Projektowanie mechatroniczne. Graficzna specyfikacja systemów

Krzysztof Pietrusewicz, Michael Scopchanov

Abstract: *Nowadays, when more and more complexity is introduced into control systems in many areas, old-fashioned document-based development is a „one-way ticket to fail".*

Model-based development is the only possible direction for future work on control systems in automotive, process industry, machinery, and many others. Graphical specification is the baseline for any model-based development actions to be taken.

The paper explains usage of four different tools and approaches for modeling: requirements, architecture and implementation of control system designs.

Also automatic code and documentation generation issue is pointed out as the most important factor for creation of workflows with the use of discussed tools.

This is the third article from series „Mechatronic Design".

Niniejszy artykuł w całości poświęcono zagadnieniom graficznej specyfikacji systemów sterowania zgodnie z założeniami tzw. inżynierii systemów. Współczesne wymagania związane z bezpieczeństwem funkcjonalnym maszyn sprawiają, iż zagadnienia jakości tworzenia oprogramowania systemów sterowania wkraczają do coraz większej ilości branż.

Od dawna wiadomo, że graficzna reprezentacja systemów jest dużo lepsza i efektywniejsza od najprecyzyjniejszego nawet opisu słownego. Artykuł, na przykładzie zagadnienia graficznej specyfikacji systemów sterowania, utwierdzi Czytelników w tym przekonaniu jeszcze bardziej.

1. Założenia inżynierii systemów

Projektowanie mechatroniczne, czy inaczej projektowanie systemów bazujące na modelach, którego założenia przedstawiono w artykule [1], jest jedną z technik wspierających inżynierię systemów. Czym zatem jest inżynieria systemów?

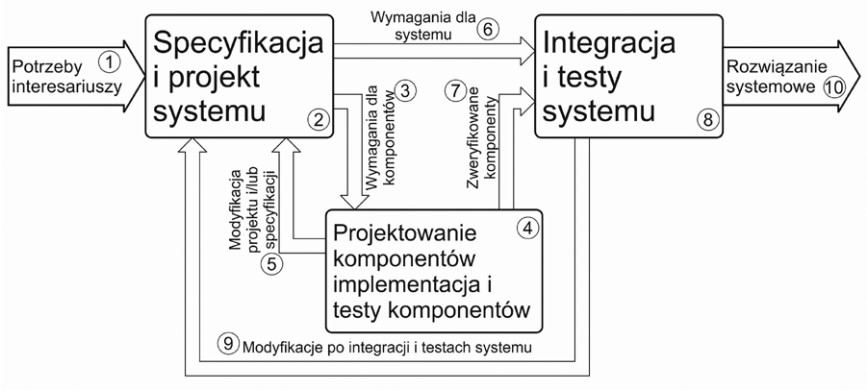
Wyczerpującą definicję zamieszczono w [2] oraz [3]: „Inżynieria systemów to wielodyscyplinarne podejście do przekształcania potrzeb i wymagań interesariuszy w rozwiązania systemowe, zaspokajające postawione wymagania i potrzeby".

Aby możliwe stało się opracowanie rozwiązania systemu, konieczne jest opracowanie specyfikacji, na podstawie której system zostanie zrealizowany. Specyfikacja odzwierciedla wymagania stawiane projektowanemu rozwiązaniu. Specyfikacja może zostać przygotowana na wiele sposobów. Jednym z najgorszych możliwych jest z pewnością opis słowny, choć z drugiej strony na początkowym etapie projektu jest zwykle jedynym dostępnym. Stanowi kombinację wyobrażeń i oczekiwań klienta nt. produktu, który ma stanowić rezultat projektu. Opis słowny wspierany jest na tym etapie schematami, rysunkami, drobnymi obliczeniami wstępnymi.

Niniejszy artykuł poświęcony jest innym metodom i sposobom opracowywania specyfikacji. Zanim przedstawione zostaną najpopularniejsze współcześnie sposoby specyfikowania systemów, najpierw należy zapoznać się z przebiegiem pracy nad projektami, zdefiniowanym w ramach inżynierii systemów. Taki uproszczony przebieg pracy zamieszczono na rysunku 1.

Z rysunku 1 wynika następujący sposób realizacji projektów. Potrzeby interesariuszy (1) po analizach i uporządkowaniu składają się na specyfikację oraz projekt systemu (2), który zdaniem interesariuszy stanowić będzie potencjalne rozwiązanie (10). Na bazie specyfikacji tworzy się szczegółowe wymagania dla komponentów projektowanego systemu (3). Po zakończeniu projektowania poszczególnych komponentów, ich implementacji oraz weryfikacji zgodności ze specyfikacją (4) możliwa jest konieczność modyfikacji projektu i/lub specyfikacji wybranego komponentu (5). Gdy wszystkie lub większość planowanych do zastosowania komponentów zostanie ukończona, można przystąpić do budowy systemu na podstawie postawionych wymagań (6), z zastosowaniem jedynie zweryfikowanych uprzednio komponentów (7). Integracja systemu w całość (8) może zakończyć się sukcesem bądź skutkować modyfikacjami specyfikacji (9). Gdy zintegrowany system wypełnia stawiane rozwiązaniu wymagania, można mówić o zakończeniu projektu i opracowaniu tzw. rozwiązania systemowego (10). Dużą wartością inżynierii systemów jest fakt, iż diagram z rysunku 1 daje się wprost zastosować do większości współczesnych zagadnień projektowania systemów mechatronicznych.

W przypadku systemów o niskiej złożoności, jak również unikalnych, niepodlegających wersjonowaniu zależnie od oczekiwań różnych użytkowników, specyfikacja opracowana w sposób słowny może zostać w prosty sposób zmodyfikowana. W przypadku systemów złożonych może się jednak zdarzyć, że aktualizacja specyfikacji przy zmianie wymagań może być trudna bądź wręcz niemożliwa bez zastosowania modelowania graficznego.



Rys. 1. Uproszczony przebieg pracy w inżynierii systemów Źródło: Opracowanie własne na podstawie [2]

Współcześnie do inżynierii systemów zaliczamy [3]: inżynierię oprogramowania, procesową, mechaniczną, chemiczną oraz elektryczną. Inżynieria systemów wymaga języka oraz narzędzi pozwalających w sposób kompleksowy ująć projektowane rozwiązanie, niezależnie od dyscypliny, w jakiej projekt jest realizowany. W niniejszym artykule skupiono się na zastosowaniu podejścia inżynierii systemów w projektowaniu mechatronicznym, łączącym inżynierię mechaniczną, elektryczną z inżynierią oprogramowania.

2. Aspekty modelowania

Z rysunku 1 wynikają następujące potrzeby, niezależnie od przyjętych narzędzi modelowania:

- opracowanie modelu wymagań;
- opracowanie modelu użycia systemu – tzw. kontekstu realizacji czy użycia rozwiązania;
- opracowanie modelu zachowania/działania systemu jako całości, jak również działania poszczególnych jego komponentów;
- opracowanie modelu architektury sprzętowo-programowej systemu, w tym interfejsów pomiędzy komponentami;
- opracowanie modelu specyfikacji implementacji poszczególnych komponentów systemu;
- opracowanie modelu parametrów i ograniczeń związanych z projektowanym rozwiązaniem, przy czym przez parametry rozumiemy zarówno ustawienia fabryczne systemu, jak i te modyfikowane z poziomu interfejsu użytkownika podczas działania systemu;
- opracowanie modelu testów, koniecznych do weryfikacji i walidacji postawionych projektowanemu systemowi wymagań.

Poza wszystkimi powyższymi aspektami modelowania systemów należy pamiętać o konieczności uwzględnienia w specyfikacji relacji pomiędzy elementami danego typu modelu, jak również relacji pomiędzy elementami różnych typów modeli. W niniejszym artykule przedstawiono, jak wybrane narzędzia, języki modelowania oraz implementacji systemów sterowania obiektami mechatronicznymi wspierają wymienione powyżej aspekty modelowania.

3. UML, SysML, Simulink, IEC6113-3. Narzędzia graficznego modelowania systemów

Z inżynierią oprogramowania od wielu lat związany jest język UML. Na jego temat napisano wiele książek [4] i artykułów. UML jest językiem modelowania graficznego, używanym najczęściej do opisu procesów biznesowych, oprogramowania oraz architektury systemów. UML jest językiem o wysokim poziomie elastyczności, a tym samym bardzo rozpowszechnionym w inżynierii oprogramowania. UML jako język graficzny posługuje się diagramami struktur oraz zachowania.

