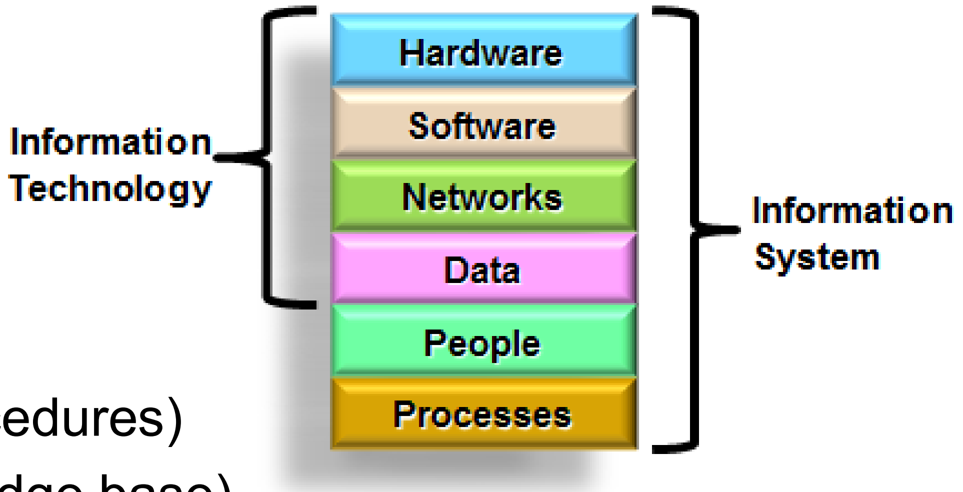
Information systems modeling

Tomasz Kubik

Information system

- A set of interrelated elements, the function of which is the processing of data using computer technology.
- Is composed of:

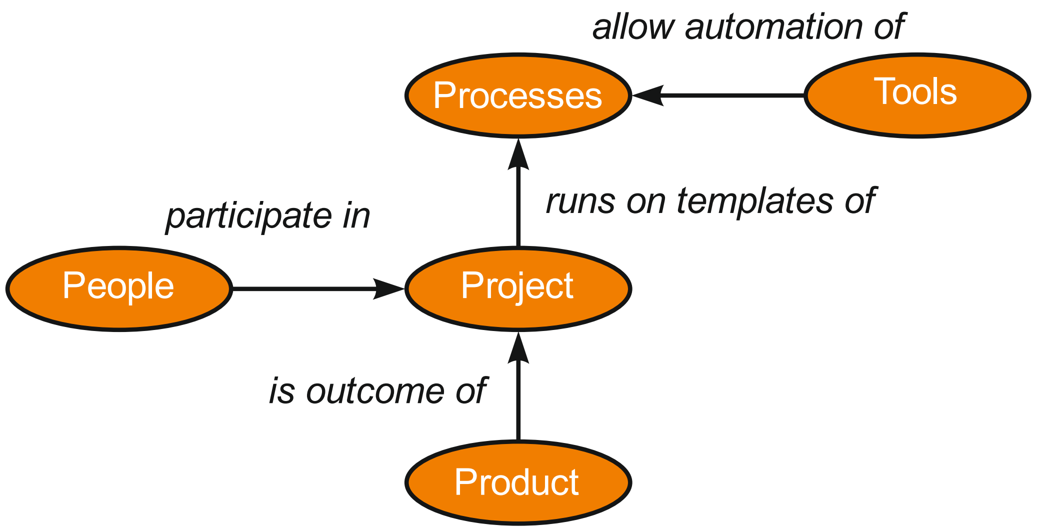


- and:
  - organizational elements (procedures)
  - information elements (knowledge base)
  - equipment (equipment for communication, devices for processing data that are not computers)

http://hajeralbuloshi.blogspot.com/

Information system - design

- Core factors



People: *Architects, developers, testers, users, customers etc*

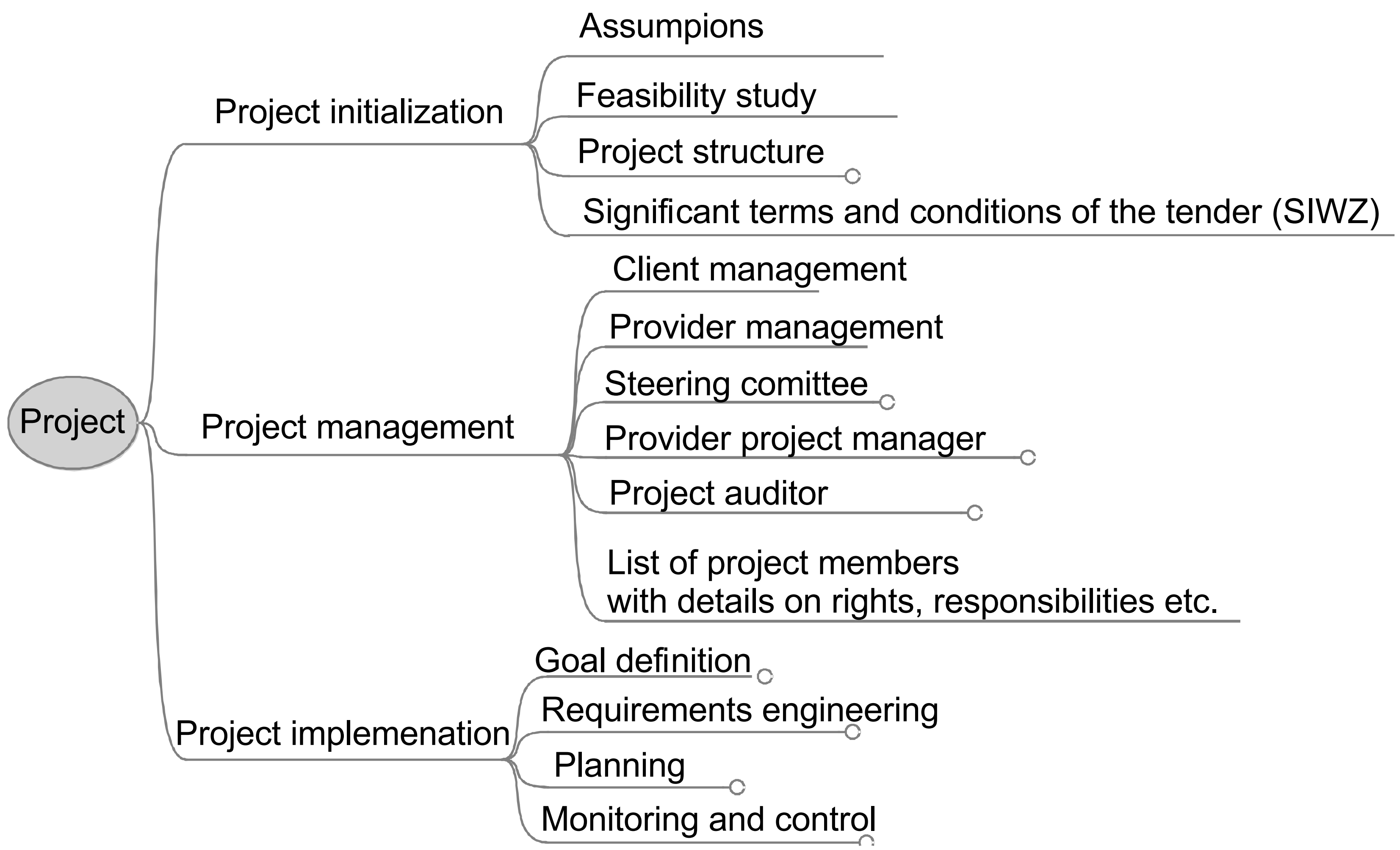
Project: The organizational element through which software development is managed

Product: Artifacts that are created during the life of the project such as models, source code, executables and documentation

Process: A software engineering process is a definition of the complete set of activities needed to transform users' requirements into a product

Tools: Software that is used to automate the activities defined in the process.

