Asupra modelelor ciclului de viaţă a unui sistem informatic

Mat.ec. Sorina-Carmen Luca
Anglo Romanian Bank Ldt.- Timişoara, România
Conf.dr. Lucian Luca, Asist. Adela Ionescu,
Universitatea „Tibiscus" Timişoara, România

ABSTRACT. The paper presents a comparative study on the patterns of the life cycle of a computer science system. There are analyzed the avantages of each pattern and presented the graphic schemes that point out each stage and step in the evolution of a computer system. In the end are discussed the classifications of the methods of projecting the computer science systems.

1. Ciclul de viaţă a unui sistem informatic

Punctul de început al acestui ciclu este reprezentat de decizia de realizare a unui nou sistem informatic mai performant, iar punctul final al ciclului de viaţă este reprezentat de momentul deciziei de înlocuire a sistemului informatic existent cu unul nou, mai bine adaptat cerinţelor, asigurând performanţe informaţionale, tehnice şi economice superioare.

În timp, au fost elaborate mai multe modele ale ciclului de viaţă al unui sistem informatic, în care se pot identifica etapele de dezvoltare a produsului software. Numărul, numele, conţinutul şi înlănţuirea fazelor propuse în cadrul acestor modele diferă, dar încercând o sinteză, putem reţine următoarele ***faze,*** ca elemente comune diferitelor abordări:

1. Definirea cerinţelor utilizatorilor: utilizatorii vor preciza obiectivele pe care urmează să le îndeplinească viitorul sistem informatic, criteriile de eficienţă, securitate, performanţă pe care acesta urmează să le asigure. Calitatea şi performanţele unui produs software depind de abilitatea echipei de analiză în colectarea de specificaţii cât mai complete de la

viitorul utilizator şi de a-I face pe acesta să "vadă" funcționalitatea viitorului sistem informatic.

2. Specificația cerințelor sistemului: prezentarea detaliată a rezultatelor pe care sistemul informatic urmează să le asigure. Se va evidenția ce anume urmează să facă sistemul, fără a se sugera în nici un fel cum va face acest lucru sistemul. Această fază este un răspuns la specificația cuprinzând cerințele utilizatorului şi va fi utilizată în faza de proiectare a sistemului.
3. Specificația cerințelor software: evidențiază ce urmează să facă produsul software şi restricțiile sub care funcționalitatea sa urmează să fie asigurată.
4. Proiectarea generală, în cadrul căreia se definesc soluții cadru, conceptuale, privind viitorul sistem informatic.
5. Proiectarea de detaliu, care rafinează soluția cadru, rezultat al proiectării generale, având ca finalitate definirea soluției finale a sistemului informatic.
6. Realizarea componentelor sistemului informatic, pe baza soluțiilor oferite de proiectarea de detaliu.
7. Testarea componentelor: verificarea modului de funcționare, modului de îndeplinire a cerințelor şi fiabilitatea în utilizare.
8. Integrarea componentelor şi testarea finală a sistemului: reunirea componentelor în cadrul produsului final şi verificarea funcționării lui în ansamblu.
9. Implementarea şi testarea produsului la beneficiar, urmate de acceptarea produsului de către acesta.
10. Exploatarea şi întreținerea sistemului: utilizarea curentă a sistemului informatic şi întreținerea lui.
11. Dezvoltarea sistemului informatic: realizarea şi integrarea de noi componente care să îmbunătățească şi/sau dezvolte funcționalitatea şi performanțele sistemului.

În cadrul diferitelor modele, aceste faze (alături de cele specifice fiecărui model) sunt reunite în cadrul unor ***etape*** şi anume:

- Analiza sistemului, care pleacă de la analiza sistemului existent şi se continuă cu fazele 1-3 mai sus-menționate;
- Proiectarea generală (faza 4);
- Proiectarea de detaliu (faza 5);
- Realizarea sistemului informatic (fazele 6-8);
- Instalarea sistemului informatic pe sistemele de calcul ale beneficiarului (faza 9);
- Exploatarea şi întreținerea sistemului informatic (faza 10);

- Dezvoltarea sistemului informatic (faza 11).

Modelele elaborate au cunoscut îmbunătăţiri permanente, încercându-se adaptarea lor la noile cerinţe ale modelării orintate obiect, precum şi inserarea unor etape specifice managementului proiectelor.

1.1. Modelul cascadă

Modelul cascadă (*Waterfall Model, figura 1*) a fost elaborat de W.W. Royce la începutul anilor '70. Este un model de referinţă în literatura de specialitate, caracterizat prin parcurgerea secvenţială a fazelor ciclului de viaţă, faze care, la rândul lor, sunt formate din activităţi, iar acestea din urmă, din subactivităţi.

Modelul prezintă următoarele avantaje:

- controlul total al fazelor, datorită modului de ordonare a acestora;
- uşor de însuşit de către membrii echipelor de analiză şi proiectare;
- fiecare fază se încheie cu o verificare a soluţiei oferite şi asigură o documentaţie prezentând soluţia elaborată.

