Mgr inż. Jacek Reiner, Mgr inż. Mariusz Mrzygłód,
Prof. zw. dr hab. inż. Jan Koch, Dr inż. Zbigniew Smalec
Politechnika Wrocławska
Instytut Technologii Maszyn i Automatyzacji

OBIEKTOWO ZORIENTOWANE MODELOWANIE STEROWANIA DLA ZATOKI MONTAŻOWEJ W ŚRODOWISKU RHAPSODY

W nowoczesnych systemach sterowania, coraz większy udział odgrywa oprogramowanie, któremu jednak stawia się znacznie bardziej rygorystyczne wymagania jak informatyce biurowej. Skomplikowane systemy sterowania porównywalne są obecnie do dużych organizmów biologicznych, jednak metody inżynierskie stosowane przez automatyków są przestarzałe i niewystarczające dla rozwiązywania takich problemów. Stąd konieczność nowego paradygmatu dla opracowywania systemów sterowania.
Poniższy artykuł przybliża język modelowania obiektowego UML zwracając uwagę na jego przydatność dla opracowywania systemów sterowania. Artykuł prezentuje ponadto przykład sterowania zatoką montażową, opracowany w środowisku modelowania UML –Rhapsody.

OBJECT ORIENTED MODELLING OF CONTROL SYSTEM FOR ASSEMBLY STATION IN RHAPSODY ENVIRONMENT

Nowadays the role of real time control systems (RT-CS) software is significantly increasing. At the same time, the requirements for CS are much more rigorous than for office software applications. As the complexity of control systems is even comparable to that large biological organism, methods used by control engineers do not suffice for such problems. And so the need for a new paradigm for developing RT-CS becomes apparent.
The article brings closer Object Oriented Approach with Unified Modeling Language with a special consideration for automation purposes. Additionally the paper presents, an example of control systems for assembly station, which is realized in the Rhapsody environment.

1. POTRZEBA NOWEGO PARADYGMATU W AUTOMATYZACJI

Podstawy wytwarzania nie zmieniły się zasadniczo od ostatniej rewolucji przemysłowej. Dążność do wyższej wydajności, jakości oraz ograniczenia zasobów i kosztów wytwarzania to główne czynniki ciągłego wprowadzania zmian w metodach oraz narzędziach produkcji. Motywacją do tego jest ekonomia, zaś katalizatorem nowe technologie czyniące pracę łatwiejszą, tańszą i szybszą.
Szczególnie dynamiczny rozwój obserwujemy w zakresie technologii sterowania produkcją. Mówi się nawet o kolejnej rewolucji w tej dziedzinie przyrównując ją do postępu wprowadzonego przez sterowniki swobodnie programowalne PLC. Idea komputerowo zintegrowanego przedsiębiorstwa CIM *(ang. Computer Integrated Manufacturing)* zaniedbywała podział na dwa odmienne obszary: przepływu i przetwarzania danych –

informatyki biurowej, oraz automatyki i sterowania procesem. Przy wprowadzaniu koncepcji CIM okazało się, że wprawdzie potrafimy fizycznie połączyć oba obszary, ale kompleksowość i logiczne odmienności obu obszarów nie pozwalają na realizację idei łatwej integracji.

Automatyka zdeterminowana bezpieczeństwem, niezawodnością i wymaganiami zachowania czasu rzeczywistego procesu oraz ograniczonymi zasobami skierowała się ku rozwiązaniom deterministycznym. Systemy synchroniczne pozwoliły się łatwo modelować, weryfikować i wdrażać. Przy wzroście kompleksowości okazało się jednak, że skala problemów nie rośnie liniowo. Wysoce deterministyczne systemy w przypadku pojedynczych zakłóceń załamują się. Wydajność systemów wytwarzania np. elektroniki spada do 30% [1]. Nie jest to wynikiem przestojów maszyn, powodowanych uszkodzeniami, lecz wąskimi gardłami powstającymi w wyniku ostrożnego planowania, braków materiałów oraz błędami ludzi.

