1. Requirement analysis

Projects fail due to insufficiencies in requirements - Standish group analysis

1. Incomplete requirements
2. Insufficient user involvement
3. Low capacities
4. Changing requirements
5. Unreliable estimates based on unstable requirements

Imp. features of req.

1. Reqs are testable
2. Reqs describe a problem to be solved
3. In level details, reqs are above system specifications
4. Also
    a. Consistent
    b. Implementable
    c. Necessary and prioritized
    d. Usable and useful

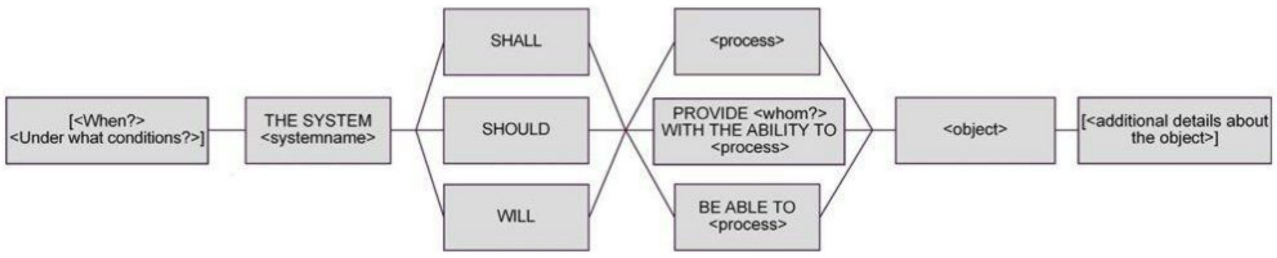
Types of reqs

1. System reqs (User, functional, Architectural, Quality)
2. Cross-sectional reqs (Operating, Migration, Deployment, Organisational)

3 steps for req. analysis

1. Identification of Stakeholders
   Anyone with possible impact on proj. Get contacts
2. Developing the high level system scope :
   Def. System Scope, Def. System context, Exclude everything else
3. Define, validate and prioritize requirements

Requirements creation with **SOPHIST template**



----

2. Quality Management

Quality: The degree to which a set of inherent characteristics fulfills requirements
Quality Management: Coordinated activities to guide and lead an organization.

Guidance and leading concerning quality typically include defining the

- quality policy
- quality objectives
- quality planning
- quality control
- quality assurance
- quality improvement

Quality assurance

1. **Constructive QA** : Prevention (spend more here for less overall QA costs)
   Norms, Standards, Project management, Soft. Eng. Training, Education
2. **Analytical QA** : Verification (spend more here for moderate overall QA costs)
   Deliverables, Processes, Audits, Documents.

Prioritization of tests (4 aspects. Test strategy imp)?

----

3. Designing Architectures

Architectural principles.

5 elementary principles

- Separation of concerns
  Divide into subparts. Each part has one task and only one.
- Minimization of dependencies
  Min. deps btwn comps. Strong cohesion, loose coupling. Avoid cyclic deps
- Hiding information
  Encapsulate internal knowledge. Expose only interfaces need for other users.
- Homogeneity
  Similar sols to solve sim. probs. Similar dims and units in same struct. level
- Zero redundancy
  Tech. and logic fn'ality by different comps in only loc. Identify and isolate shared fns in sw comps and realize in one place.

5 secondary principles

- Differentiation of software categories
  Assign precisely one software category to each software element.
- Subdivide into Layering
  Sub-divide into layers. Each layer same fns. Interacts with and below layers

- Design-by-contract
  Fully define in-terms of set of interface contracts
- Data sovereignty
  Make sure exactly 1 sw element is resp. for each element in data resources
- Re-use
  Use existing (standard) sols, design patterns and ref. architectures

Quasar component definition – six features of a component

1. Exports one or more interfaces
2. Imports other interfaces
3. Hides the implementation and can therefore be replaced by another component that exports the same interface
4. Is suitable as a unit that can be reused.
5. Can contain other components.
6. Is, together with the interface, the key unit in terms of design, implementation and planning.

