XVIII Seminarium

ZASTOSOWANIE KOMPUTERÓW W NAUCE I TECHNICE' 2009
Oddział Gdański PTETiS
Referat nr 30

WYKORZYSTANIE METOD ZORIENTOWANYCH OBIEKTOWO DO PROGRAMOWANIA STEROWNIKÓW PROGRAMOWALNYCH

Damian ZIELIŃSKI, Łukasz ŁUDZIŃSKI

Politechnika Gdańska, ul. G. Narutowicza 11/12, 80-952 Gdańsk
tel: 058 347 1347 fax: 058 347 1678 e-mail: dzielin@ely.pg.gda.pl

Streszczenie: W artykule przedstawione zostały główne udogodnienia wynikające z wykorzystywania zasad i metod podejścia obiektowego do programowania sterowników programowalnych. Omówiona została aktualna sytuacja tradycyjnych języków tekstowych i graficznych po wprowadzeniu rozszerzenia IEC 61499 normy IEC 61131. W dalszej części zaprezentowano podział cyklu tworzenia programu na trzy zasadnicze części – analizę, projektowanie oraz programowanie obiektowe. Zaproponowany podział, wraz z przedstawionymi metodami i zasadami, został wykorzystany do stworzenia programu Enkoder w oprogramowaniu narzędziowym CoDeSys v3. W celu stworzenia kolejnego programu Transport, wykorzystano oprogramowanie narzędziowe Step 7. W końcowej części, przedstawiono wady i zalety wynikające ze stosowania zasad i metod zorientowanych obiektowo.

Słowa kluczowe: programowanie zorientowane obiektowo, sterowniki programowalne, IEC 61131

1. WPROWADZENIE

Podejście obiektowe w językach programistycznych komputerów osobistych jest obecnie powszechnym stylem programowania, zarówno w przemyśle jak i na wyższych uczelniach technicznych. Najbardziej popularne języki wysokiego poziomu to C++ oraz Java. Języki programowania stosowane w komputerach osobistych nie zawsze były zorientowane obiektowo. Przez wiele lat stosowano języki strukturalne i proceduralne, tj. Cobol, Pascal czy Fortran. Ich funkcjonalność została jednak zakwestionowana z powodu sposobu tworzenia programów, nastawienia na wykonalność projektu poprzez składnie języka zamiast na dokładność odwzorowania, trudności przy modernizacji, brak możliwości ponownego wykorzystania fragmentów programu, duża nieczytelność programu, etc. Rozwijane równolegle do języków strukturalnych i proceduralnych języki zorientowane obiektowo, nieposiadające wymienionych wad, zostały z powodzeniem przyjęte przez programistów oraz inwestorów, i są stosowane do dziś.
W odmienny sposób sytuacja prezentuje się w środowisku programistów sterowników programowalnych PLC. Od wielu lat programowanie to odbywa się w oparciu o kilka języków graficznych i tekstowych o strukturze liniowej.

2. JĘZYKI PROGRAMOWANIA DLA PLC ORAZ NORMY IEC

W procesie kształtowania się graficznych i tekstowych języków programowania sterowników programowalnych ważną role odgrywa międzynarodowa norma IEC 61131-3 z 2003r. Dzięki niej, ujednolicono sposób programowania sterowników za pomocą pięciu języków: IL, ST, LD, FBD oraz SFC. Unormowanie syntaktyki i semantyki języków umożliwiło programistom jednoczesne stosowanie wielu środowisk programistycznych do różnych sterowników programowalnych. Nie wpłynęło to jednak na sytuację, w której programy napisane w różnych środowiskach często nie są ze sobą kompatybilne i uniemożliwiają zastąpienie jednego sterownika innym. Zaletą korzystania z produktów zgodnych z normą IEC jest zmniejszenie kosztów ponoszonych w związku ze szkoleniami z zakresu programowania sterowników różnych producentów. Norma IEC 61131 definiuje wszystkie typy danych i zmiennych stosowane w podejściu proceduralnym, pomija natomiast te, które są charakterystyczne dla podejścia obiektowego. W celu zbliżenia języków tradycyjnych do języków obiektowych określono sposób definiowania bloków funkcyjnych (które można wykorzystywać jako klasy), oraz sposób przydzielania pamięci sterownika a także wydano zalecenia odnośnie utrzymywania niezależności modułów programu.