În timp, au fost propuse variante îmbunătăţite ale modelului:

- modelul cu revenire la pasul următor (waterfall model with back flow);
- modelul cu reluare de la faza iniţială ("Da Capo" Waterfall Model).

În versiuni mai noi ale modelului cascadă, primele faze grupează activităţi specifice gestiunii proiectului, aceste elemente lipsind în modelul iniţial.

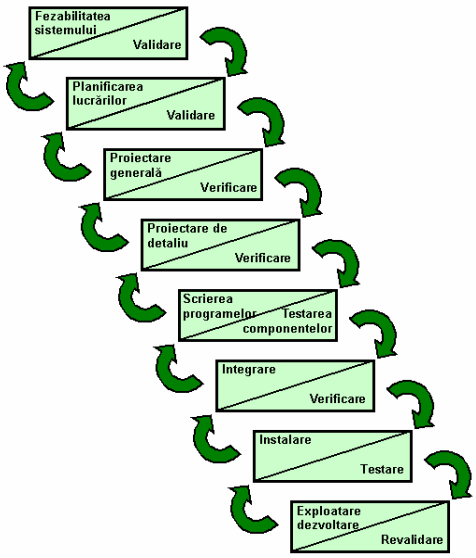

Fig. 1: *Modelul cascadă*

1.2. Modelul în V

Modelul în V este o variantă a modelului cascadă, care aduce elemente calitative noi importante. Un element caracteristic al modelului este introducerea conceptelor de sistem şi componente (subsisteme), aplicându-se teste explicite pentru creşterea controlului asupra modului în care se desfăşoară etapele. Fazele plasate în partea superioară a modelului se caracterizează prin implicarea directă a viitorului utilizator.

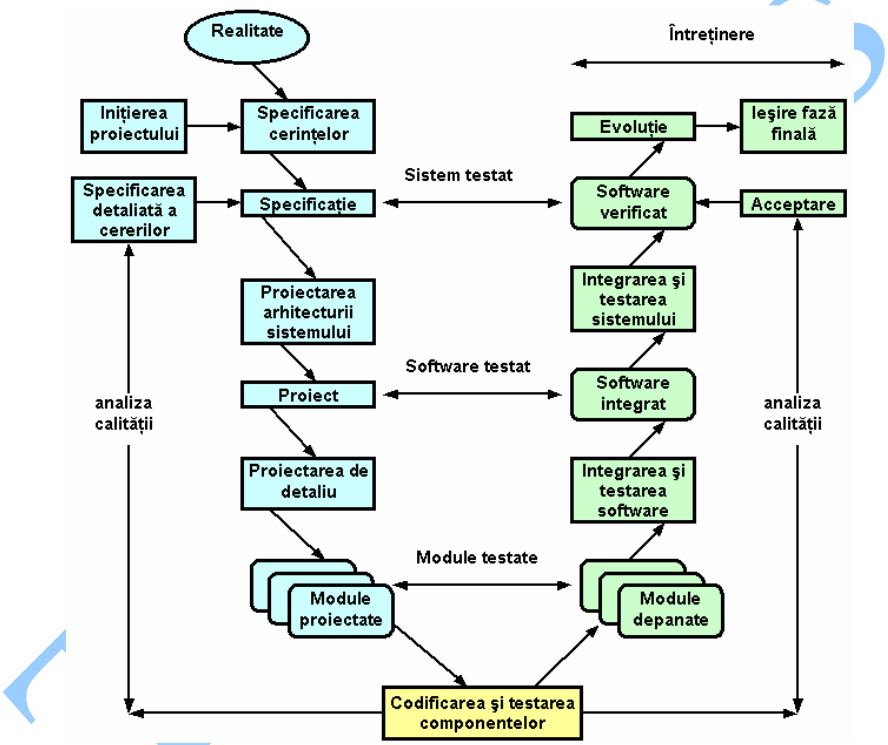

Fig. 2: *Modelul în V*

Brațul stâng al diagramei, parcurs descendent, reuneşte fazele în cadrul cărora se realizează, pas cu pas, proiectarea şi realizarea sistemului informatic. Detalierea activităților de proiectare, codificare şi asamblare a componentelor se realizează gradual. Dealtfel, Ould, creatorul modelului în forma lui consacrată, a prevăzut doar latura din stânga, unde efortul principal de proiectare se focalizează pe descompunerea sistemului pe componente.

Brațul drept al diagramei cuprinde reprezentarea fazelor asigurând

asamblarea progresivă a componentelor sistemului, pe măsura testării lor individuale, până la obținerea sistemului global şi acceptarea acestuia de către beneficiar.

În cadrul modelului, se remarcă realizarea distincției dintre verificare şi validare. Prima se referă la testarea sistemului în diversele stadii pe care le parcurge, iar validarea urmăreşte să identifice în ce măsură sistemul corespunde cerințelor inițiale, ceea ce constituie un punct slab al modelului, datorită întârzierii cu care se produce această validare.

Modelul în V i-a oferit lui Rumbaugh elementele necesare pentru modelul pe care I-a elaborat în 1991 şi, totodată, este punct de plecare pentru modelele în W şi X.

