

# Software Requirements and Architecture (SENG404)

Matthias Galster

Lecture 2 – Requirements and requirements engineering processes

February 23, 2023

---

# Schedule 2023

Lecture	Week	Date	Topic
1	1	February 22	Kick-off; Introduction
2	1	February 23	Instead of May 3; Requirements and requirements engineering processes
3	2	March 1	
4	2	March 2	Instead of May 17
5	3	March 8	
6	3	March 9	Backup (Matthias <i>might</i> be away)
7	4	March 15	
8	4	March 16	Backup
9	5	March 22	Assignment 1
10	6	March 29	
<b><i>Term break</i></b>			
11	7	April 26	
12	8	May 3	Matthias away
13	9	May 10	
14	10	May 17	Matthias away
15	11	May 24	Assignment 2: presentations + report
16	12	May 31	
		TBD	<b>Final exam</b>

# Previous lecture

## 1. Requirements engineering in a nutshell

process to gather requirements to understand customer desire and needs, managing requirement change throughout the project and make sure it get handled well.

## 2. Software architecture in a nutshell

# Reading for this session

- Z. S. H. Abad, A. Shymka, S. Pant, A. Currie and G. Ruhe, *What are Practitioners Asking about Requirements Engineering? An Exploratory Analysis of Social Q&A Sites*, IEEE 24th International Requirements Engineering Conference Workshops (REW), 2016, pp. 334-343, doi: 10.1109/REW.2016.061

there is high risk in way we elicit requirements we can give waste a lot of time. Stakeholder's pressure for time.

it is difficult to show value early on. solution could be eliciting some requirements to show what has happened to this input. worst if their input doesn't go anywhere

developers become disheartened in communication, which is interesting as this was found to be important

# Questions and lessons

# Reading for next session

- S. Wagner, D. Mendez Fernandez, M. Kalinowski, M. Felderer, P. Mafra, A. Vetrò, T. Conte, M.-T. Christiansson, D. Greer, C. Lassenius, T. Männistö, M. Nayebi, M. Oivo, B. Penzenstadler, D. Pfahl, R. Prikladnicki, G. Ruhe, A. Schekelmann, S. Sen, R. Spinola, J.L. de la Vara, A. Tuzcu, R. Wieringa, and D. Winkler, *Status Quo in Requirements Engineering: A Theory and a Global Family of Surveys*, Transactions on Software Engineering and Methodology, 2019, pp. 1–48, doi.org/10.1145/3306607

interesting paper for reflection on the practises of requirement engineering. also interesting for research method that can be used for an assignment

# Agenda

1. A closer look at requirements
2. Requirements engineering activities
3. Feasibility studies financially feasibly, is it too risky? reason is because it links to requirement engineering to establish a base of the project
4. Stakeholder analysis

# Agenda

1. A closer look at requirements
2. Requirements engineering activities
3. Feasibility studies
4. Stakeholder analysis



# Requirements

in past course looked at requirements as user stories. now in this course going to look at RE as layers, context and motivation why we are building a system in the first place.

eg. business want a solution that may not be end users

- **Documented** representation of **need** of **stakeholder(s)**
  - Plus **properties** that system shall have (not perceived by stakeholders)
- Typical layers (considering the bigger picture)
  - **Business** requirements

“More people than today shall be transported using the existing railway tracks.”

Need to understand  
cause

- **System** requirements

“Minimum distance between two trains shall always be larger than current maximum braking distance of following train.”

A “system” is only  
one type of “solution”

- **Software** requirements

“The current maximum braking distance shall be computed every 100ms.”

# Another example

- **Business** requirement

“I want a safe house.”

if understood the business as  
“i want a safe house” as in from floods

**But:** Are there other (cheaper, faster) ways  
to achieve this business requirement?

**Therefore:** Avoid “solution-centric” view  
(in particular early in a project)



- **System** requirements

“The HSS must support video surveillance.”

“The HSS must activate an alarm.”

- **Software** requirements

“The alarm shall start immediately after detecting an open window or door.”

“The alarm signal shall be deactivated by police or home owner.”

# Software-centric solution alternatives



Fully-fledged high-end surveillance system with modern hardware and customized software



Out of the box solution of a set of networked video cameras



Understand **goals, context and resources** of customer and problem

Not offering software-centric or “big” solution may not be in interest of vendor

# Example problem

An airport controls access to planes through an automated gate (“self-boarding”). Whenever the gate reads a valid barcode on a boarding pass, it unlocks for one entry.



# Questions

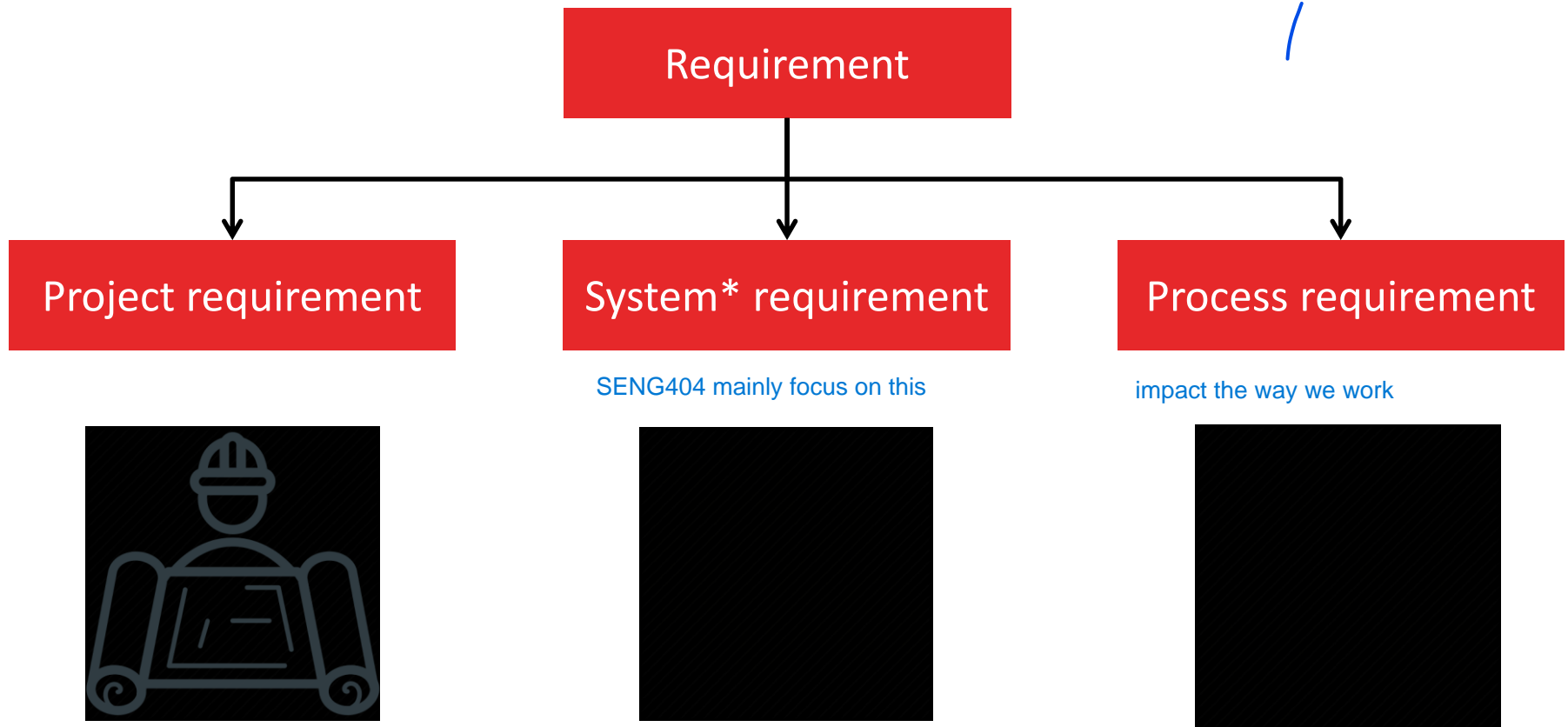
How do we know what kind of things to capture as requirements and that we don't forget important things?

