CYKL TWORZENIA OPROGRAMOWANIA NA PRZYKŁADZIE PROJEKTU SOFIA

Anna Gałach
Instytut Lotnictwa

Streszczenie

Artykuł przedstawia fazy tworzenia oprogramowania w dużym projekcie lotniczym, realizowanym przez wiele współpracujących ze sobą firm europejskich. W pracy zostały poruszone kwestie procesu tworzenia planu projektu, jego realizacji sprzętowej i programowej oraz jego integracji i testowania na różnych platformach testowych. Omówiono sposób wyboru narzędzi i platform sprzętowych i programistycznych wykorzystywanych w projekcie, uzgodnienie protokołów komunikacyjnych i proces integrowania stworzonego systemu na docelowych urządzeniach w celu przetestowania. Proces realizacji projektu przedstawiono w oparciu o doświadczenia zdobyte podczas pracy przy projekcie FP6 SOFIA (Safe Automatic Flight Back and Landing of Aircraft), który był tworzony w latach 2006-2009.

1. WSTĘP

Tworzenie oprogramowania, szczególnie w wypadku dużych systemów informatycznych, w których bierze udział wiele firm nie mających doświadczeń we współpracy ze sobą, jest procesem skomplikowanym, który potrafi zmieniać się dynamicznie wraz z rozwojem projektu, często niezależnie od wstępnych ustaleń. W przypadku, gdy każda firma zajmuje się inną częścią oprogramowania, trudno jest znaleźć rozwiązania programistyczne zadowalające wszystkich uczestników projektu, ponieważ każda firma ma swoje preferencje i/lub ograniczenia sprzętowe i programowe. Stworzenie lotniczego systemu informatycznego narzuca dodatkowe wymagania na spójność i niezawodność oprogramowania, gdyż w grę wchodzi zagrożenie życia. Te i inne zagadnienie tworzenia oprogramowania zostały przedyskutowane w artykule w oparciu o doświadczenia zdobyte podczas pracy w projekcie europejskim SOFIA.

2. ZAŁOŻENIA I CELE PROJEKTU SOFIA

Celem projektu SOFIA było opracowanie koncepcji i algorytmów pozwalających na automatyczny powrót samolotu na ziemię w przypadku pojawienia się niebezpieczeństwa ataku terrorystycznego na pokładzie. Głównym celem pracy było zaprojektowanie funkcji rekonfiguracji lotu FRF (Flight Reconfiguration Function) mających za zadanie: stworzenie bezpiecznego planu lotu do bezpiecznego lotniska, uzgodnienie dostępu do niego z władzami na ziemi oraz automatyczne doprowadzenie samolotu do wyznaczonego miejsca lądowania bez udziału załogi w celu uniemożliwienia porywaczom przejęcia kontroli nas samolotem. System FRF zapewniał zatem:

- wyznaczenie najbliższego autoryzowanego lotniska,
- obliczenie nowego planu lotu,
- negocjacje planu lotu ze stacją naziemną lotniska,
- kontrolę wszystkich systemów awionicznych związanych z lotem i lądowaniem samolotu.

Stworzony system rekonfiguracji lotu miał zostać przebadany podczas prób na symulatorach lotu i ruchu lotniczego oraz podczas prób w locie.

W projekcie wzięło udział dziewięć firm z całej Europy: ISDEFE z Hiszpanii, będące koordynatorem projektu, Alenia SIA i GALILEO Avionica z Włoch, SKYSOFT z Portugalii, THALES Avionics z Francji, Deutsche Flugsicherung i Rheinmetal Deffence Electronics z Niemiec, Diamond Aircraft Industry z Austrii i Instytut Lotnictwa.

Za stworzenie oprogramowania realizującego system rekonfiguracji lotu odpowiedzialne były dwie firmy: Alenia SIA i SKYSOFT. Stworzone oprogramowanie miało zostać zintegrowane na dostępnych platformach testowych w celu sprawdzenia poprawności jego działania. Do walidacji systemu w projekcie SOFIA zostało wykorzystane 5 platform:

- Symulator lotów ATENA opracowany przez GALILEO Avionica,
- Symulator lotów AIRLABTM opracowany przez THALES Avionics,
- Symulator naziemnej stacji kontroli ruchu lotniczego firmy DFS,
- Samolot I-23 Instytutu Lotnictwa,
- Symulator lotu i samolot Twin Star DA42 firmy Diamond Aircraft Industry.