Project phases



Feasibility study

- Decides whether or not implement a system
- Answers the questions:
  - Will the system contribute to the organizational objectives while maintaining all the technical, economic and legal constrains?
  - What problems currently exist and whether the system will help solve them?
  - Which of the problems are critical and which are not?
  - Can the system be created using available technologies within budget and time constraints?
  - Can the system be integrated with existing systems, and if so, what will be the integration problems?
  - What happens if the system does not rise?

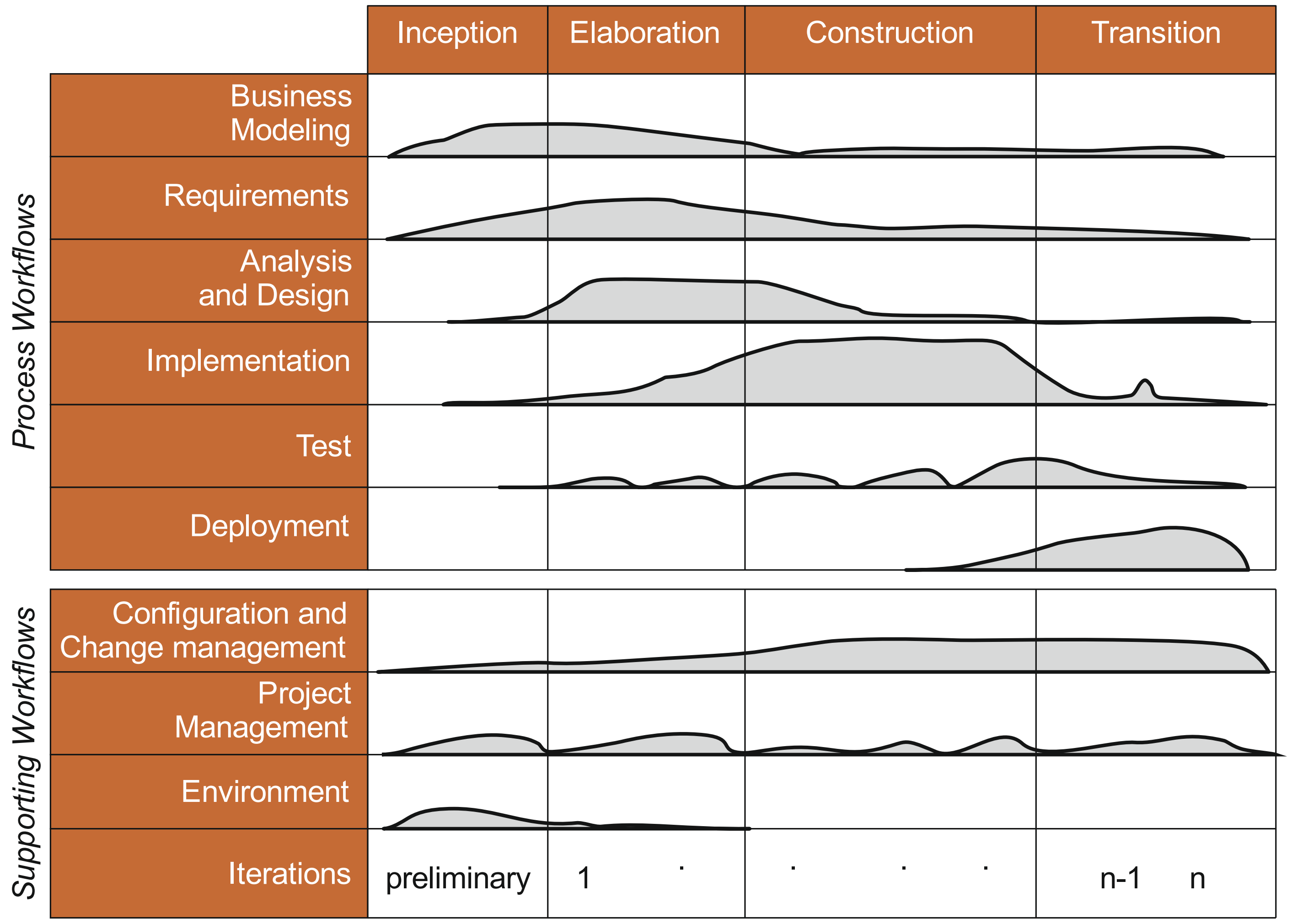
Information system of the project

- Is related to:
  - communication infrastructure
  - information acquisition
    - documentation, messages, notes
    - reports on project state
    - reports on projects efficiency
    - reports on exceptional cases
  - distributing information
  - authoring and archiving

Software tools supporting information system

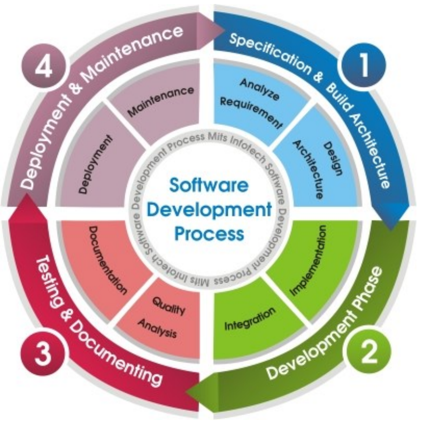
- Document management systems (DMS)
- Version control systems (CVS, SVN, GIT)
- Continous integration tools (Gitlab CI)
- IDE and various editors (mind maps, UML diagrams, mockups)
- Groupware tools (SOGo)
- Project management (ticketing systems like Track, Mantis, wiki)
- others ...