Kolejnym ważnym krokiem jest wprowadzenie rozwinięcia normy IEC 61131 przez normę IEC 61499 w 2008r. Zwrócono w niej przede wszystkim uwagę na ułatwienia modyfikacji programu oraz na możliwość jego ponownego wykorzystania. Norma IEC 61499 w szczególności zaleca budowanie bloków funkcyjnych, opartych na standardzie IEC 61131 lecz o poszerzonych możliwościach, skierowanych głównie w celu współpracy urządzeń w rozproszonych systemach sterowania. Zaleca się także stosowanie mechanizmów hermetyzacji, tak aby możliwe było tworzenie bloków funkcyjnych dla każdego z elementów układu automatyki z osobna, oraz pisania aplikacji opartych o kontrolę zdarzeń. Istnieje kilka programów narzędziowych, zgodnych z normą IEC 61499 tj. FBDK (oparty o środowisko Java), ISaGRAF 5.0, lub FBench. [1].

Recenzent: Dr inż. Ireneusz Mosoń – Wydział Elektrotechniki i Automatyki
Politechnika Gdańska

3. TWORZENIE PROGRAMU

Ponieważ cykl rozwoju oprogramowania ma charakter ewolucyjny i jest procesem złożonym, najlepszym rozwiązaniem jest podzielenie go na kilka części. W artykule zakłada się, że sam etap programowania powinien być poprzedzony analizą i projektowaniem obiektowym. Jest to najczęściej spotykany, klasyczny, trójstopniowy podział cyklu rozwoju oprogramowania.

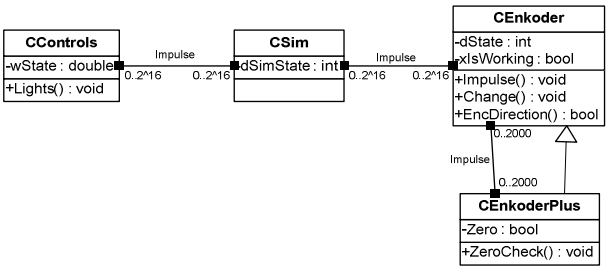
W kolejnych podrozdziałach przedstawione zostały poszczególne etapy tworzenia programu Enkoder, którego zadaniem jest symulacja pracy enkoderów inkrementalnych, możliwość wyboru odpowiedniego typu enkodera oraz prezentowanie stanów i wyników pracy. W tym celu została zaprojektowana i zaimplementowana klasa odpowiadająca działaniu enkoderowi inkrementalnemu oraz podklasy, które uszczegółowiały klasę enkodera. Stworzona w ten sposób hierarchia klas posłużyła do zbadania innych mechanizmów programowania zorientowanego obiektowo tj. dziedziczenie, hermetyzacja lub abstrakcja. Dalsze rozbudowanie hierarchii klas umożliwiło dodatkowo zastosowanie mechanizmu polimorfizmu. Wprowadzanie różnych modyfikacji do projektu np. poprzez zmianę typu enkodera umożliwia sprawdzenie możliwości ponownego wykorzystania fragmentów programu.

W pierwszej części rozdziału został zaprezentowany sposób tworzenia modeli UML (ang. Unified Modeling Language) dla etapu analizy i projektowania, a następnie w dalszej części przeprowadzona została implementacja projektu w oprogramowaniu narzędziowym CoDeSys V3 firmy 3S-Smart Software Solution GmbH. W części dotyczącej programowania omówiono wady i zalety oprogramowania narzędziowego CoDeSys V3 oraz Step 7 firmy Siemens na przykładzie programu Transport, który został stworzony przez autora dla potrzeb systemu transportowego zakładu przemysłowego we Francji. Ze względu na wielkość programu, podczas jego tworzenia pominięto analizę oraz projektowanie, skupiając się na mechanizmach programowania obiektowego. Stworzony program opiera się na rozbudowanej strukturze klas i obsłudze połączeń między nimi.

