zarządzania dokumentami XML,
- łatwość przeniesienia aplikacji z wersji okienkowej na sieciową i mobilną,
- wsparcie tworzenia aplikacji w wielu językach,
- implementację typów generycznych,
- dostępność profesjonalnego środowiska do tworzenia aplikacji.

Aplikacja będzie umożliwiała nie tylko edycję i weryfikację, ale również analizę kosztów, a na jej podstawie dekompozycję na część sprzętową i programową [2].

8. Literatura

[1] Adamski M., Chodań M.: Modelowanie układów sterowania dyskretnego z wykorzystaniem sieci SFC, Wydawnictwo PZ, Zielona Góra 2000
[2] Adamski M., Skowroński Z.: Interpretowane sieci Petriego - model formalny w zintegrowanym projektowaniu mikroprocesorowych systemów sprzętowo-programowych, Pomiary Automatyka Kontrola, nr 2-3, wyd. spec., 17-20
[3] Adamski M., Węgrzyn M. Implementacja współbieżnych algorytmów sterowania w reprogramowalnych sterownikach logicznych Pomiary Automatyka Kontrola nr 2-3, wyd. spec., 21-25
[4] Adamski M., Karatkevich A., Węgrzyn M. (Red.): Design of embedded control systems, Springer, New York, 2005
[5] David R., Alla H.: Petri Nets and Grafcet, Prentice Hall Int., USA, 1992
[6] Halang W.A.: Languages and Tools for the Graphical and Textual System Independent Programming of Programmable Logic Controllers, Microprocessing and Microprogramming 27, 1989 583 – 590
[7] Lowy J.: An Introduction to C# Generics, http://msdn.microsoft.com
[8] John K., Tiegelkamp M.: IEC 61131-3: Programming Industrial Automation Systems, Springer Berlin, 2001
[9] XML Formats for IIEC 61131-3 version 1.01 – Official Release, www.plcopen.org/

Artykuł recenzowany