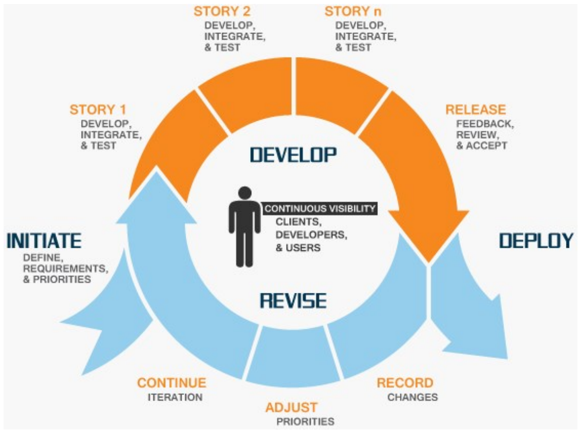
Software development process

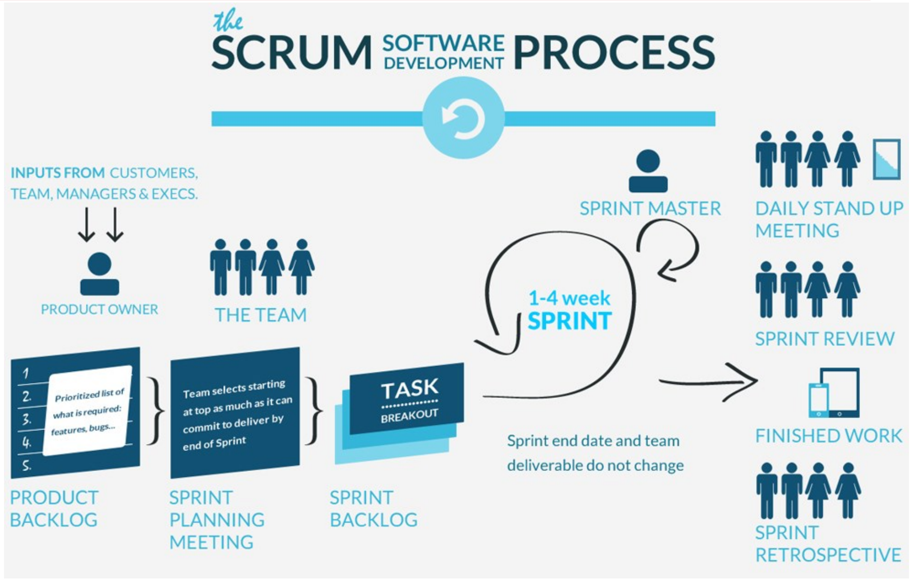


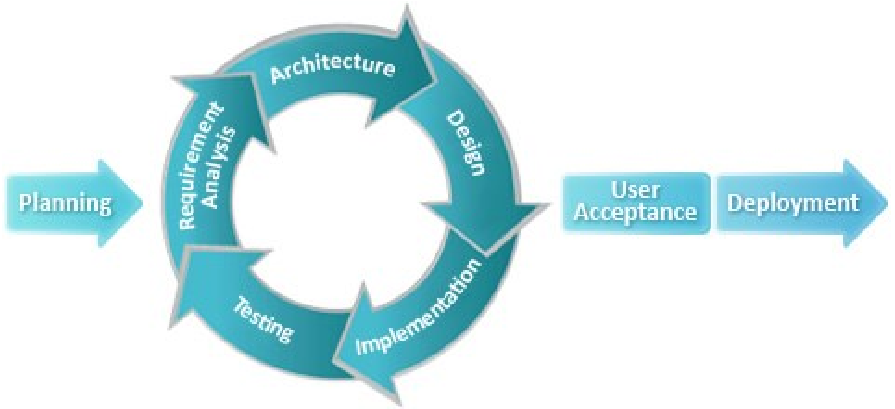
Jacobson I., Booch G, Rumbaugh J. The unified software development process

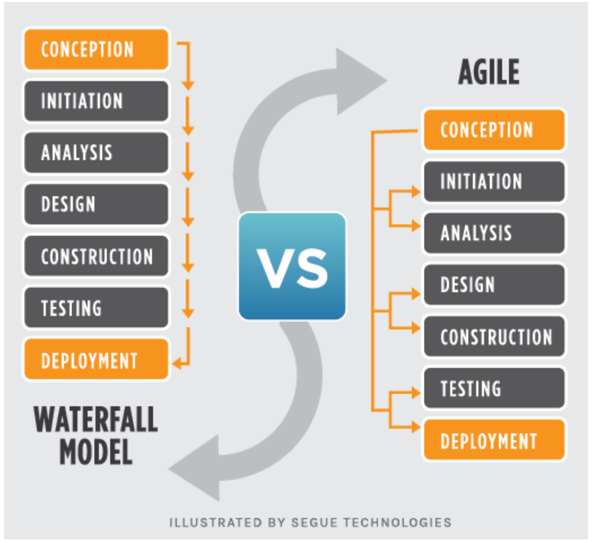
Software Development Life Cycle




STORY 1
DEVELOP, INTEGRATE, & TEST
STORY 2
DEVELOP, INTEGRATE, & TEST
STORY n
DEVELOP, INTEGRATE, & TEST
RELEASE
FEEDBACK, REVIEW, & ACCEPT
INITIATE
DEFINE, REQUIREMENTS, & PRIORITIES
DEVELOP
CONTINUOUS VISIBILITY
CLIENTS, DEVELOPERS, & USERS
REVISE
DEPLOY
CONTINUE
ITERATION
ADJUST
PRIORITIES
RECORD
CHANGES

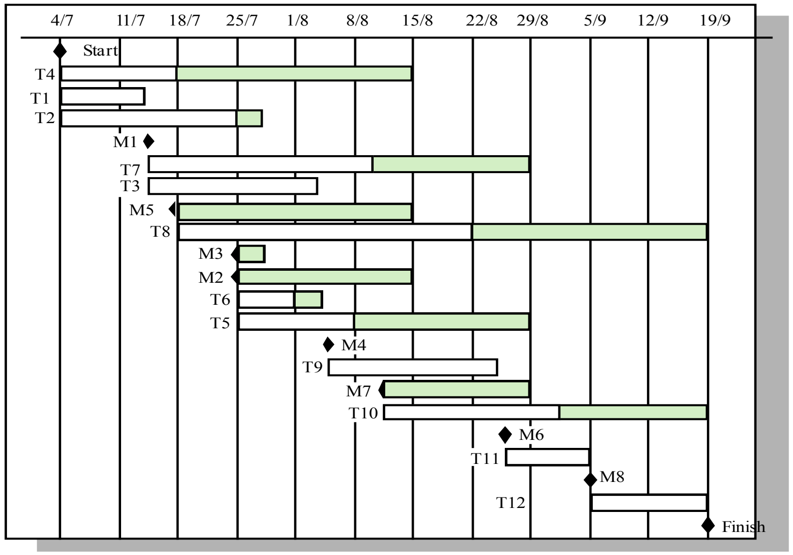

the SCRUM SOFTWARE DEVELOPMENT PROCESS
INPUTS FROM CUSTOMERS, TEAM, MANAGERS & EXECS.
PRODUCT OWNER
THE TEAM
SPRINT MASTER
DAILY STAND UP MEETING
1-4 week SPRINT
SPRINT REVIEW
1. 2. 3. 4. 5.
Prioritized list of what is required: features, bugs...
Team selects starting at top as much as it can commit to deliver by end of Sprint
TASK BREAKOUT
PRODUCT BACKLOG
SPRINT PLANNING MEETING
SPRINT BACKLOG
Sprint end date and team deliverable do not change
FINISHED WORK
SPRINT RETROSPECTIVE



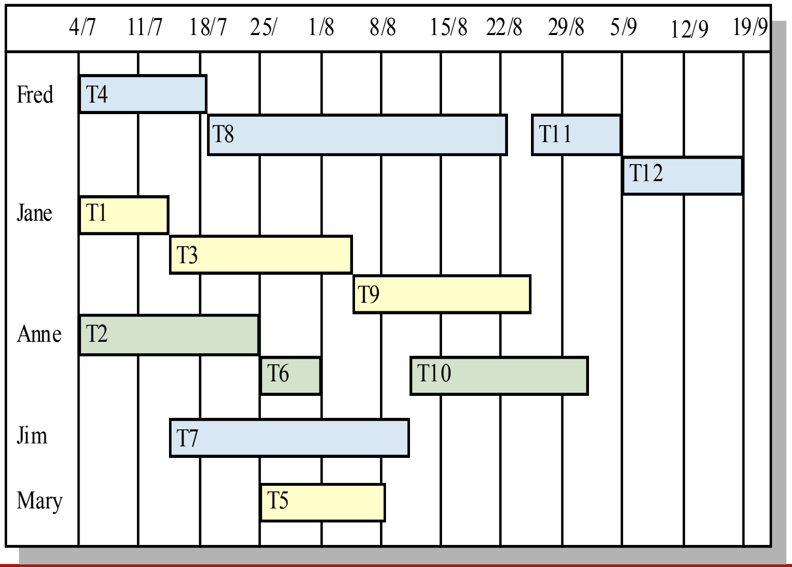


https://pl.pinterest.com/nahumespinosa/software-develoment/
http://www.netcentix.com/Web/Web-App-Development.aspx
http://victorylogic.com/
https://pl.pinterest.com/katybyrne/project-management/
http://www.seguetech.com/waterfall-vs-agile-which-is-the-right-development-methodology-for-your-project/

