**A Simplified Software Development Process & Others Guide**

* In process-

Prepared and curated by Robert-George Simion

[simion.robertgm95@gmail.com](mailto:simion.robertgm95@gmail.com)

(<https://github.com/simionrobert>)

Contents

[**Glossary** 3](#_Toc77101195)

[**Roles in a project** 5](#_Toc77101196)

[**Change Management Organisation** 6](#_Toc77101197)

[**1.** **A Business Analyst Perspective** 6](#_Toc77101198)

[**1.1.** **Presentation** 6](#_Toc77101199)

[**1.2.** **Definitions** 7](#_Toc77101200)

[**1.3.** **Business Analysis Stages. Etapele Analizei de business** 10](#_Toc77101201)

[**1.4.** **Descrierea etapelor Analizei de Business** 11](#_Toc77101202)

[**1.5.** **Note** 13](#_Toc77101203)

[**2.** **A Software Architect Perspective** 14](#_Toc77101204)

[**1.6.** **Presentation** 14](#_Toc77101205)

[**1.7.** **Soft skills** 14](#_Toc77101206)

[**1.8.** **Domain Modeling DDD** 15](#_Toc77101207)

[**1.9.** **Architecting Process Stages. Etapele procesului de arhitectura** 18](#_Toc77101208)

[**3.** **A Project Manager Perspective.** 19](#_Toc77101209)

[**1.1.** **Fundatiile managementului de proiect** 19](#_Toc77101210)

[**4.1.** **Alegerea metodologiei** 21](#_Toc77101211)

[**4.** **A Development Team Perspective.** 23](#_Toc77101212)

[**4.2.** **Etapele SDLC** 23](#_Toc77101213)

[**Marketing** 28](#_Toc77101214)

[**Communication** 28](#_Toc77101215)

[**Time management** 28](#_Toc77101216)

[**Finance** 28](#_Toc77101217)

[**Documentation** 28](#_Toc77101218)

# **Glossary**

|  |  |
| --- | --- |
| **Concept** | Descriere |
| **Stakeholders**  (parti interesate) | A stakeholder is anyone who has an interest in the successful outcome of the project, including project sponsors, users, business executives, managers, developers, clients, customers, vendors, and government or regulatory agencies. |
| **Project Sponsor** |  |
| **Business Team** | * + - CEO: Most authority     - CTO / CIO : Chief Technical Officer, Chief Innovation/Information Officer   **They focus on the financial side. Not technical**  Avoid: Technical Buzzwords (Microservices, Java)  Use: “The architecture I designed will ensure the high loads expected on Black Friday sales” |
| **Project Manager** | * + - Tine cont de managementul proiectului: planificare timp     - Se asigura ca totul merge cum trebuie     - Deleaga taskuri     - Recruteaza     - Removing bariers   **They focus on project success. Not technical**  **Avoid**: This is the latest and greatest pattern. We’ll be the first to test it. We can write a blog post too.  **Use**: This new tech can help us write code twice as fast and it cuts our schedule and budget. |
| **Business Analyst** | * + - Este podul care face legatura intre clienti si echipa tehnica. Traduce logica unui bussiness in cerinte tehnice. Traduce idei si nevoi pentru developer. Comunica, faciliteaza comunicarea, negociaza, organizeaza, gandeste critic. Ca BA, trebuie sa conduci intanirea.     - Cerceteaza si cauta cerintele unui business case     - Analizeaza cerintele: modeleaza (starea curenta, starea finala) si decide cum sunt acele cerinte: facubile, pot fi cumparate, exista resurse necesare     - Tipuri de Business Analyst   **Business Process Analyst**   * + - Analizeaza modelele actuale si le creeaza pe cele viitoare     - Analizeaza decalajul dintre procesul present si cel viitor   **Requirements Analyst**   * + - Face legatura intre business si departamentul IT     - Intelege procesul current     - Platit pentru a crea cerintele   **System Analyst**   * + - Proiecteaza solutia   **Data Analyst/Business Intelligence Analyst**   * + - Proiecteaza solutia   **User Experience Analyst (usability)**   * + - Proiecteaza solutia |
| **Software Architect / Solution Architect** | **Tipuri de arhitecti:**   * + - **Infrastructure Architect**     - **Software Architect**     - **Enterprise Architect**   Analysts work with the client.  Architects work with the company |
| **Product Manager /**  **Team Leader /**  **Technical Lead** | * + - Calitatea codului, code reviews, ensure standards |
| **Developers** |  |
| **UX Designer** |  |
| **QA Engineer / Testers** | * + - Eliminare bug-uri, asigurarea securitatii, asigurarea performanteri, usabilitate, calitate |
| **Marketing Manager** |  |

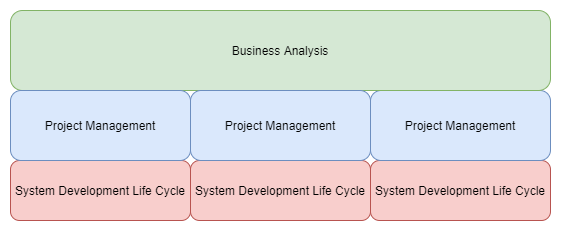
# **Roles in a project**

# **Change Management Organisation**

Exista cel putin 4 perspective de management a schimbarilor intr-o organizatie:

* perspectiva unui business analyst,
* perspectiva arhitectului
* perspectiva project managerului
* perspectiva echipei de dezvoltare, administrare, testare.

