BIULETYN INSTYTUTU AUTOMATYKI I ROBOTYKI
NR 16, 2001

Analiza i projektowanie systemów wbudowanych z wykorzystaniem Rational Unified Process (RUP)

Andrzej STASIAK, Zbigniew ZIELIŃSKI

Zakład Systemów Komputerowych, Instytut Automatyki i Robotyki WAT, ul. Kaliskiego 2, 00-908 Warszawa

STRESZCZENIE: W pracy przedstawiono przykład wykorzystania procesu wytwórczego RUP (ang. Rational Unified Process) do produkcji oprogramowania systemów wbudowanych. Na bazie projektu pokładowego systemu sterowania uzbrojeniem samolotu szkolno-bojowego zaprezentowano proces analizy i projektowania oprogramowania z wykorzystaniem języka UML oraz narzędzi Rational Suite Enterprise.

1. Wprowadzenie

Systemy wbudowane stanowią klasę systemów komputerowych, w których istnieje bardzo ścisłe powiązanie z obiektem, na rzecz którego pracują (na przykład system pokładowy samolotu, śmigłowca itp.). Projektowanie wbudowanych systemów sterowania, zwłaszcza systemów pokładowych samolotu, przedstawia znaczne trudności technologiczne i trudno poddaje się automatyzacji. Cechą wyróżniającą proces wytwarzania tego typu systemów jest zwykle konieczność zintegrowanego podejścia do projektowania sprzętu i oprogramowania. Sytuację komplikują nierzadko formułowane zwiększone wymagania niezawodnościowe. Wymusza to stosowanie systematycznego podejścia wspartego odpowiednią metodyką wytwarzania systemów wbudowanych.

RUP jest zunifikowanym procesem wytwórczym oprogramowania dostarczającym praktycznych wskazówek, wzorców dokumentów i narzędzi, szablonów dokumentów oraz przykładów postępowania dla niemalże wszystkich

działań związanych z procesem wytwarzania oprogramowania. Jest ściśle zintegrowany z zestawem narzędzi Rational Suite Enterprise i Rational ClearCase. RUP pozwala na wykorzystanie pełnych możliwości tych narzędzi i języka UML (ang. Unified Modeling Language) oraz innych sprawdzonych rozwiązań przemysłowych.

Podstawę procesu wytwarzania oprogramowania zgodnie z RUP stanowią następujące zasady ogólne:

1) wytwarzanie iteracyjne (przyrostowe),
2) efektywne zarządzanie zasobami,
3) modelowanie wizualne,
4) wykorzystywanie architektury komponentowej,
5) weryfikacja jakości w całym cyklu rozwojowym,
6) kontrolowanie i nadzór nad zmianami wprowadzonymi do oprogramowania.

RUP może być zaadoptowany do wytwarzania różnego rodzaju systemów. W pracy podjęto próbę przystosowania RUP do wytwarzania oprogramowania systemów wbudowanych. Na przykładzie systemu sterowania podwieszeniami samolotu szkolno-bojowego zaprezentowano jedynie podstawowe czynności analityka i projektanta, dotyczące definiowania wymagań i opracowania projektu systemu. Celem pracy była ocena możliwości RUP oraz narzędzi w analizie i projektowaniu oprogramowania systemów wbudowanych.

Zestaw narzędzi pakietu Rational Suite Enterprise obejmuje (między innymi) [6], [10], [11]:

Rational RequisitePro, Rational Rose, Rational Purify, Rational Visual PureCoverage, Rational Visual Quantify, Rational ClearQuest, Rational Robot, Rational SoDA.

2. Sformułowanie wymagań na system

Proces analizy i projektowania oprogramowania z zastosowaniem metodyki RUP i narzędzi CASE Rational Enterprise pokażemy na przykładzie projektu systemu sterowania podwieszeniami w samolocie szkolno-bojowym. Przykład ten, ze względów oczywistych, zostanie przedstawiony w dużym uproszczeniu i nieco fragmentarycznie.

2.1. Przykład systemu

System przeznaczony jest do zastosowania na pokładzie samolotu szkolno-bojowego. Zasadniczym celem zastosowania tego systemu jest zwiększenie efektywności wykonywania zadania postawionego pilotowi.
W otoczeniu systemu można wydzielić następujące elementy:

- stanowiska ogniowe i bloki zasilania, które przedstawiają sobą sam obiekt sterowania,
- pokładowe urządzenia pilotażowo-nawigacyjne,
- blok sterowania umożliwiający pilotowi sterowanie i kontrolę stanu środków rażenia,
- moduł nawigacji satelitarnej GPS.