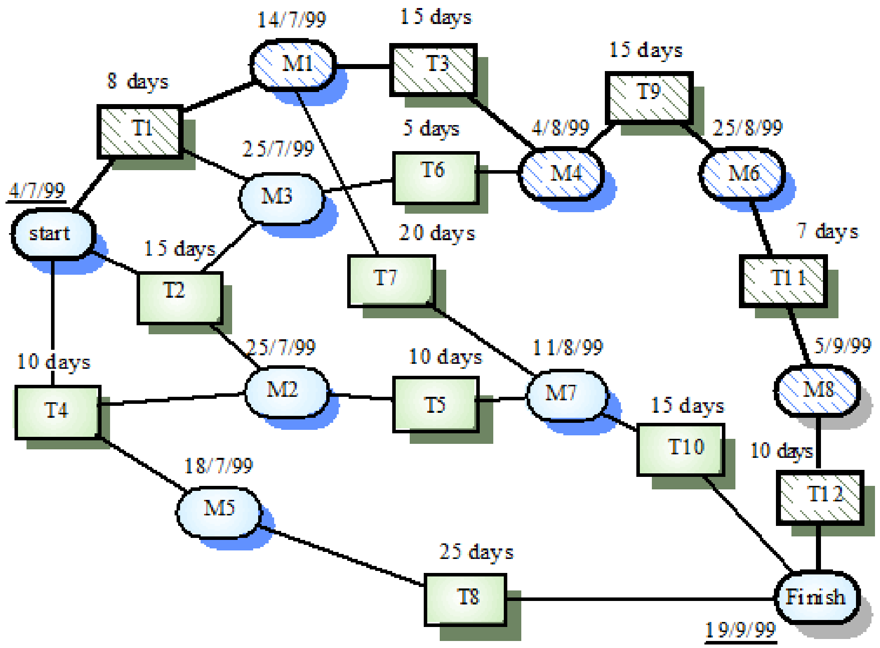
Planning



4/7
11/7
18/7
25/7
1/8
8/8
15/8
22/8
29/8
5/9
12/9
19/9
Start
T4
T1
T2
M1
T7
T3
M5
T8
M3
M2
T6
T5
M4
T9
M7
T10
M6
T11
M8
T12
Finish



4/7
11/7
18/7
25/
1/8
8/8
15/8
22/8
29/8
5/9
12/9
19/9
Fred
T4
T8
T11
T12
Jane
T1
T3
T9
Anne
T2
T6
T10
Jim
T7
Mary
T5



start
4/7/99
8 days
T1
14/7/99
M1
15 days
T3
15 days
T9
25/7/99
M3
5 days
T6
4/8/99
M4
25/8/99
M6
15 days
T2
20 days
T7
7 days
T11
10 days
T4
25/7/99
M2
10 days
T5
11/8/99
M7
5/9/99
M8
15 days
T10
10 days
T12
18/7/99
M5
25 days
T8
Finish
19/9/99

Documents template (IEEE)

Table of Contents

http://www.cse.msu.edu/~cse870/IEEEXplore-SRS-template.pdf

Documents template (IEEE) System Requirements Specification (SyRS)

1. Introduction
    - 1.1 System purpose
    - 1.2 System scope
    - 1.3 System overview
    - 1.3.1 System context
    - 1.3.2 System functions
    - 1.3.3 User characteristics
    - 1.4 Definitions
2. References
3. System requirements
    - 3.1 Functional requirements
    - 3.2 Usability requirements
    - 3.3 Performance requirements
    - 3.4 System interface
    - 3.5 System operations
    - 3.6 System modes and states
    - 3.7 Physical characteristics
    - 3.8 Environmental conditions
    - 3.9 System security
    - 3.10 Information management
    - 3.11 Policies and regulations
    - 3.12 System life cycle sustainment
    - 3.13Packaging, handling, shipping and transportation
4. Verification
    - (parallel to subsections in Section 3)
5. Appendices
    - Assumptions and dependencies
    - Acronyms and abbreviations

Systems and software engineering — Life cycle processes — Requirements engineering (ISO/IEC/IEEE 29148:2011(E))

Documents template (IEEE) Stakeholder Requirements Specification (StRS)

1. Introduction
   - 1.1 Business purpose
   - 1.2 Business scope
   - 1.3 Business overview
   - 1.4 Definitions
   - 1.5 Stakeholders
2. References
3. Business management requirements
   - 3.1 Business environment
   - 3.2 Goal and objective
   - 3.3 Business model
   - 3.4 Information environment
4. Business operational requirements
   - 4.1 Business processes
   - 4.2 Business operational policies and rules
   - 4.3 Business operational constraints
   - 4.4 Business operational modes
   - 4.5 Business operational quality
   - 4.6 Business structure
5. User requirements
6. Concept of proposed system
   - 6.1 Operational concept
   - 6.2 Operational scenario

7 Project Constraints

8. Appendix
   - 8.1 Acronyms and abbreviations

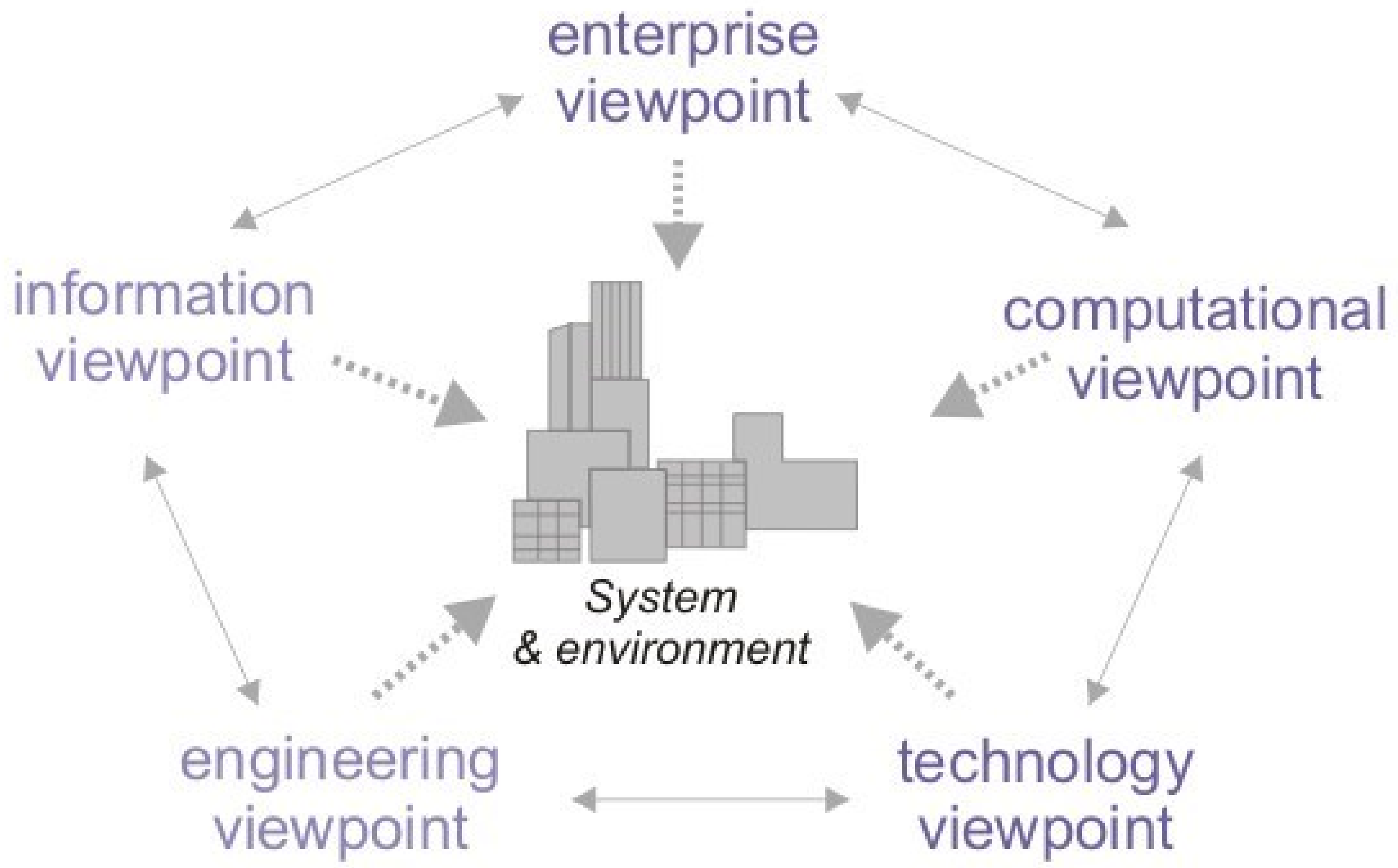
Systems and software engineering — Life cycle processes — Requirements engineering (ISO/IEC/IEEE 29148:2011(E))