Do diagramów UML opisujących strukturę modelowanego systemu zaliczamy diagramy: klas, komponentów, obiektów, wdrożenia, struktur złożonych oraz pakietów. Do diagramów UML definiujących zachowanie komponentów systemu zaliczamy diagramy: przypadków użycia, aktywności, maszyny stanów, komunikacji, sekwencji, czasowe oraz interakcji.

W książce [5] zaproponowano zastosowanie UML do wsparcia prac nad projektami mechatronicznymi. Niestety z uwagi na zbyt dużą elastyczność, swobodę modelowania trudne i niejednoznaczne staje się znalezienie zrozumiałych dla szerszej grupy użytkowników analogii pomiędzy elementami języka UML a komponentami systemów sterowania czasu rzeczywistego.

SysML, drugi z opisywanych tutaj języków graficznej specyfikacji, wywodzi się z UML, jednakże dostarcza dużo precyzyjniejszych, a tym samym bardziej zrozumiałych narzędzi modelowania systemów mechatronicznych (w tym systemów sterowania). SysML stanowi rozszerzenie UML ukierunkowane na wsparcie specyfikacji systemów wszędzie tam, gdzie projektowane oprogramowanie (inżynieria oprogramowania) wykonywane jest przez rzeczywiste systemy sterowania (inżynieria elektryczna), dokonujące pomiarów oraz wymuszające ruch elementów mechanicznych, konstrukcyjnych (inżynieria mechaniczna). SysML jest zatem doskonałym językiem opisu maszyn oraz wszelkich systemów mechatronicznych [6].

Język SysML składa się z następujących typów diagramów, opisujących systemy w zakresie:

- modelowania wymagań dla systemu: diagram wymagań (req);
- modelowania struktury i architektury systemu: diagramy pakietów (pkg), definicji bloków (bdd), struktury wewnętrznej bloków (ibd), parametrów i ograniczeń (par);
- modelowania zachowania: diagramy aktywności (act), sekwencji (seq), maszyny stanów (sm) oraz przypadków użycia (uc).

Dość często spotykaną praktyką jest rozpoczynanie definiowania wymagań dla projektowanego systemu właśnie od diagramów przypadków użycia. Stanowią one bowiem stosunkowo czytelną formę opisu systemu z perspektywy użytkownika końcowego.

Istotna zmiana specyfikacji SysML w stosunku do UML polega na uzupełnieniu języka o modelowanie wymagań, stawianych projektowanemu systemowi. Dzięki temu możliwe jest w ramach jednego modelu powiązanie zarówno celów biznesowych rozwiązania, jak i proponowanej architektury systemu. Z drugiej strony SysML doprecyzowuje sporo definicji, które w przypadku UML pozostawiały wiele swobody użytkownikowi. Dzięki temu SysML jest znacznie lepszym narzędziem modelowania systemów mechatronicznych aniżeli UML.

Współcześnie oprogramowanie Matlab/Simulink firmy Mathworks postrzegane jest jako jedno z najpopularniejszych narzędzi projektowania i badań symulacyjnych systemów sterowania. Narzędzie Simulink w naturalny sposób wspiera graficzne modelowanie systemów, zarówno w zakresie architektury, jak i implementacji systemu sterowania.

Projektowane elementy architektury mogą w czytelny sposób zostać grupowane w pakiety (tzw. przyborniki) czy biblioteki. Zastosowanie od niedawna mechanizmu tzw. referencji do modeli utworzonych w zewnętrznych plikach istotnie ułatwia projektowanie i implementację złożonych systemów, jak również wersjonowanie opracowanych modeli.

Norma IEC61131-3 [7] porządkuje proces powstawania oprogramowania dla sterowników PLC. W roku 2013 [8] została rozszerzona o pojęcie programowania obiektowego, co z kolei ułatwia tworzenie projektów systemów sterowania w branżach, które charakteryzuje mnogość wariantów podobnych produktów. Narzędzia programowania sterowników PLC zgodne ze standardem IEC61131-3 wspierają użytkowników podczas projektowania systemów sterowania w następujących obszarach: modelowania architektury oprogramowania, modelowania i konfiguracji architektury sprzętowej, implementacji funkcjonalności z użyciem języków normy IEC61131-3 oraz dokumentowania implementacji. Języki wymienione w tejże normie można podzielić na dwie podstawowe grupy: języki tekstowe oraz graficzne. Do języków tekstowych zaliczamy język IL (*Instruction List*) oraz ST (*Structured Text*). Języki graficzne to: LAD (*LADdder diagram*), FBD (*Function Block Diagram*), SFC (*Sequential Function Chart*) oraz CFC (*Continuous Function Chart*).

Współcześnie coraz większą uwagę przywiązuje się do tzw. wersjonowania kodu źródłowego systemów sterowania, m.in. z zastosowaniem technologii informatycznych jak SVN (*SubVersioN control*).

Weryfikacja poprawności opracowywanego kodu w zakresie zgodności definicji i deklaracji użytych zmiennych stanowi współcześnie jedną z podstawowych funkcji narzędzi do programowania sterowników PLC.

Istotną słabością zarówno oprogramowania Matlab/Simulink, jak i narzędzi do programowania sterowników jest brak możliwości modelowania (istotnych z punktu widzenia cyklu życia produktów mechatronicznych) relacji pomiędzy wymaganiami, architekturą, implementacją a przypadkami użycia projektowanego systemu.

W jednym z kolejnych artykułów niniejszego cyklu poruszone zostanie zagadnienie testowania aplikacji. Współcześnie jest to najintensywniej rozwijany obszar inżynierii systemów, wspierany na różne sposoby tak w UML, SysML, jak i oprogramowaniu Matlab/Simulink oraz przez narzędzia programowe dla sterowników PLC.

Podsumowanie możliwości poszczególnych notacji i narzędzi w zakresie modelowania złożonych systemów mechatronicznych przedstawiono w poniższej tabeli. W kolejnych podrozdziałach niniejszego artykułu omówiono poszczególne aspekty modelowania z perspektywy tychże narzędzi.

4. Modelowanie wymagań

Zgodnie z definicją przedstawioną w specyfikacji SysML [9]: „wymaganie opisuje zdolność systemu bądź warunki, które muszą (lub powinny) zostać spełnione. Wymaganie może specyfikować funkcje, jakie musi realizować system, bądź poziom wydajności, jaki system musi osiągać". Jak przedstawiono w poprzednim podrozdziale, język UML pozwala modelować oczekiwaną funkcjonalność (jej zakres) za pomocą tzw. diagramów przypadków użycia. Na rysunku poniżej przedstawiono trywialny

Tabela 1. Podsumowanie możliwości opisywanych narzędzi i języków w zakresie modelowania systemów