How do we know how to treat requirements? Or are all (types of) requirements equal?

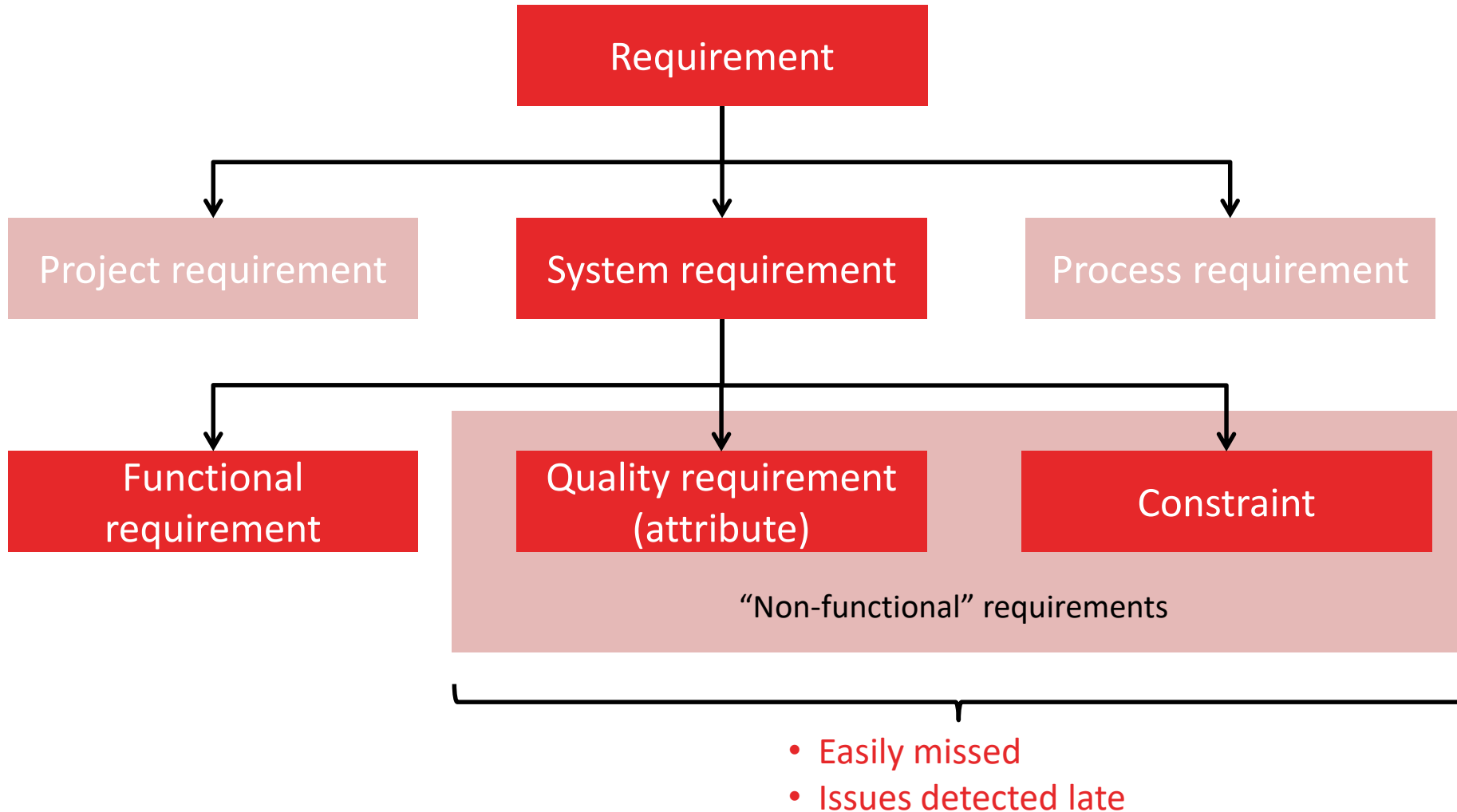
# Possible solution

Classifications of requirements (or: *what* to look for)

# Typical classification (1)



# Typical classification (2)





# Functional requirements

- **Functionality and behavior**

- Interactions with environment, including inputs, desired result
- In which state shall this function be available
- Dynamic behavior observed by users
- Which events lead to or terminate behavior

“The system shall run in three modes: normal (gate unlocked for one passenger with boarding pass), locked, and open (all gates unlocked).”

- **Data**

- Usage and structure

“A boarding pass must show the name of the passenger, the booking number, time and date of their flight as well as the flight number.”