1.3. Modelul în W

Acest model reia ideea modelului în V, pe care îl dezvoltă şi perfecționează prin integrarea activităților de validare la nivelul fazelor de proiectare.

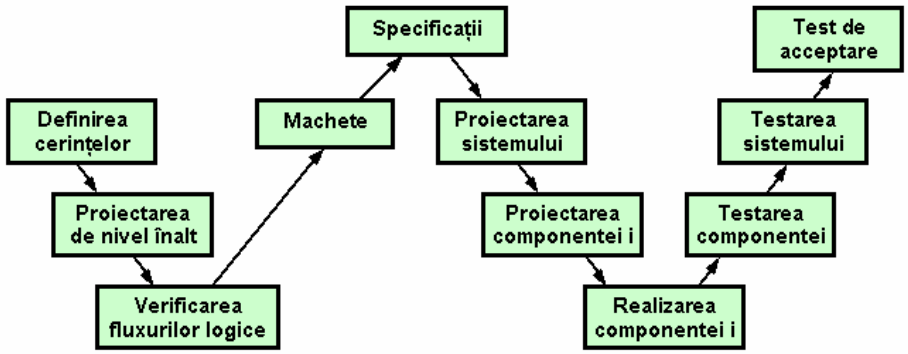

Fig. 3: *Modelul în W*

1.4. Modelul incremental

Modelul incremental este o altă variantă a modelului cascadă, care promovează ideea proiectării şi realizării independente a componentelor după definirea arhitecturii globale a sistemului informatic. Proiectarea şi realizarea sistemului informatic se realizează, astfel, în conformitate cu principiile metodelor *top-down*. Sistemul va putea fi livrat beneficiarului şi etapizat, pe măsura realizării componentelor (în functie de prioritățile formulate de beneficiar), dar într-o astfel de abordare pot apărea dificultăți legate de integrarea componentelor în sistemul final.

Din figura de mai jos se observă faptul că primele două etape - definirea cerințelor şi analiza - sunt identice cu cele două etape de început ale modelului cascadă, însă din momentul definirii arhitecturii sistemului informatic, fiecare componentă îşi urmează propriul ciclu de viață.

Spre deosebire de modelul în V, care presupunea integrarea componentelor, testarea şi validarea acestuia, de această dată se oferă şi posibilitatea livrării independente a componentelor sistemului informatic către beneficiar, fără a se exclude şi posibilitatea livrării sistemului informatic final, având toate componentele integrate.

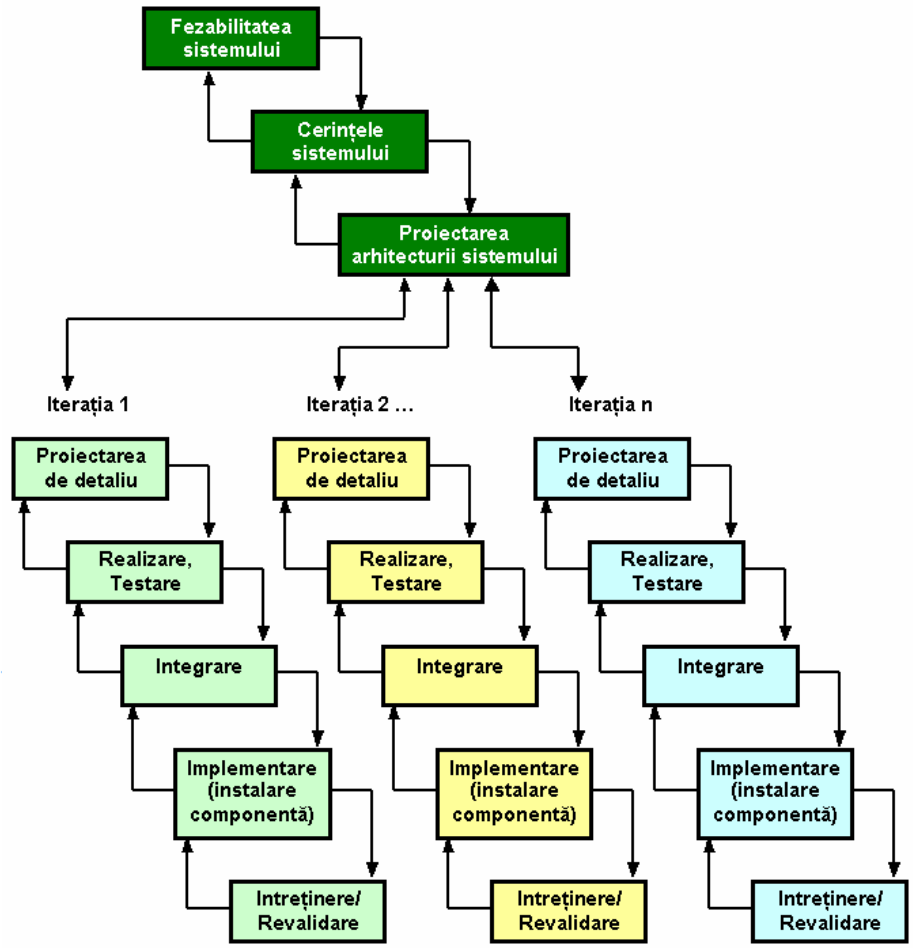

Fig. 4: *Modelul incremental*

1.5. Modelul spirală

Modelul spirală, elaborat de Barry Boehm, se bazează pe acelaşi principiu ca şi modelul evolutiv. Modelul presupune construirea mai multor prototipuri succesive, în condiţiile realizării unei analize a riscului pe fiecare nivel.