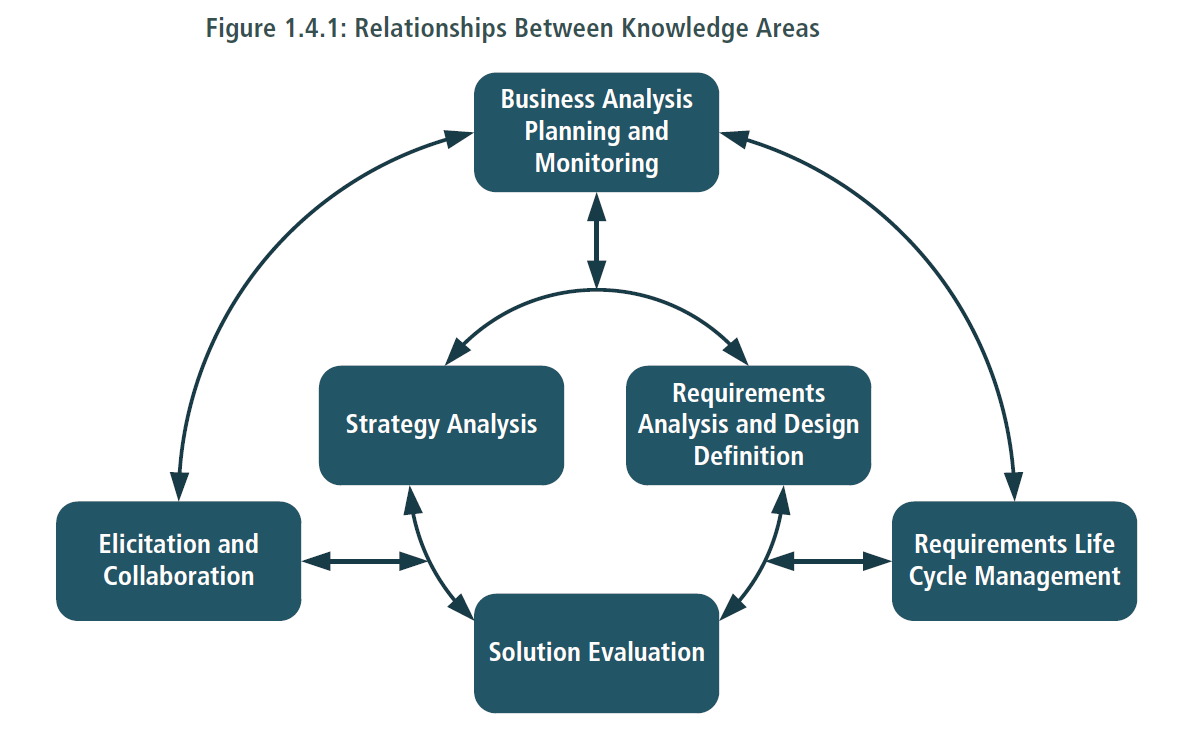
Parti din analiza de business se continua si se intrepatrund cu etapele de inceput ale celorlalte doua perspective de management.



## **A Business Analyst Perspective**

### **Presentation**

Knowledge Areas specifice expertizei analistului de business sunt:



### **Definitions**

|  |  |
| --- | --- |
| **Concept** | **Descriere** |
| **Stakeholder** (parti interesate) | A stakeholder is anyone who has an interest in the successful outcome of the project, including project sponsors, users, business executives, managers, developers, clients, customers, vendors, and government or regulatory agencies. |
| **Business Need** (nevoie, necesitate) | O problema sau o oportunitate. Nevoile de business sunt intotdeauna exprimate din **perspectiva instiutitei** si nu a unui stakeholder particular. Nevoile pot cauza schimbari prin motivarea stakeholderilor in a actiona.  O nevoie poate fi identificata la mai multe nivele ale institutiei:   * **From the top-down**: a strategic goal that needs to be achieved. * **From the bottom-up**: a problem with the current state of a process, function or system. * **From middle management**: a manager needs additional information to make sound decisions or must perform additional functions to meet business objectives. * **From external drivers**: customer demand or business competition in the marketplace. |
| **Requirement** | O cerinta este o reprezentarea utilizabila a unei nevoi.  **Goals**: efectele sistemului asupra organizatiei. Beneficii. Here is the big picture.  **Requirements**: c ear trebui sa faca sistemul  Tipuri de cerinte:   * + 1. Business requirements     2. Stakeholder requirements (User Requirements)     3. Solution requirements * Functional requirements (comportamentul solutiei) * Non-functional requirements (caracteristicile solutiei) (QoS)   + 1. Transitional requirements (temporare)       article_Martin_Table1 |
| **Business Requirement** | 1. The problem, opportunity, or constraint which is defined based on an understanding of the current state.  2. A represantation of goals, objectives and outcomes that describe why a change has been initiated and how success will be assessed.  (1 si 2 reprezinta, semantic, acelasi obiect)  They can apply to the whole of an enterprise, a business area, or a specific initiative. Business requirements recognize what is critical to the business and why it is critical before defining a solution. In some organisations, a business requirement is considered to be the high-level requirement, for which user or stakeholder requirements are then used to document the solution. Other organisations use the term business requirements to refer to any requirement that is not a system requirement.  Example: We would like to automate our customer relationship management system so that we can offer better customer services so that the customer response time improves by 70% in the next 6 months. |
| **Business Case** | Un document rezultat al etapei Change Strategy |
| **Solution Scope** |  |
| **Business Requirements Document (BRD)** |  |
| **User Requirements Document (URD)** | Stakeholders Requirements Document |
| **Software Requirements Specification (SRS)** | A Software Requirements Specification (also known as a System Requirements Specification) describes the behavior and implementation of a software application. The primary target audience for a SRS is the development team that will be required to implement the solution. |
| **Functional Requirements Document (FRD)** |  |

### **Business Analysis Stages. Etapele Analizei de business**

* + - 1. Needs Assessment
* Intelegerea proceselor
* Cerinte ce tin de process
* Identificarea solutiilor
* Identificarea celei mai bune solutii ca PROCES
  + - 1. Business Analysis Planning
* Setarea documentelor de realizat si alte planuri management
  + - 1. Requirements
* Ce face solutia
  + - 1. Proiectare
* Cum face solutia? Design baza de date, Design interfete, Design retea (ip-uri etc.)
  + - 1. Develop
      2. Test

1. **Business Analysis Planning and Monitoring**
   * + 1. **Needs Assessment / Strategic Analysis**
   1. Identify Problem or Oportunity