		UML	SysML	Simulink	IEC61131-3
Wymagania	Model wymagań dla systemu	–	diagram wymagań (req)	–	–
	Model sposobu użycia systemu	diagram przypadków użycia	diagram przypadków użycia (uc)	–	–
	Relacje między wymaganiami a elementami systemu	diagram przypadków użycia, powiązanie z dokumentami zewnętrznymi	deriveReqt, refine, satisfy, verify, copy, trace	–	–
Architektura	Architektura sprzętu	diagramy komponentów, klas, obiektów	diagramy definicji bloków (bdd), struktury wewnętrznej bloków (ibd)	bloki Simulink	konfiguracja sprzętowa sterownika, wejścia/wyjścia fizyczne, protokoły komunikacyjne
	Architektura oprogramowania			bloki Simulink	konfiguracja oprogramowania (czas cyklu, priorytet, mapowanie programów)
	Parametry systemu		diagramy parametrów i ograniczeń (par)	stałe	stałe globalne/lokalne
	Proste typy danych		diagramy definicji bloków (bdd), struktury wewnętrznej bloków (ibd), porty proste/złożone	połączenia pomiędzy blokami, porty proste	typy danych normy IEC
	Złożone typy danych			połączenia pomiędzy blokami, porty typu szyna/struktura, multipleksowanie sygnałów	struktury użytkownika, typy wyliczeniowe
Implementacja	Biblioteki komponentów programowych	obiekty UML	obiekty SysML	przyborniki, biblioteki użytkownika, zewnętrzne pliki modeli	biblioteki standardowe, producenta, użytkownika
	Działanie komponentów	diagramy aktywności, sekwencji, maszyny stanów, czasowe, interakcji, komunikacji	diagramy aktywności (act), sekwencji (seq), maszyny stanów (sm)	modele Simulink	bloki funkcyjne, funkcje, programy
	Działanie systemu		diagramy aktywności (act), sekwencji (seq), maszyny stanów (sm)	modele Simulink, mechanizm referencji do modeli	bloki funkcyjne, funkcje, programy
	Testy implementacji		diagramy sekwencji (seq)	modele Simulink	programy
Dokumentacja	Specyfikacja wymagań	–	diagramy wymagań (req)	–	–
	Specyfikacja architektury	diagramy klas, obiektów	diagramy definicji bloków (bdd), struktury wewnętrznej bloków (ibd)	MATLAB/Simulink Report Generator	projekty, funkcje drukowania projektu wbudowane w IDE
	Specyfikacja implementacji		diagramy aktywności (act), sekwencji (seq), maszyny stanów (sm)		
	Specyfikacja weryfikacji	–			
	Dokumentacja projektu	diagramy pakietów	diagramy pakietów (pkg)		

przykład modelu włączenia i wyłączenia sterownika (systemu) wraz ze szczegółowymi przypadkami użycia (Start-up test, Inicjalizacja programu, Zapisanie zmiennych oraz Sekwencja zatrzymania).

Język UML nie dostarcza innych narzędzi dla modelowania wymagań. W przypadku modeli Matlab/Simulink oraz projektów dla sterowników PLC (zgodnych z normą IEC61131-3) wymagania stawiane projektowanym systemom opracowywane są w formie dokumentów.

Normy IEEE definiują sposoby opracowania i zakres wymagań funkcjonalnych dla oprogramowania.

Standard ogólny:

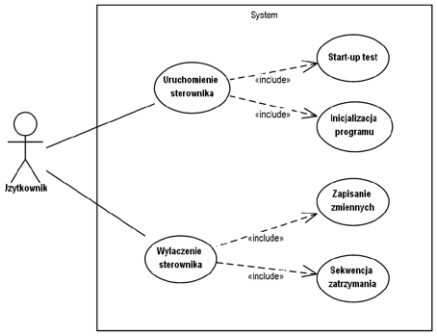
1. ISO/IEC/IEEE 24765:2010, Systems and software engineering – Vocabulary. 2010 [10].

Standardy wprowadzające wytyczne opracowywania wymagań dla systemów:

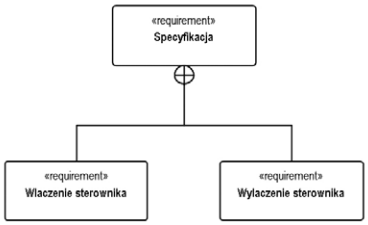
2. IEEE Std 830:1998, IEEE Recommended Practice for Software Requirements Specifications. 1998 [11];
3. IEEE Std 1233:1998, IEEE Guide for Developing System Requirements Specifications. 1998 [12];
4. IEEE Std 1362:1998, IEEE Guide for Information Technology – System Definition – Concept of Operations (ConOps) Document. 1998 [13].

Trzy wyżej wymienione standardy w 2011 roku zostały zastąpione przez normę: ISO/IEC/IEEE 29148:2011, Systems and software engineering – Life cycle processes – Requirements engineering. 2011 [14].

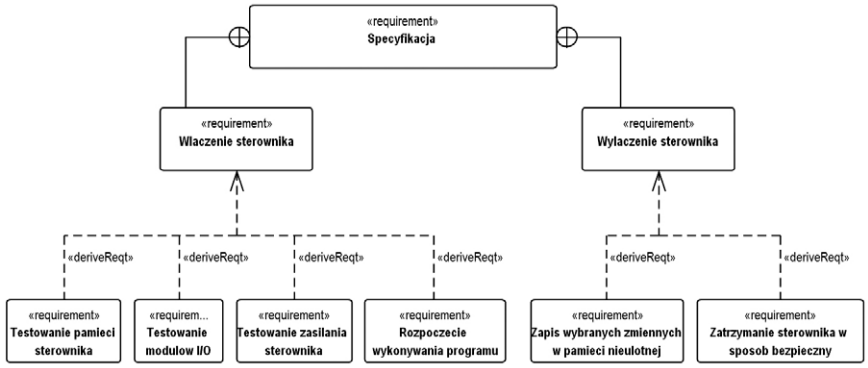
Wadą tych wszystkich rozwiązań jest dokumentowanie wymagań w formie tekstowej. Od wady tej wolny jest język SysML. Umożliwia bowiem graficzne modelowanie wymagań dla projektowanego systemu. Zawiera również istotne dla systemów relacje pomiędzy wymaganiami, przypadkami użycia a architekturą systemu oraz implementacją wybranych jego elementów. Dzięki temu



Rys. 2. Modelowanie wymagań przez model przypadków użycia z zastosowaniem języka UML



Rys. 3. Model wymagań – relacja „nesting" lub inaczej „containment"



Rys. 4. Model wymagań – relacja „deriveReqt"

modelowanie wymagań staje się czytelne, zaś specyfikacje pełniejsze.

Model wymagań dla systemu z rysunku 2 w notacji SysML przedstawiono na kolejnym rysunku.

Relacja przedstawiona na rysunku 3 opisuje uszczegółowienie wymagania wyższego poziomu na wymagania bardziej szczegółowe. Interpretacja schematu z rysunku 3: „System spełnia założenia Specyfikacji"; „Włączenie sterownika" oraz „Wyłączenie sterownika" to wymagania funkcjonalne dla projektowanego systemu. Kolejnym typem relacji pomiędzy wymaganiami, stawianymi projektowanemu systemowi jest „deriveReqt". Relacja ta opisywana jest strzałką skierowaną od wymagania niższego poziomu do wymagania poziomu wyższego, z którego czerpie swoje wystąpienie.

Przykład diagramu wymagań dla systemu przedstawiono na rysunku 4.

Na kolejnym rysunku przedstawiono przykład relacji „refine". Relacja ta wskazuje, które z elementów systemu spełniają stawiane projektowanemu systemowi wymagania. Diagram z rysunku 5 można odczytywać w następujący sposób. Przypadek użycia „Uruchomienie sterownika" (jego implementacja zgodnie z opisem tegoż przypadku) spełnia założenia wymagania „Włączenie sterownika".

Przykładowa, przedstawiona tutaj specyfikacja zakłada, iż projektowany system zawierać będzie: testowanie zasilania sterownika podczas włączania, testowanie modułów wejść/wyjść oraz testowanie pamięci sterownika. Implementacja przypadku użycia „Start-up test" musi spełniać postawione w ten sposób wymagania.

Pozostałe relacje pomiędzy wymaganiami a elementami systemów związane są z architekturą przyjętego rozwiązania i jako takie zostaną przedstawione w dalszej części niniejszego artykułu. Język SysML umożliwia, jak widać, powiązanie istotnych dla projektu elementów modelowanego systemu, również w wygodny dla użytkownika sposób graficzny.

W ramach oprogramowania Matlab/Simulink podejmowane są próby integracji modeli, a nawet całych projektów z komercyjnymi narzędziami modelowania wymagań. Przybornik Simulink Verification and Validation pozwala, by opracowane w Simulink modele można było powiązać z wymaganiami opracowanymi w formie dokumentów różnego formatu, przechowywanych na dysku twardym komputera bądź na ogólnodostępnych lokalizacjach sieciowych. Istotną wadą rozwiązania jest konieczność aktualizacji

powiązań pomiędzy modelami a dokumentami w przypadku, gdy dokumenty te powstają w wyniku procedur generowania specyfikacji wymagań z narzędzi do ich graficznego modelowania.

