Dr hab. inż. prof. Jan Werewka
Katedra Informatyki Stosowanej, Wydział EAIiIB
Akademia Górniczo-Hutnicza im. St. Staszica
31-059 Kraków, Al. Mickiewicza 30.
Email: werewka@agh.edu.pl
WWW: home.agh.edu.pl/werewka

Recenzja rozprawy doktorskiej mgr inż. Marcina Jamro pt.: „Metodyka modelowania, testowania i implementacji oprogramowania sterującego przy użyciu diagramów SysML oraz testów jednostkowych"

1. Ogólna charakterystyka wyboru tematu rozprawy

Temat recenzowanej rozprawy dotyczy opracowania metodyki modelowania, testowania i implementacji oprogramowania sterującego. Metodyka ta ma być wspomagana poprzez użycie języka SysML oraz testów jednostkowych.

Analizując tytuł rozprawy można się spodziewać zakresu obejmującego zagadnienia:

- opracowania metodyki, służącej do rozwoju oprogramowania.
- bazowanie na czynnościach rozwoju oprogramowania ograniczonych głównie do czynności modelowania, testowania i implementacji.
- koncentracji się na klasie oprogramowania, która w rozprawie nazywana jest oprogramowaniem sterującym
- modelowaniu systemów przy użyciu języka SysML (System Modeling Language)
- ukierunkowanie testów na testy jednostkowe.

Przy definiowaniu celów, autor rozprawy zaznacza, że zajmuje się budowaniem systemów sterowania w oparciu o normę IEC 61131-3. Norma ta dotyczy sterowników programowalnych PLC (Programmable Logic Controller), rozproszonych systemy sterowania DCS (Distributed Control System). Zatem rozpatrywany w pracy obszar, dotyczy informatyki przemysłowej, systemów wbudowanych, rozproszonych systemów sterowania. Projektowanie systemów informatyki przemysłowej ma duże znaczenie badawcze i praktyczne. Budowa takich systemów jest procesem złożonym i będzie bardziej efektywna, jeżeli część prac wykonywanych ręcznie będzie zautomatyzowana. Wykorzystanie półformalnego języka SysML jest uzasadnione, gdyż może on zostać wykorzystany do budowy rzeczywistych i złożonych systemów. Metody formalne co prawda dają formalny dowód poprawności systemów, jednak ze względu na możliwość modelowania małych systemów nie posiadają praktycznego znaczenia. Podsumowując tematyka rozprawy jest aktualna i obejmuje obszar, który ma duże znaczenie praktyczne, gdyż dotyczy on ważnej klasy systemów informatyki przemysłowej. Prowadzone prace badawcze z jednej strony polegają na wykorzystaniu nowych narzędzi i technologii, oraz na dobór rozwiązań związanych z produktem i metodyk rozwoju tych systemów.

2. Charakterystyka przedmiotu rozprawy

Ideą przewodnią pracy jest opracowanie metodyki modelowania, testowania i implementacji oprogramowania sterującego tworzonego zgodnie z normą IEC 61131-3 z 2013 roku. Zaproponowano tworzenie oprogramowania opartego o dwa paradygmaty MDD (Model-Driven Development) w obrębie modelowania i TDD (Test Driven Development) w obrębie testowania.

Teza pracy polega na stwierdzeniu, że możliwe jest utworzenie metodyki modelowania, testowania i implementacji oprogramowania sterującego łączącej wybrane zasady paradygmatów tworzenia oprogramowania bazującego na modelu oraz testach.

Potwierdzeniem słuszności tezy głównej (teza 1) ma być zaproponowanie całościowej metodyki modelowania, testowania i implementacji oprogramowania sterującego. Istotne jest również sprawdzenie jej na przykładzie, w celu upewnienia się co do efektywności działań dotyczących rozwoju systemu sterowania w ramach metodyki.

Praca posiada trzy tezy poboczne, których potwierdzenie będzie wynikać z odpowiedniego zaprojektowania poszczególnych faz metodyki.

Teza 2. Możliwe jest wykorzystanie diagramów graficznego języka modelowania SysML podczas projektowania poszczególnych elementów systemu sterowania.

