Lecture 10: Architecting for the eVoter

Jan Martijn van der Werf, Nishant Saurabh

Imagine the following situation

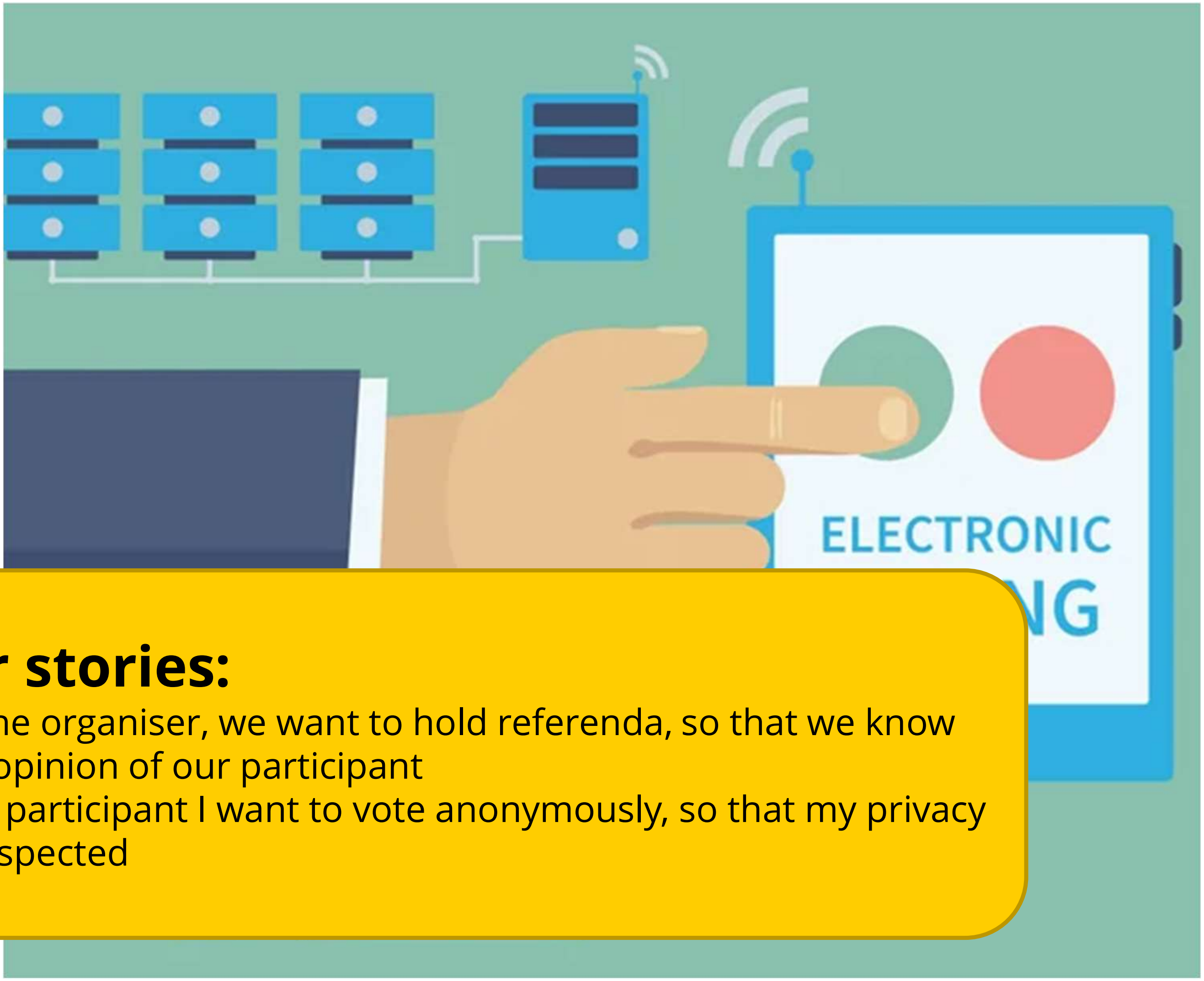
Build me an ‘eVoting system’!

What will you build?

How will you start?

Gather requirements?

System design?