3.1. Analiza obiektowa

Wszystkie czynności projektowe wykonane zostały w „spiralnym cyklu" [4].

Jednym z zadań analizy obiektowej jest stworzenie odpowiedniego opisu problemu w taki sposób, aby mógł on być podstawą do sporządzenia modelu analitycznego. W tym celu należy zebrać niezbędne informacje odnośnie danej dziedziny i przedstawić je w sposób umożliwiający zrozumienie celów i głównych założeń programu. Dokładny opis analizy obiektowej prezentowanego programu Enkoder można znaleźć w opracowaniu [3].



Rys. 1. Model UML programu Enkoder dla analizy

Model analityczny, będący efektem prac na etapie analizy, przedstawia rzeczywistą strukturę systemu w postaci graficznej. Służą do tego różne rodzaje diagramów.

Tworzenie modelu korzystnie jest podzielić na pięć etapów. Poszczególne etapy zostaną przedstawione w artykule, natomiast szczegółowe informacje można znaleźć w literaturze [3] oraz [4]. W wyniku pracy nad poszczególnymi etapami można zbudować wielowarstwowy model analizy dla programu Enkoder, przedstawiony na rysunku 1.

Na etapie tworzenia warstwy klas i obiektów powstają pierwsze główne klasy przedstawione na rysunku w postaci prostokątów.

Podczas tworzenia warstwy struktur, korzystając przy tym z zasad dziedziczenia, tworzy się strukturę klas, zobrazowaną za pomocą pionowych linii łączących klasy.

Najwyższa warstwa tematów, umożliwiająca wykorzystanie zasady „skali" służy do przedstawienia modelu w ogólnym zarysie.

W warstwie atrybutów analityk rozbudowuje model o atrybuty, opisujące stany obiektów. Konstruując model wykorzystuje się zbudowane wcześniej struktury i umieszcza atrybuty wewnątrz symboli klas. W warstwie tej można także wprowadzić ograniczenia atrybutów, które widoczne są na rysunku w sąsiedztwie powiązań.

Warstwa usług dostarcza natomiast informacji odnośnie funkcji poszczególnych klas. Powiązania pomiędzy obiektami przedstawia się liniami poziomymi.

Przejście z analizy do projektowania może odbywać się zgodnie z zasadą równoczesnego rozwijania [4].

3.2. Projektowanie obiektowe

Zadaniem projektanta stosującego techniki obiektowe jest uszczegółowienie modelu analizy i przygotowanie go do procesu programowania. Główne cele projektowania oraz zasady odwzorowywania rzeczywistości i opanowywania złożoności zostały szczegółowo opisane w literaturze [3] oraz [5]. Należy zwrócić uwagę, że zasady stosowane w analizie mogą być stosowane w projekcie, jeśli pozwalają realizować główne cele projektowania.

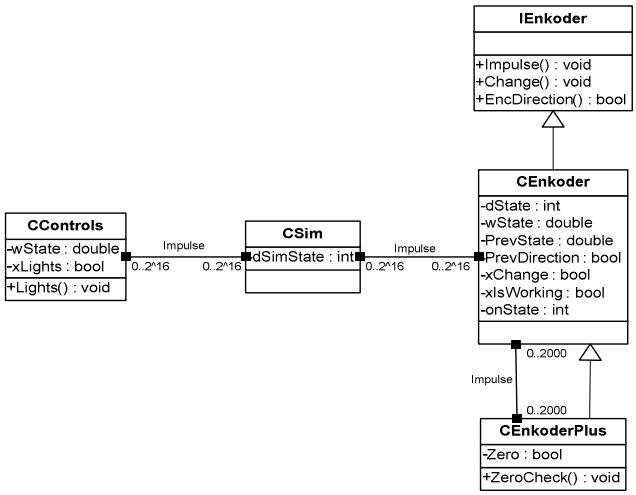
Model analizy obiektowej, poprzez zastosowanie odpowiednich strategii do jego udoskonalania i uszczegóławiania w procesie projektowania, może stać się podstawą do skutecznej implementacji na kod programu. Praca nad projektem dla programu Enkoder została podzielona na: składową dziedziny problemu, składową kontaktu z człowiekiem, składową zarządzania zadaniami oraz składową zarządzania danymi [5].