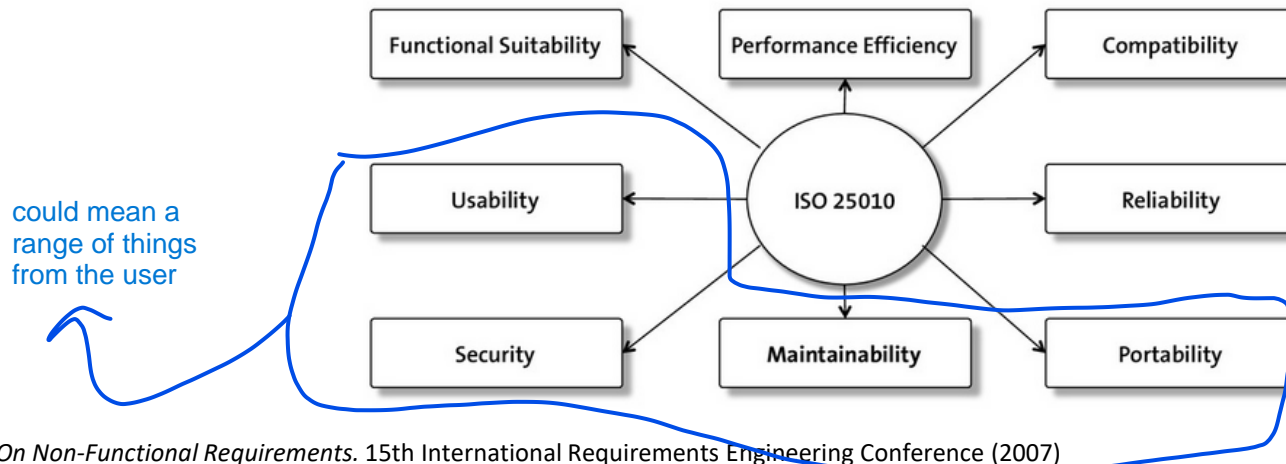
- Different ways of documenting (more later)

# Quality requirements (attributes)

- Basic types
  - Design time, e.g., portability, maintainability, etc. (often not specified)
  - Runtime, e.g., usability, security, availability

“The response time from reading a valid boarding pass to unlock must be shorter than 0.5 seconds.”

- See also ISO/IEC 9126, ISO/IEC 25010:2011, S-Cube, etc.
  - E.g., ISO 25010 – quality characteristics



# Constraints

- Restrictions or solutions that must be obeyed or satisfied
  - **Can be cross-cutting:** impact product, process and project
- Examples
  - Technical, e.g., given interfaces or protocols
  - Cultural, e.g., coloring, language, spelling
  - Organizational, e.g., structures, processes changed
  - Legal, e.g., laws, standards, regulations

“The system must comply with the privacy law of the country where the airport is located.”

# Ethical considerations

- Separate type of constraint or quality requirement?

- Broader human and societal values?

eg ppl data, trees needed, fairness and inclusiveness of software to help everybody not harm someone

how to describe human values

- How do we evaluate ethical considerations?
  - E.g., during requirements analysis?

fairness and inclusion of software. when we build systems we need to treat ppl not as second class systems.

assignment; how developers even describe human values, does this come up at all?

# Some requirements are “hard”, some are “soft”

- **Hard**
  - Requirement is fully satisfied or not at all
  - Binary acceptance criterion; value: 1 or 0
- **Soft**
  - Range of satisfaction
  - Range of acceptable values
  - Minimum acceptable, planned
- Satisfaction (hard, soft) informs
  - **Documentation**, e.g., free text, user stories, formal, etc.
  - **Validation**, e.g., testing, success and acceptance criteria

# Agenda

1. A closer look at requirements
2. Requirements engineering activities
3. Feasibility studies
4. Stakeholder analysis

NONE OF US HAS  
DESIGNED A NUCLEAR  
POWER PLANT BEFORE  
BUT WE CAN FIGURE  
IT OUT BY USING  
OUR PROCESS.



www.dilbert.com scottadams@aol.com

IN PHASE ONE WE  
WILL GATHER  
CUSTOMER REQUIRE-  
MENTS.

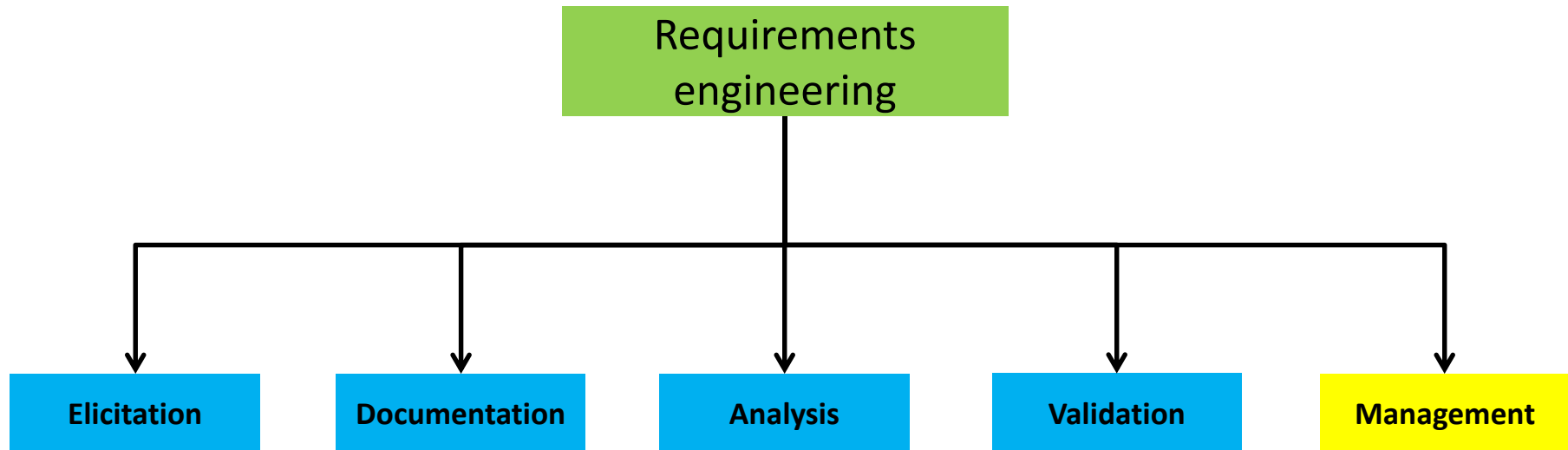


2/20/02. © 2002 United Feature Syndicate, Inc.

SO... YOU WANT FREE  
ELECTRICITY, WITHOUT  
MUTATING, UNLESS  
THE MUTATION GIVES  
YOU X-RAY  
VISION.