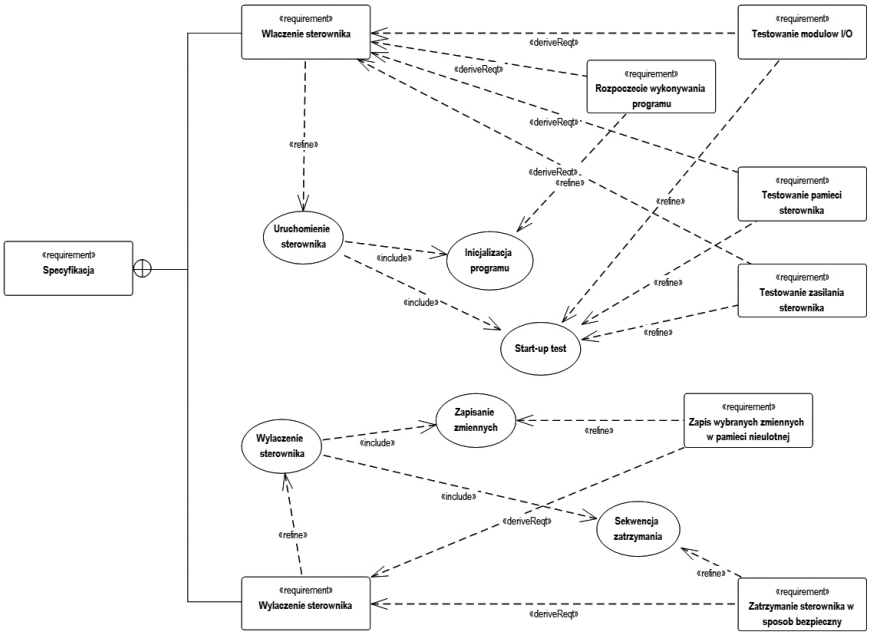
Wyjątek stanowi tutaj możliwość integracji modeli Simulink z oprogramowaniem DOORS firmy IBM, gdzie modele są powiązane z elementami modelu wymagań, a nie z wynikową dokumentacją.

Wadą aktualnego rozwiązania oferowanego przez Mathworks w tym zakresie jest brak pełnej, wygodnej dla użytkownika integracji z najpopularniejszymi współcześnie programami modelowania UML czy SysML, takimi jak Enterprise Architect, Visual Paradigm czy MagicDraw. Narzędzia oferowane na rynku pozwalają niekiedy wykorzystać silnik obliczeniowy Matlab/Simulink do weryfikacji działania modeli w nich opracowanych. Wydaje się to jednak rozwiązaniem, które posiada swój odpowiednik w silnikach obliczeniowych, wbudowywanych zwykle w wymienione tutaj EA, VP czy MD.

Można z pełną świadomością stwierdzić, iż dla większości inżynierów oprogramowanie Matlab/Simulink służy jedynie do modelowania architektury oraz implementacji komponentów projektowanego systemu (o czym więcej w dalszej części artykułu). Wymienione oprogramowanie realizuje to w bardzo wygodny dla użytkownika sposób. Zastosowanie bibliotek pozwala na czytelną strukturyzację projektów oraz na ponowne wykorzystanie w wielu projektach raz opracowanych i zaakceptowanych komponentów tworzonych systemów.

W narzędziach do programowania sterowników PLC zgodnie z normą IEC61131-3 modelowanie wymagań realizowane jest z zastosowaniem statycznych powiązań z zewnętrznymi dokumentami. Niestety narzędzia IDE (ang. *Integrated Development Environment*) dla sterowników PLC, pomimo pełnej zgodności z normą IEC61131-3, w niskim stopniu wspierają zagadnienia modelowania i zarządzania wymaganiami, a tym samym całym cyklem życia produktów.

Przyjrzyjmy się zatem, jak poszczególne notacje, języki i grupy narzędzi wspierają tworzenie modeli architektury systemów sterowania.



Rys. 5. Model wymagań i model przypadków użycia – relacja „refine"

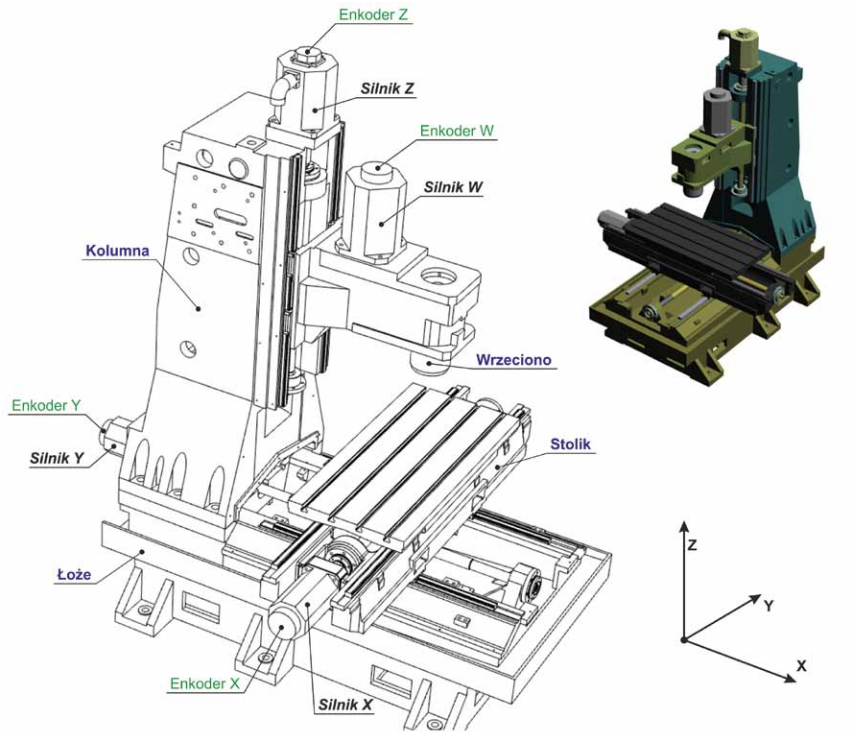
5. Modelowanie architektury

Znając wymagania stawiane systemowi sterowania układem mechatronicznym, można przystąpić do projektowania architektury systemu, który postawione wymagania spełni. Zgodnie z podejściem charakterystycznym dla inżynierii systemów (rysunek 1), w pierwszej kolejności na podstawie wymagań stawianych poszczególnym komponentom są one modelowane, implementowane i weryfikowane. W ten sposób powstają biblioteki – zestawy elementów, z których na bazie wymagań wyższego poziomu można przystąpić do budowania spełniającego wymagania klienta rozwiązania. Taka kolejność pracy zapewnia, iż budowane systemy wolne są od większości błędów wynikających z ludzkich pomyłek (których liczba rośnie zwykle wraz z poziomem złożoności implementowanego komponentu/systemu).

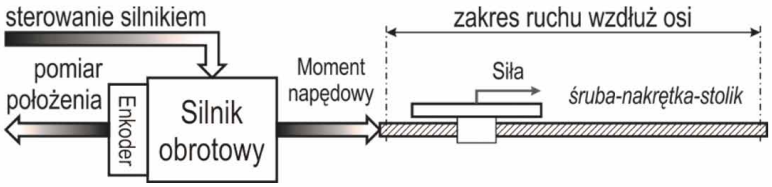
Zaletą podejścia obiektowego modelowania architektury systemów, którego różne realizacje obecne są zarówno w UML, SysML, narzędziach zgodnych z IEC61131-3, jak i w oprogramowaniu Matlab/Simulink, jest fakt, iż użytkownicy mają możliwość rozbicia prac nad bardzo złożonymi systemami na elementarne jednostki – komponenty systemu.

W niniejszym artykule zagadnienie modelowania architektury przedstawiono na przykładzie systemu sterowania trójosiowej frezarki sterowanej numerycznie z układu CNC o otwartej architekturze funkcjonalnej, opracowanego w ramach projektu badawczego „Opracowanie i badania prototypu obrabiarkowego zespołu posuwowego z napędami liniowymi, sterowanego w dwóch osiach z układu CNC o otwartej architekturze", realizowanego pod kierownictwem prof. dra hab. inż. Stefana Domka na Wydziale Elektrycznym oraz Inżynierii Mechanicznej i Mechatroniki ZUT w Szczecinie w latach 2007–2010. Ogólny schemat architektury sterowania napędami posuwu zamieszczono już w artykule [15]. Na rysunku 6 przedstawiono elementy wykonawcze i konstrukcyjne obiektu sterowania.