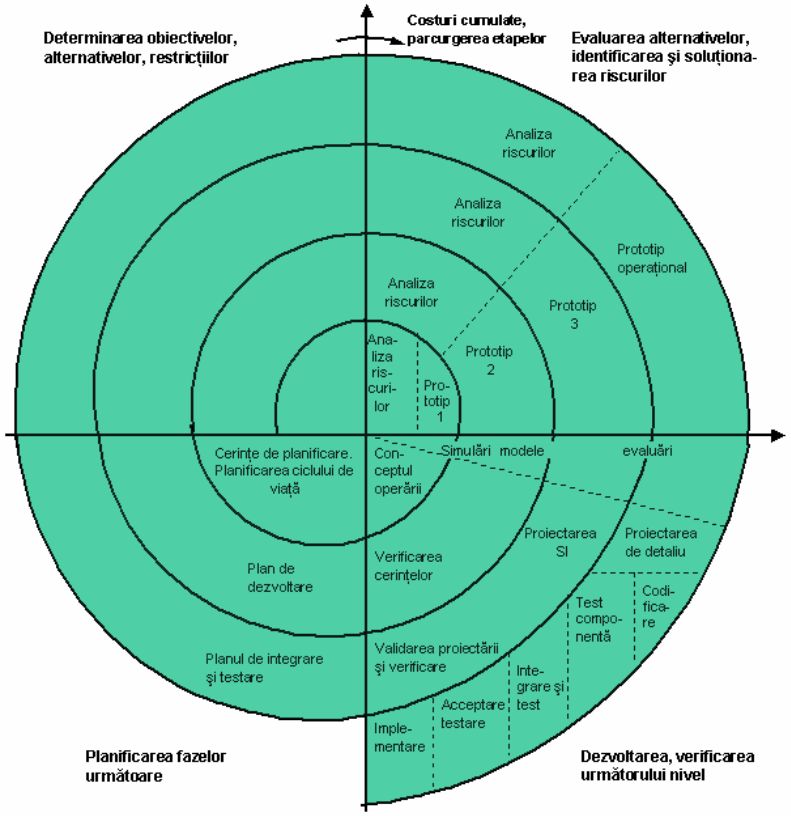

Fig. 5: *Modelul spirală*

Fazele de dezvoltare sunt reluate la fiecare iteraţie, în aceeaşi succesiune şi presupun:

1. analiza riscurilor;
2. realizarea unui prototip;
3. simularea şi testarea prototipului;
4. determinarea cerinţelor în urma rezultatelor testării;
5. validarea cerinţelor;
6. planificarea ciclului următor.

Ultimul ciclu conduce la realizarea versiunii finale a sistemului

informatic.

În centrul spiralei este plasată cunoaşterea cerinţelor şi estimarea costurilor la nivel preliminar. Evoluţia sistemului informatic urmează desfăşurarea spiralei, înregistrând acumulări succesive ale costurilor şi este marcată de succesiunea prototipurilor, fiecare dintre acestea valorificând acumulările realizate al nivelul prototipului anterior. Interacţiunea dintre faze nu este reliefată direct, atâta timp cât modelul prevede o succesiune continuă a rafinării, legate de decizii pe care riscurile proiectului le asociază cu următoarea detaliere.

Modelul evidenţiază atenţia acordată planificării, căutării de soluţii alternative, evaluării riscurilor şi validării soluţiilor pentru fiecare prototip, văzut ca un stadiu distinct în realizarea sistemului informatic.

În ingineria software, un prototip este folosit atât pentru validarea cât şi pentru identificarea cererilor utilizatorilor, pentru verificarea soluţiei de proiectare şi oferirea bazei dezvoltării ulterioare a proiectului de sistem informatic. Apelarea la utilizarea prototipului este consecinţa faptului că un model funcţional este mai uşor de înţeles de către viitorul utilizator, decât un set de diagrame însoţite de documentaţie.

Prototipul funcţional presupune proiectarea sistemului, realizarea primului prototip funcţional, verificarea măsurii în care răspunde cererilor formulate de utilizator şi rafinarea acestei prime soluţii, prin dezvoltări viitoare care adaugă noi funcţionalităţi până la obţinerea variantei finale a sistemului.

1.6. Modelul fântână arteziană

Modelul fântână arteziană îşi are izvoarele în modelul spirală (ierarhic) şi modelul "vârtej de apă". Porneşte de la cunoaşterea lumii reale, a cerinţelor şi elaborarea studiului de fezabilitate. Se parcurg apoi etapele de: analiză, proiectarea sistemului, proiectare componentă, codificare, testare componentă, testare sistem, utilizare, întreţinere, dezvoltare.