Sytuacja taka jest zupełnie zrozumiała intuicyjnie. Świat nie jest sterowany nadrzędnym zegarem, lecz zdarzeniami pojawiającymi się zupełnie niezależnie i często w najmniej odpowiednich momentach. Do dobrego opisu świata rzeczywistego potrzebny jest, zatem nowy paradygmat pojęciowy oraz modelowanie jako abstrakcyjna forma rozumienia i rozwiązania problemu.

Aktualnie najodpowiedniejszy wydaje się paradygmat obiektowy znany z informatyki. Jako podstawę definiuje on obiekt – abstrakcyjną formę przedstawienia otaczającego świata. Obiekty jako byty niezależne, znają swe przeznaczenie, potrafią reagować na przychodzące do nich komunikaty (wejścia), wykonują swoje funkcje (algorytm, metody) i wysyłają komunikaty do innych obiektów (wyjścia). Ich aktywność jest niezależna i ożywiana przez zdarzenia, których interpretacja nie zależy od źródła pochodzenia. Prostota takiej koncepcji daje jej niebywałą siłę. Szczególna łatwość modyfikowania, rozszerzania i skalowania systemów budowanych według takich założeń zapewnia ich elastyczność. Dla systemów sterowania szczególnie istotna jest możliwość rozproszenia obiektów oraz osadzenia na różnych zasobach.

Pojęcie „obiekt" jest niestety często nadużywane przez „twórców" oprogramowania chcących podnieść w ten sposób rangę swoich rozwiązań. Wielu z nich mówi o podejściu obiektowym opisując interfejs użytkownika zbudowany na bazie graficznych okien. Podejście obiektowe stosowane w informatyce jest znacznie szerzej rozumiane, prócz abstrakcyjności i kapsułowania *(ang. encapsulation)* ważną rolę odgrywa dziedziczenie i polimorfizm.

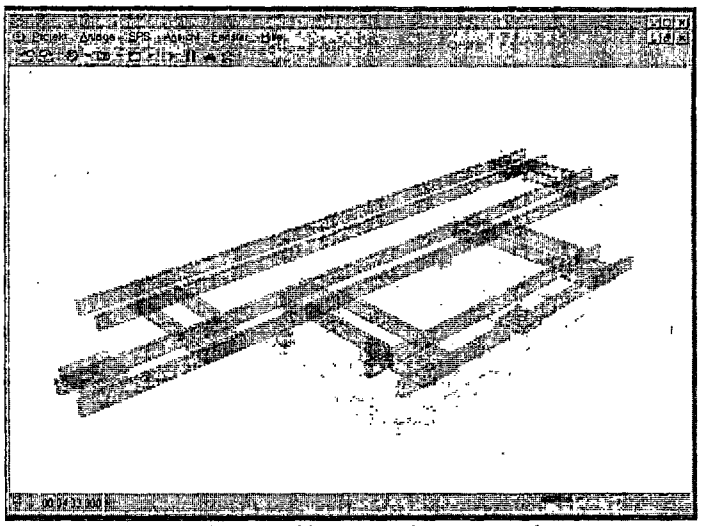
2. PARADYGMAT OBIEKTOWY

Paradygmat obiektowy jako podstawę definiuje obiekt będący programistycznym odwzorowaniem rzeczywistości (np.: obrabiarka, robot, część, zamówienie). Każdy z obiektów cechują dwa rodzaje właściwości: umiejętności bądź operacje (np.: włącz, wyłącz, zamknij chwytak), których implementacje nazywa się metodami, oraz atrybuty, czyli dane (np. liczba osi obrabiarki, maksymalne przyspieszenie w danej osi). Dane i algorytmy działania na nich zostały powiązane i osłonięte od otoczenia. Dostęp do danych odbywa się poprzez interfejs operacji. Dzięki temu, używanie obiektów staje się bezpieczne, a świat zewnętrzny kierując komunikaty nie musi zwracać uwagi na sposób ich realizacji. Obiekty o podobnych cechach (interfejsach) tworzą klasy. Obiekty są zatem instancjami (wywołaniami) bądź realizacjami klas.