* Identify Stakeholders
* Investigate Problem or Oportunity
* Gather relevant data to Evaluate the Situation
* Draft the Situation Statement
* Obtain Stakeholder Approval for Situation Statement
  1. Assess the Current State of the Organization
* Assess Organizational Goals and Objectives
* Perform Root Cause Analysis on the Situation
* Assess Current Capabilities of the Organization
  1. Assess the Future State of the Organization
* Determine Required Capabilities Needed to Address the Situation
  1. Define Change Strategy. Recommend Action to Address Business Needs
* Identify Gaps in Organizational Capabilities
* Provide Alternative Options for Satisfying the Business need
* Identify Constraints, Assumptions and Risks for each option
* Assess Fesability and Organizational Impact of each option
* Recommend the Most Viable Option
* Conduct Cost-Benefit Analysis for Recommended Option
* Assemble the Business Case
  + - 1. **Planning (Part of Business Analysis Planning and Monitoring)**

2.1. Refine the Stakeholder Analysis

2.2. Create the Business Analysis Plan. Define and plan Processes

2.3. Understand the Project Context

2.3. Plan the Business Analysis Work

* + - 1. **Requirements Elicitation and Analysis** (Elicitation and Collaboration and Requirments Analysis and Design Definition)
  1. Plan for Elicitation
  2. Prepare for Elicitation
  3. Conduct Elicitation Activities
  4. Document outputs from Elicitation Activities
  5. Analyze Requirements
  6. Model and Refine Requirements
  7. Document the Solution Requirements
  8. Validate Requirements
  9. Verify Requirements
     + 1. **Solution Evaluation**
* Requirements Life Cycle Management
* Support implementation through SDLC
* Proof of concept
* Pilot
* Beta

### **Descrierea etapelor Analizei de Business**

1. **Business Analysis Planning and Monitoring**
2. **Strategic Analysis**
   1. Identify Problem or Oportunity

* Identify Stakeholders
* Investigate Problem or Oportunity
* Gather relevant data to Evaluate the Situation
* Draft the Situation Statement
* Obtain Stakeholder Approval for Situation Statement
  1. Assess the Current State of the Organization
* Assess Organizational Goals and Objectives
* Perform Root Cause Analysis on the Situation
* Assess Current Capabilities of the Organization
  1. Assess the Future State of the Organization
* Determine Required Capabilities Needed to Address the Situation
  1. Define Change Strategy. Recommend Action to Address Business Needs
* Identify Gaps in Organizational Capabilities
* Provide Alternative Options for Satisfying the Business need
* Identify Constraints, Assumptions and Risks for each option
* Assess Fesability and Organizational Impact of each option
* Recommend the Most Viable Option
* Conduct Cost-Benefit Analysis for Recommended Option
* Assemble the Business Case
  + **Analyse Current State**

**INPUT:** Business Need **OUTPUT**: Business Requirements + Current State Description

1 Intelegem problema/oportunitatea. Ce problema rezolvam? Why there is need to change?Analizam magnitudinea

2 Problema ... are efectul de ... cu impactul ...

- The need drives us to explore different aspects of the current state in order to validate the need to change

3 Intelegem goals si obj organizatiei

4 Intelegem procesele din org.

5 Root Cause Analysis: Un analist pune sub semnul intrebarii presupunerile si constrangerile inglobate in business need si se asigura ca se adreseaza problema corecta. Five Whys technique

* + **Define Future State**

**INPUT:** Business Requirements **OUTPUT**: Business Objectives + Future State Description

- Scopul este definitrea capacitatilor (conditions, goals, objectives, results) necesare pentru a realiza Business Need

- Furnizeaza o definitie clara a rezultatelor care vor satisface Business Needs

- Descrierea Future state include ce parti trebuie sa se schimbe pentru a satisface Business Needs; intrarea intr-o noua piata, achizitie, schimbarea unui pas intr-un proces

- Formulata si descrisa astfel incat sa permita identificarea mai multor strategii

- Un Future State poate fi definit in termeni de obiective si goals SMART: increase customer satisfaction, comply with new regulationsm, increase number of high revenue customers in the 30-45 age bracket by 30% within 6 months

* + **Define Change Strategy**

- High level plan of key activities and events that will be used to transform the Enterprise from Current to Future State

- Gap Analysis between Current and Future State

1 How to add future state capabilities?

2 Provide alternative options for satisfying the business need

3 identify constraints, assumptions and risks for each option

4 assess risk for each option

5assess feasability impact for each option

6 recommend the most viable option

7 conduct cost/benefit for recommended option

8assemble the business case

1. **Planning (Part of Business Analysis Planning and Monitoring)**

2.1. Refine the Stakeholder Analysis

2.2. Create the Business Analysis Plan. Define and plan Processes

2.3. Understand the Project Context

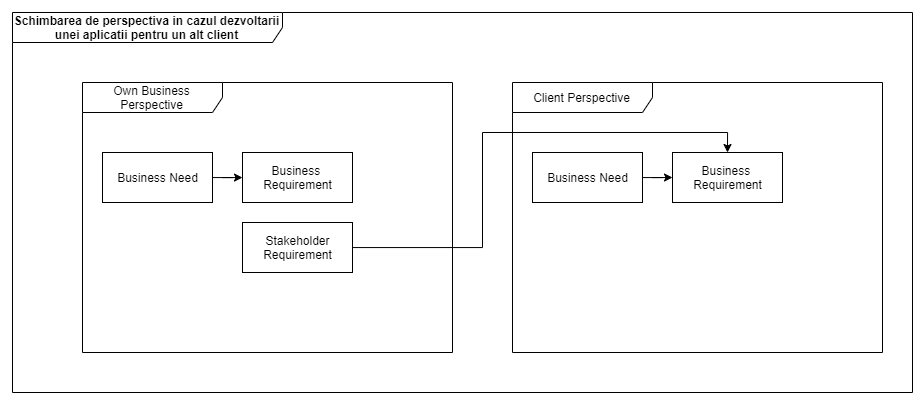
2.3. Plan the Business Analysis Work

1. **Requirements Elicitation and Analysis** (Elicitation and Collaboration and Requirments Analysis and Design Definition)
   1. Plan for Elicitation
   2. Prepare for Elicitation
   3. Conduct Elicitation Activities
   4. Document outputs from Elicitation Activities
   5. Analyze Requirements
   6. Model and Refine Requirements
   7. Document the Solution Requirements
   8. Validate Requirements
   9. Verify Requirements
2. **Solution Evaluation**

* Requirements Life Cycle Management
* Support implementation through SDLC
* Proof of concept
* Pilot
* Beta

### **Note**

1. Business Requirements, Stakeholder Requirements, Solution Requirements, Transitional Requirements sunt cerinte vazute din perspectiva fiecarui grup in parte. **Aceasta perspectiva se adapteaza pentru fiecare proiect in parte**. Exemplu: Daca firma mea dezvolta sisteme pentru alte firme, cerintele de business ale proiectului in cauza sunt vazute din perspectiva firmei pentru care dezvolt, nu din perspectiva firmei mele. In acest caz, crearea proiectului este, in fapt, o cerinta de business sau de tranzitie care aduce valoare/rezolva un business requirement al propriei firme.



