University of Oviedo

EN
English

Universidad de Oviedo

School of Computer Science

# Software architect role and stakeholders

SOFTWARE ARCHITECTURE

Course 2020/21

Jose E. Labra Gayo

School of Computer Science

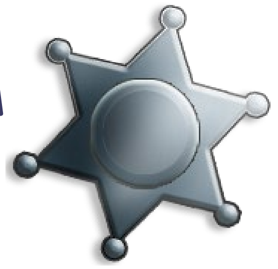
## Role of software architect

# Role of software architect

Expectations of an architect

- Make architectural decisions
- Continually analyse the architecture
- Keep current with existing trends
- Ensure compliance with existing decisions
- Diverse exposure and experience
- Have business domain knowledge
- Possess interpersonal skills
- Understand and navigate politics

Software architect is a role, not a rank



## Make architectural decisions

Define architecture decisions and design principles

Architect should guide technology decisions

Keep decision records

Analyse pros and cons

## Continually analyse the architecture

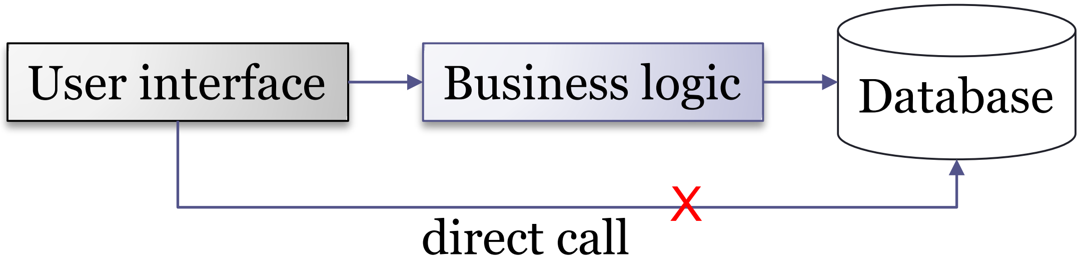
- Continually analyse the architecture and technology
- Being responsible for technical success of project
- Be aware of structural decay
- Strive for consistency
  - Organize the code into packages, folders, modules, ...
  - Define boundaries, guidelines, principles,...
  - Include testing and release environments into projects

## Ensure compliance with existing decisions

Architects usually impose some constraints

Example:

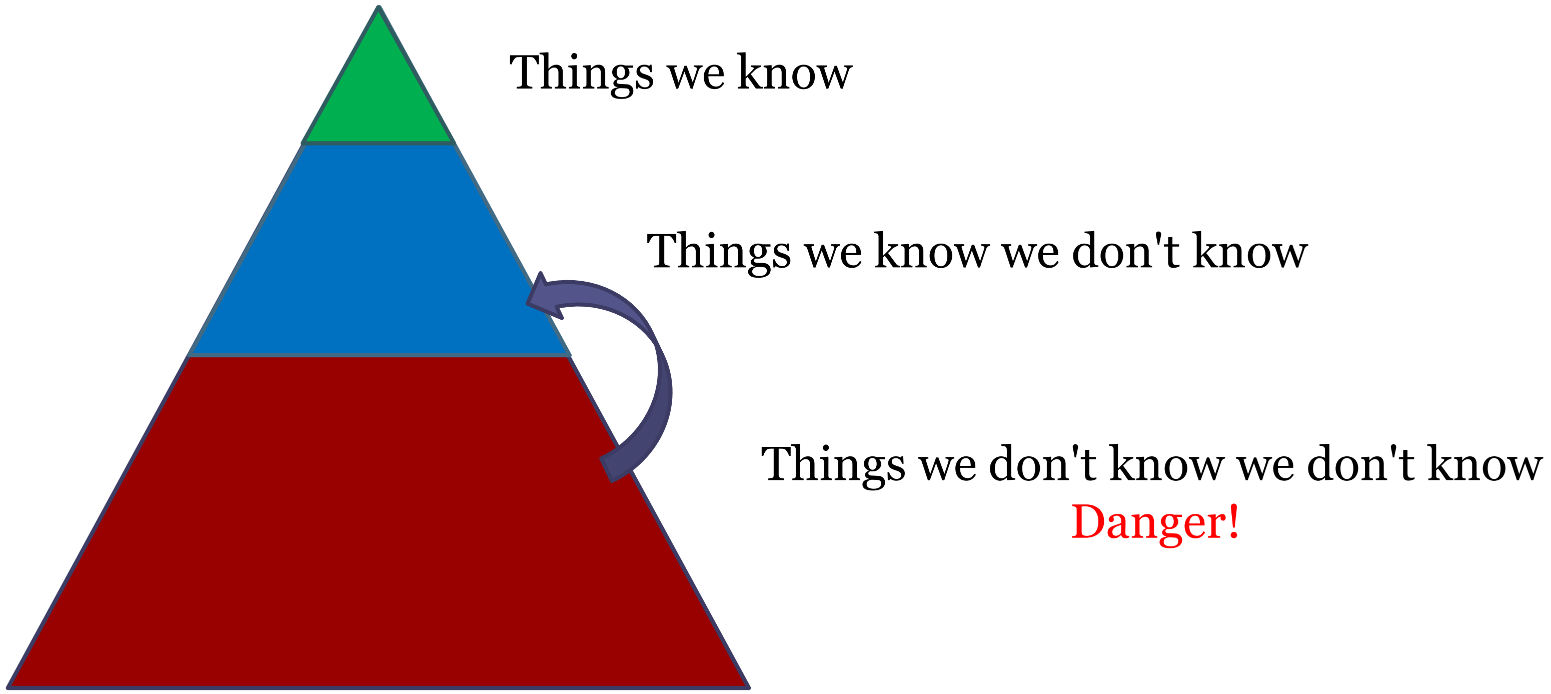
- Database access from User Interface constraint
- Developers could bypass it