# Generic RE activities



elicitation getting requirements out. eg talking to ppl in interviews

documentations is capturing requirements that are useful

analysis.

validation of having the right and meaningful requirements

management to deal with changing requirements to deal with this iteratively

## MAIN FACTORS THAT CONTRIBUTE TO SEQUENTIAL/LOW ITERATION NEED

- \* small project
- \* not expected for long time
- \* isolated project, and no need for other features to integrate
- \* low complexity
- \* state of domain is well understood for no surprises
- \* certainty
- \* less iterations
- \* less regulations
- \* less constraints
- \* low changes
- \* goal is to purely automate

example of sequentially is okay. One's that don't need more management afterwards. and not expecting another project to integrate. no surprises. low complexity that have a domain that is very well understood. so depend on state of domain. system is to automating manual process - this is linear process with low complexity, less uncertainties, and not lots of changes happening



# Processes vary

- Between organizations and projects

- Organizational culture
- Application domain
- Project size
- Etc.

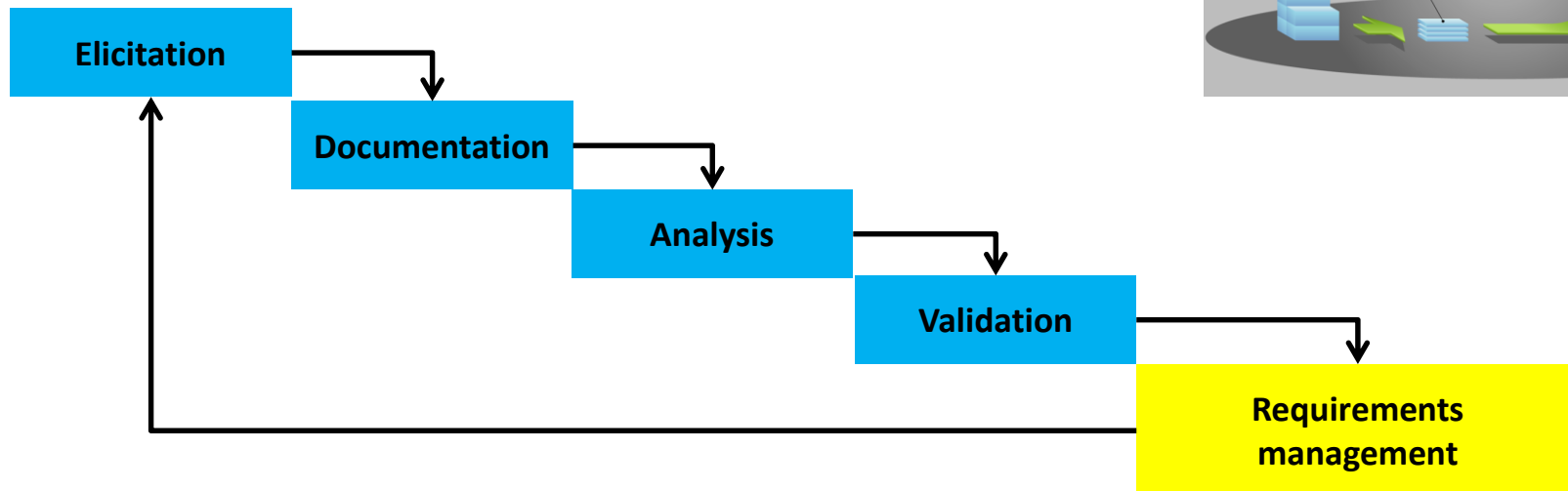
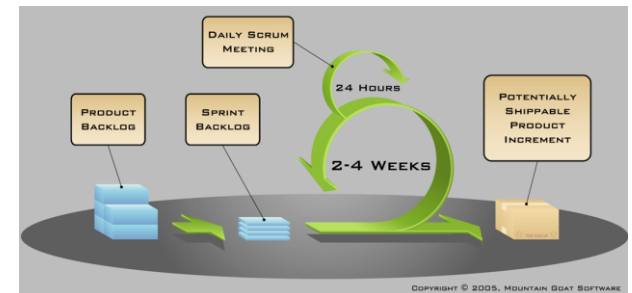
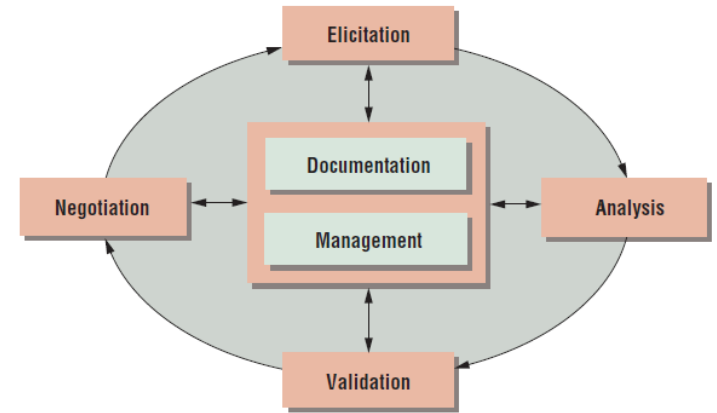
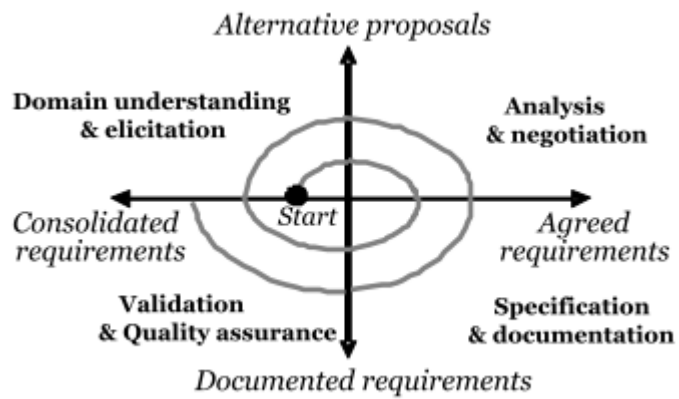
eg medical app or banking section may have regulations to follow more to the book compared to a game app



- Processes come in different “flavors”

- Linear versus incremental  
clear idea of what customer want
- Prescriptive versus exploratory  
engineer has more place to explore idea
- Customer versus market-oriented  
know our specific customer to talk to
- Etc.

eg. develop a text processing app, got market status, but don't have access to users.