Dziedziczenie jest szczególnie mocnym narzędziem. To w świecie rzeczywistym większość nowych rozwiązań powstaje na bazie już istniejących poprzez ich rozbudowę i modyfikację. Dziedziczenie pozwala przejąć operacje oraz atrybuty (definicje atrybutów a nie ich wartości) klasy nadrzędnej (np.: obrabiarka jest klasą nadrzędną, z której dziedziczą, tokarka i frezarka). Dzięki dziedziczeniu i wspólnemu przodkowi możliwe jest różne rozumienie

jednakowych komunikatów - stosownie do funkcji obiektu. Tak rozumianą wielopostaciowość obiektów określamy polimorfizmem.

3. SYSTEMY STEROWANIA I WYMAGANIA CZASU RZECZYWISTEGO



Rys.1 Model 3D zatoki montażowej wraz z trasą główną

Mimo tego, że komputery PC coraz intensywniej wypierają klasyczne, sekwencyjne, logiczne sterowniki programowalne PLC, to zadania stawiane im w obszarze biura i procesu produkcyjnego są wyraźnie różne. Najistotniejsza cecha systemów sterowania to czas-rzeczywisty *(Real-Time)*. Mówiąc obrazowo, dla systemów o krytycznych wymaganiach czasowych: *spóźniona odpowiedź jest błędną odpowiedzią.* Szersze rozumienie pojęcia RT, prócz wymagań czasowych, podkreśla niezawodność i bezpieczeństwo systemów. Okazuje się, że wzrost wydajności i zasobów komputerów PC wprawdzie upraszczają spełnienie wymagań czasowych, jednak wiążą się z wzrostem złożoności i rozbudowy oprogramowania, co jest następstwem rosnących wymagań funkcjonalnych dla systemów sterowania. Cechą różniącą aplikacje biurowe od sterujących, jest również rozproszenie instalacji technologicznych. Powoduje to konieczność umieszczania poszczególnych zadań sterowania na różnych platformach. Integracja różnych pakietów aplikacji biurowych typowo realizowana jest poprzez bazę danych, gdy w przypadku systemów sterowania konieczna jest platforma komunikacyjna. Szeroko dostępne sieci komunikacyjne Fieldbus [4] nie spełniają jednak takiej roli, gdyż stanowią interfejs danych, a nie funkcji czy operacji.
Użycie najnowszych technologii elektroniki i informatyki dla systemów RT wymaga jednak nowych metod ich stosowania. Wyrafinowane metodyki zostaną zaakceptowane zaś jedynie wtedy, gdy będą wsparte przez odpowiednie narzędzia w postaci oprogramowania CASE.