Tezę 2 autor rozprawy zamierza potwierdzić przez dostępność reguł modelowania określających sposób przedstawienia elementów systemu sterowania na diagramach SysML.

Tezę tę można rozumieć, że język SysML można adaptować do potrzeb modelowania rozpatrywanej w rozprawie klasy systemów. Zagadnienie to może być zrealizowane poprzez budowę rozszerzeń SysML (np. typu De Oliveira, K.S., Franca, J.M.S.; Soares, M.S: Extensions of SysML for Modeling an Aspect Oriented Software Architecture with Multiple Views.) lub poprzez wykazanie, że pełny model dla wybranej klasy systemów może być zamodelowany bezpośrednio w SysML. W rozprawie teza jest wykazywana poprzez zdefiniowanie podzbioru reguł określających sposób projektowania różnych elementów rozproszonych systemów sterowania,

Teza 3. Możliwe jest zaproponowanie różnych metod testowania oprogramowania sterującego, w tym korzystając z dedykowanego języka definiowania testów, testowania tabelowego, wizualnego i wydajnościowego.

Zdaniem autora rozprawy tą tezę powinna potwierdzić odpowiednia specyfikacja lub opis metod testujących, a także zaproponowanie dedykowanego języka wykorzystywanego do tworzenia testów.

Teza 4. Możliwe jest przyspieszenie i ułatwienie procesu wytwarzania oprogramowania poprzez automatyczne generowanie implementacji na podstawie modelu. Podejście to może zostać rozwinięte przez zastosowanie koncepcji inżynierii wahadłowej.

Tezę 4 autor rozprawy chce potwierdzić poprzez opracowanie algorytmów związanych z generowaniem implementacji na podstawie modelu, jak również przez rozszerzenie tego podejścia do inżynierii wahadłowej w celu propagacji zmian pomiędzy różnymi częściami projektu, tj. modelem, testami i implementacją.

Podsumowując na przedmiot rozprawy składa się wiele nurtów badawczych o charakterze teoretycznym i praktycznym. Autor postawił przed sobą ambitne zadanie, gdyż zakres prowadzonych prac jest bardzo obszerny i wymaga odpowiedniego przygotowania dotyczącego zarówno metod, modeli jak i narzędzi.

3. Ocena konstrukcji rozprawy

Praca doktorska zawiera 189 stron i składa się z 6 rozdziałów zawierających wprowadzenie i podsumowanie. Na końcu pracy znajdują się dodatki, bibliografia, oraz spisy rysunków i tablic.

W pierwszym rozdziale są omówiono cele i tezę pracy, problemy badawcze i techniczne, zawartość pracy, informacje redakcyjne.

W rozdziale drugim przedstawiono koncepcję metodyki. Główne założenia metodyki obejmują zagadnienia:

1) Wsparcie dla normy IEC 61131 (zadana struktura oprogramowania)
2) Bazowanie na modelu. Zastosowano podejście MDD (Model-Driven Development), a budowa modelu tworzona jest przy wykorzystaniu języka SysML.
3) Oparcie się na testach definiowanych przed implementacją oprogramowania (TDD - Test-Driven Development).
4) Wykorzystania komponentów (komponenty są niezależne i prostsze do ponownego użycia). *Uwaga recenzenta: Powstaje pytanie co jest komponentem w rozprawie. Zwykle się uważa, że komponenty to wykonywalne elementy oprogramowania.*
5) Zwinność. Ułatwienie rozwoju systemu sterowania realizowane jest przez wsparcie iteracyjnego procesu tworzenia oprogramowania. *Uwaga recenzenta: Raczej zastosowałbym słowo elastyczność, modyfikowalność. Zwinność oznacza zdolność do reakcji na zmiany, co oznaczałoby, że oprogramowanie powinno być samo adaptacyjne.*
6) Ponowne użycie zarówno dla fragmentów modelu, jak i implementacji. Metodyka umożliwia eksportowanie jednostek organizacyjnych oprogramowania w postaci bibliotek i używanie ich w innych projektach. *Uwaga recenzenta: Drugie zdanie dotyczy bardziej interopercyjności niż ponownego użycia.*
7) Automatyczne generowanie implementacji na podstawie danych zgromadzonych w modelu.
8) Wykorzystanie już dostępnego kodu poprzez zastosowanie języków normy IEC 61131-3.
9) Wsparcie dla inżynierii wahadłowej, skutkujące tym, że zmiany w poszczególnych częściach są wykrywane automatycznie, a następnie propagowane do odpowiednich fragmentów projektu, czyli modelu, implementacji lub testów.

