Zespołowy projekt CASE, etap modelowania i analizy

W fazie analizy są ustalane wszystkie te czynniki lub warunki w dziedzinie przedmiotowej, w otoczeniu realizatorów projektu i w istniejących lub planowanych systemach komputerowych, które mogą wpłynąć na przebieg procesu projektowego i na jakość realizacji wymagań. Wynikiem tej fazy jest logiczny model systemu, opisujący sposób realizacji przez system postawionych wymagań, lecz generalnie abstrahujący się od szczegółów implementacyjnych.

Do czynności wykonywanych w fazie analizy odnosimy: rozpoznanie, wyjaśnianie, modelowanie, specyfikowanie i dokumentowanie rzeczywistości lub problemu będącego przedmiotem projektu; ustalenie kontekstu projektu; ustalenie wymagań użytkowników; ustalenie wymagań organizacyjnych; ustalenia, dotyczące preferencji sprzętowych, preferencji w zakresie oprogramowania, ograniczeń finansowych, ograniczeń czasowych.

Etap modelowania i analizy częściowo pokrywa się z etapem projektowania, tak że w tym samym czasie zespół może już podejmować niektóre decyzje projektowe.

Cel etapu

Rozpoznanie wszystkich czynników dziedziny przedmiotowej, które mogą wpłynąć na realizację projektu oraz stworzenie logicznego modelu systemu. Wykrycie tych fragmentów dziedziny problemu, których wspomaganie za pomocą oprogramowania będzie szczególnie przydatne (priorytetowe).

Zadania do wykonania

- Podjąć decyzję: która metodyka analizy – strukturalna czy obiektowa – zostanie zastosowana w wykonywanym projekcie. Wybrać odpowiednie narzędzie CASE[1]. Zainstalować oprogramowanie CASE, opanować podstawowe tryby pracy z tym narzędziem.
- W uzgodnieniu z prowadzącym wybrać odpowiednie dla wykonywanego projektu techniki analizy[2], np.:
    - analizę funkcji i przypadków użycia;
    - budowę statycznego modelu klas;
    - identyfikację i definiowanie metod oraz komunikatów;
    - modelowanie stanów i przejść między stanami;
    - modelowanie procesów i przepływów danych;
    - modelowanie przepływu sterowania.
- Opracować właściwy zestaw diagramów, takich jak: diagram przypadków użycia, diagram klas, diagram przepływu danych, diagram związków encji, diagram przejść stanów, diagram aktywności, diagram sekwencji, diagram współpracy, diagram komponentów.
- Opracować słownik danych zawierający specyfikację modelu.
- Skorygować SWS, harmonogram i kosztorys projektu w oparciu o wiedzę uzyskaną na etapie modelowania i analizy.

Wymagana dokumentacja etapu

1. Notatka z uzasadnieniem wyboru metodologii, narzędzia CASE i techniki modelowania, zastosowanych w projekcie.

---

[1] Zaleca się wykorzystanie narzędzi CASE z listy wolnych (otwartych) narzędzi UML.

[2] Dokonany wybór metodologii, narzędzi i technik modelowania należy uzasadnić, notatka z uzasadnieniem ma być dołączona do dokumentacji projektowej.

2. Komplet diagram do projektu.
3. Opis tekstowy stworzonego logicznego modelu aplikacji.
4. Poprawione wersje SWS, harmonogramu i kosztorysu.

Źródła wiedzy

1. Artykuł o UML z Wikipedii, http://pl.wikipedia.org/wiki/UML
2. Krzysztof Goczyła, Unified Modeling Language (UML), http://www.mif.pg.gda.pl/homepages/sylas/students/io/uml/index.htm
3. Randy Miller, Język UML™ w praktyce: wprowadzenie dla programistów, http://www.borland.pl/tech/poradnik_uml.shtml
4. Przegląd diagramów języka UML, http://www.erudis.pl/index.php?page_id=56&lang=pl

Porady, wskazówki i podpowiedzi

- Należy pamiętać, że w inżynierii oprogramowania modele tworzone są dla lepszego zrozumienia dziedziny problemu oraz umożliwienia wymiany informacji pomiędzy członkami zespołu projektowego oraz zespołu i klienta. Ponieważ opis problemu jest podstawą do podjęcia decyzji projektowych, kluczowym zagadnieniem jest ścisłe i odpowiadające rzeczywistości sformułowanie tego opisu. Modelowanie pomaga zidentyfikować problem, określić jego skalę, wypracować rozwiązanie. Potencjalne korzyści z posiadania dobrych modeli to:
    - podniesienie jakości rozwiązania dzięki lepszemu zrozumieniu problemu;
    - możliwość wykorzystania doświadczeń wynikających z pracy innych ludzi (wzorce projektowe);
    - wsparcie narzędziowe przy tworzeniu kodu aplikacji (CASE generatory kodu).
- Booch sformułował następne zasady modelowania:
    - Podjęcie decyzji, jakie modele tworzyć, ma wielki wpływ na to, w jaki sposób zaatakujemy problem i jaki kształt przyjmie rozwiązanie.
    - Każdy model może być opracowany na różnych poziomach szczegółowości.
    - Najlepsze modele odpowiadają rzeczywistości.
    - Żaden model nie jest wystarczający. Niewielka liczba niemal niezależnych modeli to najlepsze rozwiązanie w przypadku każdego niebanalnego systemu.
- Oto znane w inżynierii oprogramowania metodyki i techniki modelowania:
    - Metodyki strukturalne:
        - Metodyka Yourdona (DeMarco i Ward/Mellor);
        - Structured System Analysis and Design Methodology (SSADM);
        - Structured Analysis and Design Technique (SADT).
    - Techniki analizy strukturalnej:
        - Diagramy Przepływu Danych (Data Flow Diagrams, DFD)
        - Słownik Danych (Data Dictionary)
        - Strukturalny Angielski (Structured English) -> strukturalny polski
        - Tablice Decyzyjne (Decision Tables)
        - Drzewa Decyzyjne (Decision Trees)
        - Schemat Transformacyjny (Transformation Schema)
        - Diagram Przejść Stanów (State Transition Diagram)
        - Lista Zdarzeń (Event List)
        - Schemat Danych (Data Schema)
        - Pre- i post- warunki (Pre- and Post-Conditions)
        - Diagramy Encja-Związek (występują w SSADM)

        - Historie Życia Encji (występują w SSADM)
    - Metodyki obiektowe:
        - Yourdon-Coad
        - OMT (Rumbaugh)
        - OOSE (Jacobson)
        - OOAD (Booch)
        - Class-Responsibility-Collaboration card (CRC)
        - Shlaer-Mellor
    - Techniki analizy obiektowej:
        - Express
        - OODA (Booch)
        - OMT (Rumbaugh)
        - OOSA (Shlaer-Mellor)
        - Objectory (Jacobson)
        - MOSES/OPEN
        - OOA/OOD (Coad/Yourdon)
        - Notacja UML
        - RUP