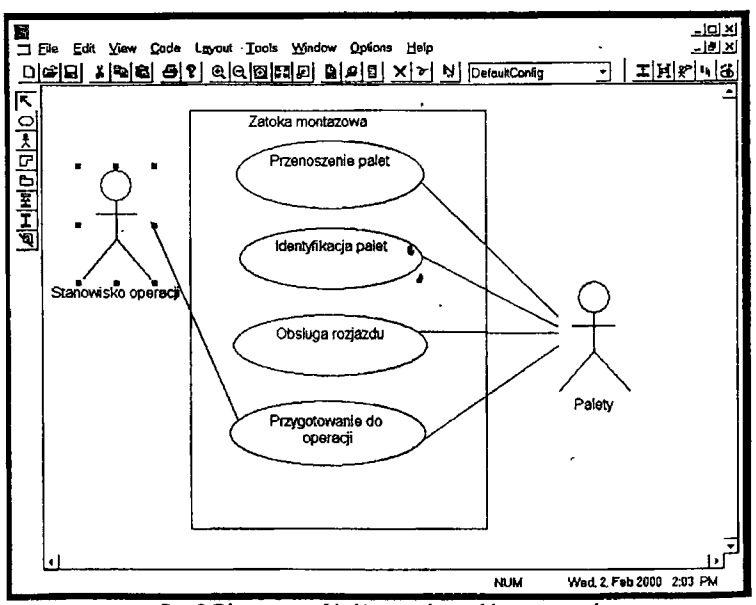
4. ZASTOSOWANIE PODEJŚCIA OBIEKTOWO-ZORIENTOWANEGO DLA MODELOWANIA STEROWANIA

Abstrakcja stanowi podstawę radzenia sobie z kompleksowością. Dzięki niej potrafimy lepiej rozumieć problem i rozwiązywać go na określonym poziomie szczegółowości. Do zapisu abstrakcyjności nieoceniony jest model, który wyrażamy poprzez różne diagramy.

Standardem modelowania obiektowego, zyskującym obecnie największą akceptację jest UML *(Unified Modeling Language)*. Stanowi on wypracowany kompromis między trzema przodującymi podejściami: OMT *(Object Modelling Technique)* Rumbaugh, OOSE *(Object Oriented Software Engineering)* Jacobson oraz Booch. Język UML jest zestawem kilku różnych technik diagramowania: przypadków użycia *(use cases)*, scenariuszy *(sequence and colaboration diagrams)*, modelu obiektowego *(object modeling diagram)*, diagramu stanów *(state charts)*, komponentów oraz rozkładu. Dzięki swojej spójności umożliwia on kompletne przedstawienie problemu zgodnie z paradygmatem obiektowym.

1.1. Analiza wymagań

Pierwszym etapem opracowania nowego rozwiązania jest szczegółowe zapoznanie się z dziedziną problemu, wymaganiami stawianymi przez proces technologiczny oraz użytkownika. Faza analizy wymagań jest często skracana a jej znaczenie doceniane zbyt późno podczas problemów w fazie integracji, uruchomienia i walidacji projektu. Ponieważ wyniki analizy stanowią podstawę projektu, którego realizacja może przybrać różne rozwiązania, koszty błędów lub zmian rosną bardzo znacznie wraz z postępem prac. Trudność fazy analizy polega na znalezieniu wspólnego języka między technologami znającymi doskonale proces technologiczny a automatykami, elektronikami i informatykami współpracującymi przy opracowywaniu sterowania i wizualizacji. Każdy z nich używa innego języka specjalistycznego i posiada różne doświadczenie. Specjaliści często kierują dyskusje do problemów szczegółowych, tzw. symptom „bitów i bajtów". Dla ułatwienia pracy w początkowej fazie projektu język UML definiuje bardzo prosty, intuicyjnie zrozumiały diagram przypadków zastosowania *(ang. use-case)*. Diagram ten przedstawia podstawowe funkcje systemu, ważne zależności funkcjonalne oraz otoczenie sytemu, czyli ludzi i urządzenia zewnętrzne*(ang. actors)* (Rys. 2).