Ze względu na to, że docelowe urządzenie miało być wykorzystywane na rzeczywistych samolotach, a podczas tworzenia projektu miały zostać przeprowadzone próby w locie, system FRF, poza określoną w założeniach funkcjonalnością, musiał spełniać następujące warunki:

- oprogramowanie systemu powinno być niezawodne i bezpieczne,
- algorytmy powinny być wykonywane w odpowiednich ramach czasowych i uniemożliwiać wystąpienie stanów nieokreślonych i błędów,
- system powinien działać wydajnie i efektywnie wykorzystywać posiadane zasoby.

Stworzone oprogramowanie musiało również:

- być łatwe do zaadaptowania na dostępnych platformach programistycznych i sprzętowych wszystkich firm w celu szczegółowego przetestowania na poprawność działania,
- umożliwiać szybkie wprowadzania ewentualnych zmian po wykryciu błędów lub braków,
- mieć uzgodnione i opracowane niezawodne protokoły komunikacji i przekazywania danych między modułami poszczególnych firm przy założeniu, że różne firmy wykorzystują różne platformy i środowiska programistyczne i że większość firm ze względu na politykę prywatności nie będzie mogła udostępniać kodu pozostałym uczestnikom projektu,
- mieć przygotowane jednolite bazy danych (lotnisk, pasów startowych, obszarów zabronionych i przeszkód, rzeźby terenu i danych o samolocie) i ustalony sposób dostępu do nich.

3. PROCES TWORZENIA OPROGRAMOWANIA

Proces produkcji oprogramowania składa się z kilku faz. Ich definicja, porządek, interakcje między poszczególnymi fazami specyfikują tzw. model cyklu życia oprogramowania. Powstało wiele różnych modeli cyklu życia oprogramowania, jednak we wszystkich można wyróżnić fazy: definicji wymagań, projektowania, implementacji, testowania i pielęgnacji. Często zdarza się sytuacja, w której wyjście z jednej fazy tworzenia oprogramowania jest wejściem do następnej. Podczas fazy definicji wymagań jest określane co system ma robić i w jakich warunkach i ograniczeniach ma działać. Typowa definicja wymagań zawiera: przegląd dostępnych produktów, specyfikację rozwoju, działania i pielęgnacji środowiska działania produktu, wysokopoziomowy model koncepcyjny systemu, specyfikację interfejsu użytkownika, specyfikację wymagań funkcjonalnych (definicje zachowań i funkcji systemu) i niefunkcjonalnych (kryteria, które mogą zostać wykorzystane do oceny działania całego systemu), specyfikację

interfejsów do innych systemów, specyfikację obsługi błędów, listę możliwych zmian i poprawek systemu.

W fazie projektowania jest tworzony plan realizacji wymagań systemu. Plan jest tworzony używając odpowiednich metod projektowania i oznaczeń, skupiając się kolejno na poszczególnych aspektach systemu.

Faza implementacji skupia się na tworzeniu kodu, według stworzonego wcześniej planu systemu. Kod jest zapisywany w języku formalnym nazywanym językiem programowania. W dzisiejszych czasach większość kodu jest tworzona w językach wysokiego poziomu jak C/C++ czy Java. W przypadku języków wysokiego poziomu kod jest zamieniany na język asemblera i maszynowy przy pomocy kompilatora. Opierając się na zasadzie modularności oprogramowania, kod w dużych systemach jest dzielony na moduły, które przypisywane są programistom lub grupom programistów.

Testowanie jest procesem egzaminowania oprogramowania w celu znalezienia w nim błędów i braków. Jest niezbędne nie tylko w celu przetestowania samego kodu ale również produktów całego cyklu życia oprogramowania i dokumentacji. Proces testowania jest najczęściej dzielony na fazy. Pierwszą fazą jest testowania małych jednostek oprogramowania przez programistów. Drugą fazą jest testowanie integracyjne, gdzie jednostki są łączone i testowane w grupach. Testowanie całego systemu odbywa się przy pomocy przypadków testowych (test case), czyli zestawu danych testowych, zmiennych, warunków wykonywania i oczekiwanych rezultatów opracowanych w konkretnym przypadku użycia systemu. Przypadki testowe są tworzone na podstawie założeń systemu. Testowanie akceptujące system jest najczęściej przeprowadzane na platformach docelowych systemu.