1. 2. Q: Este necesar sa avem documente separate pentru cerinte business, cerinte functionale si cerinte non-functionale?

R: In unele cazuri se face un document BRD care le cuprinde pe toate, in alte cazuri acestea sunt separate in documente separate. Scopul documentelor este transmiterea de knowledge. Este greu sa manageriem aceste documente astfel incat sa pastram proprietatea de traceable si contextul fiecaruia.

1. The stakeholders are the main source of requirements. They have specific needs that the analyst must identify. This is easier said than done: often stakeholders are not quite sure what they need and they often don't know how to express what they need. It is the analyst's job to help uncover the requirements of the stakeholders.
2. Requirements are focused on the need. Designs are focused on the solution. The same

techniques are used to elicit, model, and analyze both. A requirement leads to a

design which in turn may drive the discovery and analysis of more requirements. The shift in focus is often subtle.

## **A Software Architect Perspective**

### **Presentation**

Developers know what can be done

Software Architects knows what should be done

Architect should test new technologies as a proof of concept for the development team and should support them.

### **Soft skills**

1. Always understand the business before architecting. Actual systems are not standalone. They are deeply integrated in the organization.
2. Work for your client’s client.
3. Always keep in mind what is the thing that really matters to the person you are talking to. Try to be in their shoes.
4. Should focus on goals (benefits) not requirements.
5. The architecture will change
6. Be smart, not right!
7. The architect does not have any authority. He must be able to influence without authority through listening, assuming you are not the smartest in the room, not attacking back, coming with logic and facts
8. Should be able to speak to large audiences. Don’t read.

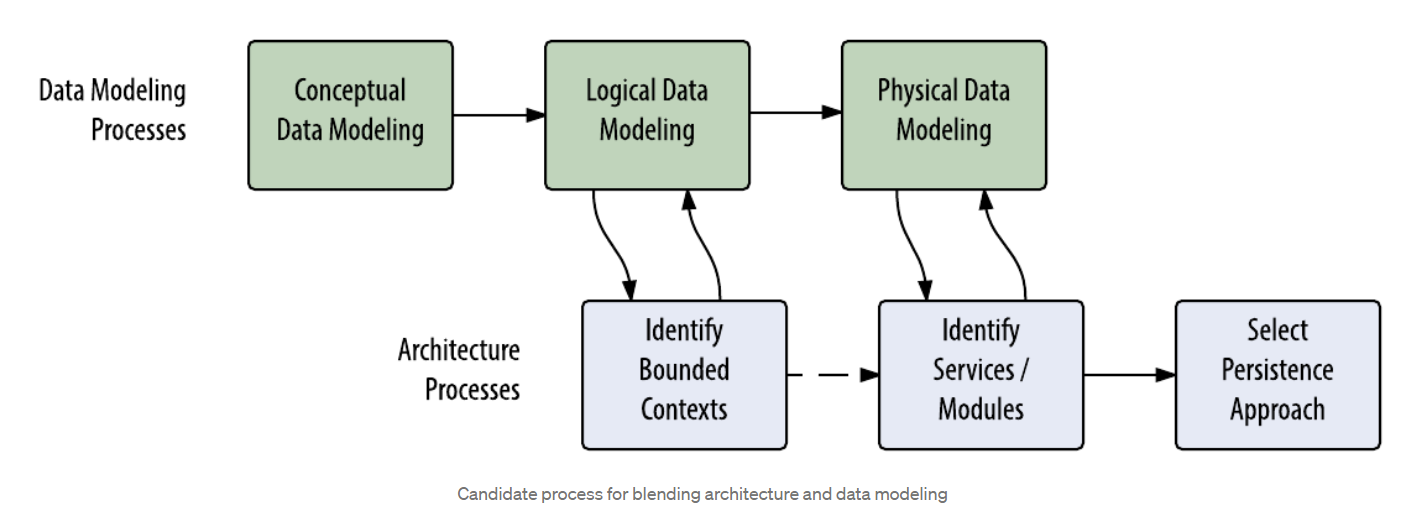
### **Domain Modeling DDD**

Domain modeling is done to ensure business rules.

Take into consideration that DDD is useful on projects that have a lot of business rules.

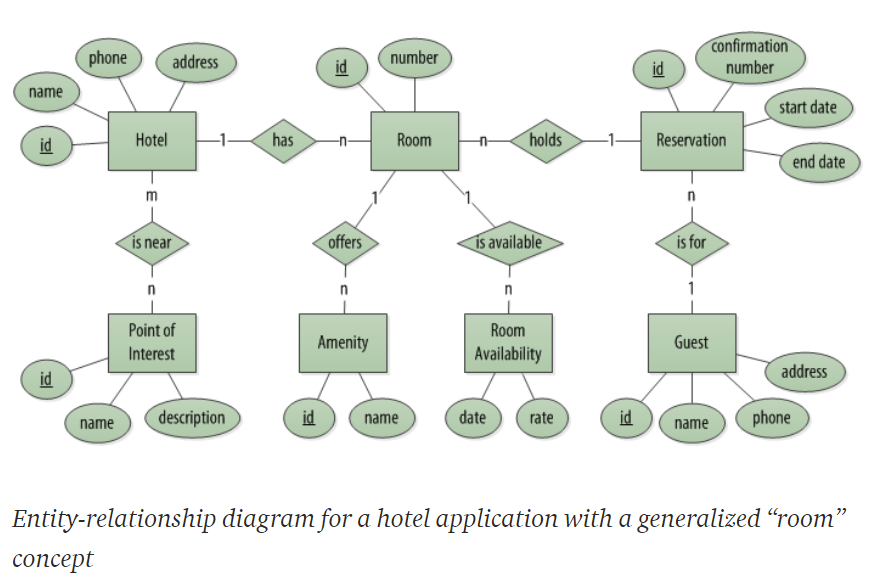
Otherwise, use the service layer and transactional scripts to enforce them.