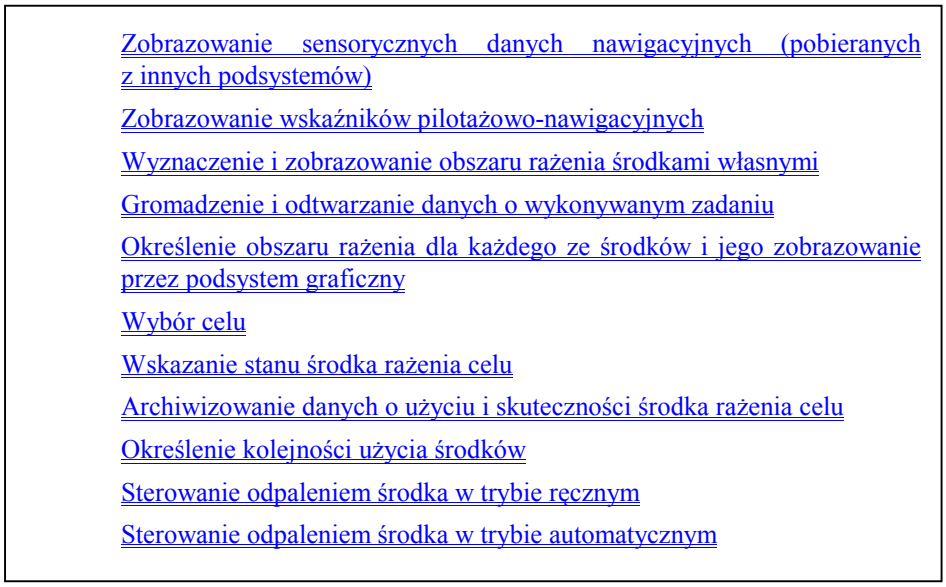
System komputerowy ma umożliwiać sterowanie środkami rażenia w dwóch zakresach: ręcznym i automatycznym. W zakresie automatycznym sterowanie odbywa się na podstawie danych zapisanych na tzw. kasecie danych operacyjnych. System komputerowy ma za zadanie realizować obliczenia nawigacyjne i balistyczne oraz bezpośrednio sterować przygotowaniem stanowisk ogniowych i odpalaniem środków rażenia. Sterowanie jest realizowane poprzez moduły wejść-wyjść dwustanowych. System komputerowy powinien dokonywać zobrazowania mapy terenu i położenia samolotu oraz wybranych punktów związanych z realizacją zadania.

Zasadniczym zadaniem systemu ma być wspomaganie pilota myśliwca w zakresie sterowania środkami rażenia. Pilot w zależności od trybu pracy powinien mieć możliwość wyboru trybu niszczenia celów określonych w zadaniu bojowym.

2.2. Wymagania

Pierwszym etapem inicjującym prace projektowe jest sformułowanie wymagań. W Rational Suite Enterprise wymagania na system gromadzone są w repozytorium projektu i udostępniane z poziomu narzędzia RequisitePro. RequisitePro współpracuje z narzędziami do modelowania, zarządzania projektem, zarządzania zmianami oraz testowania, bowiem wymagania mają bezpośredni wpływ na cały projekt informatyczny. Wymagania te mogą być prezentowane w różnych przekrojach, ilustrując nie tylko werbalnie wymagania stawiane projektowanemu systemowi, ale również sposoby zapewniania ich realizacji przez obiekty projektowe, od modeli do kodu. Przykładowe

wymagania poziomu realizacji[1] sformułowano poniżej. Wymagania te należy poprzedzić tekstem „system zapewni:”. Zarys przykładowych wymagań przedstawia rysunek 1.



Rys. 1. Szkic wymagań sporządzonych z wykorzystaniem narzędzia RequistePro.

3. Modelowanie systemu

Do modelowania systemu wykorzystuje się język UML. Modele UML reprezentują obiekty modelowanego fragmentu rzeczywistości wraz z ich wszystkimi powiązaniami. Wybrany fragment rzeczywistego systemu może być opisywany w kilku perspektywach:

- projektowej (Design view),
- komponentów (Implementation view),
- procesowej (Process view),
- rozmieszczenia (Deployment view),
- użytkownika (Use Case view).

