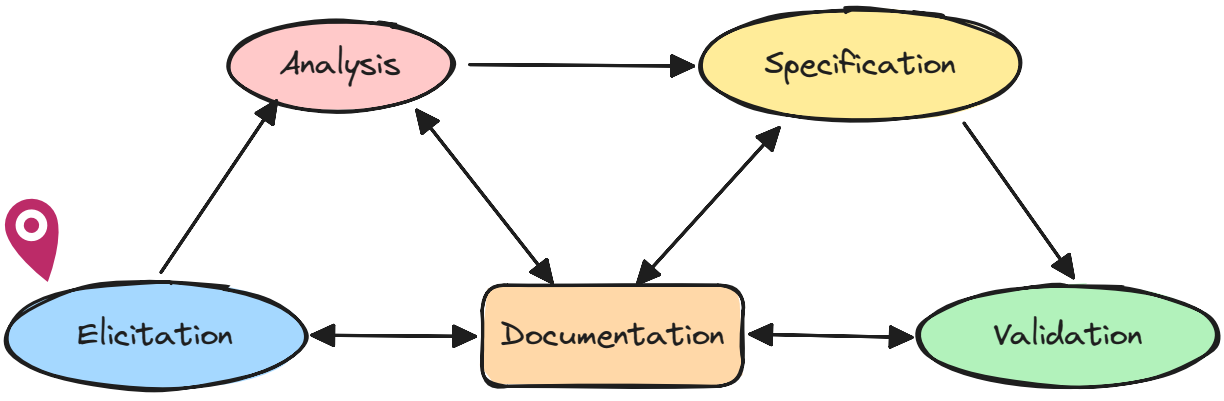
McMaster University

Stakeholders & Requirement Sources

Dr. Sébastien Mosser
Lecture #05
CS/SE 3RA3 - Fall 2023

McSCert

Where are we?



1 Stakeholders

2 Requirements Sources

3 Elicitation in practice

1

(Fantastic) Stakeholders

And where to find them

Stakeholders & Requirements Analysts

- **Stakeholders**:
  - **Source of knowledge** about the work
  - Visionaries of **what the work should be**
  - **Evaluators** of requirements analyst's ideas for the product

Two fundamentally different roles

- **Requirements analyst**:
  - **Translator** of stakeholders' input **into product specification**
  - Visionary of a **new or better product**
    - *Observe and learn from stakeholders*
    - *Interpret stakeholders' work*
    - *Invent new or better ways to do the work*
    - *Record the results in requirements documents, analysis models, …*

Who is a stakeholder?

- Long story short: **Anyone who brings value to the project**
- Can be "**directly**" involved
  - Clients, customers, users (past and future),
- Or be "**indirectly**" involved
  - Buyers, managers, domain experts, developers, marketing and QA people, lawyers, people involved in related systems
- Like anything in RE, **some "patterns" exists**, but overall it **needs to be tailored precisely** to the product you're building

Identifying the right pool of stakeholders requires expertise

Stakeholder — Owner/Client

- **Who?**
  - The person **ultimately paying** for the software to be developed
  - Almost always, **the client has the last say** in what the product does
- **Where to find them?**
  - "Bespoke"/customized system: the person with the **chequebook**
  - Mass market: company developing the software/**marketing** department
  - In-house: **manager** of the product's users
- Often **accessed through a proxy** (e.g., hierarchy, "champion")

Stakeholder — Customer/Buyer

- **Who?**
  - The person who **buys the software** after it is developed
- **Where to find them?**
  - May be the same as the **client**
  - May be the same as the **user**
    - For what requirements will they pay? Which are trivial? Excessive?
- Must be an **active participant** in the project (or be actively represented)