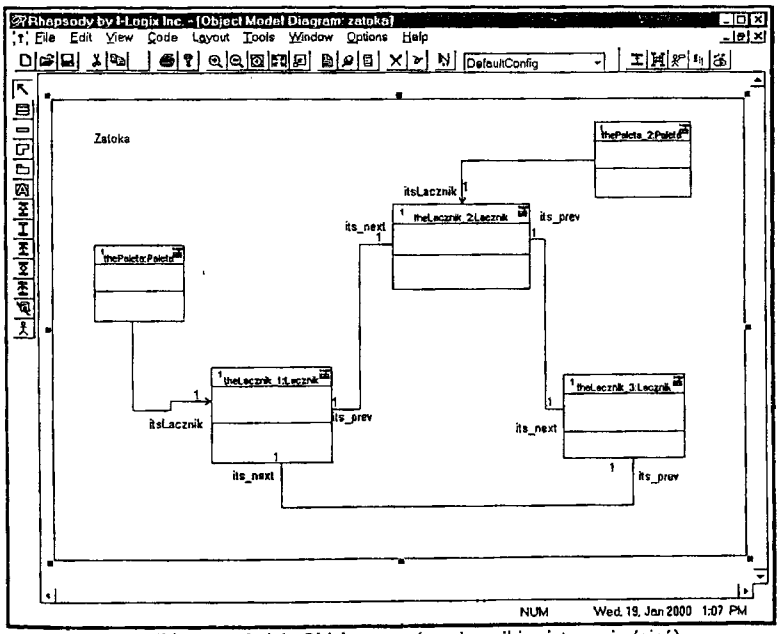


Rys.2 Diagram przykładów użycia zatoki montażowej

Taki model przybliża jedynie statyczny obraz wymagań, dlatego jest on rozszerzany o diagram scenariuszy łatwych do sformułowania przy opisie problemu. W UML scenariusze zapisuje się za pomocą diagramu sekwencji *(sequence diagram).* Pionowe linie przedstawiają obiekty, a strzałki komunikaty wymieniane między nimi w funkcji czasu (jak na Rys.5). Opisanie interakcji w systemie musi oczywiście ograniczyć się jedynie do najistotniejszych komunikacji z otoczeniem i wewnątrz systemu. Diagramy sekwencji pozwalają podkreślić wymagania nie tylko jakościowe, ale również ilościowe (np.: czas odpowiedzi na daną reakcję nie może być dłuższy niż 25ms).

1.2. Projekt

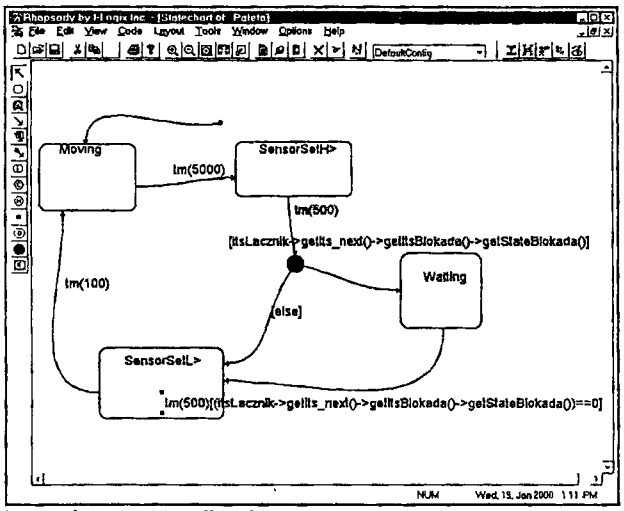
Kolejnym zadaniem pośrednim z zakresu analizy i projektowania jest identyfikacja obiektów, czyli składowych systemu. Niestety nie wszystkie byty są oczywiste, często występują różne postaci obiektów, które trzeba umiejętnie określić w zakresie odpowiedzialności, przechowywanych danych i reakcji na zdarzenia systemowe. Istnieje wiele sposobów ułatwiających identyfikację obiektów, a najprostsze z nich proponują wyłowienie rzeczowników z opisu problemu, a później uporządkowanie ich funkcjonalności np.: metodą kartkową [2].