Następnie omówiono trzy zasadnicze etapy metodyki, czyli modelowanie, testowanie i implementacje. Rozdział drugi kończy opis przykładu rozwijanego w następnych rozdziałach.

W rozdziale trzecim przedstawiono sposób modelowania systemów sterowania. Podejście to korzysta z dedykowanego języka domenowego opartego na języku SysML, zaś tworzone modele posiadają odniesienia do wybranych zagadnień normy IEC 61131-3. Proponowana koncepcja pozwala na tworzenie oprogramowania w oparciu o model MDD (Model-Driven Development), co umożliwia zwiększenie poziomu abstrakcji, ułatwienie wprowadzania zmian, uproszczenie zrozumienia budowy i funkcjonowania systemu, jak również automatyczne generowanie fragmentów implementacji.

W rozdziale czwartym przedstawiono kompleksowe podejście do testowania systemów sterowania ukierunkowane na rozwiązania tworzone w oparciu o normę IEC 61131-3. Wspiera ono zarówno testowanie jednostkowe, jak i integracyjne oraz systemowe.

W pracy podawanych jest szereg metod testowania TDD (Test driven development), testowanie bazujące na modelu MBT (Model-Based Testing), modelowanie oparte na testach

TDM (Test-Driven Modeling), testowanie jednostkowe oparte na modelu MDUT (Model-Driven Unit Testing) i stanowi rozszerzenie monitorowania opartego na modelu MDM (Model-Driven Monitoring) wykorzystującego warunki początkowe i końcowe do określania pożądanego zachowania.

Istotne jest również powiązanie testów z modelowaniem w języku SysML, gdyż pozwala to na zarządzanie testami i ich zestawami bezpośrednio na diagramach graficznych, jak również generowanie szablonów testów na podstawie modelu. Powoduje to, ze proponowane podejście łączy cechy tworzenia oprogramowania bazującego na testach TDD (Test-Driven Development) oraz testowania opartego na modelu MBT (Model-Based Testing).

Zaprezentowano kilka sposobów tworzenia testów, w tym testy w postaci tabelowej, wizualnej oraz w dedykowanym języku definiowania testów CPTest+. Oprócz sprawdzania wymagań funkcjonalnych, metodyka wspiera weryfikacje wymagań poza funkcjonalnych, ze szczególnym uwzględnieniem wydajności systemu. Do tego celu wykorzystano testy wydajności jednostek POU oraz komunikacji pomiędzy urządzeniami w systemie rozproszonym. *Uwaga recenzenta: W pracy brak jest informacji o powiązaniu stosowanych sposobów testowania z metodami testowania.*

W rozdziale piątym przedstawiono cześć metodyki dotyczącą implementacji systemów sterowania, która realizowana jest za pomocą języków normy IEC 61131-3. Istotnym założeniem jest wykorzystanie wcześniej opracowanego modelu w języku SysML do automatycznego wygenerowania fragmentów implementacji, w tym szablonów jednostek organizacyjnych oprogramowania oraz pełnej implementacji jednostek funkcjonujących w oparciu o maszyny stanowe. Metodyka pozwala na automatyczne generowanie różnych części implementacji, w tym szablonów jednostek POU, implementacji dla jednostek funkcjonujących w oparciu o maszyny stanowe, szablonów ekranów i programów interfejsu operatorskiego, a także szablonów i implementacji testów.