Documents template (IEEE) Software Requirements Specification (SRS)

1. Introduction
    - 1.1 Purpose
    - 1.2 Scope
    - 1.3 Product overview
        - 1.3.1 Product perspective
        - 1.3.2 Product functions
        - 1.3.3 User characteristics
        - 1.3.4 Limitations
    - 1.4 Definitions
2. References
3. Specific requirements
    - 3.1 External interfaces
    - 3.2 Functions
    - 3.3 Usability Requirements
    - 3.4 Performance requirements
    - 3.5 Logical database requirements
    - 3.6 Design constraints
    - 3.7 Software system attributes
    - 3.8 Supporting information
4. Verification
(parallel to subsections in Section 3)
5. Appendices
    - 5.1 Assumptions and dependencies
    - 5.2 Acronyms and abbreviations

Systems and software engineering — Life cycle processes — Requirements engineering (ISO/IEC/IEEE 29148:2011(E))

RM-ODP



ISO/IEC 10746-1:1998, Basic Reference Model of Open Distributed Processing (ODP)

Requirements engineering

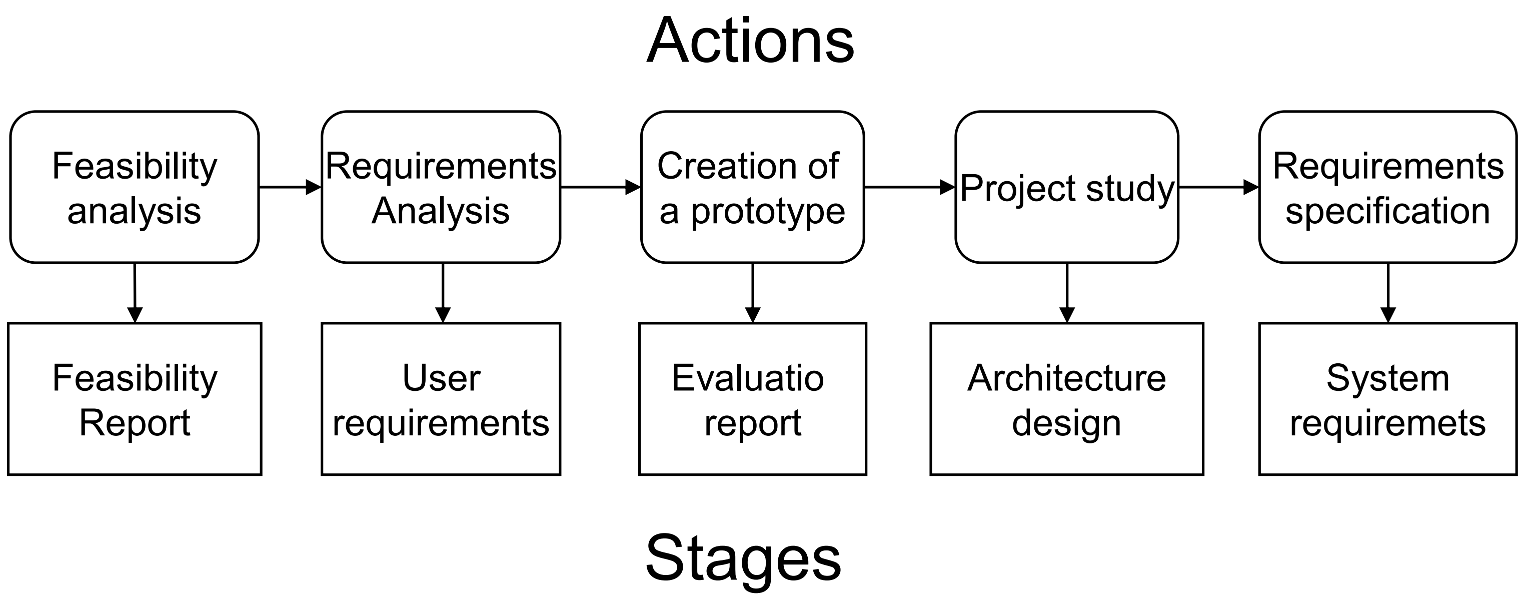
- Requirements
  - statements of what the system must do, how it must behave, the properties it must exhibit, the qualities it must possess, and the constraints that the system and its development must satisfy.
- Requirement according to The Institute of Electrical and Electronics Engineers (IEEE)
  - a condition or capability needed by a user to solve a problem or achieve an objective
  - a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document
  - a documented representation of a condition or capability as in definition 1 or 2

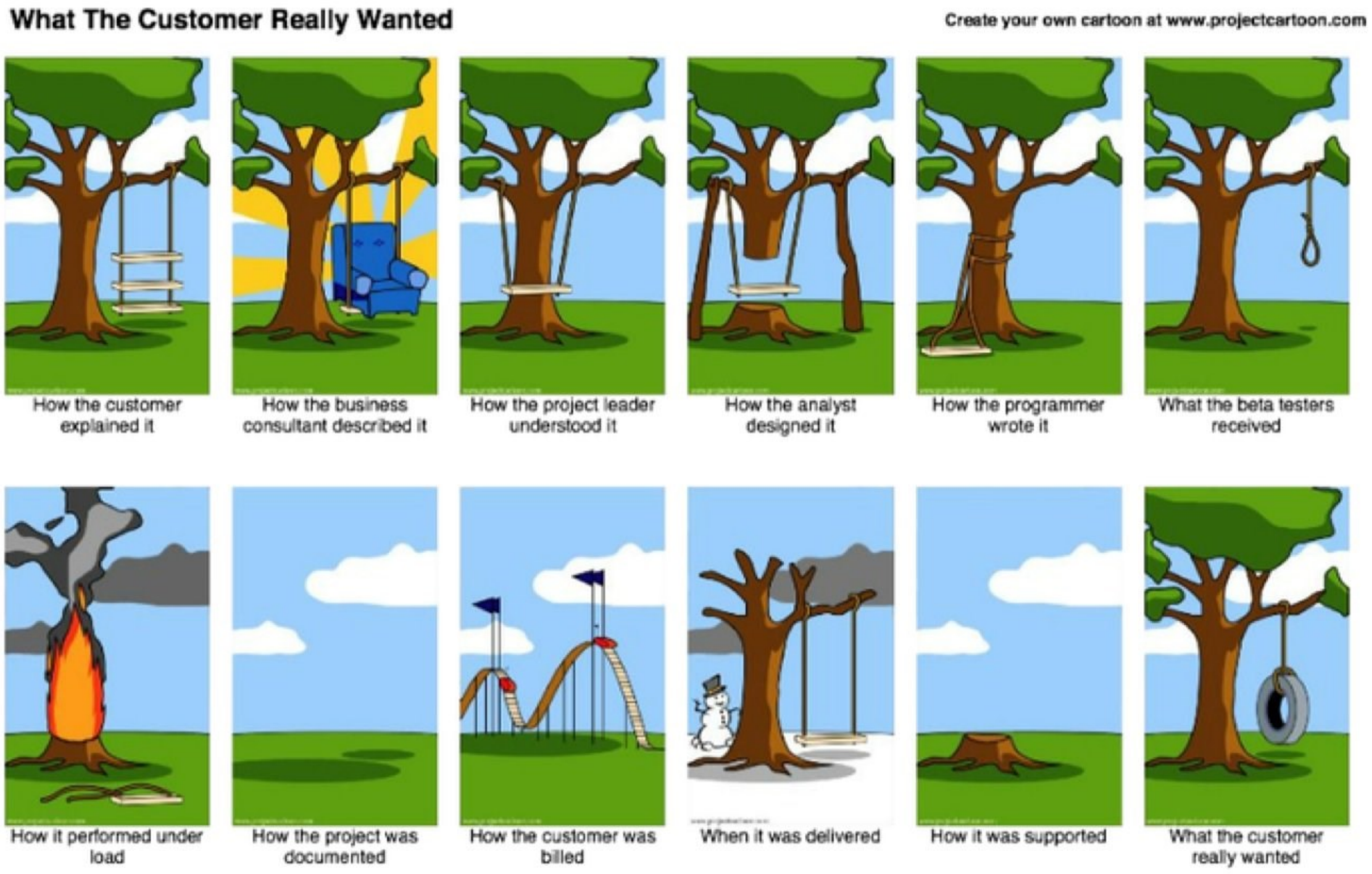
http://www.sei.cmu.edu/productlines/frame_report/req_eng.htm