Ostatnią fazą procesu tworzenia oprogramowania jest faza pielęgnacji. Duże systemy nie są statyczne, są często modyfikowane podczas tworzenia jak również po jego ukończeniu. Pielęgnacja jest fazą życia oprogramowania następującą po stworzeniu produktu. Składają się na nią trzy rodzaje działań: adaptacja, korekcja i ulepszanie. Ulepszanie jest procesem polegającym na dodawaniu nowych funkcjonalności do systemu i odbywa się najczęściej na życzenie użytkowników systemu. Adaptacja jest procesem polegającym na zmianach systemu w celu adaptacji do nowego środowiska czy platformy. Korekcja jest procesem naprawiania błędów po wypuszczeniu systemu na rynek.

Informacje z poszczególnych faz tworzenia oprogramowania są zawsze odpowiednio dokumentowane.

W projekcie SOFIA praca nad stworzeniem systemu FRF została podzielona na 5 etapów. Pierwszy obejmował zadania operacyjne systemu, drugi – tworzenie projektu systemu, trzeci – tworzenie systemu, czwarty – testy i walidację systemu, piąty poruszał kwestie rozpowszechniania i eksploatacji systemu.

4. PROCES PROJEKTOWANIA I TWORZENIA OPROGRAMOWANIA W PROJEKCIE SOFIA

Pierwszy etap pracy w projekcie SOFIA obejmował określenie podstawowych definicji systemu FRF. Wstępnie określone funkcje zostały odniesione do środowiska pracy systemu – samolotu znajdującego się w przestrzeni powietrznej i naziemnych stacji kontroli lotu. W odniesieniu do środowiska przestrzeni powietrznej zostały przeanalizowane dostępne technologie i systemy lotnicze, z którymi system FRF musiałby współpracować. Zostały scharakteryzowane niezbędne procedury związane z automatycznym lotem samolotu i zmianami jakie ze względu na nie należałoby wprowadzić do obecnie obowiązujących procedur. Środowisko naziemne zostało przeanalizowane ze względu na technologie i procedury kontroli ruchu lotniczego związane z działaniem systemu. Zostały określone efekty wprowadzenia zmian w procedurach lotniczych ze względu na nowy system - kwestie zasad i przepisów, które muszą zostać zmienione, żeby procedury FRFa i samolot z systemem FRF został dopuszczony do lotu.

Oszacowano również poziom bezpieczeństwa systemu i stworzono zbiór wskazań dla specyfikacji i architektury systemu i jego podsystemów w celu zminimalizowania ryzyka kolizji z terenem, przeszkodami i bezpiecznego doprowadzenia samolotu na ziemię do wyznaczonego lotniska.

Po dokonaniu definicji wymagań rozpoczęto etap projektowania systemu. Została stworzona specyfikacja funkcji i ich interfejsów z systemem awionicznym samolotu. Określono:

- sposób inicjalizacji FRFa,
- zbiór stanów w jakich może się znajdować system,
- sposób zarządzania funkcjami,
- definicję interfejsów z urządzeniami zewnętrznymi,
- specyfikację wykorzystywanych baz danych i definicję interfejsów do nich,
- definicję interfejsów z FMSem/autopilotem.

W projekcie systemu znalazł się również szczegółowy opis wszystkich funkcji FRFa: za co są odpowiedzialne, ich dane wejściowe i wyjściowe oraz ich struktura.