Interface contains : Interfaces, Types, Results, Contracts

Basic software categories (sw blood groups)

1. 0 software : Independent of application and technology; ideal for reuse; Lib for Strings and Containers
2. A software : Determined by the logical application; Independent of technology; Booking, employees.
3. T software : Independent of the logical application; Reusable if the same technical component is used; Db access layer
4. AT software : Concerned with technology and application; difficult to maintain; resists changes. reuse unlikely.
5. R software : Pure transformation; screen format in XML
6. C software : Configuration: brings the different categories together; ex. main

Strong cohesion : Within component
Put elements with related content into a single component. Create a clearly defined, precise area of responsibility.

Loose coupling : Between components
Minimize dependencies (narrow interfaces, few imports). Minimize assumptions about other components.

Typical errors while breaking to comps.

- Mixing of A and T : Does business logic and depends on T
- God components : more than one cat, >1 purpose. Can’t put in one cat
- Distributed resps : Resps dist across comps. Comps can’t be dev’ed seperately
- Spaghetti design : Dependencies have not been reduced

Transport objects : containers for data across layers, processes and systems. Reduces method calls and decouples comps.

TOs vs Entites
- Decouples vs higher coupling
- High control over which data sent vs low control
- Low perf vs less secrecy of internals.

Quasar Architecture views - designing s/w arch. 3 main arch. views!
- A arch : Free of technical, product related practical constraints. Dev'ed anew for each proj. Structure of sw from app perspective.
- T arch : Describes the VM on which the sw made with A arch runs. Reuse possible. Combines A and TI. Templates for A code.
- TI arch : System sw. Hardware, prog langs used. Product with versions.

Standard (bottom) AT -> R -> T -> A -> 0
3 layers : Dialog, App. core, Persistence management. : Dev'ed in parallel

Best design practices : KISS, ME. Brace yourself on reqs., Work iteratively., DRY!

---

4. Content Management and High performance scenarios

Load : number of arriving requests per time
Response time : System perspective: time for processing of a request
Throughput : number of (successfully) processed requests per unit of time

Scalability :
Vertical scalability : Increasing of peak throughput by increasing the system components' performance
Horizontal scalability : Increasing of peak throughput by distribution of tasks around more system components

Load balancing:
L4 vs L7

Two kinds of state : Resource state (changes slowly, not by norm user), Session state.
Session handling alts : Former client state (hidden vars, save auth headers, cookies) vs Server state (session key).

Alts in Server side session:
- Session stickiness : Sessions reqs to same server. By IP (bad) or Session ID (L7 load balancer). Terminate on server loss. Simple concept.
- Central sessions : Central comp. saves session states. Read vs Write access.
- Mobile sessions : Session data address in cookie. Get it from it.

SCM : Software Configuration management.

Has Build, Process, Change, Release, Environment management.

SCM used for both Source Code Mang. and Soft. Config. mang.

Continuous integration vs Bigbang integration.