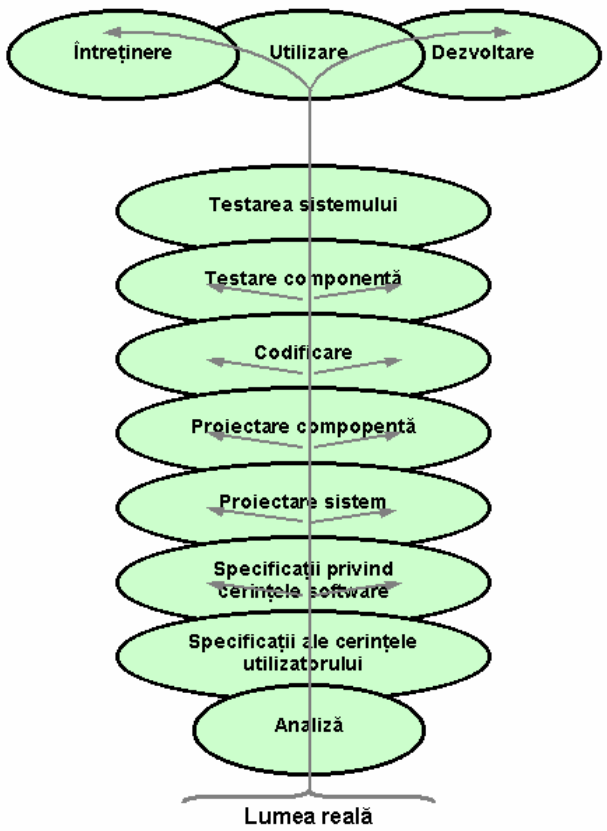

Fig. 6: *Modelul fântână arteziană*

În cazul sistemelor informatice realizate pe baza modelelor de proiectare orientată object (POO), modelul fântână arteziană este preferat modelului cascadă şi acesta datorită necesităţii fuzionării unor etape ale ciclului de viaţă şi creşterii gradului de iteraţie.

De această data se pune accentul mai degrabă pe clase decât pe sistem. Se consideră mai adecvat să nu se aibă în vedere doar întregul sistem la fiecare etapă a ciclului de viată, ci mai degrabă la identificarea claselor ce-şi urmează propriul ciclu de viată.

1.7. Modelul tridimensional

Modelul tridimensional, promovat de metoda de proiectare MERISE, se caracterizează prin reprezentarea grafică pe trei axe, fiecare dintre acestea corespunzând ciclului de viaţă al sistemului, ciclului de decizie şi respectiv ciclului abstractizării.

Ciclul abstractizării se dezvoltă pe trei niveluri: conceptual, logic şi fizic.

Nivelul conceptual se aplică independent datelor şi prelucrărilor generând modelul conceptual al datelor şi respectiv modelul conceptual al prelucrărilor. Se realizează succesiv modelul logic al datelor şi modelul fizic al datelor, modelul logic al prelucrărilor sau organizațional, modelul operațional al prelucrărilor.

Ciclul de decizie cuprinde ansamblul deciziilor legate de proiectarea, realizarea şi exploatarea sistemului informatic:

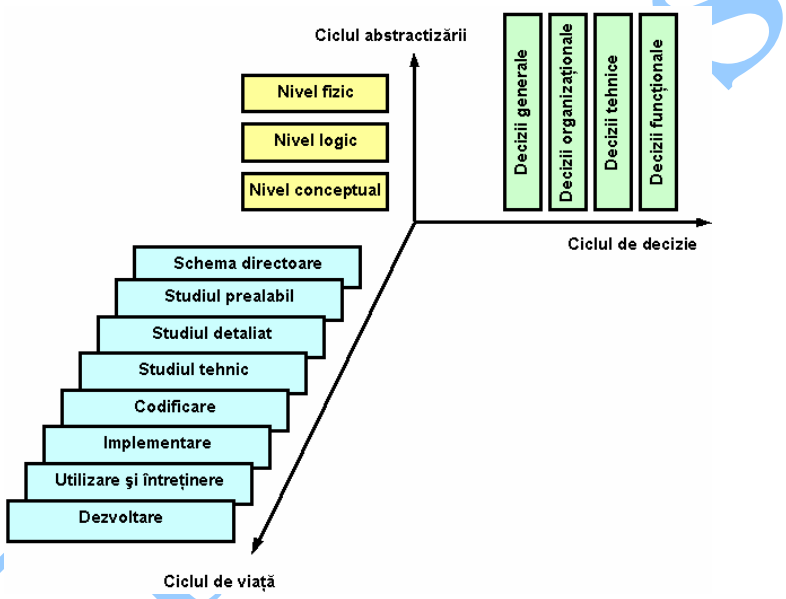

Fig. 7: *Modelul tridimensional*

- *Deciziile globale* vizează problemele cadru privind obiectivele sistemului informatic şi funcționalitatea lui, domeniile de informatizat, prioritățile, planificarea lucrărilor, etc.
- *Deciziile de organizare* vizează arhitectura şi interacțiunile dintre componente, strategia de gestiune a datelor, planificarea activităților de proiectare generală, de detaliu şi realizare a sistemului informatic.
- *Deciziile tehnice* vizează suportul hardware şi de comunicație, suportul software şi mediul de exploatare a sistemului informatic, etc.
- *Deciziile legate de exploatare şi întreținere* sunt decizii care asigură buna funcționare a sistemului informatic şi aparțin de această dată utilizatorului (celelalte decizii aparțin, în principal, proiectantului iar

validarea soluțiilor se realizează de beneficiar).