# Linear versus incremental

- **Linear** when
  - Clear, stable, a priori known requirements
  - Low risk (of developing wrong system)
  - Relatively short duration of project
  - Often produce a formal requirements document
- **Incremental** when
  - Evolving requirements
  - High risk (of developing wrong system)
  - Long duration of project
  - Ability to change requirements is important
  - Often deal with evolving product backlogs, user stories, epics

# Prescriptive versus explorative



- **Prescriptive**
  - Specification is a contract: requirements binding, must be implemented
  - When tendering design and implementation
  - Development of specified system may be outsourced
  - Functionality determines cost and deadlines
  - Frequently combined with linear processes
- **Explorative**
  - Only goals are known: concrete requirements have to be explored
  - Stakeholders strongly involved: continuous feedback
  - Requirements can be prioritized and negotiated
  - Deadlines and cost constrain functionality
  - Typically only with incremental processes (issues, tickets, change requests)

# Customer versus market-oriented



- **Customer-specific product development**
  - System is ordered by customer and developed by vendor
  - Individual persons can be identified for all stakeholder roles
  - Stakeholders on customer side are main source for requirements
- **Market-oriented product development**
  - System developed as a product for the market
  - Prospective users typically not individually identifiable
  - Requirements are specified by vendor
  - Marketing and system architects are primary stakeholders
  - Vendor has to guess / estimate / elicit the needs of potential customers

# Another view



Self-study

- “Traditional” RE
  - Result: A requirements specification (document)
  - Basis for contracting and implementation
  - Typically found in customer/supplier situations
- “Agile” RE
  - Result: Product backlog, user stories, epics
  - Basis for sprint planning
  - Typically found in “in-house” development projects
- “Change-based” RE
  - Result: Issues, tickets, change requests
  - Basis for product maintenance and evolution
  - Typically found in in many “long-lived systems”
- “Code-driven” RE
  - Results: Code comments, man pages, mailing lists
  - Basis for understanding and changing code
  - Typically found in open source projects

# Some typical RE process configurations



Self-study

	Participatory	Contractual	Product-oriented
Linear		✓	
Incremental	✓	✓	✓
Prescriptive		✓	
Explorative	✓		✓
Customer-specific	✓	✓	
Market-oriented			✓

all have their trade offs



## Participatory

- Vendor + customer collaborate closely
- Customer stakeholders heavily involved in specification + development

# Some typical RE process configurations



	Participatory	Contractual	Product-oriented
Linear		✓	
Incremental	✓	✓	✓
Prescriptive		✓	
Explorative	✓		✓
Customer-specific	✓	✓	
Market-oriented			✓



## Contractual

- Spec: contract for development by people not involved in specification
- Little stakeholder interaction after initial requirements phase



# Some typical RE process configurations



Self-study

	Participatory	Contractual	Product-oriented
Linear		✓	
Incremental	✓	✓	✓
Prescriptive		✓	
Explorative	✓		✓
Customer-specific	✓	✓	
Market-oriented			✓



## Product-oriented

- Vendor specifies and develops software to sell it as product

# Example process configuration: start-ups

The Journal of Systems and Software 146 (2018) 130–151



Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: [www.elsevier.com/locate/jss](http://www.elsevier.com/locate/jss)



Recommended reading

## An anatomy of requirements engineering in software startups using multi-vocal literature and case survey



Nirnaya Tripathi<sup>a,\*</sup>, Eriks Klotins<sup>c</sup>, Rafael Prikladnicki<sup>b</sup>, Markku Oivo<sup>a</sup>,  
Leandro Bento Pompermaier<sup>b</sup>, Arun Sojan Kudakacheril<sup>a</sup>, Michael Unterkalmsteiner<sup>c</sup>,  
Kari Liukkunen<sup>a</sup>, Tony Gorschek<sup>c</sup>

<sup>a</sup> M3S Research Group, University of Oulu, Oulu, Finland

<sup>b</sup> Computer Science School, PUCRS Pontifical Catholic University of Rio Grande do Sul, Brazil

<sup>c</sup> Software Engineering Research Lab, Blekinge Institute of Technology, Karlskrona, Sweden

### ARTICLE INFO

#### Article history:

Received 8 February 2018

Revised 13 July 2018

Accepted 27 August 2018

Available online 28 August 2018

#### Keywords:

Requirements engineering

Software startups

Multi-vocal literature review

Case survey

### ABSTRACT

**Context:** Software startups aim to develop innovative products, grow rapidly, and thus become important in the development of economy and jobs. Requirements engineering (RE) is a key process area in software development, but its effects on software startups are unclear.

**Objective:** The main objective of this study was to explore how RE (elicitation, documentation, prioritization and validation) is used in software startups.

**Method:** A multi-vocal literature review (MLR) was used to find scientific and gray literature. In addition, a case survey was employed to gather empirical data to reach this study's objective.

**Results:** In the MLR, 36 primary articles were selected out of 28,643 articles. In the case survey, 80 respondents provided information about software startup cases across the globe. Data analysis revealed that during RE processes, internal sources (e.g., for source), analyses of similar products (e.g., elicitation), uses of informal notes (e.g., for documentation), values to customers, products and stakeholders (e.g., for prioritization) and internal reviews/prototypes (e.g., for validation) were the most used techniques.

**Conclusion:** After an analysis of primary literature, it was concluded that research on this topic is still in early stages and more systematic research is needed. Furthermore, few topics were suggested for future research.

© 2018 Elsevier Inc. All rights reserved.

# Agenda

1. A closer look at requirements
2. Requirements engineering activities
3. Feasibility studies
4. Stakeholder analysis

# Feasibility studies (1)

- “Short” and focused study before committing to a project
  - Decides whether or not a proposed system is worthwhile
  - Outcome: go ahead, do not go ahead, think again
  - Example questions
    - What are current process problems?
    - How will proposed solution help?
    - What if the solution wasn’t implemented?
    - What will be the integration problems?  
integrate it into the ecosystem
    - Is new technology needed? What skills?

Might offer input into  
other RE activities

- Types of feasibility

- Technical/operational feasibility
- Economic feasibility
- Schedule feasibility



# Feasibility studies (2)

too much on feasibility, need to balance how much time and money being invested and getting an outcome, eg a 6month for app is alot, 6K for govenment

- Often come under different names

- Scoping study

put in too much time and no outcome isnt good, need to stop investing in that time earlier on

- Often part of business plans, project proposals, market studies

could go to point of no return where you can't justify how much resources you do have.

- Duration and depth vary significantly

- Depending on resources, **criticality of project, risk of failure**

- Variations between organizations, projects, etc.