----

5. Quasar Enterprise Architecture

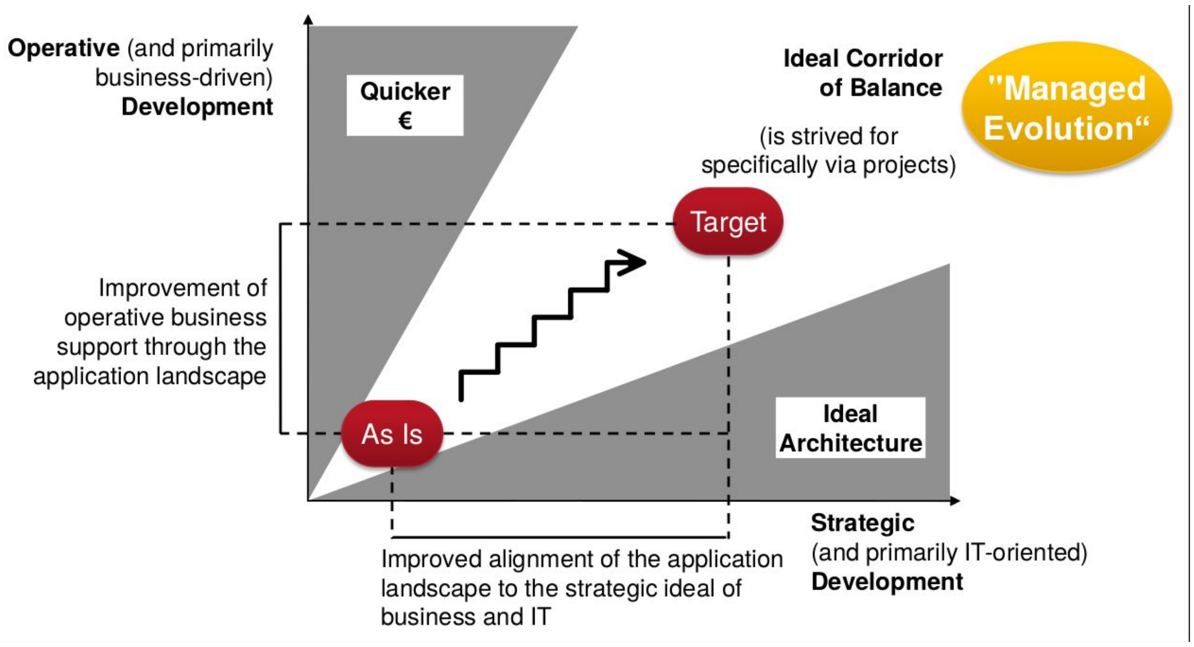
Enterprise Architecture Framework : provides principles and practices for creating and using the architecture description of a system.

- It structures architects' thinking by dividing the architecture description into domains, layers or views, and offers models ... for documenting each view.
- IAF abstraction layers : Physical, Logical, Conceptual, Contextual

4 step approach to create a service oriented corporate architecture.

1. Understanding the business
2. Creating the ideal
3. Ascertaining and appraising the actual situation
4. Creating the target architecture

The architect plans the development of the application landscape as a balance between these two requirements


Operative (and primarily business-driven) Development
Quicker €
Ideal Corridor of Balance
(is strived for specifically via projects)
"Managed Evolution"
Target
Improvement of operative business support through the application landscape
As Is
Ideal Architecture
Strategic (and primarily IT-oriented) Development
Improved alignment of the application landscape to the strategic ideal of business and IT