Mechanizm generowania kodu, diagramów i ekranów bazuje na sekwencyjnym wykonywaniu uporządkowanej listy operacji. Sposób wprowadzania modyfikacji określony jest z kolei przez reguły synchronizacji powiązane z operacjami. W ramach metodyki zaproponowano zestaw algorytmów używanych do wyszukania list operacji na podstawie analizy diagramów w języku SysML.

W rozdziale zaprezentowano również rozszerzenie podejścia generowania implementacji do idei inżynierii wahadłowej RTE (Round-Trip Engineering). Pozwala ona na wykrywanie zmian wprowadzanych w modelu, testach lub implementacji i ich propagacje do odpowiednich części projektu wraz z zachowaniem wcześniej wprowadzonych modyfikacji.

Rozdział szósty zawiera podsumowanie pracy, obejmujące opis najbardziej istotnych rezultatów pracy, podano wnioski końcowe oraz przedstawiono perspektywy dalszych badań.

Podsumowując ocenę konstrukcji rozprawy, można stwierdzić, że struktura pracy jest poprawna. Praca napisana jest w sposób zwięzły i z dużą dbałością edytorską. Autor przedstawił wyniki pracy w sposób spójny. Przy pisaniu pracy dokonano analizy wielu rozwiązań podawanych w literaturze światowej.

4. Ocena wartości merytorycznej rozprawy

Prezentowana praca doktorska realizuje cel polegający na zaproponowaniu metodyki modelowania, testowania i implementacji oprogramowania sterującego tworzonego zgodnie z normą IEC 61131-3.

W pracy udowadniana jest teza, mówiąca, iż możliwe jest utworzenie metodyki modelowania, testowania i implementacji oprogramowania sterującego łączącej wybrane zasady paradygmatów tworzenia oprogramowania bazującego na modelu oraz na testach.

Tezę pracy można uznać za udowodnioną na podstawie przygotowanej metodyki modelowania, testowania i implementacji oprogramowania sterującego, zgodnego z normą IEC 61131-3 (IEC, 2013).

W wyniku realizacji celów autor rozprawy osiągnął wiele interesujących oryginalnych rezultatów, na które składają się:

1) Sformułowanie koncepcji metodyki modelowania, testowania i implementacji oprogramowania sterującego.
2) Opracowanie przykładu prezentującego możliwości metodyki.
3) Zastosowanie języka SysML dla modelowania rozpatrywanej klasy systemów.
4) Przedstawienie zbioru reguł modelowania poszczególnych elementów systemu.
5) Zaproponowanie sposobów testowania jednostek organizacyjnych oprogramowania.
6) Przygotowanie mechanizmów testowania wydajności systemu sterowania.
7) Przedstawienie sposobu testowania ekranów interfejsu operatorskiego.
8) Opracowanie mechanizmów automatycznego generowania testów, implementacji i interfejsu operatorskiego na podstawie modelu.
9) Sformułowanie koncepcji inżynierii wahadłowej dla systemów sterowania.
10) Opracowanie dodatkowych narzędzi wspierających metodykę. Środowisko uruchomieniowe CPDev wyposażone zostało w dedykowana aplikacje do modelowania, platformę testująca oraz wieloplatformowy mechanizm interfejsu operatorskiego.

Tematyka zawarta w rozprawie jest obszerna i przy jej czytaniu brakuje spojrzenia na rozpatrywane zagadnienia w sposób bardziej ogólny. Dotyczy to w szczególności trzech kwestii:

Przy rozwoju oprogramowania w rozprawie ograniczono się do modelowania, testowania i implementacji. W tradycyjnym rozwoju oprogramowania mamy do czynienia z czynnościami: analizy wymagań systemowych, projektowaniem systemu, analizy wymagań wobec oprogramowania, projektowanie oprogramowania, implementacji, testowania, wdrożenia. Zatem autor przy rozwoju oprogramowania wybrał tylko modelowanie, testowanie i implementację. Powstaje pytanie jak można zagwarantować spójność metodyki wybierając tylko niektóre czynności z rozwoju oprogramowania.