W pierwszej składowej dziedziny problemu, dokonuje się nowych podziałów klas ze względu na wspólne cechy i wprowadza się nowe składniki w celu ich uszczegółowienia.

Mechanizm dziedziczenia wielokrotnego oraz polimorfizmu został pokazany poprzez wprowadzenie dodatkowej klasy IEnkoder, która stanowi klasę bazową dla klasy CEnkoder.

Model projektu wzbogacony został także o nowe składniki klas, które będą niezbędne do działania programu, np. o zmienną onState (udostępniającą stan wybranego enkodera).

Podczas pracy przy składowej kontaktu z człowiekiem, należy zaprojektować wszystkie elementy programu odpowiedzialne za organizację kontaktu z użytkownikiem. Główny nacisk kładzie się na sposób wprowadzania danych przez użytkownika do programu.

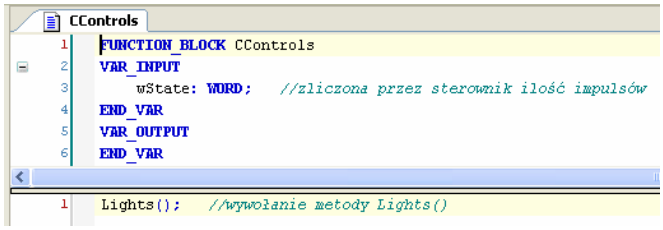


Rys. 2. Model UML programu Enkoder dla projektu

W przypadku programu Enkodera, w celu wyboru typu enkodera, należy posłużyć się zmienną wejściową dType obiektu CEncoder, która może przybierać wartości:1 dla 1000, 2 dla 1500 i 3 dla 2000 obrotów na minutę.
Użytkownik korzystający z programu Enkoder, informowany jest o procesie i jego aktualnym stanie, poprzez zmienne typu BOOL zapisane w tablicy xLights w metodzie Lights.
Składowa zarządzania zadaniami zajmuje się sporządzeniem i przedstawieniem listy i opisu zadań, które należy wykonać w programie. Ze względu na dużą objętość, lista z opisem zdarzeń oraz podział zdarzeń w programie Enkoder znajduje się w opracowaniu [3].
Ostatnia część projektu, składowa zarządzania danymi zajmuje się procesem zapisu i odczytu stanów poszczególnych obiektów programu. Programy sterownikowe wykorzystują w tym celu pamięć wewnętrzną lub zewnętrzną (np. typu Flash).
Uszczegółowiony model projektu dla programu Enkoder przedstawiony jest na rysunku 2.
Do wyników projektowania obiektowego należy także dołączyć listę lub tabelę ze wszystkimi klasami i atrybutami, w całości pokazaną w [3].