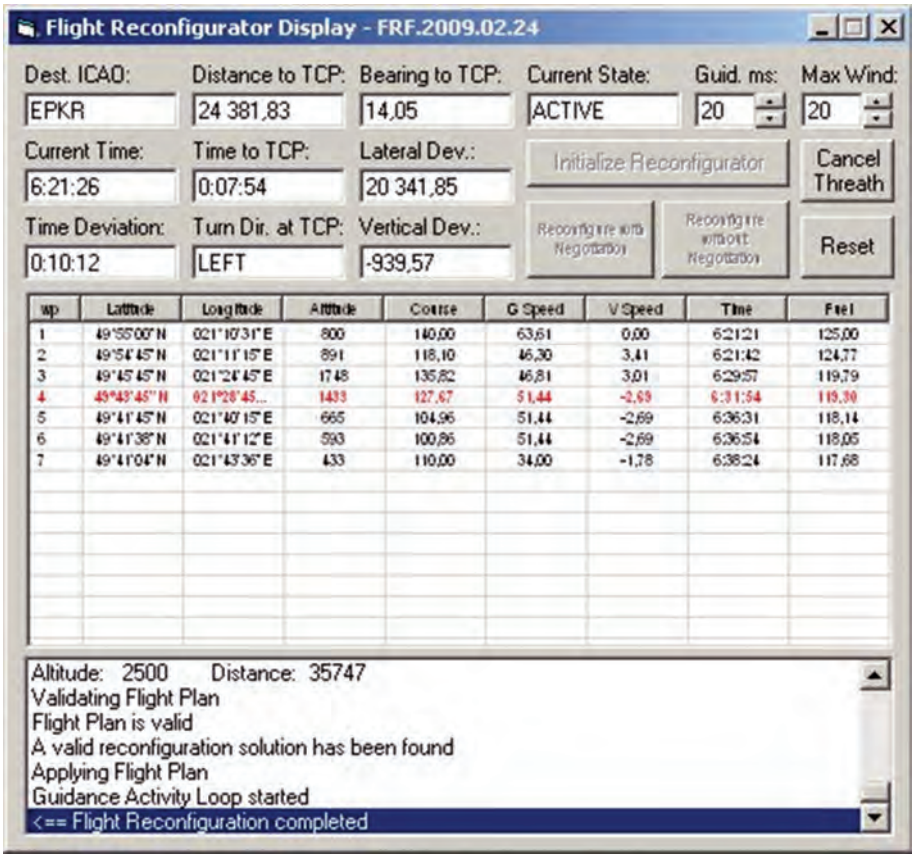
W projekcie SOFIA przyjęto, że w fazie tworzenia i testowania prototypowego oprogramowania, system FRF będzie pracował pod systemem operacyjnym Windows XP (SP2). System operacyjny został wybrany ze względu na jego popularność tak by żadna firma nie miała problemów z dostępem do niego. W późniejszej fazie rozwoju, oprogramowanie powinno korzystać z systemu o gwarantowanym czasie wykonania np. z jednego z systemów czasu rzeczywistego, jednak na czas tworzenia prototypu i sprawdzania jego funkcjonalności system Windows XP był wystarczający. Do tworzenie oprogramowania zostało wykorzystane zintegrowane środowisko programistyczne firmy Microsoft − Visual Studio 6.0 dla języków C++ lub Basic, w zależności od preferencji uczestników projektu. Środowisko programistyczne zostało zaproponowane przez firmy odpowiedzialne za tworzenie oprogramowania do systemu FRF ze względu na doświadczenie w pracy i tworzeniu projektów w tym środowisku a także jego dostępność.

Struktura stworzonego systemu została oparta na architekturze klient-serwer. W systemie aplikacja klienta korzystała z usług zapewnianych przez funkcje systemu FRF zawarte w modułach programowych (komponentach), za które odpowiedzialne były poszczególne firmy tworzące oprogramowanie i których kod nie był udostępniany innym uczestnikom projektu ze względu na politykę firmy. Moduły programistyczne zamknięte były w klasach, bibliotekach lub programach i udostępniały jedynie swoją funkcjonalność bez szczegółów dotyczących implementacji czy rozwiązań algorytmicznych. Komunikacja pomiędzy poszczególnymi częściami oprogramowania odbywa się poprzez klasy COM/DCOM.

Klasy COM (Component Object Model) są standardem binarnego interfejsu do definiowania i tworzenia komponentów programistycznych, wprowadzonym przez firmę Microsoft. Pozwalają na komunikację międzyprocesową i tworzenie obiektów dynamicznych w wielu językach programistycznych. DCOM (Distributed COM) jest rozszerzeniem modelu COM wspierającym komunikację między obiektami na różnych komputerach poprzez sieci LAN, WAN czy Internet. Zaletą technologii COM/DCOM jest to, że definiuje standard na poziomie binarnym, w oderwaniu od konkretnego narzędzia projektowego czy języka programowania. Poza tym ma przezroczysty charakter − użytkownikowi jest obojętne, gdzie fizycznie znajduje się aktualnie wykorzystywany komponent − jeżeli komponent znajduje się na odległym komputerze, aplikacja klienta korzysta z niego w taki sam sposób, jak z komponentu lokalnego.