W rozprawie doktorskiej występuję brak szczegółowego odniesienia się do klasy systemów do których tworzone jest oprogramowanie sterujące. Dotyczy to w szczególności takich systemów, których nazwy pojawiają się w pracy doktorskiej, systemy czasu rzeczywistego (str. 9) systemy sterowania (str. 9), rozproszone systemy sterowania (str. 10), systemy bazującej na normie IEC 61131-3 (sterowniki programowalne PLC (Programmable Logic Controller, rozproszone systemy sterowania DCS (Distributed Control System). Analizując treść rozprawy można odnieść wrażenie, że tą klasę systemów rozważanych w pracy autor nazywa systemami sterowania. Z drugiej strony autor mógłby zastosować podejście, że interesuje go rodzina oprogramowania odpowiadająca normie IEC 61131 i tę rodziną oprogramowania autor nazywa oprogramowaniem sterującym. Pozostaje jednak zagadnienie relacji oprogramowania i sprzętu, na którym to oprogramowanie jest wykonywane. Powstaje pytanie, czy pojęcie oprogramowania sterującego odnosi się w pracy jako oprogramowanie systemów sterowania, czy jest to część oprogramowania systemów sterowania związana z wyznaczeniem sterowania, natomiast część dotycząca monitoringu nie jest brana pod uwagę. Podsumowując, uważam, że autor pracy powinien podać definicję, że

oprogramowanie sterujące oznacza oprogramowanie rozproszonych systemów sterowania, których struktura jest zgodna z normą IEC 61131-3. Strukturę systemu sterowania powinien przedstawić w postaci prostej abstrakcji. Powstaje pytanie, dlaczego nie zbudowano pewnej abstrakcji nad normą IEC 61131-3 dla potrzeb rozpatrywanej klasy systemów. Powoływanie się na normę jest uciążliwe.

W pracy brakuje podejścia architektonicznego do klasy budowanych. Podejście to zawierałoby model referencyjny klasy systemów, widoki oprogramowania: modułowy, komponentowy i alokacyjny, punkty widzenia interesariuszy, scenariusze, atrybuty jakości rozpatrywane, stosowane taktyki dla wybranych atrybutów jakości, decyzje architektoniczne i projekt architektury dla klasy systemów. Koncepcja inżynierii wahadłowej może też odnosić się do atrybutów architektury (np. Rick Kazman, Security Pattern Assurance through Round-trip Engineering).

Podsumowując stwierdzam, że wiele części rozprawy można uznać za wyróżniające, dodatkowo do zalet zaproponowanego w pracy rozwiązań można zaliczyć ich praktyczne zastosowanie. Należy dodać, że dorobek naukowy doktoranta jest znaczący i składa się z 26 publikacji z czego większość napisana jest w języku angielskim oraz jest dostępna w obiegu międzynarodowym. Zatem wkład doktoranta w zakresie szerokiego udostępniania swego dorobku jest istotny. Znaczenie naukowe uzyskanych wyników jest istotne z punktu widzenia projektowania systemów informatyki przemysłowej.

5. Uwagi dyskusyjne

Jednym z głównych problemów przy czytaniu pracy jest podawanie różnorodnych pojęć bez ich definiowania i odwzorowania na zawartość pracy.

W pracy stosowane są paradygmaty tworzenia oprogramowania. Przy analizie zawartości rozprawy można wyłuskać następujące paradygmaty:

- MDD (Model-Driven Development)
- TDD (Test-Driven Development), TFD (Test-First Development, MBT (Model-Based Testing
- Programowanie obiektowe (norma IEC 61131-3 z 2013 r. pozwala na tworzenie systemów zgodnie z paradygmatem programowania obiektowego).

Ważnym zagadnieniem jest zdefiniowanie co oznaczają paradygmaty (wzorce) rozwoju oprogramowania w pracy i dlaczego te zostały wyróżnione.