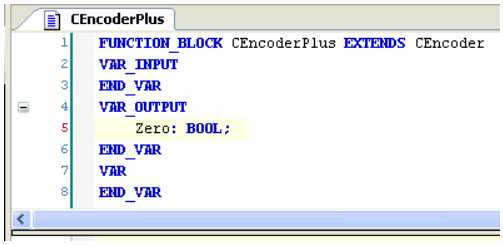
3.3. Programowanie obiektowe

Na podstawie wyników otrzymanych w procesie analizy i projektowania obiektowego, można dokonać odwzorowania modeli na działający program. Wymaga to użycia konkretnego języka programowania obiektowego wraz z odpowiednimi narzędziami wspomagającymi, takimi jak: kompilator czy edytor kodu.
W celu zaimplementowania kodu programu Enkoder wykorzystano oprogramowanie narzędziowe CoDeSys v3, zgodne z normą IEC 61131, natomiast do programu Transport, w celu zbadania normy IEC 61499, posłużono się oprogramowaniem narzędziowym Step 7.
Program Enkoder został napisany w sposób tekstowy, wykorzystując w tym celu składnię języka ST (pełen kod programu przedstawiony jest w literaturze [3]). Drugi program - Transport, napisany został w języku LD, i z powodzeniem uruchomiony na sterownikach firmy Siemens w przemyśle.
W większości programów narzędziowych do sterowników programowalnych (w tym także Step 7) klasa jest reprezentowana przez blok funkcyjny (FB), który jest częścią programu głównego.



Rys. 3. Okno definiowania zmiennych i ciało klasy CControls w CoDeSys V3

Enkoder, napisany w CoDeSys V3 różni się jednak od programu napisanego w tradycyjnym oprogramowaniu narzędziowym, przede wszystkim konstrukcją programu. Składają się na to klasy (FB – stanowiące oddzielne interfejsy) ułożone w hierarchie, które dziedziczą elementy składowe z klas nadrzędnych z uwzględnieniem zasad hermetyzacji. Interfejs klasy CControls w programie CoDeSys V3 przedstawiony jest na rysunku 3.



Rys. 4. Dziedziczenie z klasy CEnkoder w CoDeSys V3

Wywołanie mechanizmu dziedziczenia pokazane jest z kolei na przykładzie klasy CEnkoderPlus, która dziedziczy z klasy CEnkoder - rysunek 4. Dodatkowo klasa zostaje uszczegółowiona o zmienną Zero typu BOOL.
Budowanie hierarchii z zasadami dziedziczenia w CoDeSys v3 odbywa się za pomocą słowa kluczowego EXTENDS.[3] Na rysunku 2 można zauważyć, że klasa CEnkoder dziedziczy także z klasy IEnkoder, wobec czego dochodzi tutaj do dziedziczenia wielokrotnego.

4. OPROGRAMOWANIE NARZĘDZIOWE PLC

Oprogramowanie narzędziowe CoDeSys v3 umożliwia tworzenie programów, korzystając przy tym z podstawowych zasad i metod podejścia obiektowego, tj. dziedziczenie (także wielokrotne), hierarchiczność, polimorfizm, abstrakcja, hermetyzacja. Podstawowym ograniczeniem jest natomiast brak determinacji firmy 3S-Smart Software Solution w dążeniu do jego upowszechnienia i wprowadzenia do konkretnych jednostek sprzętowych, przez co brak jest w dalszym ciągu zastosowań przemysłowych.
Programy napisane w oprogramowaniu narzędziowym Step 7, znalazły już zastosowanie w wielu działających systemach automatyki przemysłowej. Główne zalety Step 7 to możliwość tworzenia obiektów w postaci FB, obsługa połączeń obiektów, okrojona hermetyzacja czy wspieranie rozszerzenia normy IEC 61499 w zakresie dzielenia programu na niezależnie działające części. Ograniczeniami pozostaje brak możliwości stosowania dziedziczenia, hierarchiczności czy polimorfizmu, co znacząco wpływa na jakość tworzonych programów.

5. PODSUMOWANIE

Korzystanie z udogodnień programowania zorientowanego obiektowo zwiększa czytelność tworzonego programu, bowiem podczas pracy przy projekcie poświęca się więcej czasu na składową kontaktu z człowiekiem. Elementy programu, na które zostaje podzielony model na etapie analizy i projektowania obiektowego, a także budowa programu oparta na obiektach i klasach, które wykorzystywane w podobny sposób do człowieka odwzorowują rzeczywistość, pozwalają lepiej zrozumieć działanie systemu, co bardzo często decyduje o możliwościach ponownego wykorzystania fragmentów programu. Z tego samego powodu wprowadzanie modyfikacji do istniejącego już programu nie jest kłopotliwe, a odnajdywanie błędów zdecydowanie ułatwione. Powyższe zalety stają się jeszcze bardziej istotne podczas adaptacji istniejących programów do nowych warunków pracy lub podczas usprawniania pisania rozbudowanych i działających systemów.