Ciclul de viață presupune parcurgerea succesivă a următoarelor etape:

- *Schema directoare şi studiul prealabil* (care poate aparține schemei directoare) presupune: analiza sistemului informatic existent, formularea cerințelor şi obiectivelor sistemului informatic, definirea priorităților în realizarea sistemului informatic, realizarea de scenarii globale alternative pentru fiecare domeniu investigat şi alegerea scenariului considerat optim.
- *Studiul detaliat* pleacă de la soluția cadru definită pentru scenariul ales, pe care o dezvoltă, abordând problemele de la general la particular. Această etapă asigură modelarea conceptuală şi organizațională a viitorului sistem informatic.
- *Studiul tehnic,* prin soluțiile concrete pe care le definește, va asigura modelarea logică şi fizică. *Codificarea* corespunde etapei de scriere şi testare a procedurilor, fiind apoi urmată de integrarea acestora şi de testarea finală a sistemului.
- *Implementarea* şi *testarea* sistemului în condițiile reale de exploatare ale beneficiarului vor fi urmate de acceptarea produsului, pe baza evaluării rezultatelor testării.
- *Exploatarea, întreținerea şi dezvoltarea* sistemului informatic au conținutuI cunoscut deja din prezentările anterioare.

1.8. Modelul evolutiv

Modelul evolutiv pornește de la realizarea unui studiu inițial privind obiectivele viitorului sistem informatic, a cărui arhitectură este definită ulterior. Fiecare componentă astfel definită îşi va urma propriul său ciclu de viață (definirea cerințelor, analiză, proiectare, realizare, testare, utilizare), urmând să fie livrată beneficiarului în momentul finalizării.

Sistemul informatic reprezintă ansamblul unor componente în interacțiunea lor, fiind rezultatul unei concepții bazate pe arhitecturi deschise şi flexible. O astfel de abordare este apropiată celei orientată obiect, caracterizate prin încapsularea datelor şi funcționalității obiectelor.

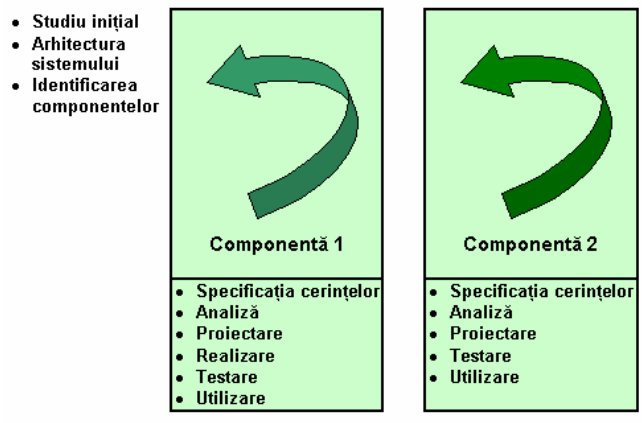

Fig. 8: *Modelul evolutiv*

Reprezentarea grafică a modelului evolutiv este influențată de modelul circular, a cărui caracteristică o reprezintă marcarea unui ciclu complet al sistemului informatic printr-un cerc.

1.9. Modelul minge de baseball

Modelul minge de baseball (dezvoltarea concurențială) propus de Codd, Yourdon şi Nicola pleacă de la ideea renunțării la paşii succesivi în realizarea sistemului, în favoarea promovării activităților desfăşurate în paralel.

Este vorba de analiza orientată obiect (AOO), proiectarea (design) orientată-obiect (DOO) şi programarea orientată-obiect (POO).

Într-o astfel de abordare, AOO ar beneficia de rezultatele DOO şi POO; DOO beneficiază de rezultatele AOO şi POO, iar POO valorifică rezultatele AOO şi DOO.

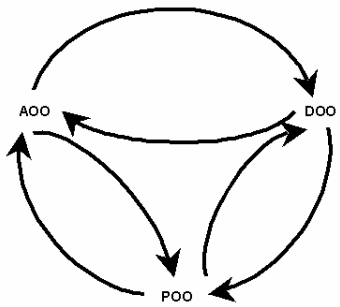

Fig. 9: *Modelul minge de baseball*

1.10. Modelul pinball

Modelul pinball, elaborat în 1994 de S.W. Ambler, este caracteristic ciclului de viaţă al produselor software realizate cu ajutorul metodologiilor orientate-obiect.

Modelul este realizat după principiul deplasării aleatoare a unei bile într-un sistem mecanic cu arc (de tip flipper), în vederea atingerii unor ţinte, reprezentate de obiective ale proiectării şi programării orientate obiect.