Taking this advice into account, the below steps are also useful to discover the entities and the business rules.



1. Write the Story
2. Discover the entities
3. Draw the relations between them

* Questions to answer: does a concept need the other? (strong association) Who cares about the other more? (loose association)



1. Write the requirements/**invariants**

Invariants = business rules that are always consistent

e.g.

What does it mean for an Order to be an Order?

A question must have at least 2 choices.

1. Create the entities.

Models change their meaning and properties depending on a Bounded Context.

Instead of exposing entities, expose only those operations thought by domain experts.

Private setters, internal classes

1. Create the aggregates.

Aggregate= an aggregate is a construct to organize BRs, not to be a representation of a state; an aggregate is a collection that communicates that everything is okay when a change happens.

Consider:

* Make aggregates small
* Performance when creating aggregates

Rules:

Aggregates should construct its entities based on values provided by their functions.

You can use ids and lazy load entities

1. Identify Bounded Contexts

Context = the setting in which a word or statement appears that determines its meaning

Bounded context = the conditions under which a particular model is defined and applicable

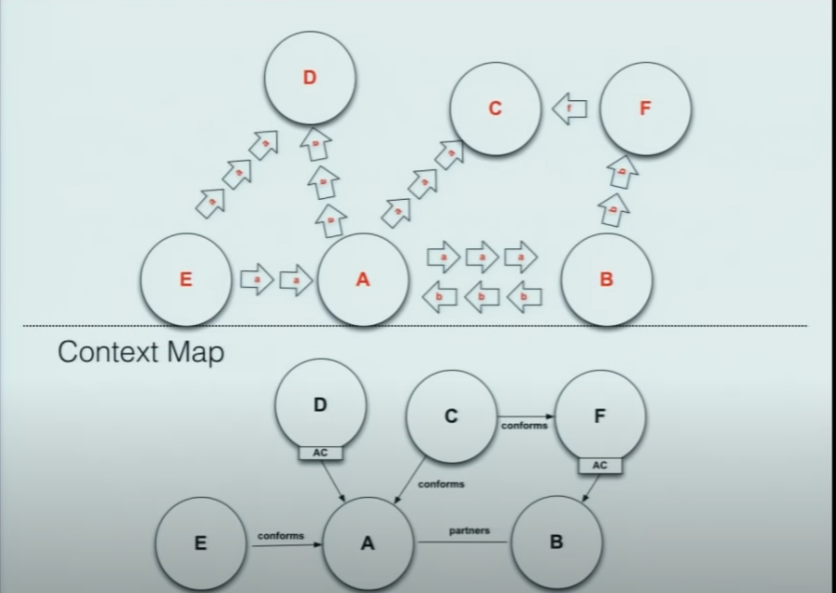
How to find bounded context? Use some heuristics

* + - * 1. Find the models that change together.
        2. Do you need transactional consistency for updating the properties?
        3. Split the model according to teams and departments: Sales, Inventory, Shipping, Marketing

Bounded Contexts communicate with each other through

* Commands (do something, 1 direction only)
* Domain Events (something of importance has happened, multiple directions)

1. Context Maps



1. Persistence Modelling

CQRS Pattern

Queries can ignore domain rules. Hence, you are free to query tables in DTOs directly.

Data Validation has less to do with data than it does with commands

1. Use Ubiquous Language. Develop an obsession for the domain language of the experts. Name methods according to the domain experts.

### **Architecting Process Stages. Etapele procesului de arhitectura**

1. **Definirea/Intelegerea cerintelor**

* Functionale:
  + definite deja de catre analist
  + Ce trebuie sa faca sistemul
  + Business flows/processes
  + De ce are nevoie clientul
* Nonfunctionale :
  + What the system should deal with
  + definite de catre architect
  + #users, load, performance, concurrent requests/users, …
  + Foarte importante
  + Comunicare intre architect, analist si client

1. **Definirea si Maparea componentelor**

* Taskurile pe care trebuie sa le faca sistemul

1. **Alegerea comunicarii intre componente**
2. **Design-ul componentelor**

* Application Type
* Technology stack
* Architecture
  + Layers: Interface Layer, Business Layer, Data Acess Layer

1. **Scrierea documentului de arhitectura**

* What should be developed and how
* Audience: everyone
* As simple as possible
* Continut:

Background

* 1 page max
* Business point of view
  + System role
  + Expected business impact
  + Reasons

Requirements

* 1 page max
* Brief requirements

Executive Summary

* 3 pages max
* High level view of the architecture
* Charts & diagrams
* Done at the final stage

Architecture Overview

* 10 page max
* Presents to the dev team the architecture components **business logic pov**

Components Drill-Down

* Most of the work, core of the document
* For the dev team
* Describe the API methods

1. **Support the team**

* The architecture will change a lot

## **A Project Manager Perspective**

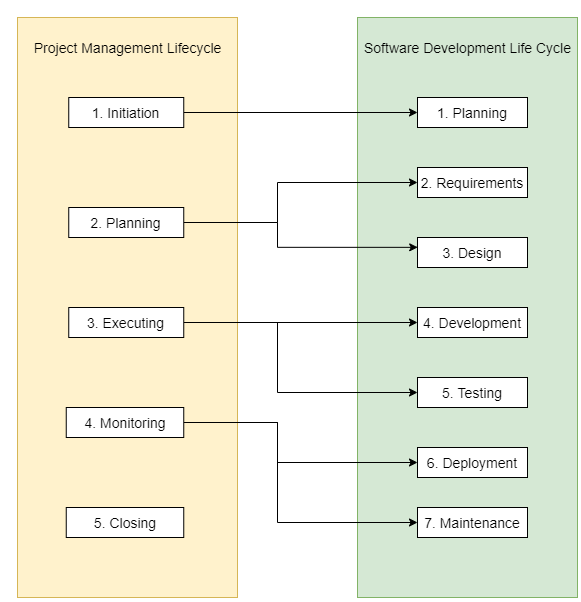
### **Fundatiile managementului de proiect**

Managemetul de proiect reprezinta aplicarea cunostintelor, abilitatilor, instrumentelor si tehnicilor pentru a indeplini cerintele proiectului atat ca buget, timp si asteptari.