Na konstrukcję obrabiarki składają się: kolumna, łoże, stolik, wrzeciono (rysunek 6). Stolik porusza się po konstrukcji łoża na tzw. prowadnicach liniowych i wózkach prowadnicowych. Wrzeciono zamontowane jest w elemencie poruszającym się pionowo w górę i w dół również na prowadnicach. Ruch poszczególnych elementów konstrukcji realizowany jest w osiach X, Y, Z z zastosowaniem



Rys. 6. Przykładowy model frezarki 3-osiowej. Widok ogólny oraz opis komponentów architektury



Rys. 7. Struktura układu pomiarowego modelu obrabiarki z rysunku 6

trzech silników typu PMSM z wbudowanymi wieloobrotowymi enkoderami absolutnymi (w prezentowanym rozwiązaniu w standardzie EnDat 2.1). Ruch translacyjny elementów konstrukcyjnych wywoływany jest z zastosowaniem przekładni śrubowej kulowo-tocznej, w każdej z trzech osi.

Rozwiązanie przedstawione na rysunku 6 z punktu widzenia techniki pomiarowej dla potrzeb realizacji procedur ruchu nazywane jest otwartym (rysunek 7). Wynika to z faktu braku bezpośredniego sprzężenia pomiarowego od elementów konstrukcyjnych obrabiarki. Układ ten charakteryzuje niepewność pomiaru położenia komponentów rzeczywistych w stosunku do estymacji na podstawie kąta wałka silnika, na którym zamontowany jest enkoder.

Na rysunku 8 przedstawiono model architektury trzyosiowej frezarki CNC z rysunku 6, zapisany w języku SysML. Jest to tzw. diagram definicji bloków (bdd).

Z rysunku 8 wynikają następujące wnioski. System CNC zawiera jeden generator trajektorii dla danej konstrukcji maszyny. Liczba elementów wykonawczych (serwonapędów, silników), jak również mechanizmów śrubowych kulowo-tocznych dla danego systemu CNC musi wynosić co najmniej 1. Ograniczeniem dla systemu jest liczba osi ruchu danego rozwiązania konstrukcyjnego. W zaprezentowanym tutaj przykładzie są trzy osie ruchu – co jest równe liczbie elementów

śrubowych kulowo-tocznych – zastosowane do budowy maszyny.

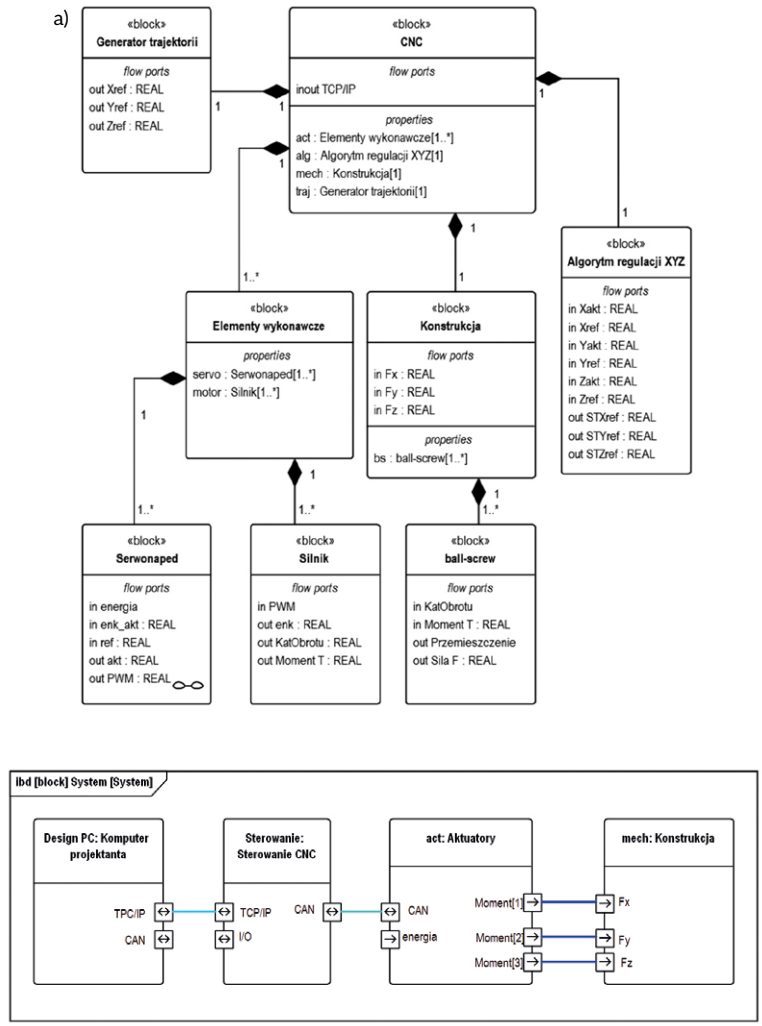
Na rysunku 9 przedstawiono diagramy wewnętrzne systemu z rysunku 8 b.

W przypadku modeli architektury zarówno SysML, jak i UML posługują się podobną notacją. W języku UML odpowiednikiem diagramu bdd z rysunku 8 a będzie niemalże identyczny diagram klas, natomiast UML-owy diagram komponentów będzie bardzo podobny do diagramu ibd z rysunku 8 b.

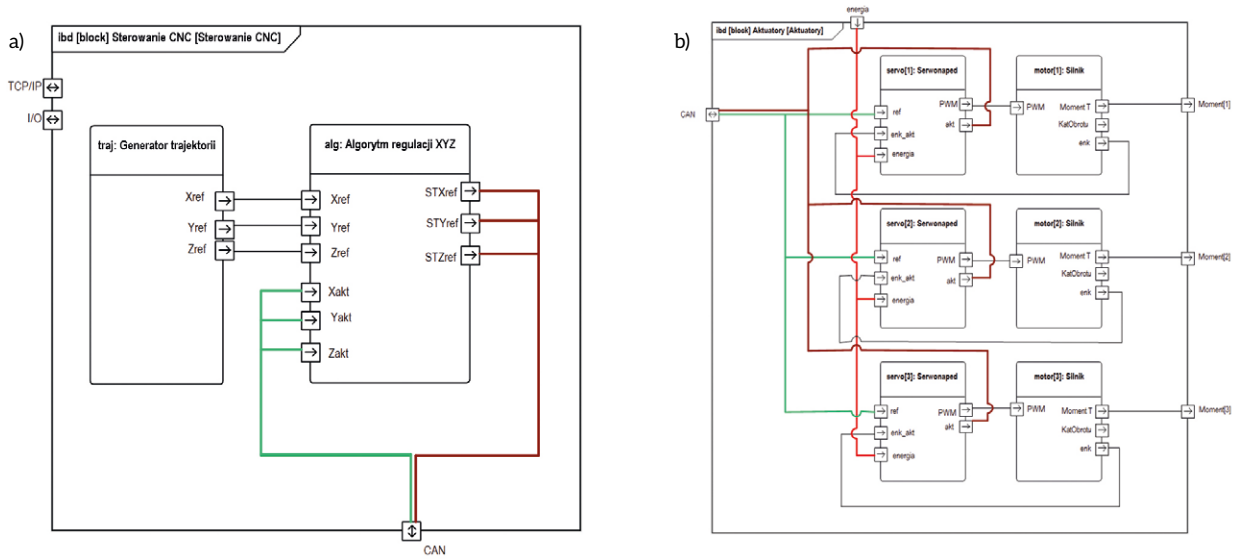
Na rysunku 10 przedstawiono widok biblioteki komponentów (CNC_lib), z których następnie w oprogramowaniu Matlab/Simulink zbudowano model architektury systemu z rysunku 6.

Analogicznie do elementów systemu CNC z rysunku 9, na kolejnym rysunku przedstawiono model architektury wewnętrznej wybranych komponentów.