Zalecane w RUP podejście obiektowe do projektowanego systemu pozwoli na uzyskanie niezbędnej elastyczność i czytelności modelu. Powiązania między wytworzonymi obiektowymi modelami zaproponowanymi w RUP

[1]) Wybrano ten poziom specyfikacji wymagań, aby czytelne stały się treści prezentowane w postaci modeli wymagań i proponowanego rozwiązania.

zapewnią bezpośredni związek z wymaganiami użytkownika. Związek ten realizowany jest za pomocą przypadków użycia (ang. *Use case*) będących modelem usług systemu, których realizacja pozwala na zaspokojenie wymagań funkcjonalnych określonych dla systemu.

Do budowy modelu systemu służą zestawy diagramów pokazujące system z różnych perspektyw:

- Diagramy przypadków użycia (ang. Use Case Diagrams);
- Diagramy klas (ang. Class Diagrams);
- Diagramy zachowania (ang. Behavior Diagrams);
    - Diagramy interakcji (ang. Interaction Diagrams);
        - Diagramy sekwencji (ang. Sequence Diagrams);
        - Diagramy współdziałania (ang. Collaboration Diagrams);
    - Diagramy stanów (ang. State Diagrams);
    - Diagramy aktywności (ang. Activity Diagrams);
- Diagramy implementacyjne (ang. Implementation Diagrams);
    - Diagramy komponentów (ang. Component Diagrams);
    - Diagramy rozmieszczenia (ang. Deployment Diagrams).

3.1. Model usług

W opisywanej metodyce, wyniki fazy analizy - w postaci specyfikacji wymagań, mogą być prezentowane z zastosowaniem zaproponowanej przez Jacobsona [5] techniki przypadków użycia. Diagramy te integrują wszystkie wyżej wymienione perspektywy systemu.

Przypadek użycia jest tu traktowany jako ciąg interakcji między aktorem a systemem oraz transakcji zachodzących w systemie i dostarczających aktorowi rezultaty o mierzalnej wartości.

Aktor to abstrakcyjny użytkownik systemu, reprezentujący grupę rzeczywistych użytkowników (np. pilot myśliwca) lub partnerów systemu o podobnych funkcjach i sposobie komunikacji z systemem. Ściślej, przez aktorów rozumie się wszystkie aktywne elementy środowiska systemu komunikujące się z systemem. Mogą to być więc nie tylko piloci, ale też inne systemy czy urządzenia czynnie wykorzystujące system.

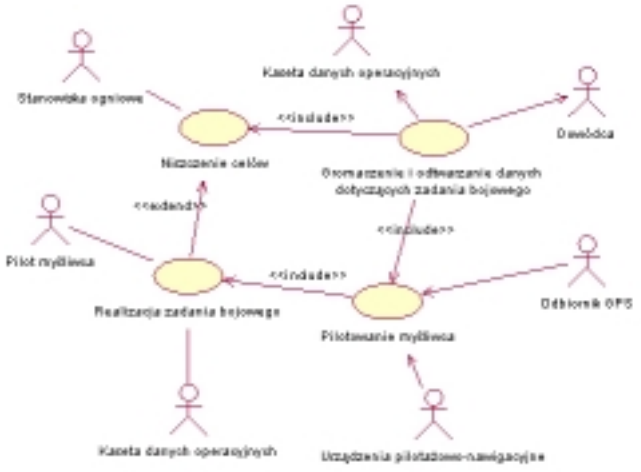
Transakcja jest operacją lub ciągiem operacji w systemie, które albo są całkowicie wykonane, albo też, w przypadku przerwania ich wykonywania, skutki ich działania są całkowicie anulowane i nie pozostawiają żadnych śladów w systemie.

Utworzenie diagramu przypadku użycia przebiega w kilku fazach:

- modelowanie aktorów,
- modelowanie przypadków użycia,

- modelowanie interakcji.