W celu dokonania oceny środków wspomagających programistę w procesie pisania programu, należy posłużyć się odpowiednimi kryteriami, zależnymi także od potrzeb aktualnego projektu. Najczęściej pojawiające się punkty to: możliwość definiowania klas i obiektów, możliwość tworzenia struktur przez dziedziczenie, obsługa połączeń obiektów, hermetyzacja.

W projektach o stosunkowo małej złożoności, stosowanie podejścia obiektowego może nie przynieść współmiernych zysków do kosztów poniesionych przy jego tworzeniu, Program tworzony w sposób liniowy, trudno poddaje się modyfikacjom przez co wykorzystywanie jego fragmentów jest mocno ograniczone, a więc zysk jest jednorazowy.

W przypadku projektów o większej złożoności, dysponując techniką obiektową, można dokonywać rozkładu procesu tworzenia programu na mniejsze fragmenty, kontrolując rozwój nawet bardzo rozbudowanego systemu.

Podejście obiektowe jest zalecane wszędzie tam gdzie istnieje potrzeba późniejszego wykorzystywania fragmentów programu, ograniczenia błędów, dokonywania planowanych lub nieplanowanych modyfikacji, równoczesnej pracy wielu programistów i analityków czy też zamierzonego ograniczenia praw dostępu do fragmentów programu. W wielu przypadkach podejście takie jest praktycznie niezbędne.

Zalety podejścia obiektowego do projektowania i programowania stają się jeszcze bardziej widoczne, kiedy spojrzymy na cykl budowy programu całościowo, od momentu kiedy program powstaje i zostaje zainstalowany w systemie, poprzez jego modyfikację, serwis, aż po zamianę na nowy.

6. BIBLIOGRAFIA

1. Chouinard J., Brennan R.: Software for next generation automation and control, IEEE International Conference on Industrial Informatics, 16-18 Aug. 2006, pp. 886-891, ISBN: 0-7803-9700-2.
2. Sunder C., Wenger M., Hanni C., Gosetti I., Steininger H., Fritsche J.: Transformation of existing IEC 61131-3, automation projects into control logic according to IEC 61499, IEEE International Conference on Emerging Technologies and Factory Automation, 15-18 Sept. 2008, pp. 369-376, ISBN: 1-4144-1506-3.
3. Trojanowski P., Zieliński D.: Programowanie obiektowe w oprogramowaniach narzędziowych do sterowników programowalnych, Praca dyplomowa, Politechnika Gdańska, 2008
4. Coad P., Yourdon E.: Analiza obiektowa, Oficyna wydawnicza READ ME, Warszawa 1994, ISBN: 83-85769-17-1.
5. Coad P., Yourdon E.: Projektowanie obiektowe, Oficyna wydawnicza READ ME, Warszawa 1994, ISBN: 83-85769-16-1.
6. Dai W.W., Vyatkin V.: A case study on migration from IEC 61131 PLC to IEC 61499 function block control, IEEE International Conference on Industrial Informatics, 23-26 June 2009, pp. 79-84.
7. Pna B., Seisdedos V., Loperana V.: Including object-oriented properties in the PLC's programming languages, IEEE International Conference on Emerging Technologies and Factory Automation, 18-21 Oct. 1999, pp. 1029-1034, ISBN: 0-7803-5670-5.

THE USE OF OBJECT-ORIENTED METHODS FOR PROGRAMABLE LOGIC CONTROLLERS PROGRAMMING

Key-words: Object-oriented programming, Programmable logic controllers (PLC), IEC 61131-3

This paper shows main conveniences of using object-oriented principles and methods for PLC's programming. Actual situation of text and graphic languages after International Electrotechnical Commission introduction IEC 61131 and its extend IEC 61499 are discussed. Next, division of the cycle of creating program into three fundamental parts, analysis, design and programming are presented. At the end, advantages and disadvantages of using object-oriented principles and methods are shown.