System FRF został podzielony na 3 moduły:

- moduł rekonfiguracji lotu (Flight Reconfiguration) - składający się ze zbioru klas i bibliotek zawierających implementację funkcji rekonfiguracji lotu. W odniesieniu do funkcjonalności systemu FRF moduł zawierał: centrum podejmowania decyzji (inicjalizował i zarządzał poszczególnymi etapami działania systemu FRF), planowanie trasy i statyczne monitorowanie lotu, prowadzenie samolotu wzdłuż zaplanowanej trasy i weryfikację planowanej trasy. Dane rekonfiguratora lotu potrzebne pilotom były wyświetlane na interfejsie graficznym pokazanym na rysunku 1.



Rys. 1. Interfejs graficzny modułu rekonfiguracji lotu systemu FRF (własność konsorcjum SOFIA kontrakt AST5-CT-2006-030911)

Moduł rekonfiguracji lotu zawierał również jednostką adaptacyjną rekonfiguratora dla konkretnej platformy testowej – zapewniał komunikację między systemem FRF a platformą. Platforma dostarczała do systemu parametry lotu i samolotu, rekonfigurator dostarczał informacje o planie lotu i podawał nowe dane nawigacyjne. Rekonfigurator lotu zapewniał również dostęp do potrzebnych baz danych.

- moduł RPL (RoutePlanning and Static Flight Monitoring) i GLM (Guidance and Leg Managment) – będący biblioteką zawierającą funkcje przewidywania trajektorii w kontekście systemu FRF (Trajectory prediction) i zajmujący się procesem wykonywania planu lotu w kontekście systemu FRF (Guidance and Leg Management).
- moduł Routings – moduł zawierający oprogramowanie zawierające koncepcję automatycznego planowanie lotu stworzoną przez firmę THALES, opartą na parametrach samolotu i uwzględniające ukształtowanie terenu, przeszkody i obszary zabronione.

Przebiegi pracy poszczególnych części oprogramowania były zapisywane w plikach, co pozwalało na kontrolę poprawności działania systemu i pomagało lokalizować ewentualne błędy oprogramowania.

Stworzony system w postaci programów i bibliotek dostarczany był firmom dysponującym platformami testowymi w celu przetestowania oprogramowania. Firmy testujące musiały zainstalować i zintegrować dostarczone oprogramowanie na swoich platformach korzystając z jednostki adaptacyjnej znajdującej się w module rekonfiguratora lotu.

5. PROCES TESTOWANIA OPROGRAMOWANIA W PROJEKCIE SOFIA

Czwarty etap pracy w projekcie SOFIA obejmował kwestie walidacji systemu FRF. Starano się oszacować czy system FRF realizuje wymagane funkcje, czy działa zgodnie z założeniami i daje spodziewane wyniki oraz czy działa poprawnie po integracji na platformach testowych i przy

współpracy z naziemną stacją kontroli lotów. Przy projektach lotniczych, których częścią są próby w locie, zaplanowanie i przeprowadzenie testów jest wyjątkowo ważne, gdyż w grę wchodzi bezpieczeństwo pilotów i ludzi znajdujących się na ziemi.