W podanym przykładzie (rysunek 2) można wskazać sześć typów aktorów systemu: pilot myśliwca, dowódca, kaseta danych operacyjnych, urządzenia pilotażowo-nawigacyjne, odbiornik GPS, stanowiska ogniowe. Jak wynika z przedstawionego diagramu (rysunek 2) działania polegające na pilotowaniu myśliwca czy sterowaniu jego środkami rażenia (niszczenie celów) są przykładami tego samego przypadku użycia: realizacja zadania bojowego.



Rys. 2. Model usług systemu.

Stosuje się trzy typy zależności pomiędzy przypadkami użycia:

- relacja „rozszerza” – (stereotyp <<extend>>),
- relacja „zawiera” – (stereotyp <<include>>),
- relacja generalizacji.

Relacja rozszerza łączy dwa przypadki użycia, z których jeden może rozszerzać funkcjonalność drugiego przypadku. Relacja ta jest używana głównie w następujących sytuacjach:

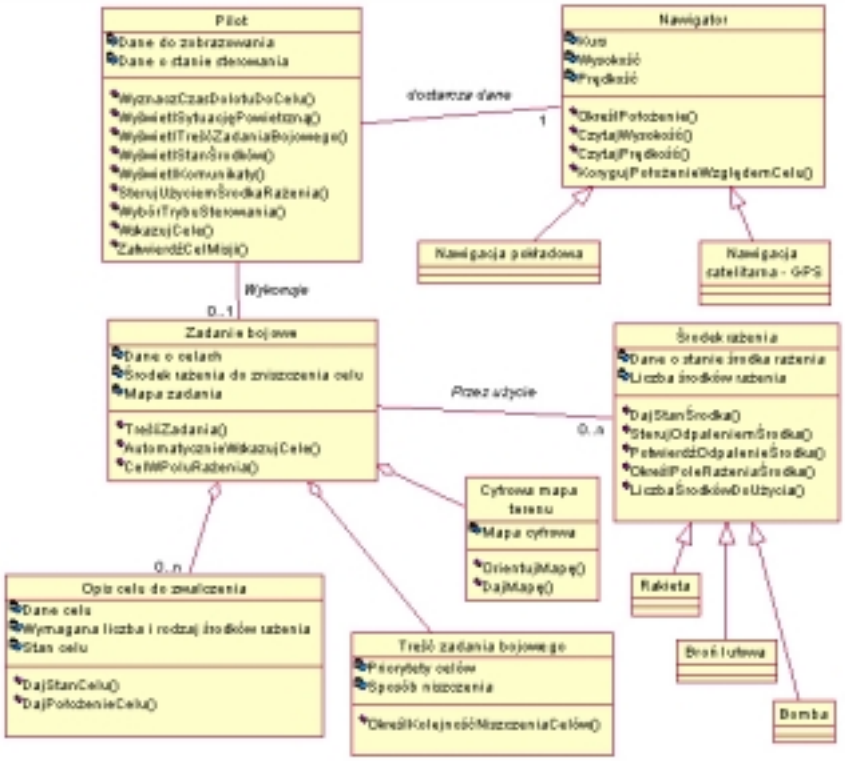
- aby pokazać warunkowe elementy przebiegu przypadku użycia,
- aby modelować złożone i (lub) alternatywne przebiegi zdarzeń w przypadku użycia.

Relacja zawiera łączy dwa przypadki użycia, z których jeden rozszerza funkcjonalność drugiego przypadku. Przy opisie różnych przypadków użycia często można zauważyć, że niektóre z nich zawierają te same elementy. Wspólne części składowe przypadków użycia wyodrębnia się jako oddzielny przypadek i wiąże się go relacją zawiera z przypadkami podstawowymi.

Zastosowanie opisanych powyżej relacji do modelu przypadku użycia systemu nazywa się strukturalizacją przypadków użycia. Jest to szczególnie wartościowa operacja, gdyż umożliwia lepsze zrozumienie modelu przypadków użycia poprzez usuwanie nadmiarów i wydzielanie części logicznie odrębnych.

3.2. Model klas

Model struktury obiektu opisuje statyczną strukturę systemu jako powiązane klasy wraz z ich atrybutami i operacjami. Graficzną reprezentacją modelu struktury obiektu są diagramy klas oraz instancji obiektów. Proces tworzenia modelu struktury rozpoczynamy od specyfikacji klas. Następnie definiujemy związki między klasami, po czym określamy ich krotność. Jeśli jednak którykolwiek związek będzie zawierał usługi, to powinien stać się obiektem. Po określeniu związków poszukujemy struktur całość-część oraz staramy się zidentyfikować „podobne klasy" tworząc nadklasy (model hierarchii klas).