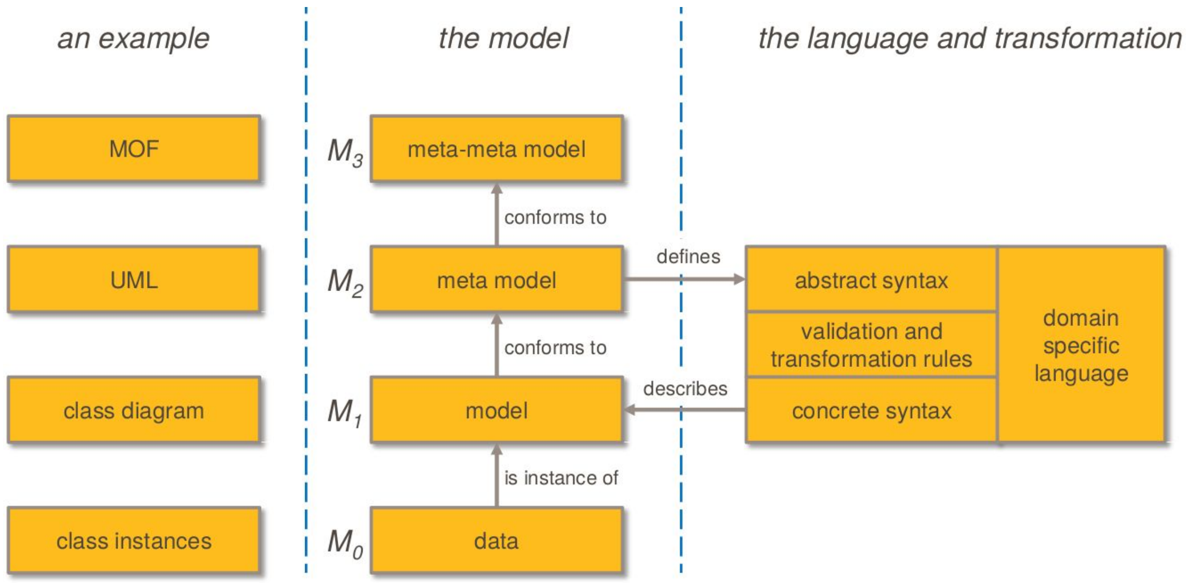
----

6. Model-Driven Software Development

5 arguments against MDDs
1. Models are models, real life is different
2. No-one knows, how to do it (right)
3. Performance woes
4. All in and no way out
5. High effort and low return

Different approaches : Top down (MDD), Closed system (vendor controlled RT), Bottom up (selected areas modelled and generated)

Model driven development uses formal models to generate derived artefacts (models or source code). A formal metamodel is required to generate artefacts.



MOF - meta object facility. MOF meta-metamodel is used to define the UML.
Exisiting langs -> Custom DSLs (key for MDD)

UML profiles offer a **lightweight extension of UML** using **stereotypes** and **tagged values**

BPM - Business Process Modelling. BPM attempts to improve processes continuously.
- Start, End, Tasks, Exclusive Gateways, Parallel Gateways.

Modelling -> Generating -> Implementation
Early phases are vital to project success with high MDD usage.
- Consistent tool chain, community support
- Customer involvement
- Perm. team mems with detailed knwldge and capable offshore team.

---

7. Modern Tech stack

- App stack, tool stack, mang. stack, Analysis stack

Dependency injection Alts.
- Instantiate deps in the component (can't swap implementations)
- Wire deps programmatically (manually maintain order of deps. pass through constr.)
- Factory pattern (central deps mang, but everything deps on Factory class)
- Dependency injection framework like Spring.
    - Setter injection
    - Constructor injection

  No invasive (classes don't know about DI framework), no deps to framework, comps reusable, implementations swappable, deps description centralized.

Aspect oriented programming : Addresses Cross-Cutting-Concerns in OOP. Encourages the Single Responsibility Principle.
- Intercepting method calls
- Manipulate behavior
- Inject new behavior
- Suppress behavior

Returns proxy instead of real impl.

ORM - Object Relational Mapping. Hibernate - DAOs.
Mapping options :
- Annotations (+ Faster, compiled java bytecode, +compact, +easy to maintain, - binary dependency, - configuration widespread)
- Mapping in XML files (+ single point of config., + easy to mod. w/o sw recompile. - poor readability)

Mapping model : Query engine translate query to SQL query. Query engine builds objects out of results.
Diff. query methods : HQL (+ Db indept., + compfort. - no standard), Criteria API (+ creation via clear API, - Hard to est. SQL query cost), SQL (+ clear control, - db dep.)