Managerii de proiect au fost invatati ca intotdeauna exista 2 tipuri de cicluri de viata:

* Ciclul de viata al proiectului (Project management)
* Ciclul de viata al produsului (SDLC)

Aceste 2 cicluri interactioneaza una cu cealalta si se intrepatrund. Ele presupun tipuri diferite de proceduri si procese, unele facute de project manager, altele de business analist si de echipa de dezvoltare.



**Project Management Lifecycle si Responsabilitatile unui manager de proiect**

Managerul de proiect este responsabil ca proiectul sa aduca valoare companiei.

1. Initierea proeictului

* capteaza asteptarile clientilor
* identifica ce trebuie facut
* identifica scopul proiectului si livrabilele
* identifica bugetul si resursele
* identifica persoanele implicate
* organizeaza informatiile intr-un singur loc pentru aprobare
* Timeline pentru documentarea procesului de crearea a proiectului
* Tooluri
* Documentatii
* Liste
* Planuri pentru riscuri si provocari
* Masurarea progresului

1. Planning

* Creaza bugetul
* Creaza un plan de proiect detaliat (major milestones + deliverabiles)
* Estimeaza timpul necesar realizarii proiectului
  + Comunica sefului ca iti trebuie timp de analiza
  + Descopune taskurile mari in bucati mici, estimeaza-le si insumeaza-le
  + Dubleaza timpul previzionat. Adauga timp pentru mici pauze.
  + Reestimeaza atunci cand o noua functionalitate apare. Si atunci o poti accepta sau refuza => Continous project management
  + Comunica sefului atunci cand intervin schimbari in timp util
  + Daca realizezi ca o sa fie timp in plus, termina proiectul intai si abia apoi adauga noi functionalitati
  + Creaza mai intai MVP (vq); ceea ce face mai important proiectul prin liste si abia apoi celelalte; regula 80-20
* Specifica taskuri clare
* determina rolurile si responsabilitatile echipei
* planuieste pentru risc si schimbare
* comunica toate aceste informatii catre echipa

1. Executia

* monitorizeaza progresul
* Mentine echipa motivata
* Mentine echipa la current cu programul si livrabilele
* indeparteaza obstacolele
* Face ajustari la proiect
* actualizeaza starea proiectului catre persoanele implicate
* Schimbarile apar foarte des
* Urmarirea problemelor si riscurilor
* Escaladarea problemelor care nu pot fi rezolvate; atunci ar trebui sa vina atat cu problema cat si cu solutia
* Pm se asigura ca lucrurile vor fi facute
* Managerul de proiect ghideaza echipa, atribuie taskuri catre personae, nu este seful lor.
* Influenteaza fara autoritate
  + Se asigura ca echipa intelege taskurile si scopurile prin incurajarea lor sa puna intrebari
  + Imputernicirea echipei pentru a-si lua propriile decizii pentru ca asta ii motiveaza si ii responsabilizeaza
  + Verifica in mod continuu cu echipa pentru a se asigura ca raman pe drumul cel bun
* Masurarea si comunicarea progresului catre echipa

Tip: Keep calm in uncertain times. This inspires your team

1. Inchiderea proiectului

* Ia feedback
* Aloca timp pentru reflectare (retrospectiva)
* Multumeste si celebreaza (email, party)
* se asigura ca taskurile sunt complete
* se asigura ca clientul e satisfacut
* documenteaza tot

### **Alegerea metodologiei**

SDLC este un framework, o “directiva”.

Metodologia este o instanta a SDLC, un model aplicat. Metodologia este un set de principii si procese ale unui proiect in ciclul lui de viata. In functie de modelul SDLC ales, fazele SDLC pot fi adoptate secvential sau in paralel. Modelul poate sa includa tranzitii repetitive sau iteratii prin fazele SDLC pana la faza finala.

* Waterfall
* Agile
* Lean
  + Eficienta, maximizarea valorii
  + Optimizarea proceselor
* Six Sigma
  + Originar din statistica; procesele trebuie sa aibe 99.9996% calitate
  + Folosit pentru a reduce variatiile prin asigurarea calitatii proceselor
  + Procesele care nu se ridica la standardul Six Sigma trebuie sa fie optimizate
* Lean Six Sigma

**Titluri pentru project manager in functie de domenii**

* Project assistant
* Project manager
* Project analyst
* Project lead
* Program manager
* Operations analyst
* Operations manager
* Chief operating officer
* Scrum master
* Product owner

Managementul de proiect poate fi facut in orice domeniu fara a sti implementarea proiectelor din acel domeniu.

**Structura organizatiei**

* Clasica/ierarhica – fiecare persoana are un singur sef
* Matrice – o persoana poate avea mai multi sefi

**Cultura organizationala**

Reprezinta ansamblul valorilor companiei. Personalitatea organizatiei. Toate lucrurile care se intampla in interiorul organizatiei in afara de cele profesionalte.