Stakeholder — Software Engineer

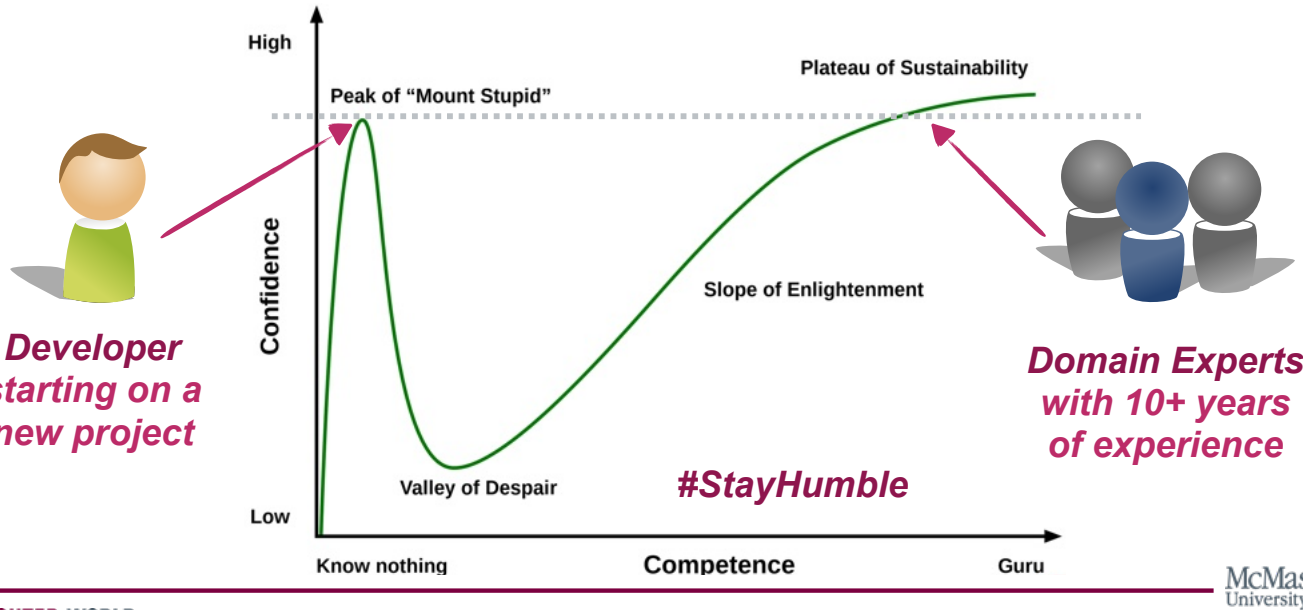
- **Who?**
  - Expert who knows **technology**, **processes**, **infrastructure**, …
  - Represent the rest of the **development team**
- **Where to find them?**
  - Look around the room 😅
- Works on **estimation**: is the project **technologically/economically** feasible?
  - They **educate** the buyer on technology, **recommend new features** that would benefit from these technologies

Not every "disruptive" idea is actually possible. Think Theranos. Or FTX.

Stakeholder — Domain Expert

- **Who?**
  - Expert who **knows the problem** domain/work involved
- **Where to find them?**
  - **Close to your application domain**!
    - *Financial expert for finance management software*
    - *Aeronautical engineers for air navigation systems*
    - *Meteorologists for weather forecasting system*
- **You are not a domain expert**. They are. **They know**. **You don't**. Be Humble.
  - As an RE, you ***must not be perceived as an enemy***.

