ZESZYTY NAUKOWE POLITECHNIKI RZESZOWSKIEJ **Nr 279**
Mechanika z. 83 (3/11) **2011**

Anna GAŁACH
Instytut Lotnictwa

PROCES I PROBLEMY TWORZENIA, INTEGRACJI I TESTOWANIA OPROGRAMOWANIA W DUŻYCH EUROPEJSKICH PROJEKTACH, NA PRZYKŁADZIE PROJEKTU SOFIA

Artykuł przedstawia zagadnienia związane z tworzeniem wspólnego oprogramowania przez wiele firm uczestniczących w dużym projekcie europejskim. W pracy zostały poruszone kwestie wyboru platform sprzętowych i programistycznych, uzgodnienia protokołów komunikacyjnych, jak również zagadnienia integracji i testowania stworzonego oprogramowania. Na podstawie doświadczeń z realizacji projektu FP6 SOFIA (ang. *Safe Automatic Flight Back and Landing of Aircraft*) opisano problemy wynikające z ograniczeń dotyczących polityki prywatności firm, dostępności i preferencji platform programistycznych i sprzętowych, jak również odległości dzielących wykonawców projektu.

Wstęp

Tworzenie oprogramowania, szczególnie w przypadku dużych systemów informatycznych, w których bierze udział wiele firm niemających doświadczeń we współpracy, jest procesem skomplikowanym i bardzo dynamicznym. Gdy poszczególne firmy zajmują się innymi częściami 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 zagadnienia tworzenia oprogramowania omówiono w artykule, a oparte są one na doświadczeniach zdobytych podczas pracy w projekcie europejskim SOFIA.

1. Założenia i cele projektu SOFIA

Celem SOFII było opracowanie koncepcji i algorytmów pozwalających na automatyczny powrót samolotu na ziemię w przypadku pojawienia się na pokła-

dzie niebezpieczeństwa ataku terrorystycznego [3]. Głównym zadaniem projektu było zaprojektowanie funkcji rekonfiguracji lotu FRF (ang. *Flight Reconfiguration Function*) odpowiedzialnych za: 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, tak aby uniemożliwić porywaczom przejęcie kontroli nad samolotem. Stworzone oprogramowanie realizujące system rekonfiguracji lotu miało być zintegrowane na platformach testowych w celu sprawdzenia poprawności jego działania.

Do walidacji systemu w projekcie SOFIA wykorzystano pięć platform: dwa symulatory lotów, symulator naziemnej stacji kontroli ruchu lotniczego i dwa samoloty klasy General Aviaton. Ponieważ 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 wprowadzanie 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.

2. Proces tworzenia oprogramowania

Proces produkcji oprogramowania składa się z kilku faz [2, 4]. Ich definicja, porządek, interakcje między poszczególnymi fazami specyfikują tzw. modele cyklu życia oprogramowania. Mimo istnienia wielu różnych modeli cyklu życia oprogramowania, we wszystkich można wyróżnić fazy: definicji wymagań, projektowania, implementacji, testowania i pielęgnacji.

1. Podczas fazy definicji wymagań jest określane, co system ma robić oraz w jakich warunkach i ograniczeniach ma działać. Typowa definicja wymagań zawiera: przegląd dostępnych produktów, specyfikację rozwoju, działania i pielęgnację środowiska działania produktu, wysokopoziomowy model koncepcyjny systemu, specyfikację interfejsu użytkownika, specyfikację wymagań funkcjonalnych i niefunkcjonalnych, specyfikację interfejsów do innych systemów, specyfikację obsługi błędów, listę możliwych zmian i poprawek systemu.

2. W fazie projektowania jest tworzony plan realizacji wymagań systemu, a faza implementacji skupia się na tworzeniu kodu według stworzonego wcześniej planu.

3. Testowanie jest procesem egzaminowania oprogramowania, w czasie którego wyszukiwane są występujące w nim błędy i braki. Proces testowania jest najczęściej dzielony na fazy. Pierwszą fazą jest testowanie małych jednostek oprogramowania przez programistów, drugą – testowanie integracyjne, gdzie jednostki są łączone i testowane w grupach. Na koniec odbywa się testowanie całego systemu.