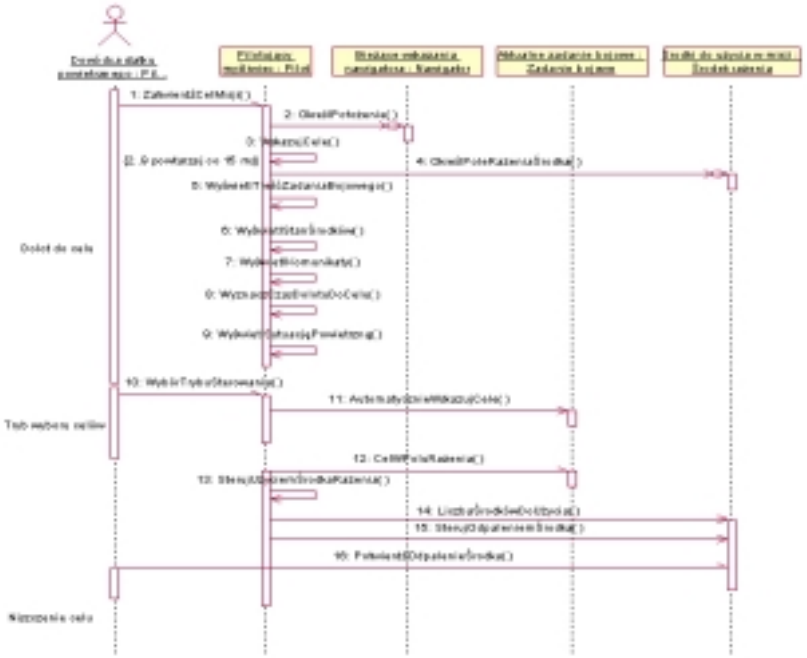


Rys. 3. Model klas systemu.

Na rysunku 3 przedstawiono model struktury systemu poprzez definicje klas: „Pilot", „Zadanie bojowe", „Nawigator", „Środek rażenia" oraz relacji między nimi. Z jego analizy wynika, że klasa „Pilot" pobiera dane z dokładnie jednej klasy „Nawigator" wykonując jedno lub więcej „Zadań bojowych" z użyciem lub bez użycia klasy „Środek rażenia" (opcja lub wiele). Zadaniem klasy „Nawigator" jest dostarczenie bieżących danych nawigacyjnych z wykorzystaniem nawigacji pokładowej lub/i satelitarnej.

Podobnie klasa „Zadanie bojowe" składa się z klas: „Opis celu do zwalczania", „Cyfrowa mapa terenu", „Treść zadania bojowego" powiązanych z nią relacją agregacji. Innym typem relacji jest generalizacja klas. Przedstawia ona hierarchię dziedziczenia obiektów. Na rysunku 3 klasy: „Bomba", „Rakieta" i „Broń lufowa" dziedziczą zarówno atrybuty, jak i metody klasy „Środek rażenia". Jak wynika z relacji między klasą „Zadanie bojowe" a „Środek rażenia" możliwe jest wykonanie zadania bojowego bez użycia środka rażenia (np. rekonesans powietrzny).

3.3. Model działania



Rys. 4. Diagram sekwencji: „automatyczne niszczenie celów".

Model działania (dynamiki) obiektu opisuje jego zachowanie w czasie. Zawiera opis tych aspektów systemu, które dotyczą: sterowania, kolejności wykonywanych operacji, jak również zdarzeń występujących w systemie oraz interakcji zachodzących między obiektami. Model ten jest ilustrowany za pomocą diagramów: interakcji, stanów i aktywności, przy czym interakcje między obiektami systemu są przedstawiane na diagramach: sekwencji i współdziałania. Proces modelowania dynamiki rozpoczyna się od przedstawienia interakcji jako specyfikacji przesyłania wiadomości pomiędzy obiektami w celu wykonania określonego zadania. Diagramy interakcji modelują pojedyncze scenariusze dla poszczególnych przypadków użycia z diagramów przypadków użycia.