## Keep current with existing trends

Be aware of latest technology and industry trends

- Decisions made by architect = long lasting and costly
- Good architects know what they know and what they don't know


Things we know
Things we know we don't know
Things we don't know we don't know
Danger!

## Diverse exposure and experience

Have exposure to multiple and diverse technologies, frameworks, platforms, environments,...

It doesn't mean being an expert in each of them

...but at least be familiar with varying technologies

Technical breadth better than technical depth

## Business domain knowledge

Architect expected to have certain level of business domain knowledge

Understand business problem, goals and requirements

Effectively communicate with executives and business users using the domain language

## Possess interpersonal skills

- Software architect = leader
- Teamwork and leadership skills
- Technical leadership
  - Be inclusive and collaborate
    - Help developers understand *the big picture*
  - Get hands-on
    - Be engaged in the delivery
    - Low-level understanding
    - Coding as part of the role
    - Code reviews and mentorship

*"no matter what they tell you, it's always a people problem", G. Weinberg*

## Understand and navigate politics

- Understand the political climate of the enterprise and be able to navigate the politics
  - Architectural decisions affect stakeholders
    - Product owners, project managers, business stakeholders, developers...
  - Almost every decision an architect makes will be challenged
- Negotiation skills are required
  - Present and defend the architecture
- The software architect's elevator
  - Communication with the different layers

## Main concerns of software architects

- Specify quality attributes
  - How to do something
- Determine trade-offs and decisions
  - Why to do something
- Contain entropy
  - Define standards, conventions, toolsets for teams

## Working in teams

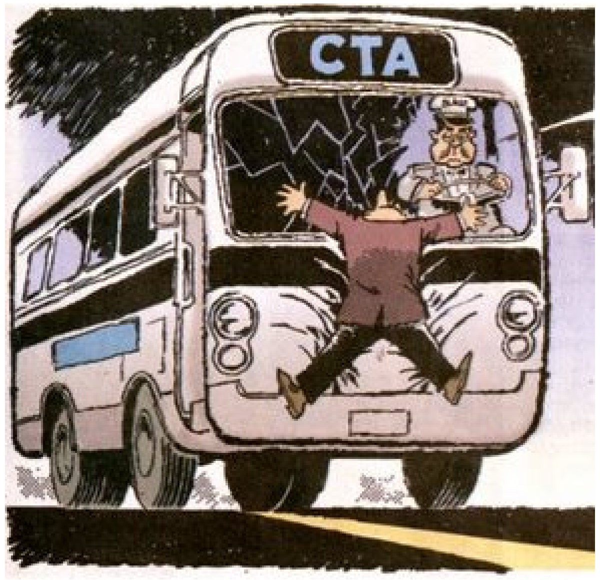
- Software engineering is a team endeavour
  - Social interactions
  - Architect personalities
  - Team topologies
  - Team size