4. Ostatnią fazą procesu tworzenia oprogramowania jest faza pielęgnacji, która polega na wprowadzaniu zmian do systemu po jego stworzeniu i oddaniu do użytkowania. Informacje z poszczególnych faz tworzenia oprogramowania są zawsze dokumentowane.

W projekcie SOFIA praca nad stworzeniem systemu FRF została podzielona na 5 etapów [3]. Pierwszy obejmował zadania operacyjne systemu, drugi – tworzenie projektu systemu, trzeci – tworzenie systemu, czwarty – testy i walidację systemu, piąty poruszał zaś kwestie rozpowszechniania i eksploatacji systemu.

3. 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 przeanalizowano dostępne technologie i systemy lotnicze, z którymi system FRF musiałby współpracować. Scharakteryzowano 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. Określono 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 zostały dopuszczone do lotu. Oszacowano również poziom bezpieczeństwa systemu i stworzono zbiór wskazań dla specyfikacji i architektury systemu oraz 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 oraz definicję interfejsów z FMSem/autopilotem. W projekcie systemu znalazł się również szczegółowy opis wszystkich funkcji FRFa, tzn. określono za co są one odpowiedzialne, ich dane wejściowe i wyjściowe oraz strukturę.

W projekcie SOFIA przyjęto, że w fazie tworzenia i testowania prototypowego oprogramowania system FRF będzie pracował w systemie operacyjnym Windows XP (SP2). System operacyjny został wybrany ze względu na jego popularność, tak aby żadna z firm nie miała trudności 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 tworzenia oprogramowania wykorzystano 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 tworzenie projektów w tym środowisku, a także jego dostępność.

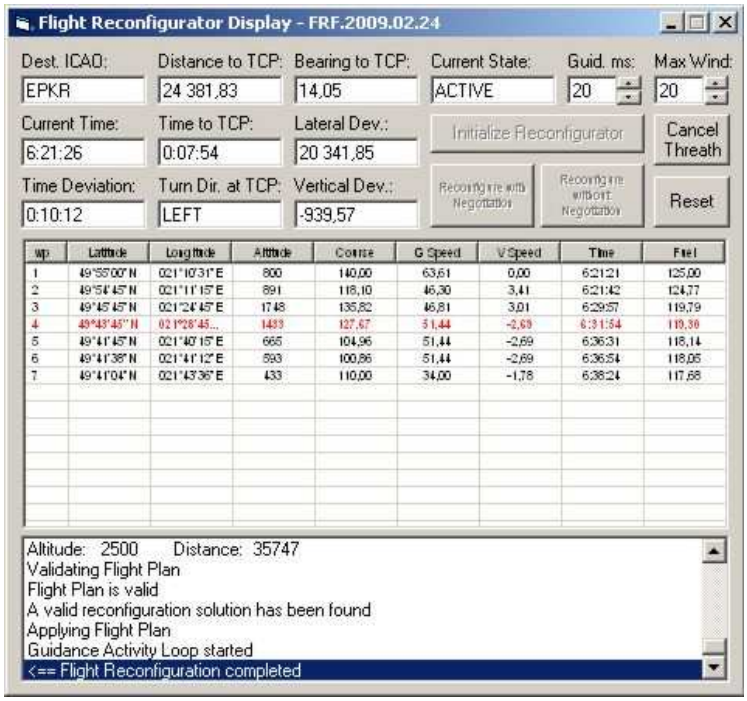
Struktura stworzonego systemu została oparta na architekturze klient–serwer. W systemie aplikacja klienta korzystała z usług zapewnianych przez funkcje systemu FRF zawartych w modułach programowych (komponentach), za które odpowiedzialne były poszczególne firmy tworzące oprogramowanie i których kod nie 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/DOM [1].

Klasy COM (ang. *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 (ang. *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.

