Arhitecturi soft

curs opțional
specializarea Informatică (liniile română şi engleză)
semestrul 8 (cursul opțional 7), cod MI099
limba de predare: engleză
titular: prof. dr. Bazil PÂRV

Odată cu creşterea complexității sistemelor soft, accentul se deplasează de la algoritmi şi structuri de date (implementate în module - componente soft) la organizarea, structura generală a sistemelor construite din respectivele componente, altfel spus la *arhitectura* sistemului. Există diverse modalități de exprimare a arhitecturii unui sistem (descrieri textuale, diagrame, limbaje de interconectare, şabloane de proiectare etc.).

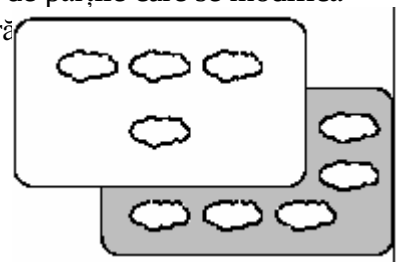
Există numeroase definiții ale arhitecturii soft, care se pot consulta pe pagina http://www.sei.cmu.edu/architecture/definitions.html. De altfel, Software Engineering Institute al Universității Carnegie Mellon din Pittsburgh, PA are preocupări serioase în domeniul arhitecturii sistemelor soft.

Scopurile arhitecturii sunt [Fussell]:

- înțelegerea a ce se doreşte să se construiască
  - o imagine generală (big, whole picture)
  - responsabilitățile subsistemelor şi dezvoltatorilor
  - cum să se gestioneze evoluția sistemului
  - unde pot apare probleme
- pregătirea sistemului pentru modificarea cerințelor şi a implementării
  - înțelegerea limitelor abordării curente
  - a cunoaşte ce trebuie schimbat/construit pentru a satisface orice modificare a cerințelor

O arhitectură bună

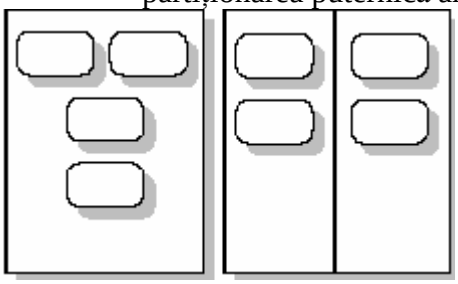
- permite dezvoltatorilor să înțeleagă ce reprezintă munca lor pentru întregul proiect
- permite gestionarea nevoilor curente ale proiectului
- este stabilă chiar dacă apar cerințe noi
- izolează părțile sistemului care nu se modifică de părțile care se modifică
- permite beneficii sporite (reutilizare) pe măsură



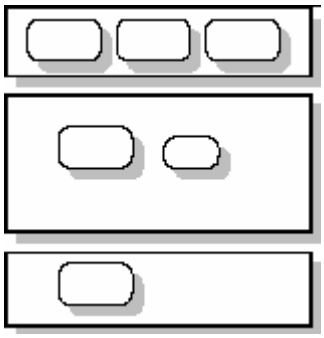
Abordări arhitecturale

- la nivelul proiectării (fizice)
  - metode şi clase
- la nivelul întregului sistem
  - subsistem:
    - pachet de clase înrudite cu o interfață bine definită
    - reduce complexitatea prin separarea funcționalității interfeței publice de implementare
    - este gestionat de o singură echipă
  - partiție
    - diviziune pe verticală a aplicației, separând zone diferite de funcționalitate
    - reduce gradul de interacțiune între subsisteme
    - simplifică interacțiunile între echipe

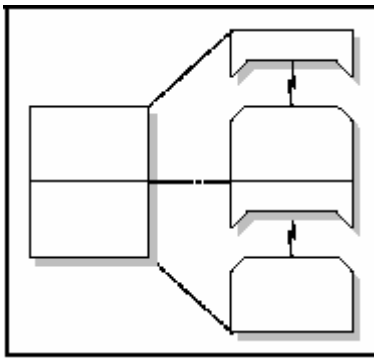
- partiționarea puternică are interfețe formale între partiții



- framework şi bibliotecă (library)
  - partiționare puternică a funcționalității comune
  - permite dezvoltatorilor să se concentreze asupra problemelor specifice ale aplicației
  - ciclu de dezvoltare specific
    - framework-ul există înainte de începerea dezvoltării aplicației (dacă este bine definit şi stabil)
    - framework-ul rezultă în urma dezvoltării aplicației (generalizare şi integrare)
    - niciodată framework-ul şi aplicația în acelaşi timp
- layer (strat)
  - stratificarea logică a subsistemelor aplicației
  - fiecare strat are o abstractizare pentru un sistem particular şi responsabilități specifice
    - este uşoară înțelegerea şi folosirea fiecărui strat
    - este uşoară dezvoltarea într-un strat
  - izolează straturile superioare de implementarea straturilor inferioare



- tier (nivel)
  - divide ierarhic o aplicație în mai multe procese
  - apare în interiorul unui strat
    - aceeaşi direcție ca şi stratificarea
    - de obicei nu se adaugă funcționalitate deasupra punctului de descompunere în nivele
  - măreşte complexitatea internă a stratului