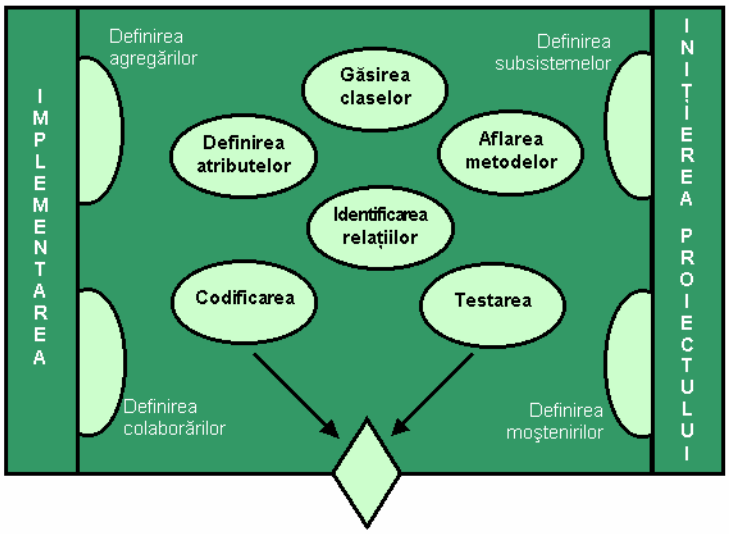

Fig. 10: *Modelul pinball*

Tampoanele, obstacolele şi braţele mobile (din partea de jos) aparţinând spaţiului de "joc", corespund următoarelor activităţi: aflarea clasei de apartenenţă a atributelor şi metodelor, determinarea relaţiilor dintre obiecte, definirea agregărilor, moştenirilor, scrierea codului, testarea, implementarea sistemului.

1.11. Modelul RAD

Modelul RAD (*Rapid Application Development*), cu varianta sa europeană ***PD*** (*Participatory Design*), se caracterizează printr-un număr redus de faze, utilizarea prototipurilor în faza de realizare şi participarea activă a viitorilor utilizatori.

Etapele prevăzute de model sunt: initializare, formularea cerințelor, proiectare, realizare, implementare.

Fiecare din etapele enunțate se descompune în faze, prezentând următoarea structură: lucrări preliminarii, sesiunea participativă (la activitatea grupului de specialişti se alătură şi utilizatorii), lucrări de sinteză (concluzii).

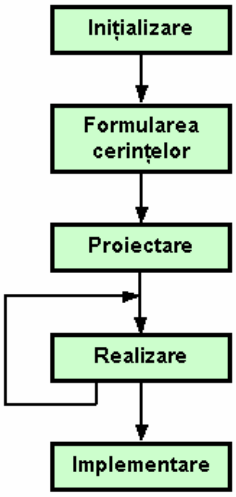

Fig. 11: *Modelul RAD*

2. Evoluția metodelor de proiectare

Evoluția metodelor de proiectare este consecința mutațiilor calitative şi cantitative în planul:

- abordării sistemelor informatice;
- dezvoltării bazei conceptuale specifice proiectării şi realizării sistemelor informatice (mai ales odată cu promovarea abordării obiectuale);
- apariției şi extinderii utilizării tehnicilor rapide de proiectare;
- evoluției permanente a limbajelor de programare;
- sporirii considerabile a complexității aplicațiilor realizate în condițiile creşterii nivelului de integrare a acestora;
- extinderii ariei de utilizare a informaticii;
- utilizării tehnicilor de gestiune în timp real.

Putem spune că, în timp, s-au conturat mai multe curente de gândire,

care au promovat şi dezvoltat anumite metode de proiectare. Este însă greu să realizăm o clasificare a acestor metode, tocmai datorită diversităţii punctelor de vedere asupra acestei probleme.

O clasificare realizată plecând de la abordările promovate de metodele de proiectare [Bro01] ne conduce la următoarea grupare:

A) Metode timpurii, metode nestructurate specifice perioadei '50 -'60.
B) Metode orientate spre ieşiri (sfârşitul anilor '60) caracterizate prin faptul că proiectarea sistemului informatic avea ca punct de plecare ieşirile pe care acesta trebuia să le asigure: rapoarte, grafice, etc. Pe baza ieşirilor identificate se determinau apoi datele de intrare şi prelucrările.
C) Metode orientate spre procese, utilizate în deceniul şapte, prezentând drept caracteristică utilizarea diagramelor fluxurilor de date.
D) Metode orientate spre date, specifice anilor '80, prezentând ca element caracteristic utilizarea diagramelor entitate-relaţie;
E) Metode orientate obiect, promovate în anii '90, caracterizate prin promovarea conceptului de obiect care încapsulează date şi metode.

Ivar Jacobson, de numele căruia sunt legate realizări deosebite în domeniul metodelor de proiectare orientate obiect, grupează în două mari categorii metodele actuale de proiectare [JCJO92]:

- Metode funcţie/date;
- Metode orientate-obiect.

O altă posibilă grupare a metodelor de proiectare ar putea fi [Opr99]:

- Metode orientate spre funcţii (metode ale descompunerii funcţionale);
- Metode orientate spre procese (metode orientate spre fluxuri de date);
- Metode orientate spre informaţii sau date (având rădăcini în ingineria informaţiei elaborată de James Martin şi în diagramele entitate-relaţie elaborate de Chen);
- Metode orientate obiect.