# Why are feasibility studies difficult

- **Uncertainty**
  - Clients unsure of scope, solution ill-defined
  - Benefits/risks hard to identify and to quantify
  - Effort and resources are rough estimates
  - Organizational changes may be needed
- **Advocacy**
  - Enthusiasm and ownership
  - **Who will pay for it?**
- **Risk to run a full scoping study or requirements analysis**

# Agenda

1. A closer look at requirements
2. Requirements engineering activities
3. Feasibility studies
4. Stakeholder analysis

stakeholders are different with diff views, and can impact the product or be impacted by the product in diff ways

# Stakeholder analysis

- Typical steps
  - Identify stakeholders: identify candidates (see later)
  - Classify stakeholders (see later) prioritize them to differentiate them. and going future to see how much we inform them - eg a funder who may want to stay in the loop but not have too much technical information.
  - Describe their concerns, e.g.,
    - What is their stake? what are stakeholders, their interest and concerns and how this is relevant to us.
    - What are their interests?
    - How are they impacted?
    - What is their impact?
    - [If no answer to these questions – maybe not a real stakeholder?]
  - Prioritize stakeholders
    - Ranking?



# Stakeholder analysis – identify stakeholders

- Identify stakeholder roles – **who are they**
  - End user, customer, operator, support staff, PM, regulator, etc.
  - **Negative stakeholders**
  - Start with exhaustive list (be broad), then “filter”
- **Identify concrete person(s) for each stakeholder role**

this matters to have a concrete person as you have a persona and personality documented. but this seem artificial as its not an actual person.

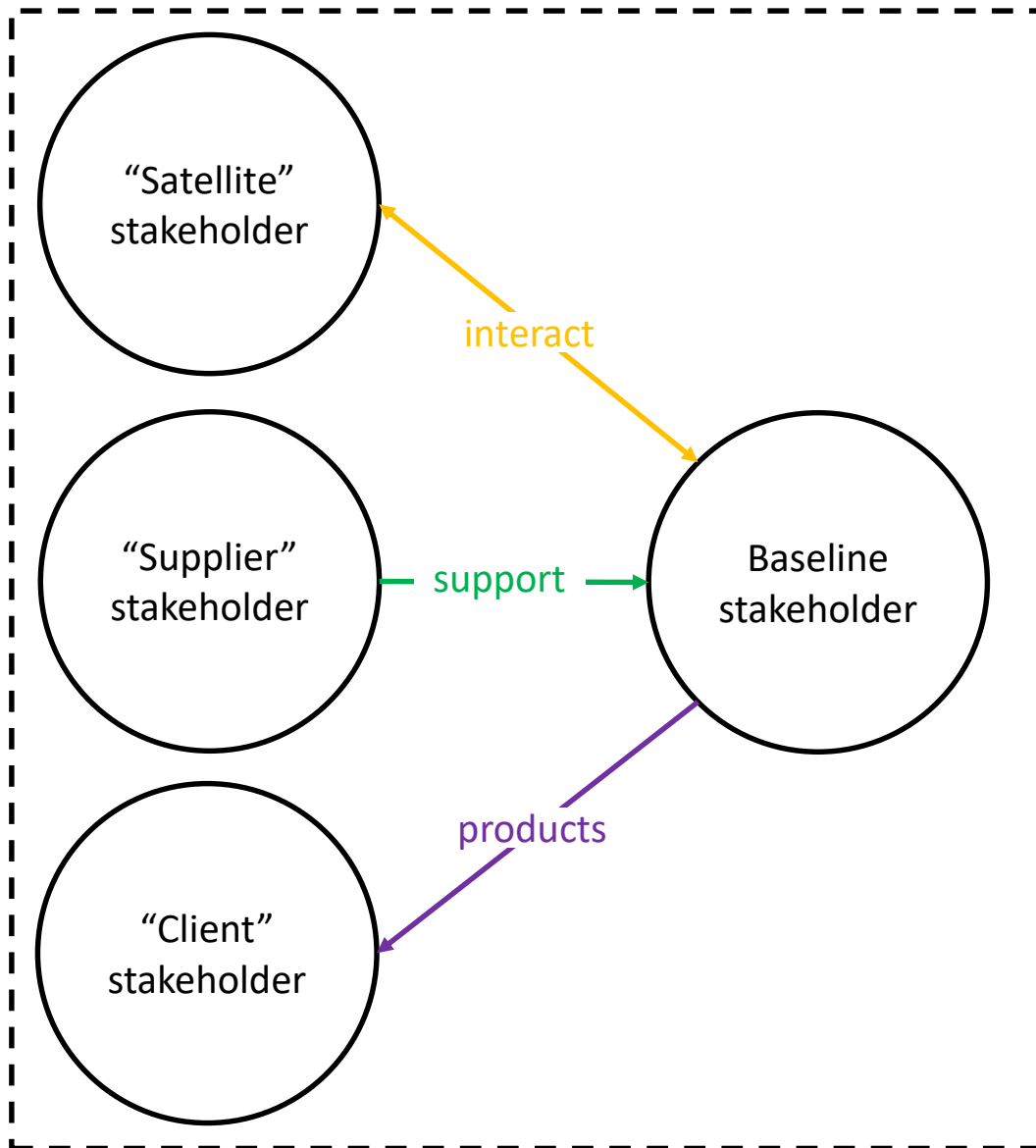
a stakeholder is real and actual existinig person that you can reach out to be in touch