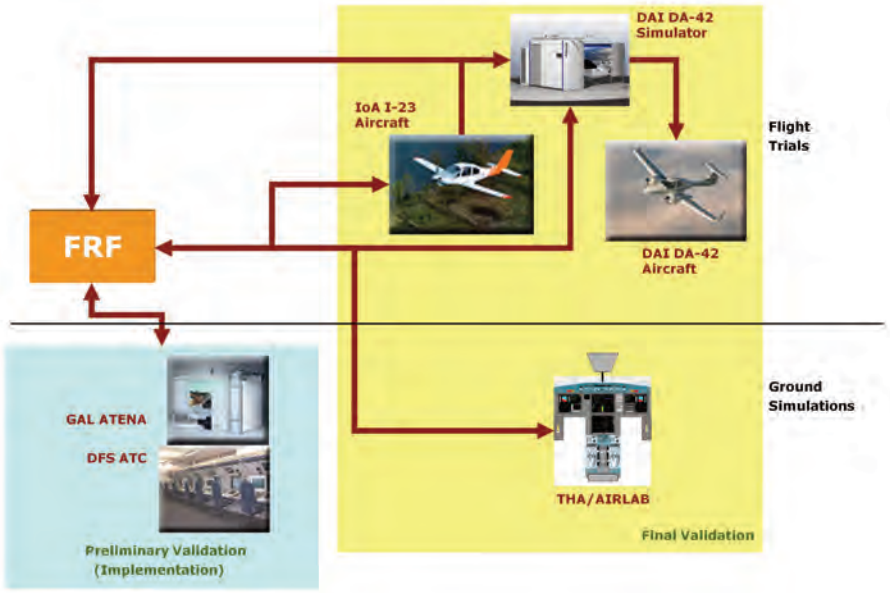
W projekcie SOFIA proces walidacji podzielono na trzy fazy (Rys. 2):

1. wstępną walidację, na którą składały się wstępne badania lotu na symulatorze lotów GAL ATENA połączonym z ATC symulatorem DFS,
2. końcowe walidacje na symulatorach kabinowych,
3. eksperymentalne próby w locie.

Przy testowaniu systemu FRF postępowano według następującego planu:

- stworzenie wiarygodnego scenariusza walidacji z logiczną sekwencją zdarzeń,
- testowanie funkcji platformy testowej, żeby wykluczyć wystąpienie błędu z jej strony,
- testowanie niezawodności nowych urządzeń, stworzonych na potrzeby walidacji systemu,
- sprawdzenie połączeń i interfejsów pomiędzy poszczególnymi częściami systemu by wykluczyć błędy komunikacyjne,
- sprawdzenie działania systemu FRF zintegrowanego na platformie.

Podczas ćwiczeń walidacyjnych, proces testowania musiał być na tyle elastyczny, żeby możliwe było wprowadzanie zmian w systemie lub przebiegu testu w zależności od przebiegu doświadczenia lub jego rezultatów.



Rys. 2. Plan walidacji systemu FRF
(własność konsorcjum SOFIA kontrakt AST5-CT-2006-030911)

W projekcie SOFIA Instytut Lotnictwa był odpowiedzialny za walidację systemu FRF na jednej z platform docelowych - na samolocie General Aviation I-23 (Rys. 2). Jego rolą było zintegrowanie systemu na samolocie i sprawdzenie poprawności algorytmów i działania oprogramowania tworzącego system FRF. Zdobyte w ten sposób doświadczenia i wyniki badań miały być wykorzystane do poprawienia systemu i przygotowania go do kolejnych prób w locie na samolocie firmy Diamond (Rys. 2).

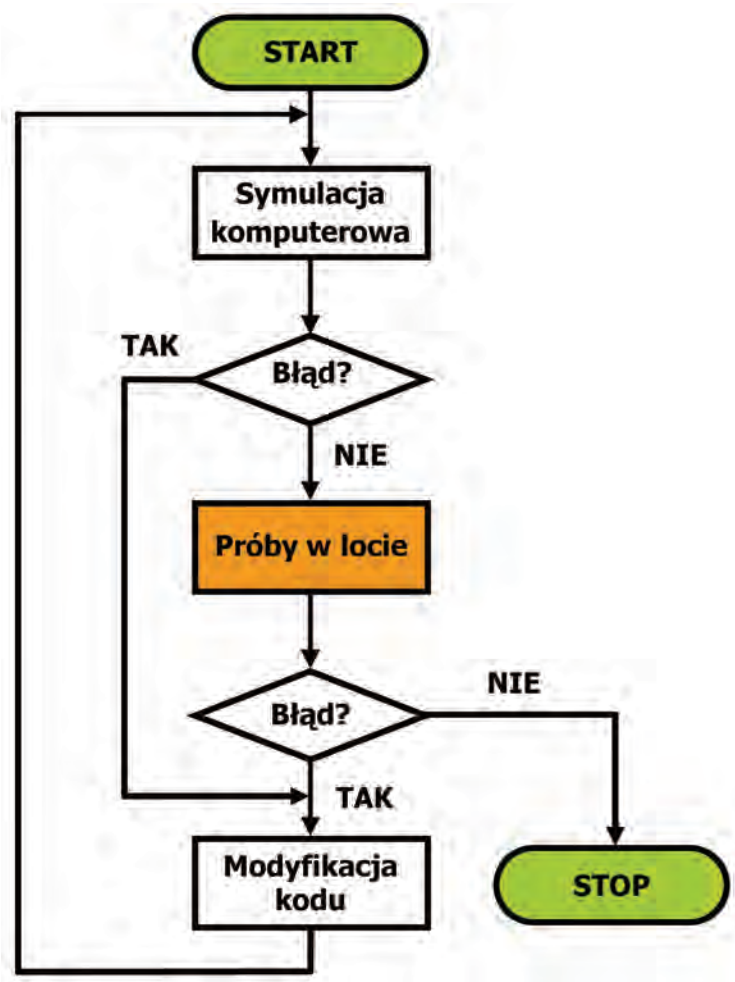
W skład sprzętowej platformy testowej będącej częścią wyposażenia samolotu testowego I-23 wchodziły:

- komputer FRF/AP – komputer PC/104 (Celeron 650MHz, 512 RAM, 4GB ATA/IDE Flash) z dodatkowymi kartami do obsługi szyny CAN i GSM/GPRS, na którym zaimplementowano oprogramowanie FRF wykonane przez partnerów projektów SOFIA oraz autopilota opracowane przez Politechnikę Rzeszowską,

- mechanizmy wykonawcze z linkami sterującymi lotek, sterów wysokości i trymerów,
- panel autopilota 2D z osobnym systemem zasilania, wyłącznikami bezpieczeństwa i bezpiecznikami,
- szyna CAN łącząca jednostki obliczeniowe z komputerem FRF/AP, mechanizmami wykonawczymi i panelem sterowania.

Przed rozpoczęciem testów systemu FRF przygotowana platforma testowa została sprawdzona osobno by uzyskać pewność, że ewentualne błędy wynikają tylko z nieprawidłowego działania testowanego oprogramowania, a nie są powodowane przez platformę.

Ze względu na specyfikę przeprowadzanych badań i koszt ich wykonywania, działanie systemu FRF było testowane na początku „na ziemi". Metodyka przeprowadzonych badań została przedstawiona na rysunku 3.



Rys. 3. Metodyka przeprowadzania testów systemu FRF

Na początku oprogramowanie FRF było testowane podczas symulacji komputerowych. Jeżeli zostały znalezione błędy, oprogramowanie było modyfikowane i testowane ponownie, jeżeli nie – odbywały się próby w locie. Jeżeli podczas prób w locie wykryte zostały kolejne błędy, proces testowania zaczynał się od początku.

Symulacje komputerowe stworzone w celu wstępnego przetestowania oprogramowania FRF na ziemi były bardzo proste, jednak nawet takie nieskomplikowane doświadczenia pozwoliły zdobyć wiele informacji na temat błędów i braków w oprogramowaniu oraz uzyskać wiele wskazówek jak poprawić jego jakość i wiarygodność. Podczas symulacji oprogramowanie było testowane przez wprowadzenie do systemu prostych tras lotu i kontrolowanie czy dane generowanie przez system są prawidłowe. W późniejszych fazach testów podawane do sytemu trasy

symulowały trasę lotu przewidzianą na rzeczywiste próby w locie, a w końcowych na wejście były podawane rzeczywiste dane z wykonanego wcześniej lotu. W wypadku wystąpienia błędu oprogramowanie przekazywane było do twórców w celu modyfikacji kodu lub działania algorytmów. Ze względu na prostotę symulatora możliwe było przekazywanie jego kodu partnerom w celu lepszego unaocznienia spostrzeżeń co do działania systemu.

Po przejściu testów na symulatorze, przeprowadzano testy na komunikację przez szynę CAN. Skontrolowano czy oprogramowanie systemu FRF po integracji z kodem autopilota dostarcza prawidłowe dane na szynę łączącą komputer FRF z innymi urządzeniami samolotu i czy prawidłowo odbiera symulowane dane wejściowe.

Przeprowadzone symulacje pozwoliły wykryć część błędów i zwiększały prawdopodobieństwo sukcesu podczas prób w powietrzu.