## Hiding & the genius myth

- Insecurity
  - People are afraid of others judging their work in progress
  - Attempts to hide code
- The genius myth:
  - Tendency to ascribe success of team to a person
  - Examples: Bill Gates, Linus Torvalds, etc.
- Hiding considered harmful
  - Working alone increases risk

# The Bus factor*

- Number of people that need to get hit by a bus before your project is completely doomed
  - Unpredictable life events can happen
- Teamwork is mandatory to reduce risk
  - Ensure to have at least 2 people
  - Good documentation

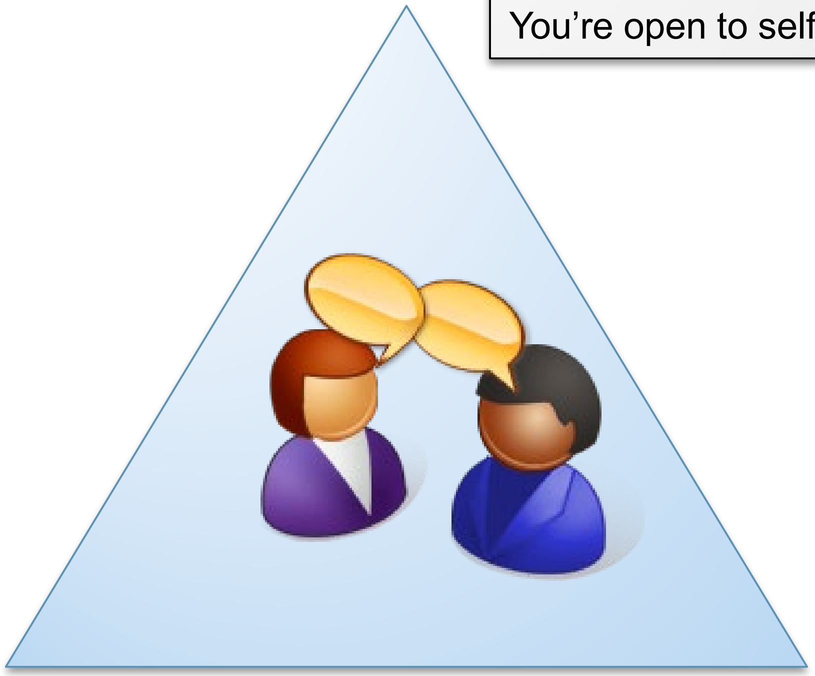


*Term coined at Google (Software Engineering at Google, 2020)

## 3 pillars of social interaction

Humility

You are not the center of the universe (nor is your code!)
You're neither omniscient nor infallible
You're open to self-improvement



Trust

You believe others are competent
You believe others will do the right thing
You're OK with letting them drive when appropriate

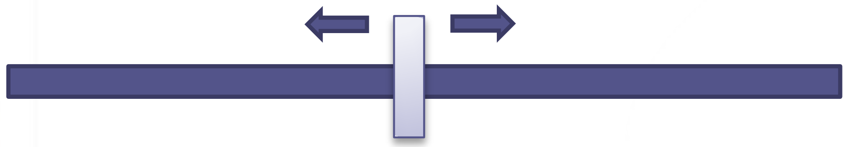
Respect

You genuinely care about others you work with
You treat them kindly
You appreciate their abilities and accomplishments

Software engineering at Google, T. Winters et al, 2020

# Architect personalities

Effective architect = trade-off between control freak and armchair architect



Control freak

Participate in all decisions
Decisions too fine-grained and low level
Participate in code development (bottleneck)

Armchair architect

Disconnected from development teams
Never around (jump from project to project)
Only participate in initial diagrams