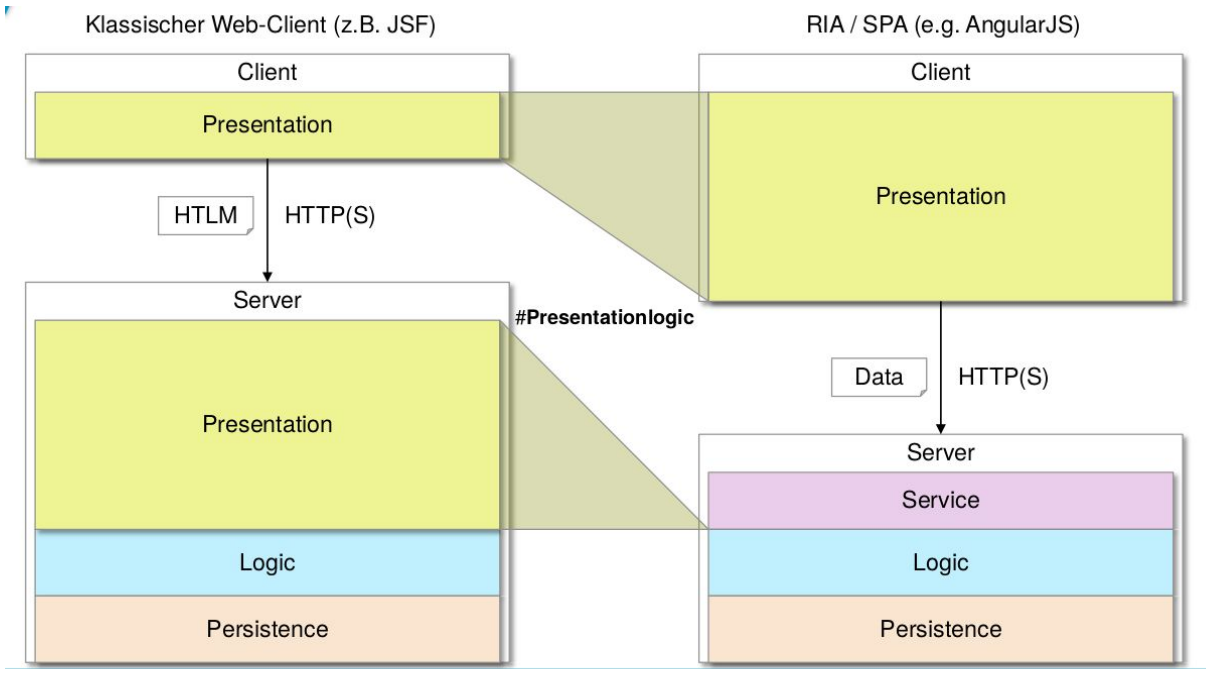
---

8. OASP

Open Application Standard Platform. Increase efficiency by reuse of Code, Knowledge, Documentation, Tools.

OASP is no framework but follows a flexible pattern-based approach
- Approach is proven
- IT security
- Non functional reqs (can be verified upfront)

- Licensing (tech stack is verified by legal department)



- Data Access Layer (Entities & DAOs)
- Logic layer (Use-Cases, TOs and Component-Interface)
- Service Layer (Expose functions as services)