System FRF został podzielony na 3 moduły [1]:

- moduł rekonfiguracji lotu (ang. *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 (inicjalizacja i zarządzanie 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 (rys. 1.); moduł rekonfiguracji lotu zawierał również jednostkę adaptacyjną rekonfiguratora dla konkretnej platformy testowej – zapewniał komunikację między systemem FRF a platformą, platforma dostarczała do systemu parametry lotu i samolotu, a rekonfigurator – informacje o planie lotu i nowych danych nawigacyjnych, rekonfigurator lotu zapewniał również dostęp do potrzebnych baz danych,



Rys. 1. Interfejs graficzny modułu rekonfiguracji lotu systemu FRF

- moduł RPL (ang. *RoutePlanning and Static Flight Monitoring*) i GLM (ang. *Guidance Managment and Leg Managment*) – będący biblioteką zawierającą funkcje przewidywania trajektorii w kontekście systemu FRF i zajmujący się procesem wykonywania planu lotu w kontekście systemu FRF,
- moduł Routings – moduł zawierający oprogramowanie z koncepcją automatycznego planowania lotu, 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 był dostarczany firmom dysponującym platformami testowymi do przetestowania oprogramowania. Firmy testujące musiały zainstalować i zintegrować dostarczone oprogramowanie na swoich platformach, korzystając z jednostki adaptacyjnej znajdującej się w module rekonfiguratora lotu.

4. 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 we współpracy z naziemną stacją kontroli lotów. W projektach lotniczych, których częścią są próby w locie, szczególnie ważne jest zaplanowanie i przeprowadzenie testów, ze względu na bezpieczeństwo pilotów i ludzi znajdujących się na ziemi. W projekcie SOFIA wyróżniono fazy: wstępną walidację, na którą składały się wstępne badania na symulatorze lotów połączonym z symulatorem naziemnej stacji kontroli ruchu lotniczego oraz końcowe walidacje na symulatorach kabinowych i eksperymentalne próby w locie.

Podczas testowania systemu FRF postępowano według następującego planu.

1. Stworzenie wiarygodnego scenariusza walidacji z logiczną sekwencją zdarzeń.
2. Testowanie funkcji platformy testowej w celu wykluczenia wystąpienia błędu z jej strony.
3. Testowanie niezawodności nowych urządzeń stworzonych na potrzeby walidacji systemu.
4. Sprawdzenie połączeń i interfejsów pomiędzy poszczególnymi częściami systemu, aby wykluczyć błędy komunikacyjne.
5. Sprawdzenie działania systemu FRF zintegrowanego na platformie.

Podczas ćwiczeń walidacyjnych proces testowania musiał być na tyle elastyczny, aby umożliwiał wprowadzenie zmian w systemie lub etapach testu, w zależności od przebiegu doświadczenia lub jego rezultatów.

W projekcie SOFIA Instytut Lotnictwa był odpowiedzialny za walidację systemu FRF na jednej z platform docelowych, tj. na samolocie I-23. Jego rolą było zintegrowanie systemu na samolocie oraz 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.

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

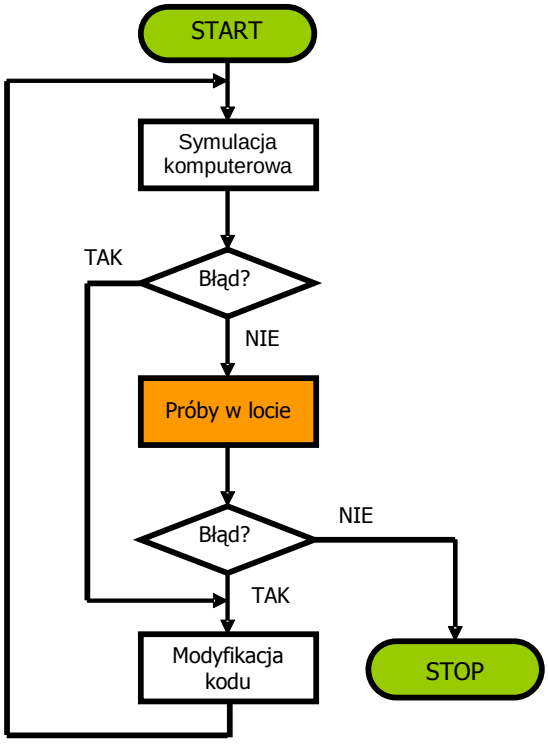
- komputer FRF/AP – komputer PC/104 (Celeron 650 MHz, 512 RAM, 4GB ATA/IDE Flash) z dodatkowymi kartami do obsługi szyny CAN i GSM/GPRS, na którym zaimplementowano oprogramowanie FRF wykonane przez partnerów projektów SOFIA oraz autopilota, opracowane przez Politechnikę Rzeszowską,
- mechanizmy wykonawcze 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 najpierw było testowane „na ziemi". Metodyka przeprowadzonych badań została przedstawiona na rys. 2.