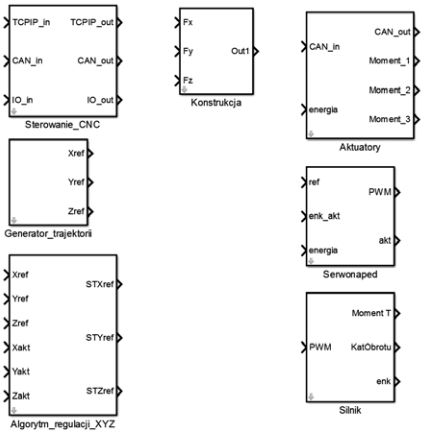
Oprogramowanie Matlab/Simulink od niedawna wspiera tzw. referencje do zewnętrznych modeli. Poza biblioteką komponentów możliwe jest utworzenie globalnego repozytorium modeli, nad którymi użytkownicy mogą pracować niezależnie od budowanej architektury. Aktualizacja interfejsu w modelach nadrzędnych dokonywana jest automatycznie po wprowadzeniu zmian w modelach niższego poziomu. Mechanizm ten usprawnia prace nad złożonymi architekturami wielopoziomowych systemów sterowania.



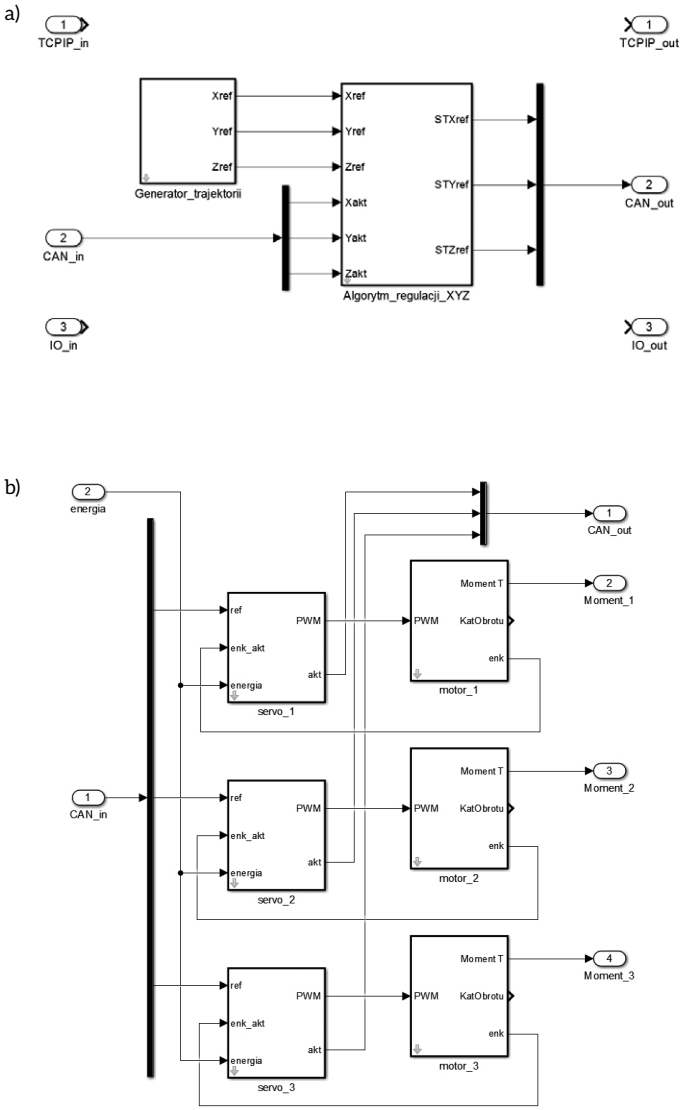
Rys. 8. Architektura systemu – obrabiarki wraz z systemem sterowania w języku SysML. Diagram zakresu architektury (a) oraz przykładowa konfiguracja uwzględniająca komputer projektanta, a także interfejs CAN dla komunikacji systemu sterowania z serwonapędami (b)



Rys. 9. Architektura wewnętrzna wybranych komponentów systemu sterowania obrabiarką. Komponenty systemu sterowania (a) oraz architektura zestawu elementów wykonawczych (b) w języku SysML



Rys. 10. Biblioteka komponentów systemu CNC



Rys. 11. Architektura wewnętrzna wybranych komponentów systemu sterowania obrabiarką. Komponenty systemu sterowania (a) oraz zestawu elementów wykonawczych (b) w oprogramowaniu Simulink

Ostatnim z opisywanych w niniejszym artykule podejść modelowania architektury systemów jest norma IEC 61131-3. Na kolejnym rysunku zamieszczono widok grupy bloków funkcyjnych POU w oprogramowaniu CodeSys v2.3.

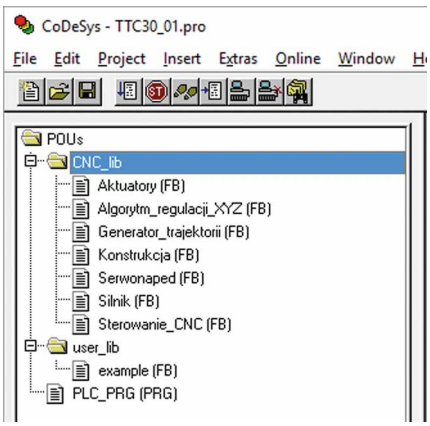
Narzędzia programowania sterowników PLC zgodnie z normą IEC61131-3 posiadają wiele rozwiązań wspierających implementację (kodowanie), jak np. kolorowanie kodu, podpowiadanie nazw zadeklarowanych uprzednio zmiennych, kontrolę poprawności nazw zmiennych, nazw wejść/wyjść bloków funkcyjnych, poprawności użytych typów danych. W zakresie modelowania architektury sprzętowej i programowej producenci sterowników prześcigają się w pomysłach usprawnienia i skrócenia czasu tych czynności, również z użyciem konfiguratorów graficznych.

Modelowanie architektury jest zagadnieniem bardzo złożonym, zwykle poziom optymalności modelu architektury zależy od doświadczenia projektanta. Ważne jest jednak nie to, jak łatwo modeluje się architekturę systemu, tylko na ile język lub narzędzia wspierają powiązanie modeli architektury z wymaganiami stawianymi projektowi tejże architektury. Jedynym rozwiązaniem wspierającym to zagadnienie w sposób bezpośredni jest język SysML.

W tabeli 1 przywołano również zagadnienie modelowania parametrów systemu, prostych oraz złożonych typów danych. Jest to o tyle ważne, że wszystkie porównywane w niniejszym artykule narzędzia i języki mają to zagadnienie rozwiązane w wygodny, specyficzny dla siebie sposób. Wszystkie udzielają wsparcia dla modelowania typów złożonych, jak tablice, struktury, typy wyliczeniowe, typy specjalne (data, czas).

6. Modelowanie implementacji

Na podstawie postawionych systemowi wymagań, po wykazaniu, iż przyjęta architektura rozwiązania spełni oczekiwania klienta, można przystąpić do implementacji systemu. UML oraz SysML nie są językami, dzięki którym implementacja jest wspierana bezpośrednio.



Rys. 12. Widok biblioteki komponentów (bloków funkcyjnych) w środowisku CodeSys v2.3 dla systemu z rysunku 6

Z kolei Simulink oraz oprogramowania dla sterowników PLC są stworzone właśnie dla tego celu (!).

Matlab/Simulink z zastosowaniem wielu bibliotek standardowych oraz złożonych, tworzonych przez użytkowników (rysunek 13 b), pozwala na implementację dowolnie skomplikowanych algorytmów.

W przypadku oprogramowania CoDeSys ciekawostką jest działalność grupy OSCAT (www.oscat.de). Na stronie internetowej znaleźć można darmowe biblioteki wielu złożonych funkcji, możliwych do zastosowania w aplikacjach opartych o sterowniki PLC.

Narzędzia IDE zgodne z IEC61131-3 współcześnie umożliwiają implementację funkcji z użyciem wszystkich zdefiniowanych w normie języków programowania, zarówno tekstowych, jak i graficznych.