Na rysunku 4 przedstawiono przykład scenariusza w postaci diagramu sekwencji „automatyczne niszczenie celów" dla przypadku „niszczenie celów" (patrz rysunek 2). Ze specyfikacji wymagań (rysunek 1) wynika, że dla tego samego przypadku należy przedstawić jeszcze jeden scenariusz, a mianowicie „ręczne sterowanie niszczeniem celów" – lub przedstawić jeden scenariusz w tzw. postaci generycznej.

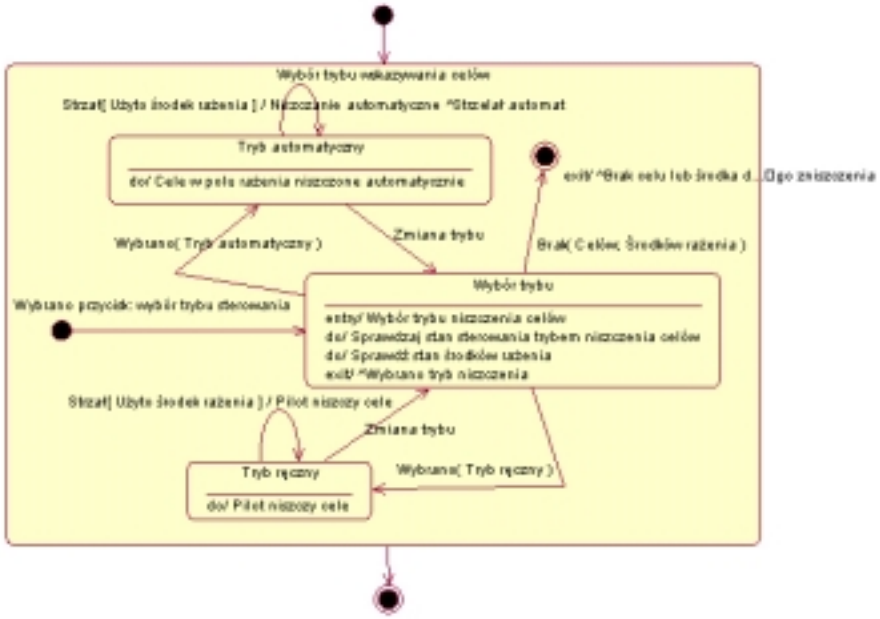
Tworzenie diagramu rozpoczyna się od specyfikacji obiektów z określeniem ich linii życia (pionowe przerywane linie określające upływ czasu. Mogą one wyznaczać czas rozpoczęcia kreowania obiektu - od góry diagramu, lub chwilę jego skasowania - na dole diagramu). Przedstawiamy na nim jedynie obiekty biorące udział w realizacji określonego zadania, dla którego był tworzony scenariusz (diagram). Podwójna linia określa okres aktywności obiektu (dla obiektu „Dowódca statku powietrznego" jego aktywności nazwano: „dolot do celu", „tryb wyboru celów", „niszczenie celów"). Przesyłane komunikaty przedstawiane są za pomocą poziomych linii zakończonych strzałką. Strzałka określa kierunek przepływu komunikatów. Opis umieszczony nad strzałką wskazuje metodę obiektu, która zostanie uruchomiona w przypadku otrzymania danego komunikatu.

Na rysunku 4 zastosowano trzy typy komunikatów: proste („Wskazuj cele"), synchroniczne („Określ położenie"), asynchroniczne („Zatwierdź cel misji").

Ograniczenia prezentowane na diagramach dotyczą komunikatów i są prezentowane w nawiasach klamrowych {} (np. {2..9 powtarzaj co 15 ms}, co oznacza: powtarzaj wywołania komunikatów o numerach od 2 do 9 co 15 ms).

Kolejnym typem diagramów interakcji są diagramy współdziałania. Przykładowy scenariusz „ręczne sterowanie niszczeniem celów" dla diagramu współdziałania przedstawiono na rysunku 5.

Rys. 5. Diagram współdziałania: „ręczne sterowanie niszczeniem celów”.



Rys. 6. Diagram stanów : „wybór trybu niszczenia celów”.

Diagramy te przedstawiają interakcje między obiektami, ale w inny sposób niż diagramy sekwencji. Brak jest na tych diagramach linii życia

obiektów, a kolejność wymienianych komunikatów określamy przez etykietowanie ich kolejnymi liczbami. Na diagramach tych bardziej czytelny wydaje się opis obiektów wielokrotnych (na przykład wyświetlacz czy pulpit).