if you can;t identify a concrete person, maybe its not a relevant

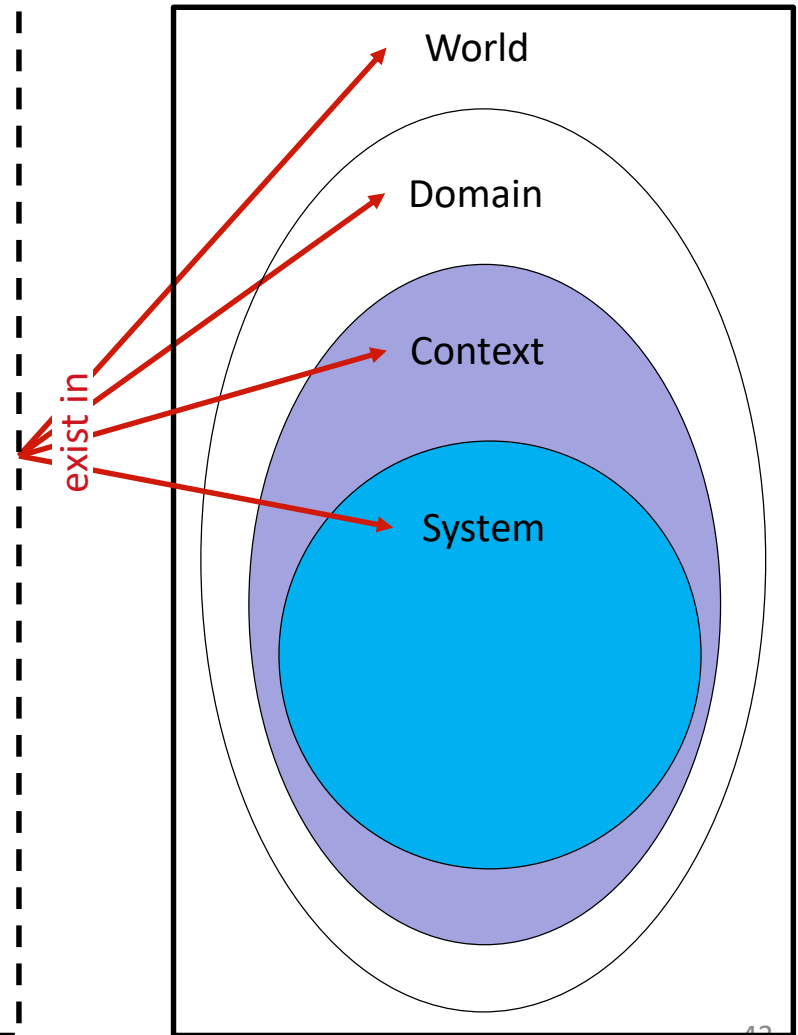
# How do identify candidate stakeholders

- Start with baseline
  - Identify all specific roles with baseline stakeholder groups, e.g.,
    - Users: e.g., primary, secondary end users
    - Developers: e.g., those who design, develop, deploy, maintain involve with delivery of product so impact product
    - Legislators: e.g., government, trade unions, legal representatives, auditors
    - Decision makers: e.g., those with financial interests, responsible for sale, etc.
- Then explore network of stakeholders around baseline

# Network of stakeholders around baseline



eg. law



# Network of stakeholders around baseline

- “Supplier” stakeholders
  - Support baseline stakeholder, provide information or supporting tasks
- “Client” stakeholders
  - Process or inspects the products of the baseline
- “Satellite” stakeholders
  - Interact with the baseline in a variety of ways

# Stakeholders and context

- Stakeholders and requirements need to be framed in a **context**
  - Part of environment relevant to understand system and requirements
- Context and domain define how and where system will be used
  - **Also often source of process + project requirements**

# Which context?

“For each gate, the system shall count the number of passengers passing through.”

Gate control software

“The system shall provide effective access control to gates.”

Everything (equipment, computers, boarding passes, software)

“The system shall operate in a temperature range of -30 to +30.”

Computer hardware and devices

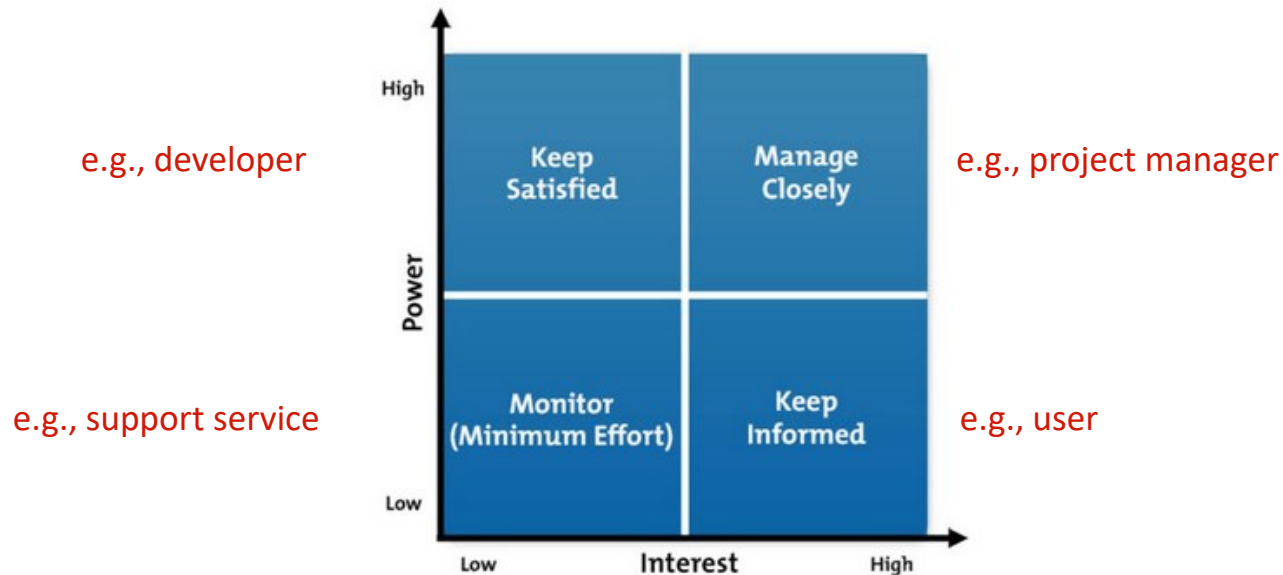
“The gate agent shall be able to run the system in three modes: normal (gate unlocked for one entry when a boarding pass is read), locked (all gates locked), and open (all gates unlocked).”