Rys.3 Diagram Modelu Obiektowego (trzy łączniki spięte w pierścień)

Podczas identyfikacji obiektów w nowym systemie należy przejść na wyższy poziom abstrakcji i zastąpić identyfikowane obiekty klasami, a następnie poszukać zależności hierarchicznych dziedziczenia między nimi. Kolejnym etapem projektowania systemu będzie pokazanie relacji między poszczególnymi klasami. Podstawowa forma relacji to asocjacja umożliwiająca wymianę komunikatów między poszczególnymi obiektami. Gdy chcemy pokazać związek całość-część (np.: sensor i blokada są częścią łącznika), to stosujemy kompozycję lub agregację. W takim przypadku obiekt bazowy jest odpowiedzialny za tworzenie i usuwanie obiektów będących jego częścią.

Diagram Modelu Obiektowego (Rys. 3) przedstawia architekturę systemu, natomiast dla przedstawienia interakcji między obiektami, stosowany jest diagram sekwencji lub współpracy *(collaboration diagram)*.
Dla zobrazowania zachowania poszczególnych obiektów UML definiuje diagram sekwencji *(sequence chart)*. Jest on opracowany na bazie modelu automatu stanów, który został rozszerzony o współbieżność, pamięć, warunki początkowe i końcowe realizacji przejścia między stanami. Istnieje naturalnie możliwość implementacji metod obiektu w językach wysokiego poziomu (np.: C, C++, Java).



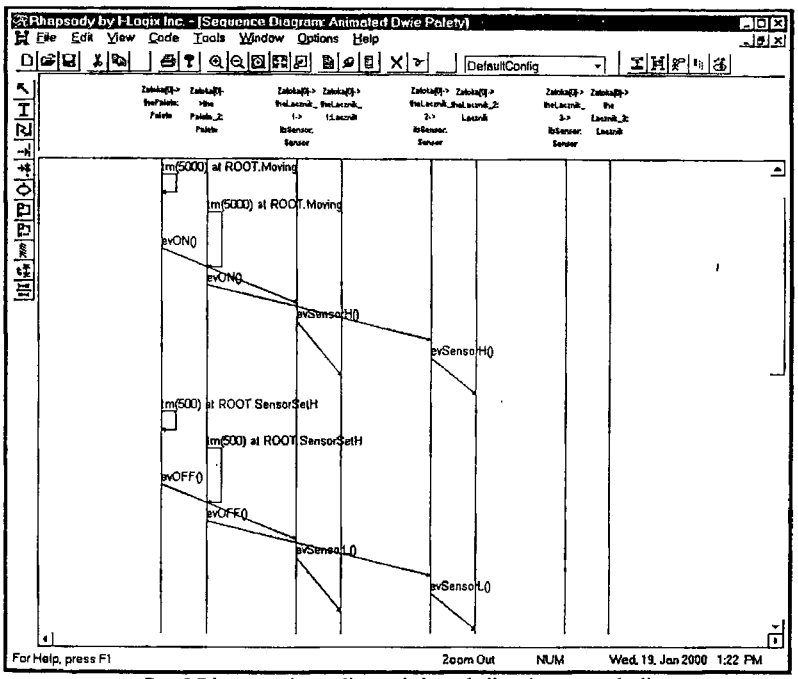
Rys.4 Diagram stanów dla palety

Dla modeli predyktywnych niewystarczające jest modelowanie tylko struktury i zachowania, a konieczne jest pokazanie także logicznych i fizycznych zasobów oraz infrastruktury inżynierskiej w ramach, której to oprogramowanie będzie działało. Wymaganie to szczególnie wyróżnia modelowanie systemów czasu rzeczywistego od innych systemów informatycznych. Infrastruktura inżynierska składa się z fizycznych urządzeń takich jak: procesory, sieci komunikacyjne oraz logicznych „urządzeń" jak procesy równoległe (*taski*), kolejki, semafory. Charakteryzują się one fizyczną wydajnością i ulokowaniem. Jedynie przez umiejętne ich wykorzystanie możliwe jest zapewnienie wszystkich wymagań systemów. W tym celu UML wprowadza diagram implementacji.