6. WNIOSKI

Praca przy projekcie SOFIA pozwoliła zdobyć wiele doświadczeń dotyczących zarówno metodyki tworzenia dużych lotniczych projektów informatycznych, jak i współpracy z firmami z innych krajów i instytucji. Wiele informacji, począwszy od sposobów definicji wstępnych założeń projektu, tak aby wszyscy uczestnicy byli zadowolenie z podziału pracy i wyboru rozwiązań programistycznych i lotniczych, poprzez prowadzenie dokumentacji w czasie trwania całego projektu aż do rozpowszechniania osiągnięć współpracy innym firmom i instytucjom z Europy, pozwoli w przyszłości na znaczne usprawnienie działań w podobnych przedsięwzięciach. Udział w SOFII unaocznił ilość i typy problemów jakie mogą powstać w czasie trwania takiego projektu, od małych, takich jak różne rozumienie tych samych pojęć przez różne firmy, po duże, takie jak uzyskiwanie certyfikatów lotu dla samolotu z nowym, eksperymentalnym systemem pokładowym.

Spośród wielu problemów, które zaistniały podczas tworzenia opisywanego projektu należy zwrócić uwagę na dwa: problem określenia i trwania poszczególnych faz tworzenia systemu i problem jego adaptacji na konkretnych platformach.

Przy przechodzeniu przez fazy tworzenia oprogramowania FRF bardzo dużo czasu poświęcono fazie określania wymagań i tworzenia planu projektu systemu. Można było zaobserwować trudności w organizacji pracy i dokładnym określeniu funkcji systemu FRF, których źródłem były różne doświadczenia i potrzeby partnerów. Dużo czasu poświęcono kwestiom proceduralnym a jednoznaczny podział obowiązków i pracy powstał w stosunkowo późnej fazie rozwoju planu projektu. Przedłużenie wstępnych faz rozwoju życia projektu spowodowało, że ograniczony został czas na jego realizację i testowanie, szczególnie, że pojawiły się również niespodziewane opóźnienia wynikające z problemów z certyfikacją samolotu z eksperymentalnym systemem FRF.

Wiele problemów wynikło również podczas adaptacji oprogramowania systemu FRF. Zróżnicowanie platform testowych, różnorodność urządzeń i oprogramowania, z którym nowy system miał współpracować spowodował, że niektóre części systemu były często modyfikowane. Prowadziło to do czasochłonnego procesu dostosowywania systemu do wymagań, któremu towarzyszyła stała wymiany informacji między firmami tworzącymi oprogramowania a firmami testującymi je na docelowych urządzeniach. Wymiana informacji oparta była o kontakt telefoniczny i mailowy, który nie jest tak efektywny jak kontakt bezpośredni i zajmował dużo czasu. W szczególnych przypadkach przy adaptacji systemu niezbędny okazał się przyjazd i pomoc twórców oprogramowania co wiązało się z dodatkowymi kosztami. W przyszłości należy zatem poświęcać bardzo dużo uwagi na dokładne określenie potrzeb i wymagań jakie powinno spełniać oprogramowanie by móc działać w różnych środowiskach.

BIBLIOGRAFIA

[1] *SOFIA reports*, 2005-2008.
[2] *McGraw-Hill Encyclopedia of Science and Technology*, 5th edition, published by The McGraw-Hill Companies, Inc.
[3] *Software engineering. The eighth edition*, Ian Sommerville.
[4] *DCOM Technical Overview. MSDN*, Microsoft Corporation, 1996.

Anna Gałach

THE CYCLE OF MANUFACTURING SOFTWARE SYSTEMS ON A BASE OF SOFIA PROJECT

Abstract

The article discusses phases of manufacturing big, aircraft software system, realized by cooperation of a few European companies. The paper is focused on requirements definition, design, coding, integration and testing of system created during the project. The article discussed the process of selection of hardware and software platforms used in the project, the choice of communication protocols, process of integration of the new system into destination platforms and process of testing. The described cycle of manufacturing software system was based on experience collected during FP6 SOFIA (Safe Automatic Flight Back and landing of Aircraft) project realized in years 2006-2009.

Praca naukowa finansowana ze środków na naukę w latach 2006-09 (projekt nr T12C 062 30)
His article is undertake within the framework FP6-2005-CT-2006-030911