Cunoasterea culturii organizationale te ajuta sa gestionezi comunicarile si sa te ajute sa stii la ce sa te astepti. Ex: O companie poate sa puna accentul pe rezultate/viteza, nu pe colaborare

**Change management**

PM este responsabil de procesul de livrare a proiectului finalizat și de a determina oamenii să-l adopte prin:

* Ownership
* Urgency and importance

Metode practive:

* Demonstratii
* Intrebari
* Implicare
* Surveys
* Training

Rezultatele se materializeaza prin persoanele “avocati” care se implica si sunt responsabili pentru terminarea taskurilor proprii

## **A Development Team Perspective**

### **Etapele SDLC**

**Planning**

**Rezultat:** Business Case Document

**Responsible**:

* + 1. Executives sau upper level management
    2. Subject matter experts
    3. Business partner(s)
    4. Bussiness Analyst

**Etape:**

1. Analiza initiala. Intelegerea problemei
2. Determina **toate** solutiile potentiale (inclusiv a nu face nimic). Vinde solutia. ROI, Risk, Timp, Bani, Analiza cost/beneficii
3. Scrie documentul
4. Prezinta documentul celor care iau decizii

**Exemplu:**

* + - Problema este ca ai nostrii client renunta la serviciile noastre intrucat dureaza prea mult si este anevoios procesul de inrolare. Dorim ca ai nostrii clienti sa parcurga procesul de inrolare mult mai usor astfel incat sa-i atragem. Solutiile sunt … ROI ale acestora sunt … Recomandarea este …

**Descriere:**

* + - Care este problema ce incercam sa o rezolvam? Care sunt componentele critice care se rezolva prin acest proiect?
    - **Problema + Solutiile propuse + ROI ale acelor solutii + Recomandare solutie**
    - Care sunt obiectivele proiectului? Unde este ROI-ul? Care sunt partile critice ale sistemului care vor fi rezolvate prin acest proiect?
    - Care se considera a fi un success? Cum il masuram?
    - Creat de cineva care nu ia el deciziile. Scopul este in a vinde solutia propusa
    - Odată ce aceste activități sunt complete, iar directorii, sponsori au examinat analiza și au luat decizia finală de a continua, managerul de proiect va începe să formeze echipa și să revizuiască lista cerintelor proiectului

**Requirements Definition**

**Responsible**:

* + - Bussiness Analyst
    - Project Manager

**Standard:** BABOOK

**Descriere:** Scopul este intelegerea procesului curent (“as-is”), ce se vrea de la cel viitor (“to-be”) si analiza a ceea ce trebuie facut (“gap-analisys”)

**Etapele:**

* 1. **Requirements Gathering/Elicitation** (Pentru a intelege cat mai bine)
     1. Alegerea tipului de interviu (oral, job shadow)
     2. Pregateste-te pentru interviu:
        1. **Pregateste o foaie cu:**
           1. numele si rolul celui intervievat
           2. Lista intrebarilor deschise (nu trebuie ca raspunsul sa fie da sau nu)
           3. Interviul trebuie sa fie ca o conversatie si nu un survey. Pune intrebari care duc la altele. Aceasta tehnica vine cu timpul
        2. Trimite agenda
        3. Best practices
           1. Trebuie sa arati ca o intalnire este absolut necesara si valideaza ca stii despre ce se vorbesti
           2. Arata-i celui intervievat ce se va face cu informatia de la acesta (ex: informatie acumulata o sa o documentez si ma ajuta foarte mult)
           3. Match the pace of the interviewee (talk slow or fast)
     3. Interviul
        1. **Intelegerea sistemului curent si ce se vrea de la cel viitor.** Culegerea informatiilor de la utilizatori, documentatii. Care sunt nevoile lor?
        2. Din acestea, analistul creaza cerinte. Cerintele trebuie sa fie specifice, clare, masurabile. Ce trebuie sa faca un sistem? Ce proprietati are sistemul? Care sunt constrangerile? Care sunt Bussiness Rules? Care sunt activitatile utilizatorilor? Care sunt rolurile lor?
        3. Pentru refacerea unui system, nu este de ajun si imitarea sistemului, altfel nu ar avea niciun sens. Vezi si ce e de imbunatatit. Asta e mai important.
        4. Rezerva timp pentru a trage concluzii
     4. Trimiteti multumiri pentru participarea la interviu ca si-a luat din timp si aratati cum informatia va ajuta la crearea unor cerinte de calitate si aratati-i pasii urmatori din proiect. Persoana sa plece cu o experienta pozitiva si nu o pierdere de timp din care nimic nu se va schimba
  2. **Requirements Analysis** (Pentru a facilita intelegerea cerintelor pentru alti oameni)
     1. Anularea ambiguitatilor cerintelor gasite in etapa anterioara
     2. Usurarea intelegerii cerintelor prin modele vizuale
     3. **Creem modele vizuale pentru starea curenta**
     4. **Creem modele vizuale pentru starea finala**
     5. **Gap Analisys:** Cerintele fill the gap btw these 2 models
     6. Dupa analiza, se poate itera procesul 1 de culegere a informatiilor
  3. **Requirements Specification** (Pentru traduerea cerintei dpdv al userului in pdv al sistemului)
     1. Elaborarea documentului **BRD**
     2. Categorisirea cerintelor: functionale, non-functionale, constrangeri
     3. Impartim cerintele in mai multe cerinte singulare (fara “si” in cerinte). Clarificarea cerintelor. Inlaturarea generalitatii cerintelor (trebuie sa fim specifici): “Sistemul trebuie sa”
     4. Atribuirea atributelor la cerinte (uid, prioritati, complexitate, valoarea adusa la business, status, sursa cerintei)
     5. Cerinte SMART (simple, clare, 1 lucru, testabile, masurabile)
     6. Cerinte acompaniate de comentarii
  4. **Requirements Approval**
     1. Aprobarea documentului **BRD**
     2. Do a meeting with the business, with the technical team and with the sponsor
     3. Create an agenda
        1. Explica motivul sis punele agenda, durata etc.
        2. Trecem prin fiecare cerinta
        3. Actualizeaza cerintele. Adauga cerinte noi doar daca sunt critice. De tinut minte: Nu este timpul mereu sa adaugi cerinte noi decat dupa ce ai terminat sistemul.
        4. Discuta intrebarile imediat.
        5. Tehnic: Identifica problemele tehnice care pot aparea
  5. **Project Review Feedback**
     1. Surveys (Ce obstacole ati intampinat? A avut succes? 3 lucruri de imbunatatit. Alte comentarii?)
     2. Meeting for feedback
     3. Post meeting review (Ce putem schimba in viitor, ce procedure se pot schimba?)