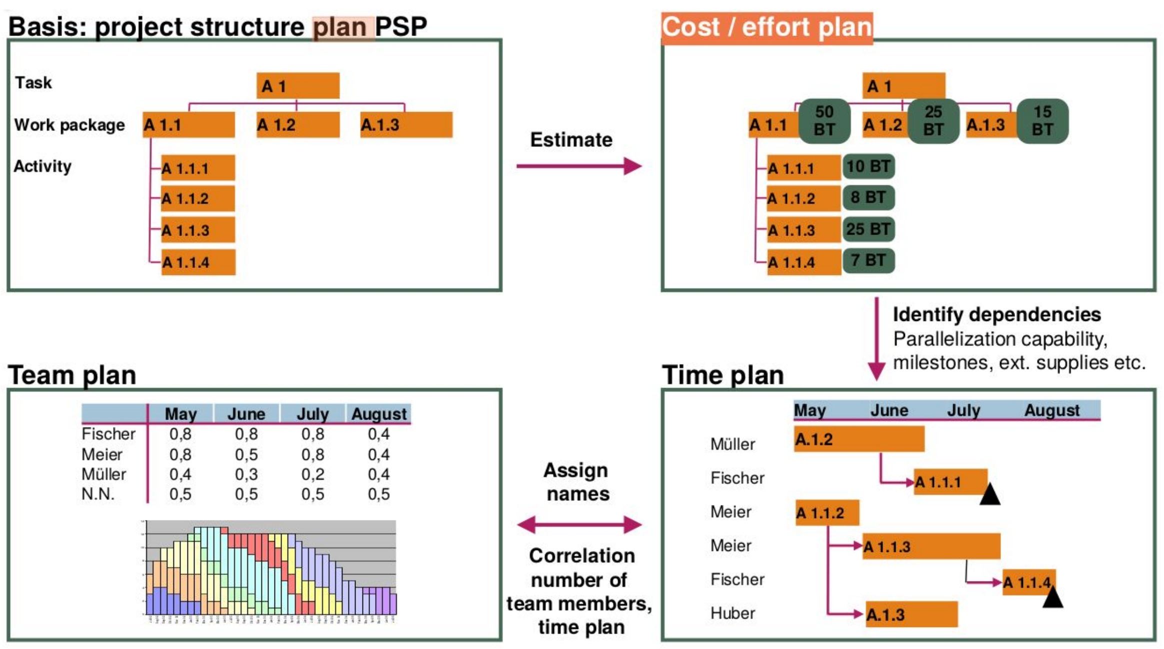
Open source licensing models. Permissive -> Weak protective -> Strong pro. -> Network pro.

----

9. Estimation and Project planning

The project plan includes multiple (sub-) plans and is created in several steps

1. project structure plan PSP
2. Cost / effort plan
3. Time plan
4. Team plan



	May	June	July	August
Fischer	0,8	0,8	0,8	0,4
Meier	0,8	0,5	0,8	0,4
Müller	0,4	0,3	0,2	0,4
N.N.	0,5	0,5	0,5	0,5

Name 3 cross-sectional activities in cost estimation

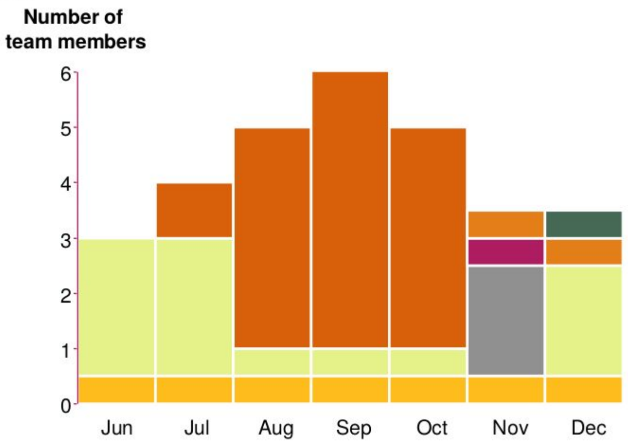
gross effort	**net effort (PI)**	effort for producing the actual products
	cross-sectional aspects (PQ)	project and quality management, architectural consulting, meetings, …
	other effects (PN)	travel-time, different locations, training

After that surcharges for risk and warranty are added. Describe both?

surcharge for fixed price („risk“)	Surcharge to insure against risks (wrong assumptions, contractual penalties, work packages which have not been estimated, …)
warranty surcharge	allowance for warranty aspects after delivery (bug fixes, etc.)

Effort estimation in Human resources possible. How this cals month period? Where does the factor 0.8 come from?

- Outline the project plan by estimated duration and team size
- Calculate the area
  here: 30 month periods (MP)
- 1 MP = 0,8 person month due to holidays, trainings, illness, non-project meetings, etc.
- The conversion from MP into person months results in:
  30 * 0,8 = 24 PM
- Does that match the effort estimation?


Number of team members
Jun
Jul
Aug
Sep
Oct
Nov
Dec