Model dynamiki przedstawiany jest także za pomocą diagramu stanów, reprezentującego maszynę skończenie-stanową. Opisuje on w formie graficznej zbiór stanów, zdarzeń, akcji i aktywności pojedynczej instancji obiektu. Jest to graf zorientowany, którego węzły oznaczają stany, a łuki przejścia pomiędzy stanami. Stany mogą być zagnieżdżane i wówczas są one rozwijane jako poddiagramy. Przykładowy diagram stanów opisujący działanie systemu w czasie „wyboru trybu niszczenia celów" przedstawia rysunek 6. Jak wynika z rysunku 6, niszczenie celów poprzedzone jest zawsze określeniem trybu ich wskazywania dokonanym przez dowódcę samolotu.

Ze względu na poprawność syntaktyczną na diagramie musi być określony jeden stan początkowy i jeden końcowy dla złożonego stanu sekwencyjnego lub po jednym dla każdego złożonego stanu równoległego. Maszyna przechodzi od jednego stanu do kolejnego wykonując akcje (atomowe operacje lub inaczej wykonywalne, niepodzielne czynności, których wynikiem jest zmiana stanu systemu lub dostarczenie mu pewnej mierzalnej wartości).

Akcje są zazwyczaj powodowane przez zdarzenia lub mogą powodować ich wystąpienie (akcje najczęściej mają zerowy czas trwania.). Na diagramie (rysunek 6) wyróżniono następujące typy akcji:

- wejściowe („Wybór trybu niszczenia celów"),
- wewnętrzne („Sprawdzaj stan sterowania trybem niszczenia celów" i „Sprawdź stan środków rażenia"),
- wyjściowe („Wybrano tryb niszczenia" - tu jako zdarzenie).

Specyficzny rodzaj diagramu stanu to diagram aktywności. W diagramach tych większość stanów to stany aktywności, a przejścia między nimi powodowane są zakończeniem wykonywania aktywności. Aktywność to złożona operacja zdefiniowana w obrębie maszyny stanów, której wykonanie wymaga pewnego czasu.

Na rysunku 7 przedstawiono przykład diagramu aktywności: „Wybór trybu niszczenia celów". Przedstawiony diagram interpretujemy jako przepływ sterowania pomiędzy aktywnościami („Wybór niszczenia celów", „Pilot niszczy cele" itd.). Na diagramie aktywności możliwe jest również przedstawienie przepływów danych.

Przepływy sterowania mogą być ze sobą synchronizowane:

- pojedynczy przepływ sterowania może być rozdzielony do kilku współbieżnych stanów,

- kilka przepływów sterowania może być łączonych w jeden przepływ lub mogą wchodzić do jednego, wspólnego stanu.

Rys. 7. Diagram aktywności: „ wybór trybu niszczenia celów".

Dodatkowo na diagramie możemy określić tzw. granice odpowiedzialności wykonawców aktywności: „Pilot", „Urządzenia sterowania trybem", „Automat".

4. Proces projektowania systemu

Opracowane w fazie analizy i modelowania diagramy uzupełnione o rozwiązania, dotyczące architektury przyszłego systemu stanowią punkt wyjścia do fazy projektowania. Celem procesu projektowania jest rozwinięcie modeli z fazy analizy, tak aby spełniały wszystkie wymagania stawiane systemowi oraz były jak najlepiej przystosowane do implementacji systemu.

W fazie projektowania system zostaje podzielony na podsystemy. Na tym etapie trzeba również zdecydować, które obiekty będą realizowane w wersji sprzętowej, a które w wersji programowej. W przypadku pokładowych

systemów lotniczych często zachodzi potrzeba zastosowania specjalizowanych jednostek sprzętowych, co najczęściej podwyższa koszty projektu.

W przypadku rozważanego systemu wbudowanego celowa była dekompozycja całego systemu na trzy podsystemy:

- podsystem wejścia-wyjścia (komunikacja i odbiór danych z urządzeń pokładowych),
- podsystem zobrazowania,
- podsystem sterowania stanowiskami ogniowymi.

Decydując o sposobie realizacji tych podsystemów należy rozważyć przede wszystkim intensywność napływu danych z innych podsystemów pokładowych, wymagane czasy wypracowania sygnałów sterujących liniami stanowisk.