Przykłady implementacji regulatora PID o transmitancji w wersji tzw. szeregowej z interakcją nastaw (K_c – wzmocnienie proporcjonalne, T_d – stała czasowa różniczkowania, T_i – stała czasowa całkowania)

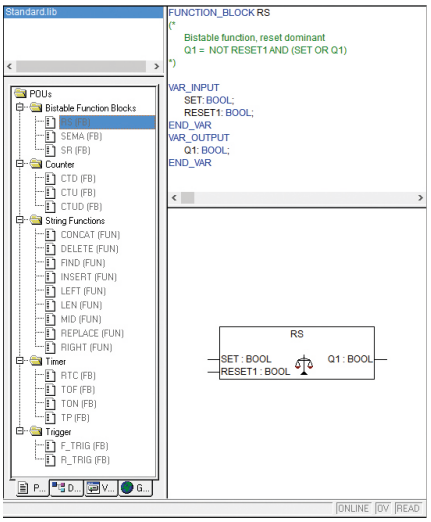
$$R(s) = \frac{K_c(1 + sT_d)(1 + sT_i)}{sT_i} \qquad (1)$$

zamieszczono na rysunku 14. W implementacji istotnym parametrem jest również czas próbkowania T_s.

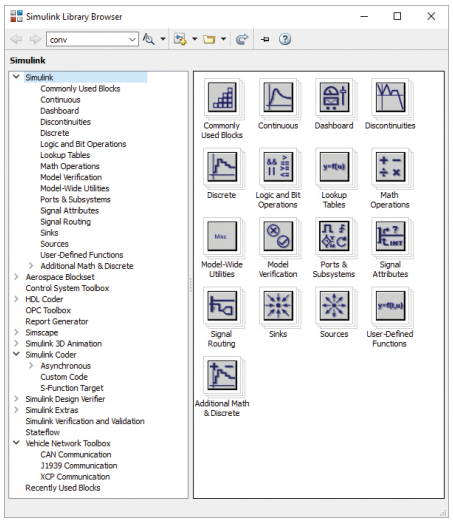
Jak widać na rysunku 14, UML oraz SysML umożliwiają zamodelowanie jedynie obiektu regulatora PID – sama implementacja musi zostać opracowana w formie kodu źródłowego dołączanego do projektu jako plik zewnętrzny. Model implementacji z rysunku 14 b (SysML ibd) tym różni się od modelu z rysunku 14 c (model Simulink), że ten drugi może zostać wykonany w trybie symulacyjnym, a następnie wygenerowany do platformy sprzętowej. Model z rysunku 14 b jest jedynie graficzną reprezentacją obiektów, użytych do implementacji. Narzędzia zgodne z IEC61131-3 oraz oprogramowanie Matlab/Simulink poza modelem obiektu/architektury rozwiązania umożliwiają jednocześnie implementację rozwiązania.

Zanim omówiony zostanie ostatni wątek niniejszego artykułu, należy zauważyć, iż współcześnie naturalnym

a)



b)



Rys. 13. Biblioteki komponentów programowych w CodeSys v.2.3 (a) oraz Simulink 2016a (b)

Tabela 2. Języki i narzędzia – wsparcie poszczególnych rozwiązań

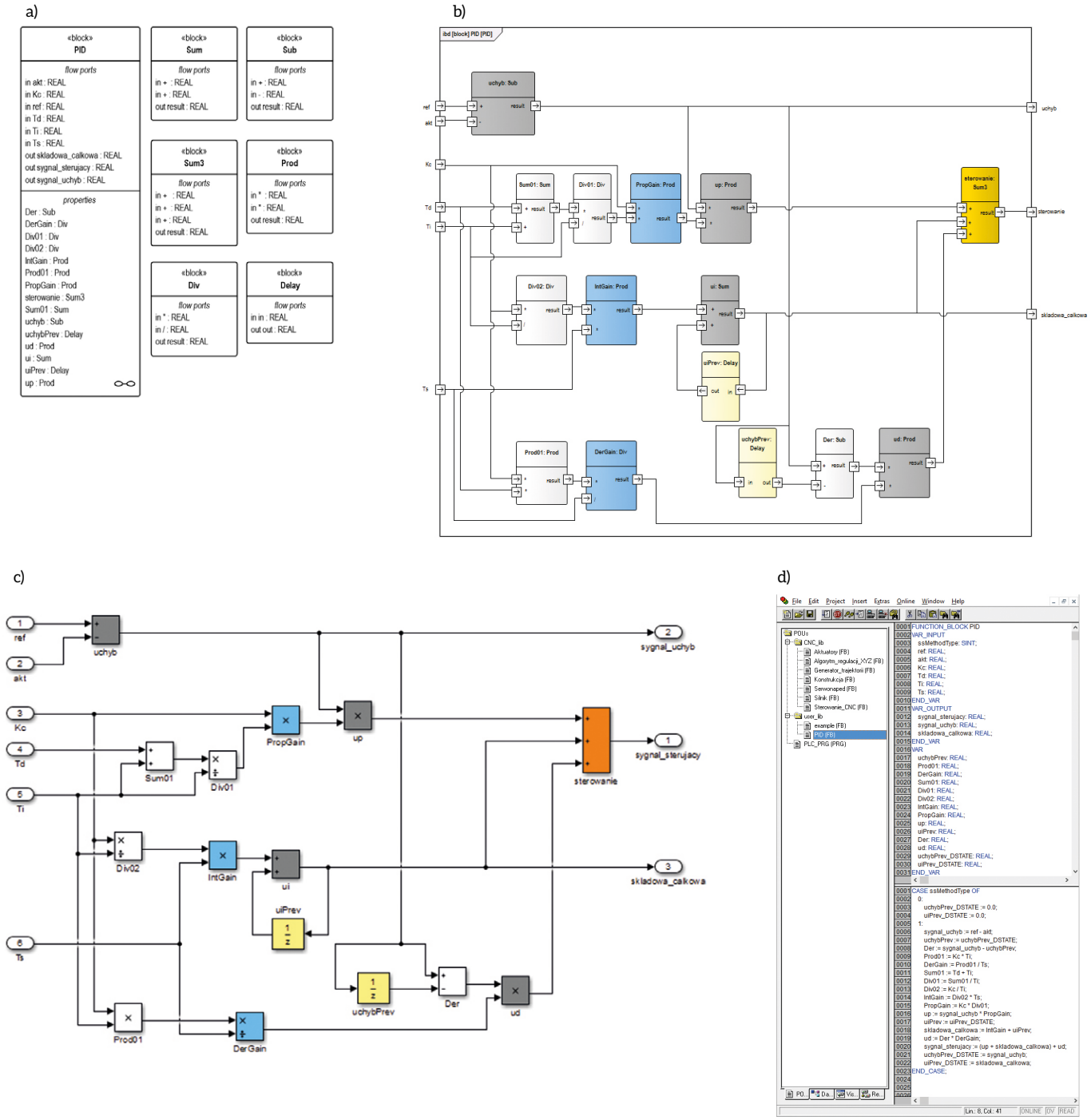
	UML	SysML	Simulink	IEC61131-3
Wymagania	–	+	–	–
Architektura	+	+	+	+
Implementacja	–	–	+	+

podejściem do efektywnego projektowania systemów sterowania może być powiązanie specyfikacji SysML z jednym z dwóch sposobów modelowania architektury i implementacji (tabela 2). Język UML wspiera modelowanie architektury systemów, jednakże dopiero SysML umożliwia efektywne powiązanie architektury ze stawianymi systemom wymaganiami. Ani Simulink ani narzędzia IEC61131-3 nie wspierają modelowania powiązań pomiędzy modelami architektury a wymaganiami. Tym samym użytkownicy zmuszeni są do budowania równolegle dwóch typów specyfikacji – z jednej strony wymagań i architektury, z drugiej zaś na jej podstawie modelowania architektury i implementacji w narzędziach przeznaczonych do tego celu.

Ciekawym i nowatorskim rozwiązaniem może być opracowanie własnych rozwiązań [6], [16], wiążących modele SysML z modelami architektury i implementacją z pomocą tzw. transformacji modeli. Więcej na temat powiązania SysML z modelowaniem architektury i implementacji zostanie przedstawione w jednym z kolejnych artykułów cyklu „Projektowanie mechatroniczne". Na Wydziale Elektrycznym ZUT w Szczecinie od blisko trzech lat prowadzone są bowiem intensywne prace badawcze w tym obszarze. Dzięki nim modele architektury opracowane w języku SysML będą mogły być w sposób automatyczny przetransformowane do modeli Simulink i/lub kodu systemu sterowania zgodnie z normą IEC61131-3.