Requirements definitions

- They can take different forms, depending on the objectives:
  - Open form at the contracting stage
  - Closed at the implementation stage
- Desirable characteristics of the definitions:
  - Consistency (without contradiction in the defined expectations)
  - Completeness (including all required expectations)
  - Accuracy (narrowing the scope for interpretation)
  - Reality (i.e. realizable)
  - Verifiability (verifiable)
  - Significance (best meet the needs of the user)

Milestones in the proces of requirements definitions




What The Customer Really Wanted
Create your own cartoon at www.projectcartoon.com
How the customer explained it
How the business consultant described it
How the project leader understood it
How the analyst designed it
How the programmer wrote it
What the beta testers received
How it performed under load
How the project was documented
How the customer was billed
When it was delivered
How it was supported
What the customer really wanted

source: *Product Management*

Requirements

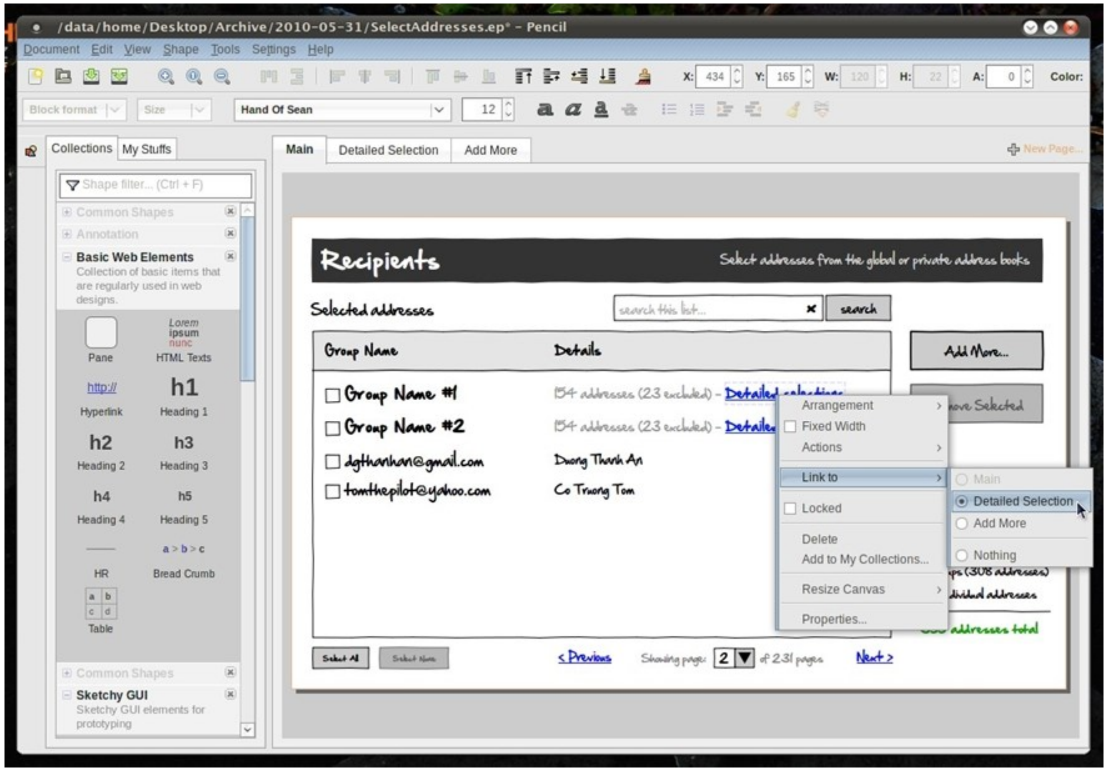
- User requirements
  - Define, in a general way, expectations for the services offered by the system and the constraints under which the system will operate
  - collection of functional and nonfunctional requirements
  - expressed in natural language, using the forms, tables and diagrams readable for users without technical background
  - formed on the basis of an interview with the client
- System requirements
  - Detailed descriptions of services and system constraints
  - They can be represented by models
  - They appear as part of the contract between the customer and the supplier
- Specification of software project
  - an abstract description of a software project, a core for detailed project description and implementation, created for developers

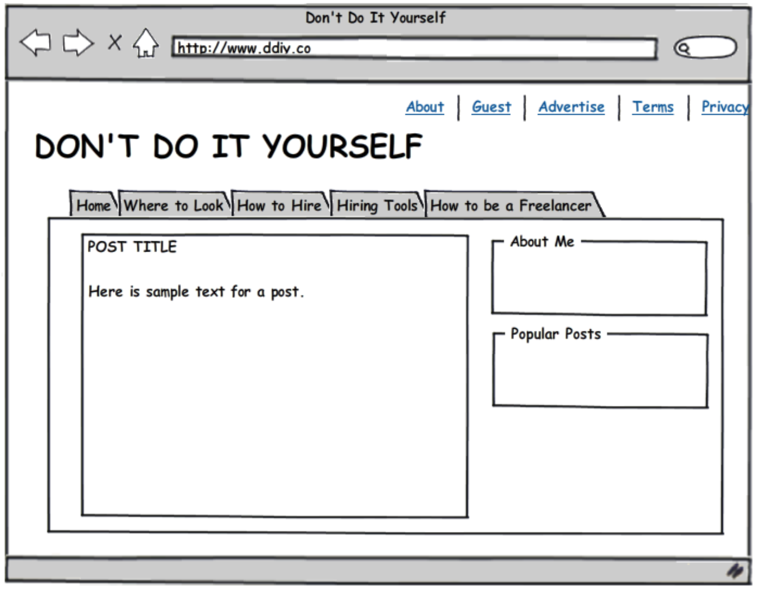
Requirements

- Functional requirements
  - planned services offered by the system
  - reaction to the particular inputs
  - behaviors in certain situations
  - protected behaviors and limits
- Non-functional requirements
  - limitations of the services and features, eg. timing constraints, constraints on the development process, standards, etc.
- Domain requirements
  - they come from the field of application
  - they can be functional or non-functional