"You know nothing": The Dunning-Kruger Effect

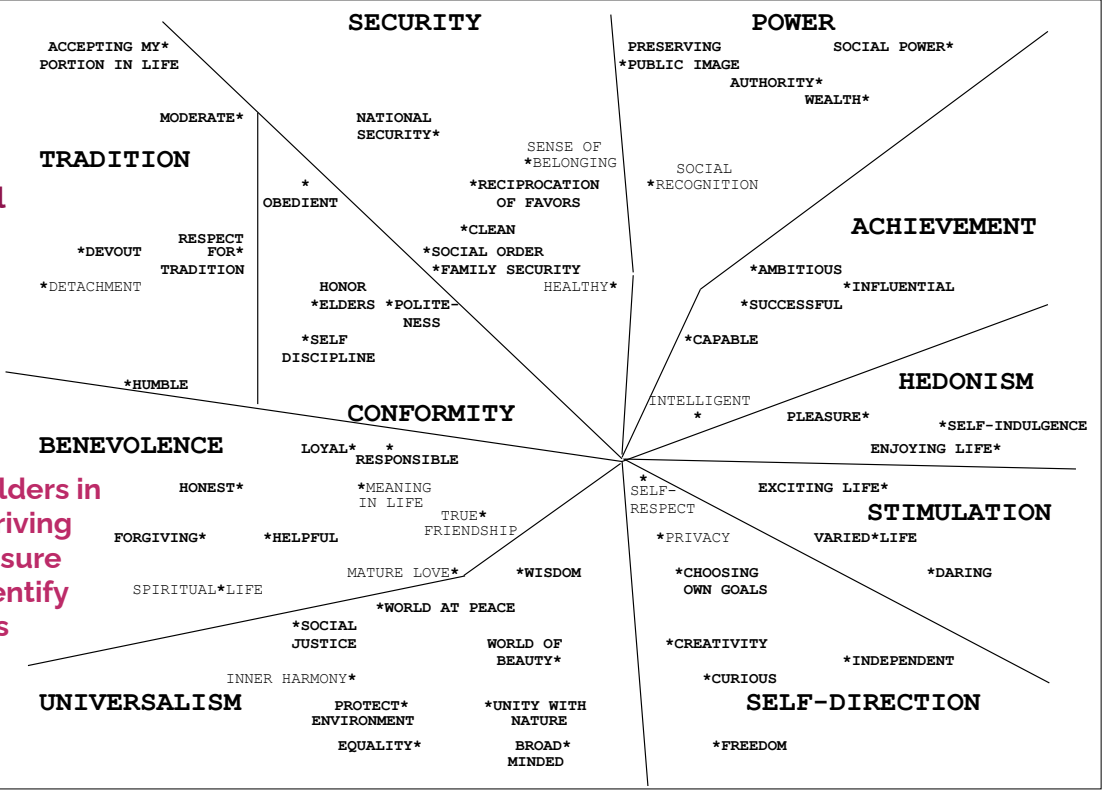


Stakeholders — Users

- **Who?**
  - Users of the **current** and **future systems**
- **Where to find them?**
  - Experts of the **current system** *(which features to keep, needs improvement, …)*
  - Experts in **competitors' products** *(At UofT, they're using XXX instead of Mosaic)*
  - Come with **special needs**/requirements *(training, usability, help, …)*
  - **Diversity is essential** *(seniority, handicap, background, …)*
- Finding the **right/representative** users' pool is hard.

The list of potential stakeholders is virtually endless

Thinking of stakeholders in terms of values driving them helps to ensure coverage, and identify redundancies



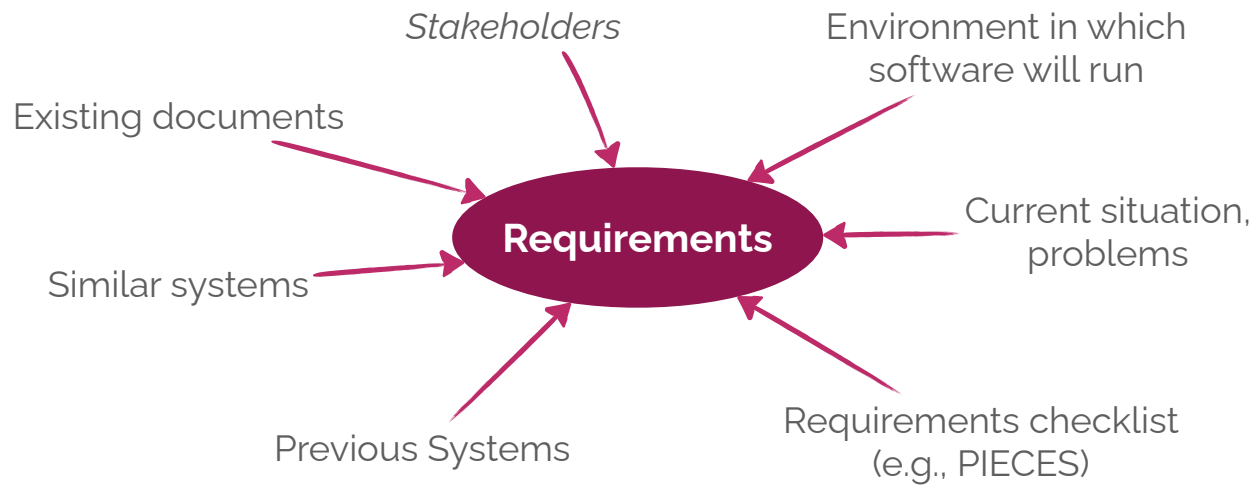
(Schwartz, 2006)

2

Requirements Sources

Not everything is human-driven

Sources of Requirements



http://www.cs.toronto.edu/~sme/CSC340F/readings/PIECES.html

Do not become the enemy!