1.3. Weryfikacja modelu

Wraz ze wzrostem złożoności modelu zwiększa się trudność jego weryfikacji. Dlatego od narzędzi – oprogramowania wspomagającego modelowanie oczekujemy nie tylko pomocy przy szkicowaniu diagramów, lecz szczególnie mechanizmów ułatwiających weryfikację. Typowo narzędzia modelowania UML (Rhapsody, Rational Rose) generują kod pośredni w postaci kodu języka wysokiego poziomu jak C++ czy Java. Dołączane są również biblioteki systemowe tworzące platformę programistyczną dla zaimplementowanych obiektów. W takiej sytuacji do testowania wykorzystujemy mechanizmy pakietów kompilatorów (*debugger*). Pakiet Rhapsody umożliwia ponadto testowanie działania modelu na poziomie graficznym. Zmiana stanów poszczególnych obiektów oraz śledzenie interakcji obrazowane są w postaci animacji. Użytkownik ma możliwość generowania dowolnych zdarzeń w systemie i analizy odpowiedzi.
Dla złożonych systemów, gdy analizie chcemy poddać zachowanie w przypadku większej liczby zdarzeń współbieżnych, niezależnych, konieczne jest rozbudowanie modelu sterowania o elementy dynamiczne procesu. Mimo, że modelowanie obiektu wiąże się z dużym dodatkowym nakładem prac, to dzięki temu możliwe jest kompletne przetestowanie działania sterowania wraz z symulowanym obiektem. Taka sposobność jest szczególnie cenna, gdyż

kosztowne instalacje technologiczne nie są zwykle gotowe w czasie powstawania sterowania. Przy modelowaniu urządzeń technologicznych, najbardziej efektywne są wirtualne modele 3D. Model 3D zatoki montażowej opracowany w środowisku oprogramowania TrySim przedstawia Rys.1. Model ten składa się z 60 obiektów, których elementy wykonawcze „ożywiane są" poprzez sygnały ze sterowania. Takie podejście pozwala wcześnie uchwycić problemy styku mechanika-sterowanie. Niestety dostępne obecnie pakiety oprogramowania nie pozwalają jeszcze na integrację modelu sterowania i modelu instalacji technologicznej.



Rys.5 Diagram sekwencji – zapis interakcji podczas symulacji

5. STEROWANIE ZATOKĄ MONTAŻOWĄ OPRACOWANE W ŚRODOWISKU RHAPSODY

System transportowy palet gniazda montażowego z Laboratorium Systemów Produkcyjnych ITMiA PWr posłużył jako obiekt doświadczalny. Z uwagi na jego rozbudowę, dzięki podobieństwu i modułowej konstrukcji zdecydowano się ograniczyć zakres prac do jednej zatoki. Celem sytemu jest *bezkolizyjne przenoszenie palet montażowych zgodnie z ich różnymi marszrutami technologicznymi.* Aktualnie do sterowania systemem wykorzystywany jest Rozproszony System Sterowania zrealizowany na bazie siedmiu sterowników PLC (S7-215). Podstawę komunikacji stanowi sieć PROFIBUS wraz ze stacją master na bazie komputera PC. Więcej szczegółów dotyczących zastosowanego rozwiązania sterowania, zawarto w artykule [3].

Dla oceny przydatności podejścia obiektowo-zorientowanego dla opracowania sterowania zatoki montażowej systemu transportowego palet oraz oceny środowiska Rhapsody, w pierwszym kroku określono funkcje zatoki jako diagram przypadków użycia (Rys.2). Najistotniejsze interakcje udokumentowano w postaci scenariuszy oraz diagramów sekwencji. Następnie zidentyfikowano obiekty systemu oraz ich uogólnienia w postaci klas: *Łącznik* -

element przenoszący palety; *Rozjazd* - skrzyżowanie wjazdowe do zatoki; *Koncentrator* - skrzyżowanie wyjazdowe z zatoki; *Blokada*; *Sensor*; *Podnośnik*; *Paleta*; *Skaner*; *Identyfikator*.