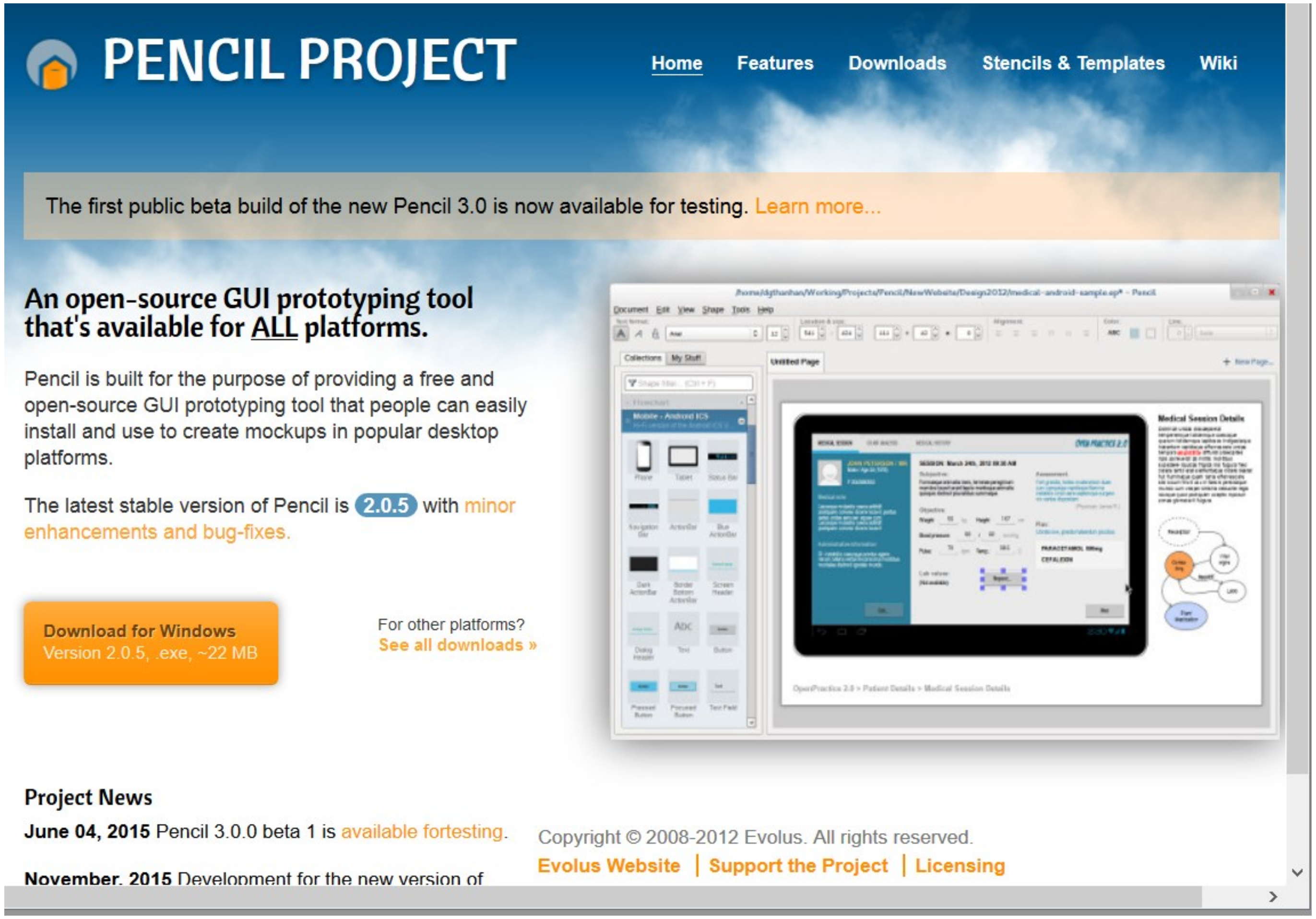
UI prototyping

- Mockups, Wireframes


/data/home/Desktop/Archive/2010-05-31/SelectAddresses.ep* - Pencil
Document Edit View Shape Tools Settings Help
Collections My Stuffs
Main Detailed Selection Add More
Basic Web Elements
Sketchy GUI
Recipients
Select addresses from the global or private address books
Selected addresses
search
Group Name Details
Add More...
Group Name #1
Group Name #2
dgthanhan@gmail.com Duong Thanh An
tomthepilot@yahoo.com Co Truong Tom
Arrangement
Fixed Width
Actions
Link to
Locked
Delete
Add to My Collections...
Resize Canvas
Properties...
Main
Detailed Selection
Add More
Nothing
&lt; Previous Showing page: 2 of 231 pages Next &gt;


Don't Do It Yourself
http://www.ddiy.co
About | Guest | Advertise | Terms | Privacy
DON'T DO IT YOURSELF
Home Where to Look How to Hire Hiring Tools How to be a Freelancer
POST TITLE
Here is sample text for a post.
About Me
Popular Posts

Pencil



PENCIL PROJECT

The first public beta build of the new Pencil 3.0 is now available for testing. Learn more...

An open-source GUI prototyping tool that's available for ALL platforms.

Pencil is built for the purpose of providing a free and open-source GUI prototyping tool that people can easily install and use to create mockups in popular desktop platforms.

The latest stable version of Pencil is 2.0.5 with minor enhancements and bug-fixes.

Project News

June 04, 2015 Pencil 3.0.0 beta 1 is available fortesting.

November, 2015 Development for the new version of

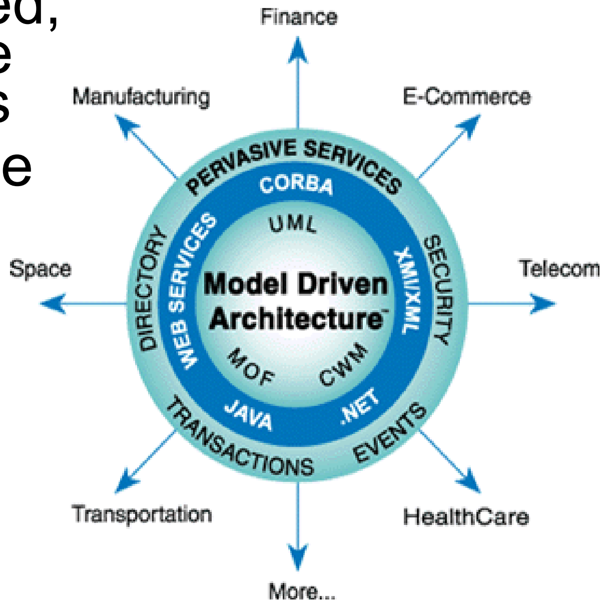
http://pencil.evolus.vn/

Various perspectives on modeling

Modeling the structure and dynamics of the system	Implementation of the structure and dynamics of the system, code generation	
Perspective of the concept what to do?	Perspective of specifications how should I use?	Perspective of implementation how to perform?
• Model of the real system (business modeling) • Requirements Analysis (conceptual model) • Conceptual model tests	• Design model (hardware and architecture software; user access; storage) • Deployment model • Design model tests • Deployment tests	• Programming (specification of the program: declarations, definitions; additional data structures: structure of containers, files, databases) • Software tests • Implementation