You're here to understand, not to judge or "know better"

Eliciting Requirements

Which engineering techniques can we use?

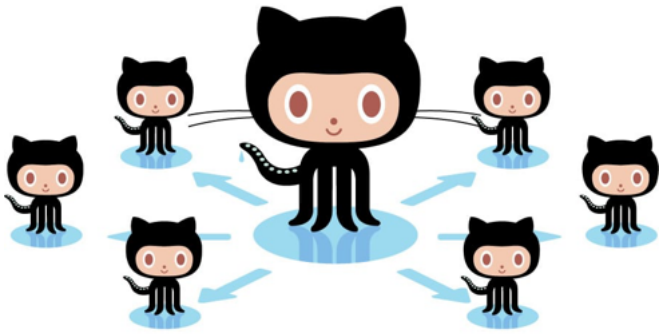
3

The Problem with Requirements

(Related to "bike shedding" at the design level)

How to Collect Requirements Efficiently?

- **Traditional** Elicitation techniques
  - *Background readings*
  - *Interviews*
  - *Questionnaire/surveys*
  - *Observing Behaviour*
- **Collaborative** Elicitation Techniques
  - *Brainstorming*
  - *Rapid Application Development*
  - *Prototyping* *(often used as a validation technique for requirements)*
- **Assisted** Elicitation Techniques
  - *Model-based approaches: Scenario-based, use cases, misuse cases*
  - *AI-driven approach: Knowledge exploration*



Background readings

- **Familiarizing** yourself with the organization
- **Sources of information**:
  - Company **report**, organization charts, policy manuals, policies
  - Job descriptions, **documentation** of existing systems
- **Advantage**:
  - Help **prepare meetings** with the right persons/knowledge
- **Disadvantage**:
  - **Time-consuming**: separating the wheat from the chaff, lots of details, …

Interviews

- **Different types** of interviews (coarse-grained; see HCI course for more)
  - **Closed interviews**: involves a pre-defined set of questions
  - **Open interviews**: No pre-defined agenda
- Good for **eliciting non-tacit** knowledge
  - **General description** of work
  - **Explicit** procedure
  - Difficulties faced, **Critical incident** reporting
- **Not good at**:
  - **Concealed** knowledge, **tacit** knowledge, domain **understanding** (jargon)

Everyone lies.
(not necessarily on purpose)

Questionnaires

- Choose an **adequate sample size** and selection of "targets"
- Defining questions is **a very difficult exercise**:
  - Avoid **ambiguous** questions (messing up your results)
  - Avoid **leading** questions (introducing bias)
  - Avoid **loaded** questions (questions presupposing conditions)
  - Avoid **open-ended** questions (interviews instead)
- **Advantage**: quickly collect **quantitative data** from **large pool** of stakeholders
- **Disadvantage**:
  - Provide **less context** than interviews. **Really hard to design**.
  - Little room for stakeholders to really convey their needs (**self-realizing prophecy**)

Observing Behaviour

- Also called "**Shadowing**" *(think "Boss undercover")*
- **Spectrum** of techniques:
  - From **silent observation** to **apprenticeship**
  - From **auditing** (external) to **acceptance** (ethnography) *(think "gorilla in the mist")*
- **Good for**:
  - **Understanding behaviour** that is **hard to describe**,
  - Differences between "**What I say**" and "**What I do**" *(not necessarily voluntarily)*
- **Difficulties**:
  - Time-consuming, **depends on who is observed** and when.

Conclusions

TL;DR: Takeaway Messages

Takeaway Message #StayHumble

- **Identifying Stakeholders**
  - Is a **tailored** job, even if we can find some "**usual suspects.**"
  - **Less is more**. But not always. It's complicated. **It depends**.
    - *How the hell is he going to grade us if "it depends"? This course is pure b*llsh*t. Gonna rant on reddit.*
- **Sources of requirements** are **more diverse** than just stakeholders
- **Engineering approaches** are necessary to sort this mess out
  - Needs to be **adapted to objectives**, goals, needed knowledge, …
  - And also **stakeholders' availabilities**!

McMaster University

ENGINEERING

Computing & Software

McSCert

Dr. Sébastien Mosser

Associate Professor

mossers@mcmaster.ca

https://mosser.github.io

ITB 131 (appointment only)