Gate access control software

# Stakeholder analysis – classify stakeholders

not all stakeholders are the same, so this is where classification comes in so we can treat each stakeholder appropriately

- Classify stakeholders – **how important are they**
  - Importance (power, interest): critical, major, minor
  - Category: user, development, environment
- Power-interest grid



# How to capture end user stakeholders

- Thinking in terms of generic users not very helpful
  - Difficult to identify precise requirements
  - Hard to define “value”: what value, for whom, how is it delivered
  - Few systems target homogenous group of people
- Understand and describe users’ characteristics
  - Put face on coherent “user data structure” to engage with during design
  - Highlight user needs, attitudes, values, goals, context

personas and user profiles





# Personas and user profiles

- Personas and user profiles
  - Support thought process to identify and test requirements
- Typical steps
  - Survey potential users
  - Categorize them into groups
  - Propose, evaluate, and create relevant set of personas

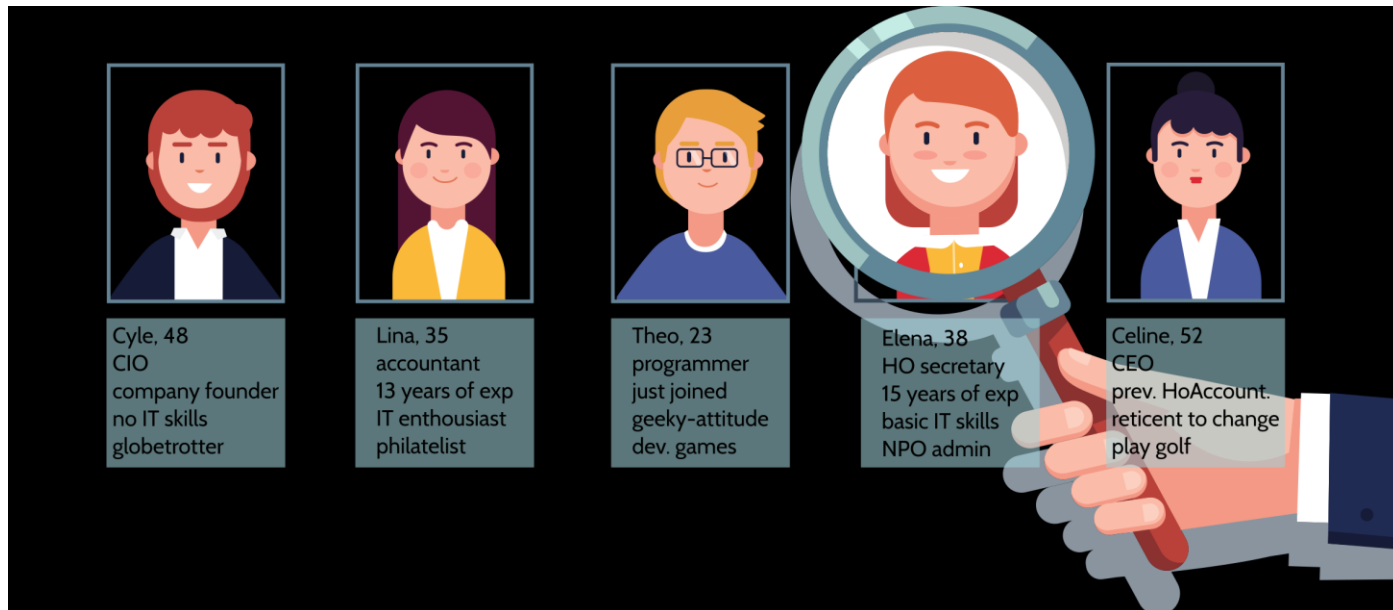
what are examples of users and personas and their relationship?

user profiles are less concrete and more aggregated class of users compared to personas

personas can be aggregated and fall into user profiles

# Personas

- Fictional (but realistic) people and characters<sup>1, 2</sup>
  - Well-defined characteristics, behavioural patterns and goals
  - Physical, cognitive characteristics (i.e., possible disabilities, motivation)
  - Social, ethnical or religious details, education, competencies, experience
    - E.g., skills, environment in which they work, how they use computers



<sup>1</sup> A. Cooper: The inmates are running the asylum (1999)

<sup>2</sup> Blog about Cooper's book: <https://blog.adilmajid.com/notes-on-the-inmates-are-running-the-asylum-by-alan-cooper/>

# User profiles



Self-study

- “Classes” of (abstract but realistic) users
  - User archetypes synthesised from common characteristics
  - Share common “use cases” and concerns
  - Highlight differences between users



# Example template for personas

How do these characteristics impact our project, process, requirements, architecture?

- Persona name
- Persona role
- Demographic information
  - Age
  - Gender
  - Income
  - Educational level
  - Residential environment
- Personal quote
  - Quote from actual customer who represents persona
- Use cases/user stories
- Biography
- Professional goals
  - Chen wants to become a better leader
- Motivators
  - Chen wants to provide the best service to her clients to raise up the ranks quickly
- Challenges
  - Chen works remotely and has trouble staying in sync with team
- Information sources
  - Chen reads about the latest industry trends on Tech Crunch

# Simple example template for personas

- Goals
- Background
- Age
- Gender
- Location
- Behaviors
- Spending habits
- Pain points
- Needs

# How to create personas




Approaches	Methodologies	Advantages	Disadvantages
Quantitative	<ul style="list-style-type: none"><li>• Factor analysis</li><li>• Principal component analysis</li><li>• Cluster analysis</li><li>• Correspondence analysis</li><li>• Association rule mining</li></ul>	<ul style="list-style-type: none"><li>• Grounded in data from large user community</li><li>• Easy to explain</li></ul>	<ul style="list-style-type: none"><li>• Lack of contextual richness</li></ul>
Qualitative	<ul style="list-style-type: none"><li>• Ethnography</li><li>• Grounded theory</li><li>• Affinity diagrams [12], [13], [17]</li><li>• Expert panels</li><li>• Latent semantic analysis</li></ul>	<ul style="list-style-type: none"><li>• Rich contextual information</li></ul>	<ul style="list-style-type: none"><li>• Potential lack of credibility and rigor</li><li>• Take very much time and many resources to develop</li><li>• Quality is inconsistent across different persona designers</li></ul>
Current Mixed Quantitative and Qualitative	<ul style="list-style-type: none"><li>• Creating groups of users quantitatively and adding richness to those groups qualitatively</li></ul>	<ul style="list-style-type: none"><li>• Grounded in data from large user community</li><li>• Rich contextual information</li></ul>	<ul style="list-style-type: none"><li>• Groups are made based solely on quantitative data</li><li>• Do not leverage the full potential of qualitative data</li></ul>

# Personas to the extreme

- Success of Netflix due to high customer retention rates
  - Collect data from their millions of subscribers
  - Implement data analytics to discover behavior and patterns
  - E.g., to generate personalized recommendations
- Models each customer and their interactions
  - User profile that goes beyond conventional persona user modelling, e.g.,
    - Time and date a subscriber watched a show
    - Device used to watch a show
    - If the show was paused, if viewers resume after pausing
    - If viewers finish an entire TV show, time it takes to finish a show
    - Etc.
- Are these user models still personas?

**NETFLIX**

# Summary

1. A closer look at requirements 
2. Requirements engineering activities 
3. Feasibility studies 
4. Stakeholder analysis 