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 wskazówki co do poprawy jego jakości i wiarygodności. Podczas symulacji oprogramowanie było testowane przez wprowadzenie do systemu prostych tras lotu i kontrolowanie, czy dane generowane 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 wejściu były podawane rzeczywiste dane z wykonanego wcześniej lotu. W przypadku wystąpienia błędu oprogramowanie przekazywane było do twórców w celu modyfikacji kodu lub działania algorytmów. Ze względu na prostotę symulatora możliwe było przekazanie jego kodu partnerom, aby lepiej unaocznić spostrzeżenia 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.



Rys. 2. Metodyka przeprowadzania testów systemu FRF

5. 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 czy instytucji. Wiele informacji – począwszy od sposobów definicji wstępnych założeń projektu, tak aby

wszyscy uczestnicy byli zadowoleni z podziału pracy i wyboru rozwiązań programistycznych i lotniczych, poprzez prowadzenie dokumentacji w czasie trwania całego projektu, aż do rozpowszechniania osiągnięć współpracy – pozwoli w przyszłości innym firmom i instytucjom z Europy na znaczne usprawnienie działań w podobnych przedsięwzięciach. Udział w SOFII unaocznił liczbę 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, jak np. uzyskiwanie certyfikatów lotu dla samolotu z nowym, eksperymentalnym systemem pokładowym.

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

Podczas faz tworzenia oprogramowania FRF bardzo dużo czasu poświęcono określaniu wymagań i tworzeniu planu projektu systemu. Można było zaobserwować trudności w organizacji pracy i dokładnym określeniu funkcji systemu FRF, których źródłem były różne doświadczenia i potrzeby partnerów. Szczególną uwagę zwrócono także na kwestie proceduralne, a jednoznaczny podział obowiązków i pracy powstał w stosunkowo późnej fazie rozwoju planu projektu. Przedłużenie wstępnych faz rozwoju projektu spowodowało, że ograniczono czas na jego realizację i testowanie, szczególnie że pojawiły się również niespodziewane opóźnienia wynikające z problemów z certyfikacją samolotu z eksperymentalnym systemem FRF.

Wiele trudności wynikło również podczas adaptacji oprogramowania systemu FRF. Zróżnicowanie platform testowych, różnorodność urządzeń i oprogramowania, z którym nowy system miał współpracować, spowodowały, że niektóre części systemu były często modyfikowane. Prowadziło to do czasochłonnego procesu dostosowywania systemu do wymagań, któremu towarzyszyła stała wymiana informacji między firmami tworzącymi oprogramowania a firmami testującymi je na docelowych urządzeniach. Wymiana informacji oparta była na kontaktach telefonicznych i mailowych, które nie są tak efektywne jak kontakt bezpośredni, i zajmowała 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.

Literatura

[1] DCOM Technical overview, MSDN, Microsoft Corporation, 1996.
[2] Encyclopedia of science and technology, 5th ed., McGraw-Hill, 2005.

[3] SOFIA reports, 2005÷2008.
[4] Sommerville I.: Software engineering, 8th ed., Addison Wesley Publishing Company, 2007.

PROCESS AND PROBLEMS OF SOFTWARE DEVELOPMENT, INTEGRATION AND TESTING IN LARGE EUROPEAN FRAME PROJECTS – LESSON LEARNED IN SOFIA PROJECT

A b s t r a c t

The article discusses the cooperation problems during the software development in large European Frame Project. The paper is focused on the choice of software and hardware platforms, selection of communication protocols, problems with integration and testing of the created software modules. Based on experience collected during FP6 SOFIA (Safe Automatic Flight Back and landing of Aircraft) project, the following problems are discussed: difficulties caused by the privacy policies of companies engaged in a project, problems with availability and preferences of hardware and software, problems caused by distance and language barriers.

Złożono w Oficynie Wydawniczej we wrześniu 2011 r.