## Team topologies

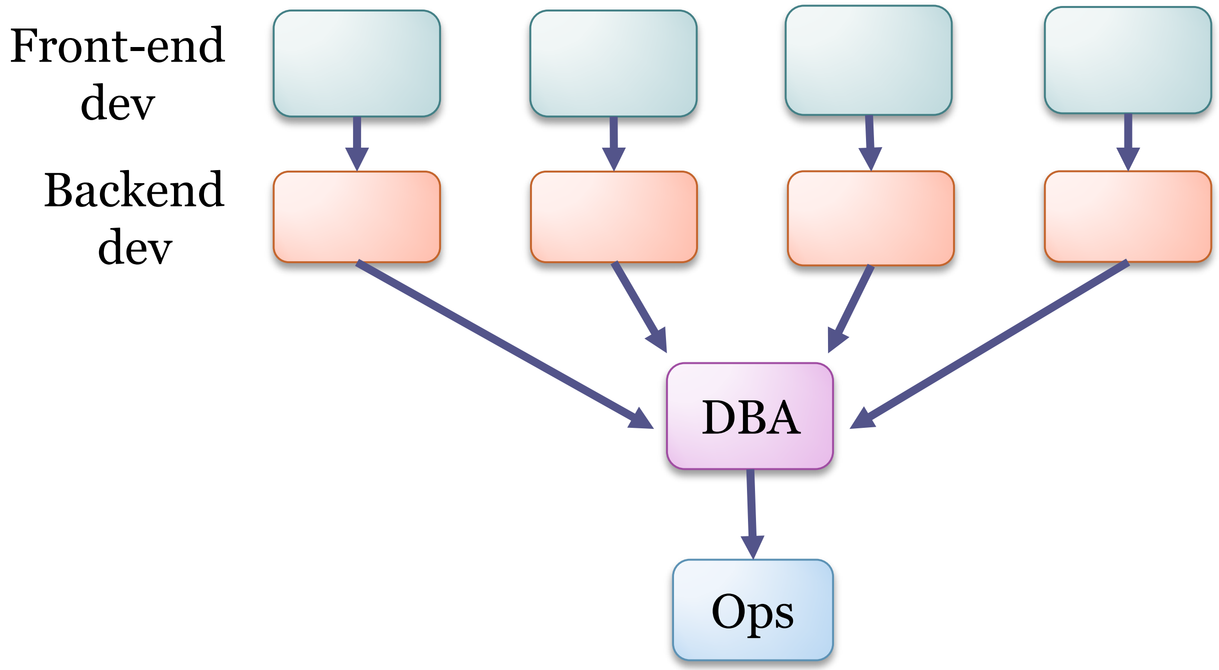
- Team topologies affect the systems
  - Communication structures
  - Team dynamics
  - Team size

"*Team assignments are the first draft of the architecture*", M. Nygaard

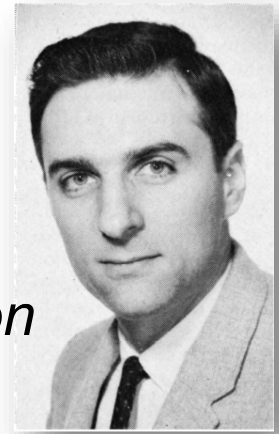
## Traditional team topology

Traditional work allocation:

- Existing teams are required for every new project
- Example: 4 teams: front-end, back-end, DBA and Ops



## Canway's law



*Organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations* [M. Conway, 1967]

Corollary:

*The best structure for a system is influenced by the social structure of the organization*