Zagadnieniem wartym przedyskutowania jest podejście bazujące na modelu MDD (Model-Driven Development). Zakłada ono, ze projektant tworzy model systemu, na podstawie którego możliwe jest automatyczne wygenerowanie całości lub części implementacji. Z kolei podejście dotyczące rozwoju architektury bazującej na modelu określane jest z kolei jako MDA (Model Driven Architecture). Podejście to posiada trzy podstawowe cele – rozdzielenie logiki biznesowej i aplikacji od platformy docelowej, skupienie uwagi programistów na tworzeniu modelu aplikacji oraz logiki biznesowej. Podejście MDA może zostać rozszerzone na cały proces projektowania systemu, co określane jest jako inżynieria bazująca na modelu MDE (Model-Driven Engineering). W ramach pracy zaproponowano metodykę modelowania, która zgodna jest z podejściem MDD, a także posiada cechy wspólne z MDA oraz MDE. W przedstawionych rozwiązaniach brak jest wydzielenia trzech podstawowych widoków architektury oprogramowania: modułów – strona implementacyjna, komponentów i konektorów (Components & Connectors) – strona wykonawcza, alokacji wdrożeniowej i instalacyjnej.

W rozprawie nie podano uzasadnienia, dlaczego przy przeprowadzaniu testów skoncentrowano się na testach jednostkowych. Zwykle testowanie obejmuje szerszy obszar: testowanie jednostkowe, integracyjne, systemowe, akceptacyjne. Z tego z kolei wynika, że autor mniej interesuje się testowaniem całych systemów.

W pracy napisano „Podejście to korzysta z dedykowanego języka domenowego opartego na języku SysML". Autor posługuje się chętnie słowem dziedzinowy (domenowy) nie wspominając o jaką dziedzinę chodzi. Język dziedzinowy [Wikipedia], także język dedykowany, język specjalizowany (ang. domain-specific language, DSL) to język programowania przystosowany do rozwiązywania określonej dziedziny problemów, określonej reprezentacji problemu lub określonej techniki ich rozwiązywania. Język SysML służy do opisu architektury systemów, podobnie jak inne języki UML, AADL, ArchiMate. Z drugiej strony SysML ukierunkowany jest na inżynierię systemów wspierając specyfikację, analizę, projektowanie, weryfikację i walidację szerokiej klasy systemów. Zastanawia fakt jakie jest podejście SysML do widoków i perspektyw (viewpoints). Klasyczne widoki to modułowy, komponentów i alokacji. Perspektywy dotyczą zazwyczaj punktów widzenia interesariuszy.

Wybrane uwagi szczegółowe:

W tezie 2, opuściłbym słowo diagramów w zdaniu „Możliwe jest wykorzystanie diagramów graficznego języka modelowania SysML podczas projektowania poszczególnych elementów systemu sterowania".

W tezie 3 zamiast słowa zaproponowanie powinno być użyte słowo zastosowanie.

W pracy występują stwierdzenia „Wykorzystuje ono język domenowy oparty na dedykowanym profilu języka SysML". Raczej powinno być Wykorzystuje ono język domenowy SysML oparty na dedykowanym profilu języka UML".

Autor rozprawy nie wyjaśnia swojego wkładu w rozwój języka CPTest+ i środowiska CPdev.

6. Wniosek końcowy

Przedstawiona do recenzji praca stanowi oryginalny i istotny wkład w dziedzinie rozwoju metodyki dotyczącej modelowania, testowania i implementacji oprogramowania systemów sterowania wybranej klasy. W pracy zaproponowano dojrzałą koncepcję metodyki rozwoju oprogramowania rozproszonych systemów sterowania.

W celu wykazania tezy pracy autor opracowania musiał wykonać szereg prac mających istotne znaczenie badawcze. Praktyczne zastosowanie metodyki wykazano na przykładowym systemie z wykorzystaniem środowiska CPDev.

Uwagi zawarte w recenzji mają charakter dyskusyjny i nie umniejszają wysokiej oceny merytorycznej.

W związku z powyższym stwierdzam, że recenzowana praca spełnia wymagania stawiane rozprawom doktorskim określone w Ustawie i wnoszę o dopuszczenie mgr inż. Marcina Jamro do dalszych etapów związanych z uzyskaniem stopnia naukowego doktora, w tym do dopuszczenia rozprawy doktorskiej do publicznej obrony.

Kraków 29.12.2014

Jan Werewka