**Rezultat: BRD (Bussiness Requirements Document)**

* + - O parte a “Caietului de sarcini”
    - Descrie procesul curent si pe cel viitor
    - Defineste cerintele de bussiness si alte tipuri de cerinte high level; diagrame use-case si diagrame high-level
    - Include toate cerintele dorite de client.Explica povestea fiecarei cerinte
    - Defineste scopul si obiectivele proiectului
    - Explica cum satisface solutia nevoile clientilor
    - Explica cerinte functionale fara jargon tehnic, ce pot fi intelese de catre clienti si business
    - Revizuit si acceptat de client la final

**Proiectare/Design**

**Responsible**:

* + - Software Architect
    - UI/UX Designers
    - Project manager

**Approval:** Clients and Stakeholders

**Descriere**:

* + - Elaborarea documentelor SRS si/sau FRD
    - Transformarea cerintelor in functionalitati si features
    - Crearea de 50+ task-uri si prioritizarea lor
    - Crearea de Diagrame, Interfete mock-up, Prototipuri
    - Design-ul bazei de date
    - Design-ul UI

**Rezultate:**

* + 1. **SRS (System Requirements Specification)**
       1. Specifica clasele de utilizatori si rolurile lor
       2. Specifica framework-ul si tehnologiile
       3. Specifica fiecare actiune a utilizatorului cu sistemul
          1. Use-case
          2. Descriere a actiunii utilizatorului (IN)
          3. Ce face sistemul
          4. Cum raspune sistemul (OUT)
       4. Specifica modulele si submodulele si o scurta descriere a functionalitatii acestora (features/cerinte functionale)
       5. Specifica cerintele non-functionale
       6. Specifica interfetele cu exteriorul (utilizator, alte sisteme)
       7. Prioritizeaza cerintele functionale
          1. Adaugarea atributelor la cerinte: ID, criteriu de acceptare, complexitate, status, valoarea adusa sistemului
          2. Prioritizarea cerintelor: care sunt cele mai importante cerinte (Mereu vor fi prea multe cerinte raportate la timpul si resursele avute): Valoarea ei (critical, important, bine de avut), Costul de dezvoltare. Reprioritizeaza dupa fiecare iteratie
       8. Ajuta echipa de dezvoltare sa inteleaga proiectul si ce sa faca si cum interactioneaza fiecare parte cu sistemul
    2. **FRD (Functional Requirements Document)** 
       1. Specifica toate cerintele functionale si descrie ce fac
       2. Unele companii nu creaza FRD, dar utilizeaza BRD fiind destul de detaliat. La fel si SRS/SRD.
       3. Nu include diagrame use-case

**Developing**

**Responsible**:

* + - Product Manager / Team Leader / Tech Lead
    - Developers
    - Security Engineer
    - Project manager

**Testing**

**Responsible**:

* + - Testers
    - QA Engineers
    - Project Manager

**Descriere**:

* + - Unit testing
    - Black box testing based on client requirements
    - User acceptance testing performed by the client

**Deployment**

**Responsible:**

* + - DevOps Engineer
    - Administrator Engineer
    - Project manager

**Descriere**:

* + - CI/CD, Kubernetes etc.

**Maintenance**

**Responsible:**

* + - DevOps Engineer
    - Administrator Engineer
    - Project manager

**Descriere**:

* + - If the client required any new features based on present situation or he may identifies any defects then he will send change request (CR) to the organization.

# **Marketing**

Furnizeaza solutii, nu cee ace stii sa faci.

Fa voluntariat pentru a intalni oameni si a construe relatii.

# **Communication**

Comunicarea cu non-tehnici

* + - Concentreaza-te pe rezultate, nu pe proces
    - Fii onest. Creaza incredere
    - Elimina jargonul

# **Time management**

Make your time intentional.

Focus on tasks with no distractions. Eg. Pomodoro technique. If they are, then put them on a waiting list.

# **Finance**

# **Documentation**

1. <https://www.sdlcforms.com/UnderstandingSDLC.html>
2. <https://thebusinessanalystjobdescription.com/brd-vs-srs-vs-frs-detailed-comparison/>
3. <https://www.batimes.com/articles/the-quest-for-good-requirements.html>
4. <https://aoteastudios.com/2012/05/the-babok-how-different-knowledge-areas-come-into-play-in-practice/>
5. The Complete Guide to Becoming a Software Architect Course, Memi Lavi
6. Business Analysis Fundamentals Course
7. Foundations of Project Management by Google
8. Tim Corey C# - Youtube
9. Manual de branding personal, Dan Schawbel
10. <https://jscarp.medium.com/data-model-meets-world-part-iii-data-vs-architecture-c2b111165b9a>
11. + \*\*[Clean Architecture and DDD](https://github.com/Sairyss/domain-driven-hexagon)\*\*: Recommendations on how to design software applications.
12. [Domain Driven Design](https://lostechies.com/jimmybogard/2010/02/04/strengthening-your-domain-a-primer/: Recommendations on how to think about models and develop with DDD patterns.
13. [Microsoft Microservices Architecture ebook](https://dotnet.microsoft.com/download/e-book/microservices-architecture/pdf): Recommendations on how to develop microservices using CQRS, Event Sourcing and Clean Architecture patterns.
14. [Strengthening your domain](https://lostechies.com/jimmybogard/2010/02/04/strengthening-your-domain-a-primer/)
15. [DDD Notions](https://blog.sapiensworks.com/tags.html): DDD notions explained
16. [DDD and Microservices: At Last, Some Boundaries!](https://www.youtube.com/watch?v=sFCgXH7DwxM&t=2266s): InfoQ DDD and Microservices: At Last, Some Boundaries!
17. [Keynote - Udi Dahan - DDD Europe 2020](https://www.youtube.com/watch?v=-iuMjjKQnhg&t=3281s)
18. [DevTernity 2019: Ian Cooper – The Clean Architecture](https://www.youtube.com/watch?v=SxJPQ5qXisw)
19. [DevTernity 2017: Ian Cooper - TDD, Where Did It All Go Wrong](https://www.youtube.com/watch?v=EZ05e7EMOLM)