W kolejnym etapie, po ustaleniu relacji miedzy poszczególnymi klasami, zdefiniowano działanie każdej z nich, w postaci diagramu stanów (patrz Rys.4). Po uzupełnieniu klas o definicje atrybutów, zdefiniowano model obiektowy sterowania wskazując poszczególne instancje klas. Tak przygotowany model poddano weryfikacji symulacyjnej wysyłając pojedyncze zdarzenia do obiektów. W celu kompleksowego przetestowania opracowanego systemu sterowania zamodelowano element dynamiczny – paletę, definiując jej połączenie z sensorem i blokadą. Rysunek 4 prezentuje diagram stanów dla palety. Tak rozszerzony system poddano następnie symulacji. Rysunek 5 przedstawia zapis interakcji *Paleta – Sensor – Łącznik*. Na podstawie analizy przekazywanych zdarzeń widzimy ruch palet.

6. PODSUMOWANIE

Oprogramowanie ma coraz większy udział w nowoczesnych systemach sterowania. Aby zapewnić ich wysoką jakość, „produkcja" musi przebiegać w sposób systematyczny i sterowany. Główna zaleta oprogramowania, jaką jest jego elastyczność, pozostaje jednak w opozycji do potrzeby łatwego zestawiania systemów sterowania z „klocków". Wyniki prowadzonych prac w zakresie wykorzystania podejścia obiektowo-zorientowanego UML dla opracowywania systemów sterowania wytwarzaniem są bardzo pozytywne. Niestety błędy oprogramowania wspomagającego modelowanie oraz literatura opisująca jedynie teoretyczne podstawy spowalniają prowadzone badania. Zdobyte doświadczenia wskazują na szczególny wzrost efektywności tej metody wraz ze wzrostem kompleksowości projektu. Dla małych projektów, nakład jest znacznie większy niż przy podejściu klasycznym. Ciągły brak efektywnych narzędzi wspomagających, przykładów zastosowań, zbyt duże możliwości i elastyczność podejścia obiektowego utrudniają jednak przeniesienie tego doskonałego pomysłu do praktyki. Dlatego też zdefiniowane zostało rozszerzenie języka UML dla zastosowań czasu rzeczywistego RT. W rzeczywistości UML-RT jest zorientowany na inżynierów a nie informatyków. Na podobieństwo obiektów definiuje on kapsuły, zezwalając na dostęp do nich jedynie poprzez porty. Jedyną formą komunikacji jest telegram. Takie ograniczenia wpływają negatywnie na elastyczność, podnoszą jednak bezpieczeństwo i niezawodność systemu.

W ramach dalszych prac, zrealizowane zostanie połączenie opracowanego sterowania z obiektem rzeczywistym – zatoką montażową. Komunikacja na bazie PROFIBUS zostanie zrealizowana poprzez kartę PROFIboard. Aktualnie zainstalowane sterowniki PLC w rozproszonym systemie sterowania gniazdem, przekonfigurowane zostaną do funkcji rozproszonych peryferiów. Opracowany model . obiektowy sterowania systemem transportowym palet jest ciągle rozwijany.

LITERATURA

[1] Smith David., *The four industrial revolution: Objects replace synchronicity*, Control software Forum 08 1999 s. 59-60.

[2] Douglass B.P., *Doing Real Time; Developing Real-Time Systems with UML, Objects, Frameworks and Patterns*, Addison-Wesley, 1999

[3] Reiner J., Koch J., *Rozproszone sterowanie dla gniazd wytwórczych*, PAN KBM, Prace Naukowe Instytutu Technologii Maszyn i Automatyzacji Politechniki Wrocławskiej Nr 71, 1998, s. 161-168

[4] Koch J., Smalec Z., Reiner J., Skura K., *Communication systems in manufacturing and their role in automation*, Human Systems Management, 18/1999, s. 253-263