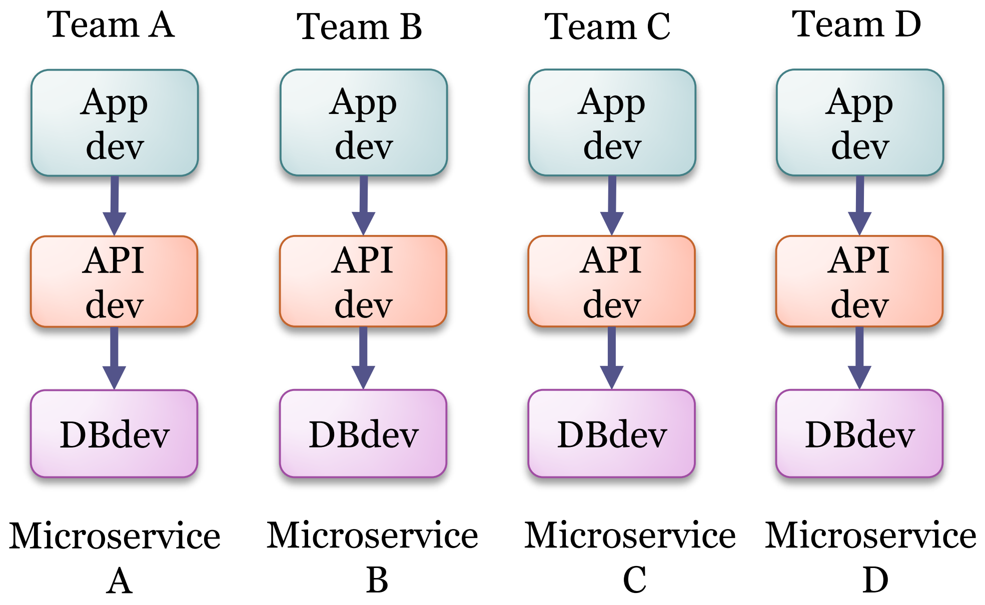
Example:

If there are 3 teams (design, programming, database), the system will naturally have 3 modules

Source: https://meekrosoft.wordpress.com/2013/06/25/conways-second-law/

# Inverse Canway Maneuver

- Evolve teams and organizational structure to promote the desired architecture
  - Create teams after the modular decomposition
  - Example with microservices



Amazon's principle: *You build it, you run it*

## Team size

Efficient team size can influence project success

Some warnings to be aware of:

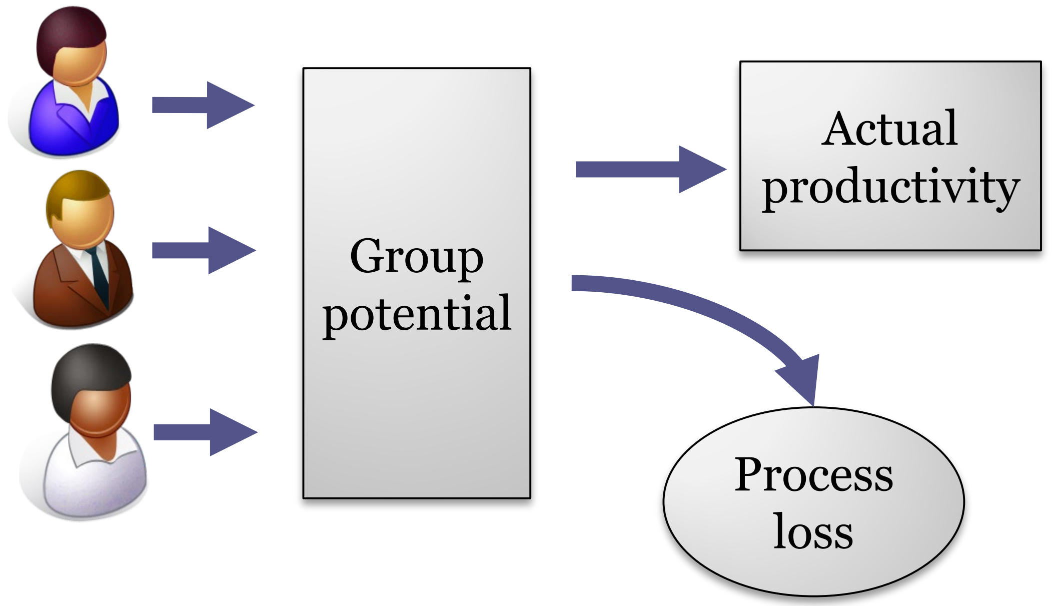
- Process loss
- Pluralistic ignorance
- Diffusion of responsibility

2-pizza rule: "if you can't feed a team with two pizzas, it's too large", J. Bezos

## Process loss

Difference between group potential and actual productivity

Some reasons: communication overhead, meetings,...