Curentul de gândire francez propune o clasificare a metodelor de proiectare plecând de la modalităţile în care este perceput sistemul: funcţional, sistemic, obiectual. S-a ajuns astfel la următoarea clasificare:

- Metode ierarhice (generaţia I-a a metodelor de proiectare);
- Metode sistemice (generaţia a II-a);
- Metode obiectuale, numite şi metode orientate obiect (generaţia a III-a)

2.1. Metodele ierarhice

Metodele ierarhice au la bază analiza funcțională a întreprinderii. Astfel, sistemul informatic cuprinde în arhitectura sa subsisteme definite la nivel de funcții ale întreprinderii. Cum fiecare funcție este subdivizată ierarhic în subfuncții, iar acestea la rândul lor, se descompun succesiv, până la definirea unor componente elementare uşor de programat, arhitectura sistemului informatic va urma această ierarhie, fiecare componentă a sa acoperind o subdiviziune funcțională.

2.2. Metodele sistemice

Reprezentative sunt metodele: Information Engeneering, MERISE, AXIAL, etc. Ceea ce este specific acestor metode este utilizarea teoriei sistemelor în abordarea întreprinderii. Sistemul informatic este abordat sub două aspecte complementare - datele şi prelucrările - analizate şi modelate independent, reunirea celor două modele realizându-se cât mai târziu cu putință.

Spre deosebire de metodele ierarhice, metodele sistemice acordă prioritate datelor față de prelucrări şi respectă cele trei niveluri de abstractizare introduse de raportul ANSI/SPARC: conceptual, logic, fizic.

2.3. Metodele obiectuale

Primele concepte ale tehnologiei orientate pe obiecte au fost formulate la începutul anilor '80. Cele mai cunoscute metode de proiectare orientată obiect sunt: OOD elaborată de Booch, OOA/OOD - Analiză OO/Proiectare OO elaborată de Coad, OOM autor Rochfeld, OMT elaborată de Rumbaugh aceasta fiind cea mai cunoscută şi utilizată.

La ora actuală s-a realizat unificarea tuturor metodelor de proiectare orientată obiect proiectul fiind reprezentat de ***UML - Unified Modeling Language*** (limbaj unificat de modelare).

Caracteristic metodelor orientate obiect este faptul că sistemul informatic este gândit ca un ansamblu de obiecte autonome, care se organizează şi cooperează între ele. Pentru prima dată, datele şi prelucrările (metodele) se găsesc implementate în cadrul aceleiaşi structuri, obiectul. Fiecărui obiect îi este implementat un anumit comportament definit prin ansamblul metodelor pe care le poate realiza. Datele şi prelucrările sunt încapsulate în cadrul obiectului.

Concluzii

Avantajele metodelor ierarhice constau în simplitate şi o bună adaptare la cerinţele formulate de utilizator.

Dezavantajele pornesc de la conceperea sistemului informatic conform cerinţelor analizei funcţionale, ceea ce determină concentrarea efortului de analiză şi proiectare asupra prelucrărilor, în condiţiile în care tocmai acestea sunt cele mai susceptible modificărilor în timp, modelarea datelor căzând pe un plan secund.

Avantajele metodelor sistemice decurg din promovarea tehnologiei bazelor de date. Dezavantajele sunt datorate deficienţelor care pot apărea în modelarea prelucrărilor şi a riscului apariţiei unor discordanţe între modelele datelor şi prelucrărilor.

Avantajele metodelor obiectuale decurg din posibilitatea reutilizării componentelor de program şi posibilitatea utilizării obiectelor complexe (un obiect complex fiind definit prin intermediul altor obiecte).

Dezavantajele acestor metode decurg din faptul că nu întotdeauna realitatea poate fi reprezentată numai prin obiecte.

Bibliografie

[Bro01] Brown, D. – *An introduction in Object-Oriented Analysis*, 2end Edition, John Wiley, 2001

[Bur01] Buraga, S. – *Proiectarea siturilor Web. Design şi funcţionalitate*, Editura Polirom, Iaşi, 2001

[JCJO92] Jacobson, I.; Christerson, M.; Jonsson, P.; Overgoard, G. – *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley Publishing Company, Wekingham, England, 1992

[LLD01] Luca, L.; Luca, S.C.; Despi, I. – *Informatică aplicată*, Editura Mirton, Timişoara, 2001

[Opr99] Oprea, D. – *Analiză şi proiectarea sistemelor informaţionale economice,* ed. Polirom, Iaşi, 1999

[Mun02] Muntean, A. – *Auditul sistemelor informaţionale contabile*, Editura Poliron, Iaşi, 2002

[Ros+93] Roşca, I., ş.a. – *Proiectarea sistemelor informatice financiar contabile*, Editura Didactică şi Pedagogică, Bucureşti, 1993

[Sta00] Stanciu, V. – *Proiectarea sistemelor informatice de gestiune*, Editura Cison, Bucureşti, 2000