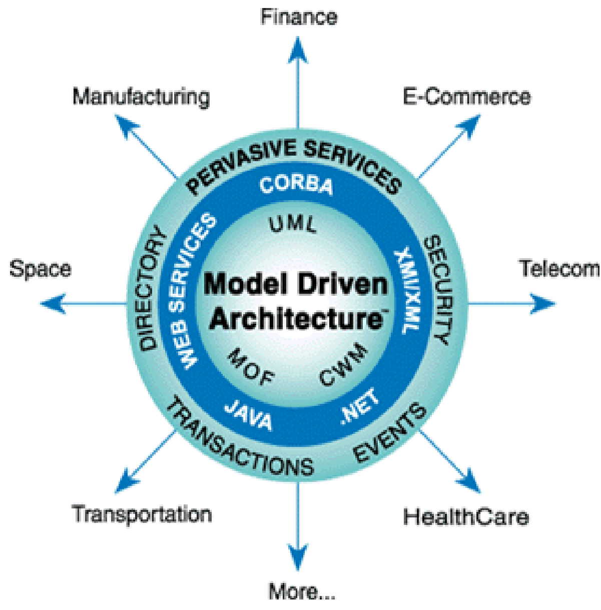
Model-driven architecture (MDA)

- an approach to using models in software development whereby models are used as the primary source for documenting, analyzing, designing, constructing, deploying and maintaining a system
- prescribes certain kinds of models to be used, how those models may be prepared and the relationships of the different kinds of models
- separates the operation of a system from the details of the way that system uses the capabilities of its platform
- basic concepts:
  - system notions
  - model, viewpoint, platform
  - platform independence,
  - pervasive services
  - application, implementation,
  - model transformation



Model Driven Architecture

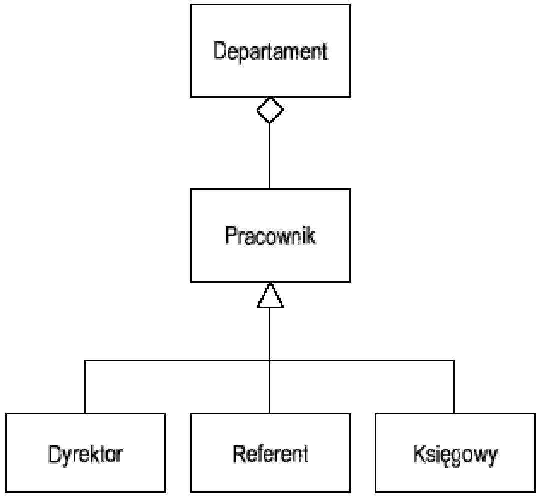
- Architecture
  - The architecture of a system is a specification of the parts and connectors of the system and the rules for the interactions of the parts using the connectors.
  - Within the context of MDA these parts, connectors and rules are expressed via a set of interrelated models.

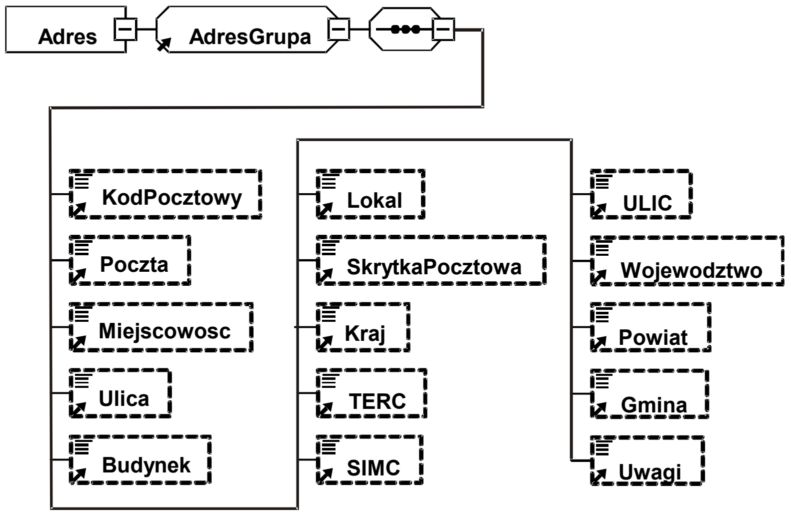


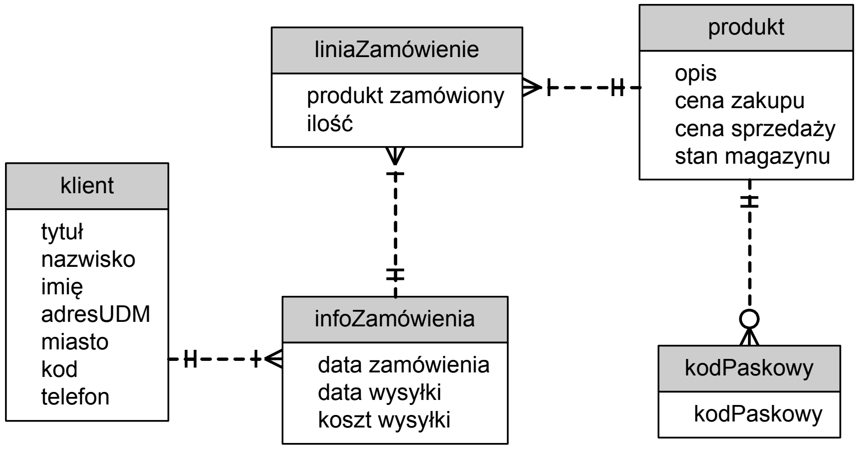
Frank Truye: The Fast Guide to Model Driven Architecture, The Basics of Model Driven Architecture (MDA). Cephas Consulting Corp, 2006

Various modeling paradigms and languages

- UML
- Entity relationship diagrams
- XML Schema
- Ontology







Programming level

- Databases

```
SELECT ST_Intersects('POINT(0 0)'::geometry, 'LINESTRING ( 2 0, 0
2 )'::geometry)
```

- Data exchange

```
<gml:Point gml:id="p21"
  srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
  <gml:coordinates>45.67, 88.56</gml:coordinates>
</gml:Point>
```

- Data processing using API

```
Geometry g = ...
BufferOp bufOp = new BufferOp(g);
bufOp.setEndCapStyle(BufferOp.CAP_BUTT);
Geometry buffer = bufOp.getResultGeometry(distance);
```

- Working with ontologies

```
PREFIX spatial: <http://jena.apache.org/spatial#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?placeName
{
    ?place spatial:nearby (51.46 2.6 10 'km') .
    ?place rdfs:label ?placeName
}
```

Object-Oriented Modeling

- Object orientation
  - an object represents an entity in the real world
  - state on an object can change (due to internal action or external interaction)
  - an object exposes its behaviors through interfaces (can only interact at interfaces – encapsulation).
  - abstraction: classification, generalization, association, and aggregation
- Structural orientation
  - no artificial decomposition into simpler parts due to technical restrictions should be necessary
- Operational orientation
  - operations on complex objects are possible without having to decompose the objects into a number of simple objects
- Behavioral orientation
  - a system must allow its objects to be accessed and modified only through a set of operations specific to an object type

S.O.L.I.D.

- An acronym for the **first five object-oriented design(OOD) principles** by Robert C. Martin, popularly known as Uncle Bob.
  - **S** – Single-responsiblity principle
    - A class should have one and only one reason to change, meaning that a class should have only one job.
  - **O** – Open-closed principle
    - Objects or entities should be open for extension, but closed for modification.
  - **L** – Liskov substitution principle
    - Let **q(x)** be a property provable about objects of **x** of type **T**. Then **q(y)** should be provable for objects **y** of type **S** where **S** is a subtype of **T** (every subclass/derived class should be substitutable for their base/parent class.)
  - **I** – Interface segregation principle
    - A client should never be forced to implement an interface that it doesn't use or clients shouldn't be forced to depend on methods they do not use.
  - **D** – Dependency Inversion Principle
    - Entities must depend on abstractions not on concretions. It states that the high level module must not depend on the low level module, but they should depend on abstractions.

https://scotch.io/bar-talk/s-o-l-i-d-the-first-five-principles-of-object-oriented-design