Brook's law. Adding manpower to a late software project makes it later

## Pluralistic ignorance

- When everybody publicly agree to something, but privately reject it because they think they are missing something obvious
  - Some architect decisions are not confronted

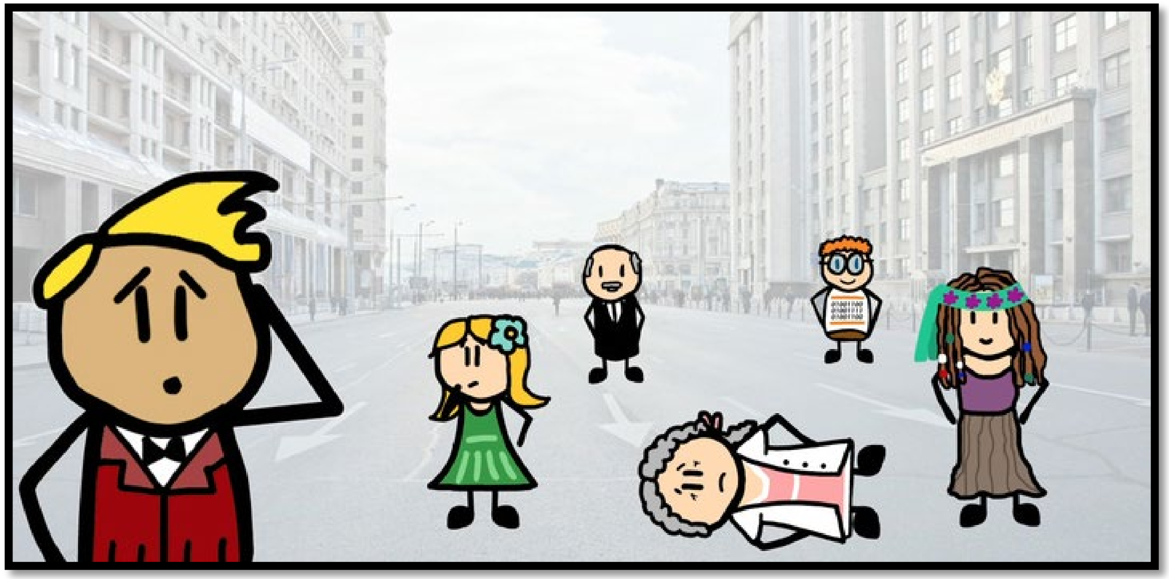
Emperor's new clothes fable
http://fablesfairytalesandsocialjustice.weebly.com/the-emperors-new-clothes.html

## Diffusion of responsibility

Bigger team size negatively impact communication

Some signs:

- Confusion about who is responsible for what
- Things getting dropped



## Leveraging checklists

Checklists = effective means to ensure some tasks are covered or addressed

- Error-prone tasks/steps that are frequently missed
- Make development teams more effective

Hawthorne effect: *If people know they are being observed their behaviour changes and will generally do the right thing*

## Stakeholders

## Stakeholders

- All parties that participate in the development or are affected by the system
- Can be person, a role or organization
- Typically have different concerns
  - Sometimes contradictory
- It's necessary to
  - Understand nature, source and priority of concerns
  - Identify and actively engage with them
  - Solicit their needs and expectations

Stakeholders (explicitly or implicitly) drive the whole shape and direction of the architecture to serve their needs

## Identifying stakeholders

All individuals, roles, organizations that:

- Should know the architecture
- Have to be convinced of the architecture
- Have to work with the architecture or with code
- Need the architecture documentation for their work
- Have to come up with decisions about the system or its development

## Identifying stakeholders

Internal

- Analyst
- Designer
- Business manager
- Developer
- Product owner
- Auditor
- UX designer
- Project manager
- . . .

External

- Customer
- End users
- Auditor
- Public authority
- Suppliers
- External service providers
- . . .

## Stakeholders' expectations

Expectations help to:

- Identify specific needs
  - Goal: achieve greater satisfaction of target audience
- Avoid unnecessary work
- Avoid documenting irrelevant things

Typical format:

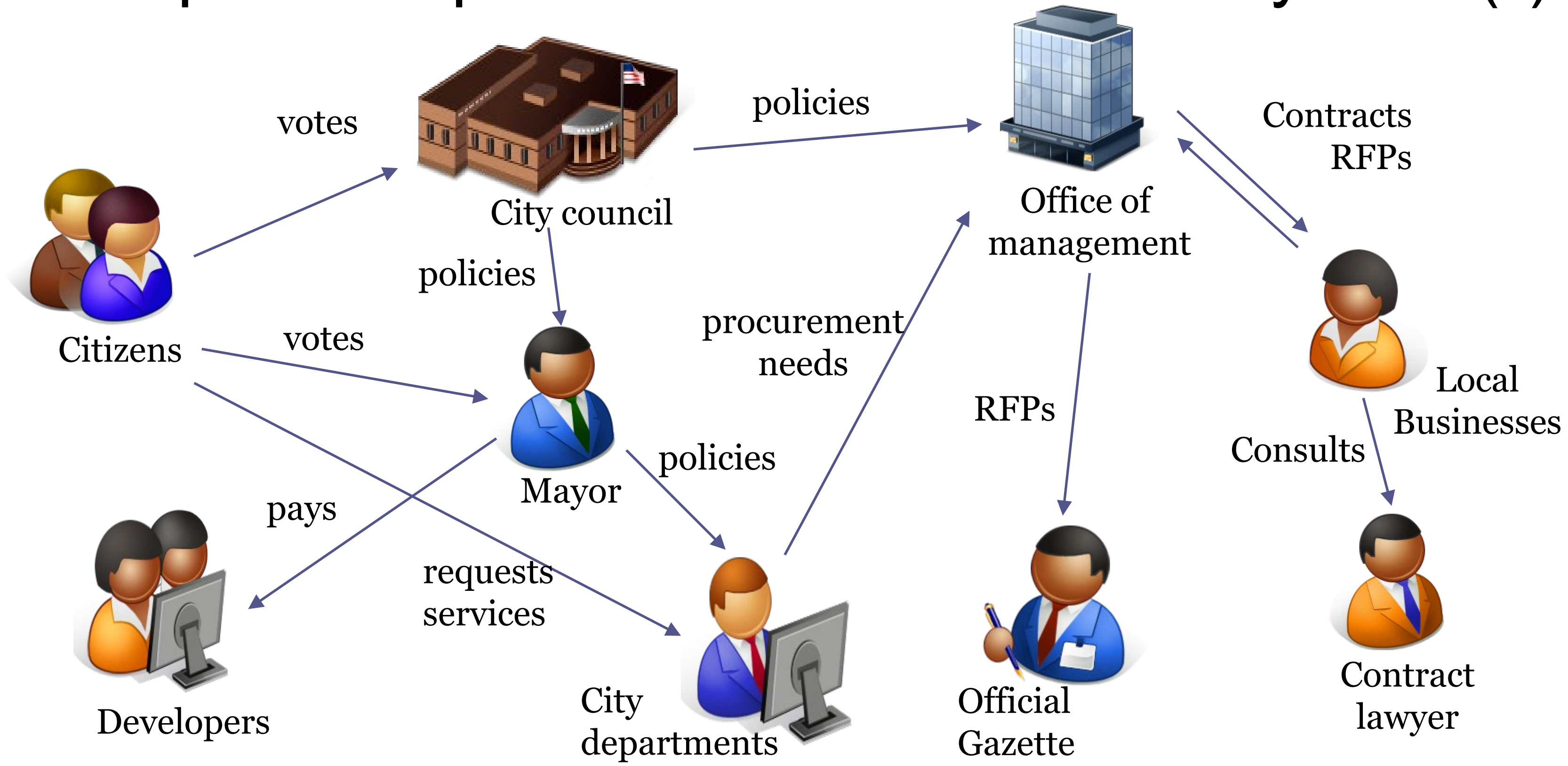
| Role/name | Contact | Expectations |
|---|---|---|
| | | |
| | | |

## Stakeholder map

Show people/roles involved or affected by system

- Include relationships/interactions

Example for a procurement automation system (*)



(*) Source: Design it!, M. Keeling

## Business goal statements

- Human-centered business goals
- Usually between 3-5
- Structure
  - Subject/stakeholder
  - Outcome: express the need as a measurable
    - How will the world change if system is successful?
  - Context
    - Some insight about the goal

| Subject | Outcome | Context |
|---|---|---|
| Mayor of the city | Reduce costs 30% | Avoid making budget cuts to essential services |
| Office of management | Review historical procurement data for the last 10 years | Historic data can help predict future contracts |