- combinarea abordărilor
  - pentru simplificarea arhitecturii sistemelor mari, complexe

Cuprinsul orientativ al cursului:

1. Introducere. De la limbajele de programare la arhitectura sistemelor soft
1.1. Definiții
1.2. Istoric. Preocupări 'clasice' legate de arhitectura sistemelor soft
1.3. Limbaje de programare de nivel inalt
1.4. Tipuri abstracte de date
1.5. Arhitectura sistemelor soft
2. Stiluri arhitecturale
2.1. Magistrale şi filtre
2.2. Structurarea programelor folosind abstractizarea datelor şi orientarea pe obiecte
2.3. Apelarea implicită, bazată pe evenimente
2.4. Sisteme stratificate
2.5. Depozite de date partajate
2.6. Interpretarea dirijată de tabele
2.7. Alte stiluri 'familiare' de arhitectură
2.8. Arhitecturi heterogene
3. Studii de caz
3.1. Problema 1: KWIC (Key Word in Context, David Parnas)
3.2. Problema 2: Soft pentru instrumente de măsură
3.3. Problema 3: Roboți mobili
3.4. Problema 4: Controlul vitezei (Cruise control)
3.5. Problema 5: Structura stratificată cu stiluri diferite pentru fiecare nivel
3.6. Problema 6: Interpretor cu diferite stiluri de componente
3.7. Problema 7: Arhitectură blackboard reconsiderată global ca interpretor
4. Sisteme soft bazate pe partajarea datelor între utilizatori
4.1. Partajarea resurselor in sistemele software
4.2. Integrarea bazelor de date
4.3. Integrarea în mediile de programare
4.4. Integrarea în aplicațiile de proiectare a construcțiilor
4.5. Structuri arhitecturale pentru sistemele informatice bazate pe partajarea datelor
5. Arhitecturi client-server
5.1. Introducere
5.2. Evoluția arhitecturilor software
5.2.1. Arhitecturi mainframe

5.2.2. Arhitecturi bazate pe partajarea fişierelor
5.2.3. Arhitecturi client-server
5.3. Exemple de arhitecturi client-server
5.3.1. Arhitecturi pe două nivele
5.3.2. Arhitecturi pe trei (mai multe) nivele
5.3.2.1. Arhitecturi pe trei nivele folosind tehnologia monitoarelor pentru prelucrarea tranzacțiilor
5.3.2.2. Arhitecturi pe trei nivele cu server de mesaje
5.3.2.3. Arhitecturi pe trei nivele cu server de aplicații
5.3.2.4. Arhitecturi de broker de cereri la obiecte (Object Request Broker, ORB)
5.3.2.5. Arhitectura distribuită/colaborativă de intreprindere
6. Modele şi elemente de arhitectură
6.1. Tipuri de elemente şi modele
6.2. Notații arhitecturale tradiționale
6.3. Limbaje de descriere a arhitecturilor
6.4. Modelul celor 4+1 vederi (Kruchten)
7. Notații arhitecturale
7.1. Diagrame informale
7.2. Limbaje de interconectare a modulelor
7.3. SGL (Software Glue Language)
7.4. ADL (Architecture Description Language)
7.5. Limbaje de modelare/specificare (UML, ModeChart)

Activități de laborator

Studenții vor lucra în echipe de 3-5 membri pentru rezolvarea unei probleme concrete, folosind diverse stiluri arhitecturale. Fiecare membru va implementa o soluție proprie, cu un stil arhitectural specific.
Principalele activități de laborator sunt:
• faza de gestiune a proiectului (săpt 1 - 3)
• faza de analiză/proiectare (săpt 4 - 7)
• faza de implementare/testare (săpt 8 - 11)
• demonstrații live şi documentație finală (săpt 12)

Materialele de curs si de laborator sunt/vor fi disponibile pe serverul Departamentului de Informatica, în directorul ..\labor\engleza\an4\SoftwareArch

Bibliografie

1. FUSSELL, MARK L.: A Good Architecture for Object-Oriented Information Systems. Structures, Designs, and Patterns, OOPSLA'96 Tutorial 23, [http://www.chimu.com/publications/oopsla96tutorial23/oopsla96tutorial23.pdf].
2. KRUCHTEN, PHILIPPE: Architectural Blueprints - The 4+1 View Model of Software Architecture, IEEE Software 12 (6), 1995, pp. 42-50.
3. MOWBRAY, THOMAS J. - MALVEAU, RAPHAEL: Software Architect Bootcamp, Prentice Hall, (1st ed., 2000, 2nd ed., 2003).
4. SHAW, MARY: The Coming-of-Age of Software Architecture Research, in Proc. of the 23rd ICSE, IEEE Comp. Soc. 2001, 656, [http://www.cs.cmu.edu/afs/cs.cmu.edu/project/vit/ftp/pdf/shaw-keynote-rev.pdf]

5. SHAW, MARY - GARLAN, DAVID: Software Architecture: Perspectives on an Emerging Discipline, Prentice-Hall, 1996.
6. Software Architecture Resources, [http://www.bredemeyer.com/papers.htm]