Szczegółowe oszacowania charakterystyk czasowych pokazały, że decydując się na wykorzystanie platformy sprzętowej do zastosowań wbudowanych (płyta procesorowa *Pentium*), niezbędna jest realizacja wydzielonych podsystemów na oddzielnych procesorach.

Zgodnie z procesem wytwórczym RUP, nowymi diagramami opracowywanymi na etapie projektu systemu są diagramy komponentów oraz diagramy rozmieszczenia. Opracowujemy je podobnie jak pozostałe diagramy z wykorzystaniem *Rational Rose*. Na etapie uściślania definicji obiektów celowe jest wiązanie obiektów z plikami opisowymi, zawierającymi szczegółowe implementacje. Postępowanie takie daje nam możliwości automatycznego wygenerowania kodu systemu i szybkiego przetestowania przyjętych rozwiązań, krytycznych dla całego projektu systemu.

5. Podsumowanie

Przedstawiona propozycja wykorzystania RUP do opracowywania oprogramowania systemów wbudowanych została zweryfikowana na przykładzie pokładowego systemu sterowania podwieszeniami samolotu.

W metodyce RUP dzięki oparciu całego modelu systemu na modelu usług i zastosowaniu podejścia obiektowego, zapewnia się jego niezbędną elastyczność, zrozumiałość i przede wszystkim bezpośredni związek z wymaganiami użytkownika.

Główną zaletą użytych narzędzi jest silna integracja definicji projektowych powstałych na etapie analizy z implementacją. Obiekty są rozwijane i uściślane w etapach i iteracjach proponowanego w RUP modelu spiralnego. Każdy obiekt projektowy dostępny jest w we wszystkich narzędziach za pomocą tej samej formatki na każdym etapie realizacji projektu. Narzędzia te ukierunkowane są na generowanie kodu wynikowego. Bardzo silnie wspierany

jest narzędziowo proces testowania, pozwalający, między innymi, na śledzenie interakcji między komponentami aplikacji (ang. *real-time UML sequence diagram tracing*), co ma szczególne znaczenie w systemach wbudowanych.

Niedostatkiem wykorzystywanych narzędzi są, między innymi, trudności w konstruowaniu oraz kontroli ograniczeń czasowych na etapie analizy. Narzędzia wspierające RUP bezpośrednio nie wspomagają kontroli sformułowanych ograniczeń czasowych (na przykład, wyspecyfikowanych na diagramach sekwencji), jednakże wykorzystanie możliwości śledzenia interakcji między takimi komponentami jak metody, funkcje, klasy i moduły daje bardzo silne narzędzie do weryfikacji ograniczeń czasowych.

Literatura:

[1] Huzar, Z.*: Wprowadzenie do języka UML*, IAiR, Warszawa 1999 .

[2] Booch, G.: *Object-Oriented Development*, IEEE Transactions on Software Engineering, February, SE-12, 12 (December 1986), pp. 211-221.

[3] Douglass B.: *Real-Time UML, Developing Efficient Objects for Embedded Systems. Second Edition,* Addison-Wesley 2000.

[4] Hatley, D. J., Pirbhai, I. A.: *Strategies for Real-Time System Specification*, Dorset House, NY, 1987.

[5] Jacobson I.: *Object-Oriented Software Engineering, A Use Case Driven Approach*, Addison-Wesley Publishing Company, New York, NY, 1992.

[6] Kruchten P.: *The Rational Unified Process an Introduction Second Edition,* Addison Wesley Longman, Inc. 2000.

[7] Marshall C.*: Enterprise Modeling with UML. Designing Successful Software Through Business Analysis*, Addison Wesley Longman, Inc. 2000.

[8] Rumaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W.: *Object-Oriented Modeling and Design*, Prentice Hall, Englewood Cliffs, NJ., 1991.

[9] Rumaugh J.: *Using Use Case to Capture Requirements*, Journal of Object-Oriented Programming, September 1994.

[10] *Rational Unified Process – Getting Started*. Rational Software Corporation, 1999.

[11] *Rational Rose RealTime Toolset Guide*. Rational Software Corporation, November 2000

Recenzent: dr inż. Krzysztof Liderman
Praca wpłynęła do redakcji: 10.10.2001