7. Generowanie kodu oraz dokumentacji na podstawie graficznej specyfikacji

Siłą graficznej specyfikacji, poza niezaprzeczalną przejrzystością przekazu funkcjonowania projektowanych w ten sposób systemów, jest usprawnienie komunikacji i przepływu pracy pomiędzy członkami interdyscyplinarnych, mechatronicznych zespołów badawczych, projektowych. Współcześnie stanowi jeden z niewielu sposobów na zapewnienie efektywnej współpracy pomiędzy zespołami inżynierów z przemysłu i świata nauki.



Rys. 14. Przykład implementacji regulatora PID. SysML bdd (a); SysML ibd (b); Simulink (c) oraz CodeSys v2.3 (d)

Jednakże najważniejsza zaleta zastosowania graficznych modeli z punktu widzenia czasu i kosztu wdrażania nowych funkcjonalności kryje się w tzw. automatyzacji generowania kodu systemu sterowania z jednej strony, z drugiej zaś tworzenia ustrukturyzowanej określonej przepisami dokumentacji projektowej. Specyfikacje oraz narzędzia modelowania UML i SysML wspierają procedury generowania kodu poprzez powiązanie zewnętrznych plików kodu źródłowego z obiektami modeli zapisanymi w tych językach. Tym samym procedura generowania kodu polega na składaniu fragmentów kodu źródłowego w całość w sposób określony strukturą modeli UML/SysML.

Oprogramowanie Matlab/Simulink wspiera procedury generowania kodu na podstawie modeli architektury oraz implementacji na kilka sposobów. Do grupy pierwszej zaliczamy narzędzia do tzw. szybkiego prototypowania, opisane pokrótce w artykule [15], jak karty dSpace czy system Opal RT. Kolejną grupę tzw. platform docelowych (ang. *target platforms*) stanowią rozwiązania typu Arduino, Raspberry Pi, BeagleBoard. Zdobywają one współcześnie olbrzymią popularność z uwagi na atrakcyjną cenę i olbrzymie wsparcie darmowych narzędzi programowych (w tym bogatych

bibliotek obsługi sprzętu dodatkowego: czujników, elementów wykonawczych). Do trzeciej grupy platform stanowiących możliwe narzędzie obliczeniowe dla systemów modelowanych z zastosowaniem Matlab/Simulink należą przemysłowe sterowniki programowalne (PLC). Przybornik Simulink PLC Coder umożliwia wygenerowanie programów oraz bloków funkcyjnych do języka Structured Text normy IEC61131-3 [8], co w praktyce oznacza możliwość generowania kodu dla dowolnego współcześnie dostępnego na rynku sterownika programowalnego zgodnego z tą normą.

Kolejny artykuł cyklu „Projektowanie mechatroniczne" w całości poświęcimy zagadnieniu automatycznego generowania kodu dla sterowników PLC. Z zagadnieniem generowania kodu nierozerwalnie związane jest pojęcie tzw. wykonywalnej specyfikacji. Wykonywalne specyfikacje to takie, które umożliwiają weryfikację postawionych wymagań i działania przed podjęciem kolejnego kroku projektowego – generowania kodu systemu sterowania.

W przypadku narzędzi programowych zgodnych z normą IEC61131-3 zagadnienie generowania kodu nie istnieje, gdyż są one same w sobie narzędziem tworzenia kodu i programowania platformy sprzętowej – sterownika PLC czy programowalnego sterownika automatyki PAC.

Każdy ze sposobów modelowania opisany w niniejszym artykule wspiera zagadnienie automatycznego generowania dokumentacji projektowej. Różne podejścia wynikają ze specyfiki modelowanych zagadnień, jak również wymagań stawianych dokumentacji projektowej na kolejnych etapach realizacji projektu. Narzędzia modelowania UML/SysML umożliwiają tworzenie skryptów automatyzujących generowanie dokumentacji do popularnych standardów, jak *.doc, *.pdf czy *.html. Elementy graficzne, tekstowe, dynamiczne powiązania pomiędzy elementami dokumentu dodatkowo ułatwiają jego dalsze wykorzystanie. Oprogramowanie Matlab/Simulink udostępnia generowanie tzw. raportów, co w praktyce oznacza tworzenie dynamicznych dokumentacji, raportów z analiz modelowanych systemów, tworzenie dokumentów, których struktura wynika z określonych uwarunkowań legislacyjnych dla systemów sterowania. Dokumentacja projektów z narzędzi programowania sterowników PLC wynika głównie z tego, co jest w tych narzędziach modelowane/implementowane: model architektury sprzętowej, programowej, implementacje funkcji, bloków funkcyjnych, programów – wszystko można wyeksportować do zewnętrznych plików *.pdf czy *.html.

8. Podsumowanie

Podsumowując, można stwierdzić, iż współcześnie możliwości modelowania złożonych systemów sterowania w sposób graficzny są coraz większe, a co ważniejsze – pełniejsze niż kiedykolwiek wcześniej.

Dzięki graficznej specyfikacji z zastosowaniem SysML możliwe jest opracowanie wymagań klienta w formie zrozumiałej dla zespołu projektowego, implementującego oczekiwane rozwiązanie, mające w efekcie spełniać stawiane wymagania (rysunek 1). Oprogramowanie takie, jak Matlab/Simulink, ułatwia efektywną graficzną implementację (a wcześniejszą symulacyjną weryfikację i walidację) komponentów programowych systemu sterowania. Norma IEC61131-3 oraz mnogość możliwych do zastosowania języków programowania, w połączeniu ze wspierającymi proces kodowania rozwiązaniami producentów sterowników programowalnych, wspomagają poprawną integrację wszystkich komponentów w ramach rozwiązania zgodnego z oczekiwaniami klienta.

W kolejnym artykule cyklu „Projektowanie mechatroniczne" omówione zostanie szczegółowo zagadnienie generowania i testowania kodu sterownika PLC.

Literatura

[1] Pietrusewicz K.: *Projektowanie mechatroniczne. Projektowanie bazujące na modelach*. „Napędy i Sterowanie" 11/2015.

[2] Friedenthal S., Moore A., Steiner R.: *A Practical Guide To SysML: The Systems Modeling Language*. 2015.

[3] Wrycza S., Marcinkowski B.: *Język inżynierii systemów SysML. Architektura i zastosowania. Profile UML 2.x w praktyce*. Helion 2010.

[4] Maksimchuk R.A., Nalburg E.J.: *UML dla zwykłych śmiertelników*. mikom 2007.

[5] Mrozek Z.: *Komputerowo wspomagane projektowanie systemów mechatronicznych*. Kraków 2002.

[6] Mhenni F., Choley J.-Y., Penas O., Plateaux R., Hammadi M.: *A SysML-based methodology for mechatronic systems architectural design*,. Adv. Eng. Informatics, vol. 28, no. 3, Aug. 2014, pp. 218–231.

[7] John K.H., Tiegelkamp M.: *IEC 61131-3: Programming Industrial Automation Systems*. Springer Berlin – Heidelberg 2010.

[8] IEC 61131-3:2013. Programmable controllers. Part 3: Programming languages. 2013.

[9] Omg, „OMG Systems Modeling Language (OMG SysML) v.1.4," Source, no. June. 2010, p. 260.

[10] ISO/IEC/IEEE 24765:2010, Systems and software engineering – Vocabulary. 2010.

[11] IEEE Std 830:1998, IEEE Recommended Practice for Software Requirements Specifications. 1998.

[12] IEEE Std 1233:1998, IEEE Guide for Developing System Requirements Specifications. 1998.

[13] IEEE Std 1362:1998, IEEE Guide for Information Technology – System Definition – Concept of Operations (ConOps) Document. 1998.

[14] ISO/IEC/IEEE 29148:2011, Systems and software engineering – Life cycle processes – Requirements engineering. 2011.

[15] Pietrusewicz K.: *Projektowanie mechatroniczne. Technika Hardware-in-the-loop a założenia Industry 4.0*. „Napędy i Sterowanie" 4/2016.

[16] Vogel-Heuser B., Schütz D., Frank T., Legat C.: *Model-driven engineering of Manufacturing Automation Software Projects – A SysML-based approach*. Mechatronics, vol. 24, no. 7, Oct. 2014, pp. 883–897.

dr hab. inż. Krzysztof Pietrusewicz;
dr inż. Michael Scopchanov –
Zachodniopomorski Uniwersytet Technologiczny w Szczecinie, Wydział Elektryczny

artykuł recenzowany