Proiectarea Sistemelor Software Complexe

Curs 14 –Documentarea Arhitecturii Unui Sistem Software Complex

14.1 Introducere

Documentarea arhitecturii unui sistem software este o sarcină dificilă. Multe proiecte software conțin foarte puțină (poate chiar deloc) documentație în ceea ce privește arhitectura, alte proiecte conțin foarte multă documentație, dar care este greu de urmărit și de cele mai multe ori nu este actualizată la zi. În continuare sunt enumerate câteva motive pentru care este bine să se realizeze o documentație de calitate a arhitecturii:

- Alte persoane pot să evalueze și să înțeleagă arhitectura.
- Ajută la înțelegerea arhitecturii atunci când se revine la ea după o perioadă de timp.
- Alți membrii ai echipei pot învăța analizând gândirea din spatele arhitecturii.
- Se pot face analize care permit de exemplu evaluarea performanței.

Principalele dificultăți în ceea ce privește realizarea unei documentații de calitate sunt:

- Nu există un standard în ceea ce privește documentarea arhitecturii.
- Arhitectura unui sistem poate fi complexă, iar documentarea ei este dificilă și consumatoare de timp.
- Există mai multe perspective din care poate fi analizată o arhitectură, documentarea tuturor acestor perspective este consumatoare de timp și scumpă din punct de vedere al costurilor.
- Arhitectura unui sistem evoluează pe măsură ce sistemul este dezvoltat, de aceea și documentația trebuie actualizată, proces care este consumator de timp și dificil de realizat.

Stabilirea conținutului pentru documentația unei arhitecturi nu este ușor de realizat, în plus acesta poate să varieze în funcție de proiect. În mod normal documentația trebuie să descrie:

- interfețele componentelor;
- constrângerile subsistemelor;
- scenariile de test;
- deciziile de achiziționare ale componentelor terțe;
- structura echipei;
- serviciile externe oferite de aplicație.

14.2 UML 2.0

Existe mai multe modalități prin care pot fi descrise diferitele perspective ale unei arhitecturi. UML 2.0 reprezintă însă un limbaj descriptiv care este folosit în toate domeniile ale dezvoltării software. Notațiile UML 2.0 pot descrie atât aspecte structurale cât și comportamentale ale unui sistem software. Diagramele structurale descriu arhitectura statică, acestea sunt:

- diagrame de clase – descriu clase din sistem și relațiile dintre acestea;

- diagramele de componente – descriu relațiile între componente cu interfețe bine definite;
- diagrame de pachete – descompun modelul în grupuri de elemente și descriu dependențele dintre ele la nivel abstract;
- diagrame de distribuire – descriu modul în care componentele și alte artefacte software sunt distribuite pe echipamentele hardware;
- diagrame de obiecte – descriu modul în care obiectele sunt relaționate și utilizate în timpul rulării;
- diagrame care descriu structura internă a claselor sau componentelor în ceea ce privește obiectele și relațiile dintre ele.

Diagramele comportamentale descriu interacțiunile și schimbările de stare ce au loc pe măsură ce elementele din model sunt executate:

- diagrame de activitate – sunt folosite pentru a defini logica unui program și procesele business;
- diagrame de comunicare – descriu secvența de apeluri ce au loc între obiecte în timpul rulării;
- diagrame de secvență – descriu secvența de mesaje schimbată între obiecte;
- diagrame de stare – descriu stările, evenimentele și condițiile care generează tranzițiile de stare;
- diagrame temporale – descriu evoluția stărilor unui obiect în timp;
- diagrame use case – redau interacțiunile dintre sistem și mediul înconjurător (utilizatori și alte sistem software).

14.3 Șabloane

Organizațiile mari obișnuiesc să dezvolte șabloane de documente. Acest lucru reduce timpul de start pentru realizarea unui document punând la dispoziție o structură de document predefinită. Șabloanele ajută de asemenea la pregătirea noilor membri ai unei echipe.

În continuare este prezentat un exemplu de astfel de șablon:

```
Șablon Pentru Documentarea Arhitecturii
Nume Proiect: ZZZ
1 Context Proiect
2 Cerințe Arhitecturale
    2.1 Privire de Ansamblu Asupra Obiectivelor Cheie
    2.2 Use Case
    2.3 Cerințe
    2.4 Constrângeri
    2.5 Cerințe Non-funcționale
    2.6 Riscuri
3 Soluție
    3.1 Modele Arhitecturale Folosite
    3.2 Privire de Ansamblu Asupra Arhitecturii
    3.3 Structură
    3.4 Comportament
    3.5 Probleme de Implementare
4 Analiza Arhitecturii
    4.1 Scenarii de Analiză
    4.2 Risc
```

Bibliografie

[1] Ian Gorton. Essential Software Architecture. Editura Springer. 2006.