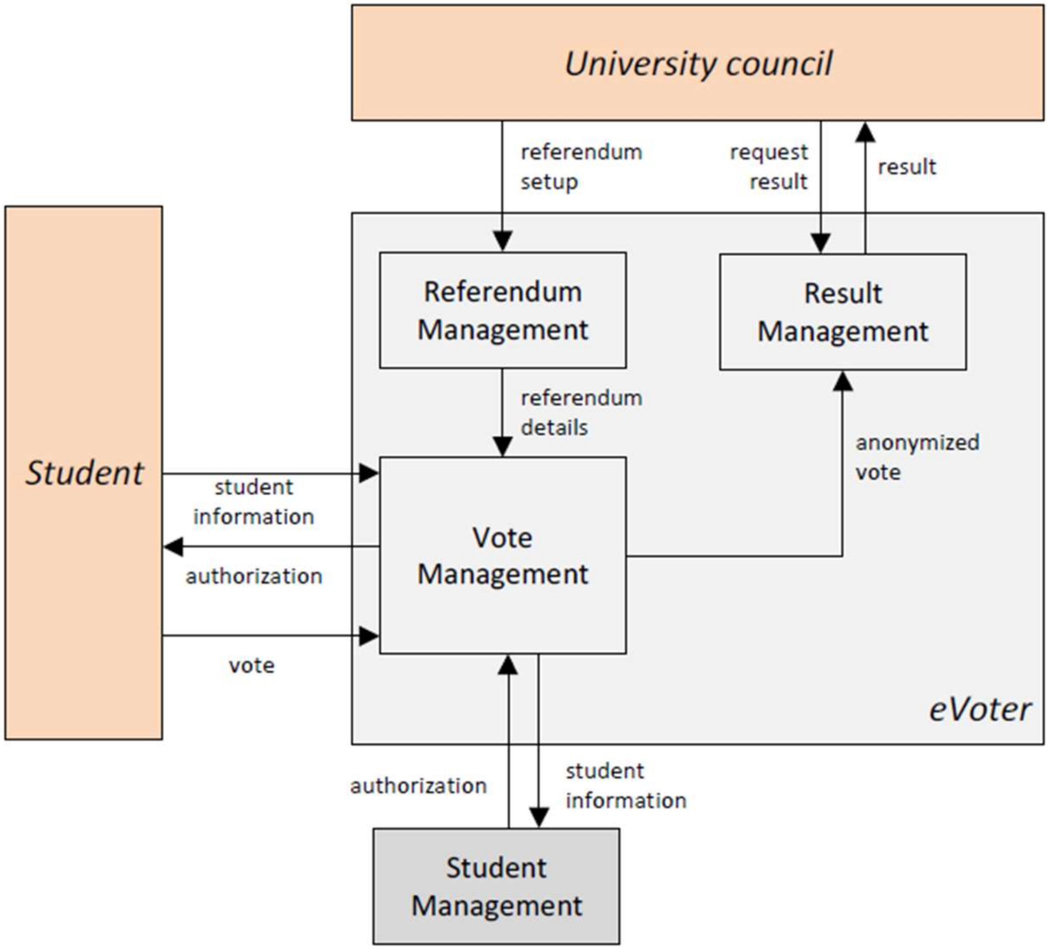
User stories:

- As the organiser, we want to hold referenda, so that we know the opinion of our participant
- As a participant I want to vote anonymously, so that my privacy is respected

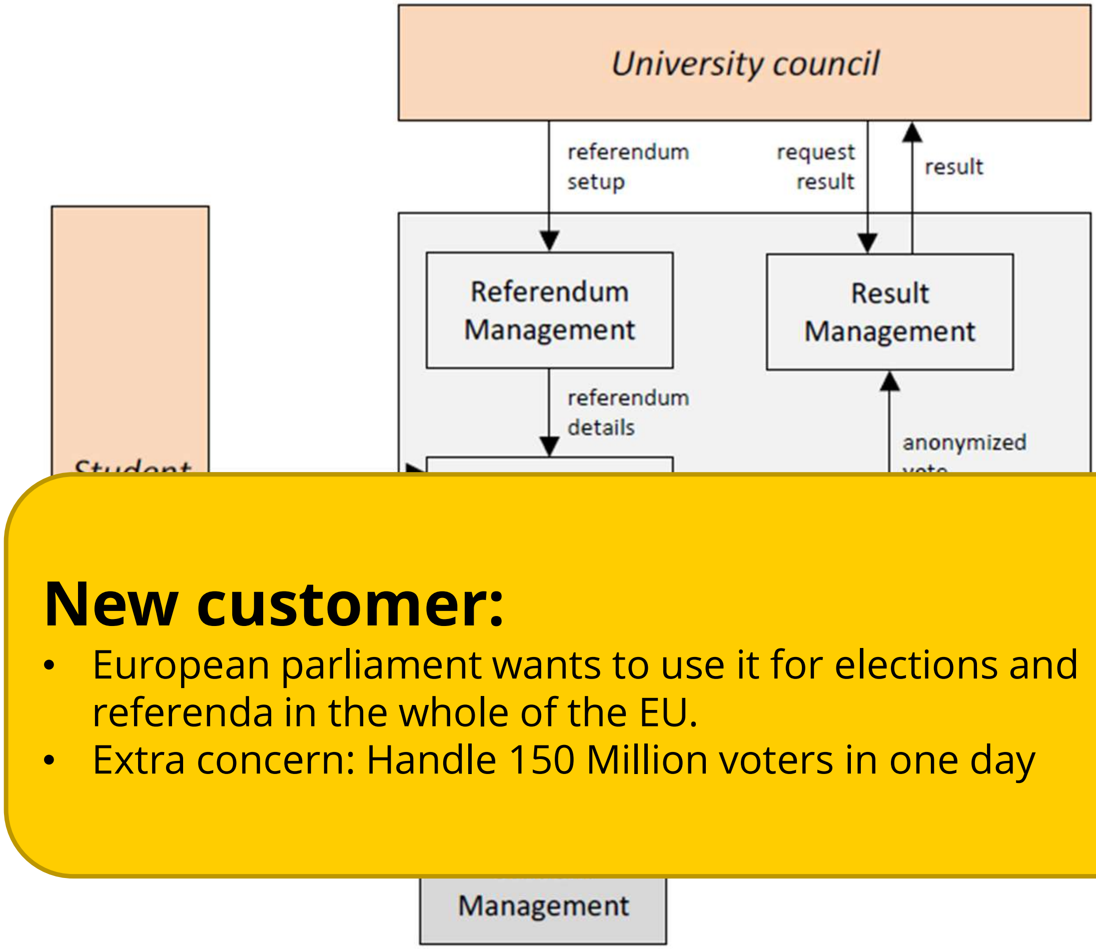
Credit: MIKKO LEMOLA Getty Images

Utrecht University

System design



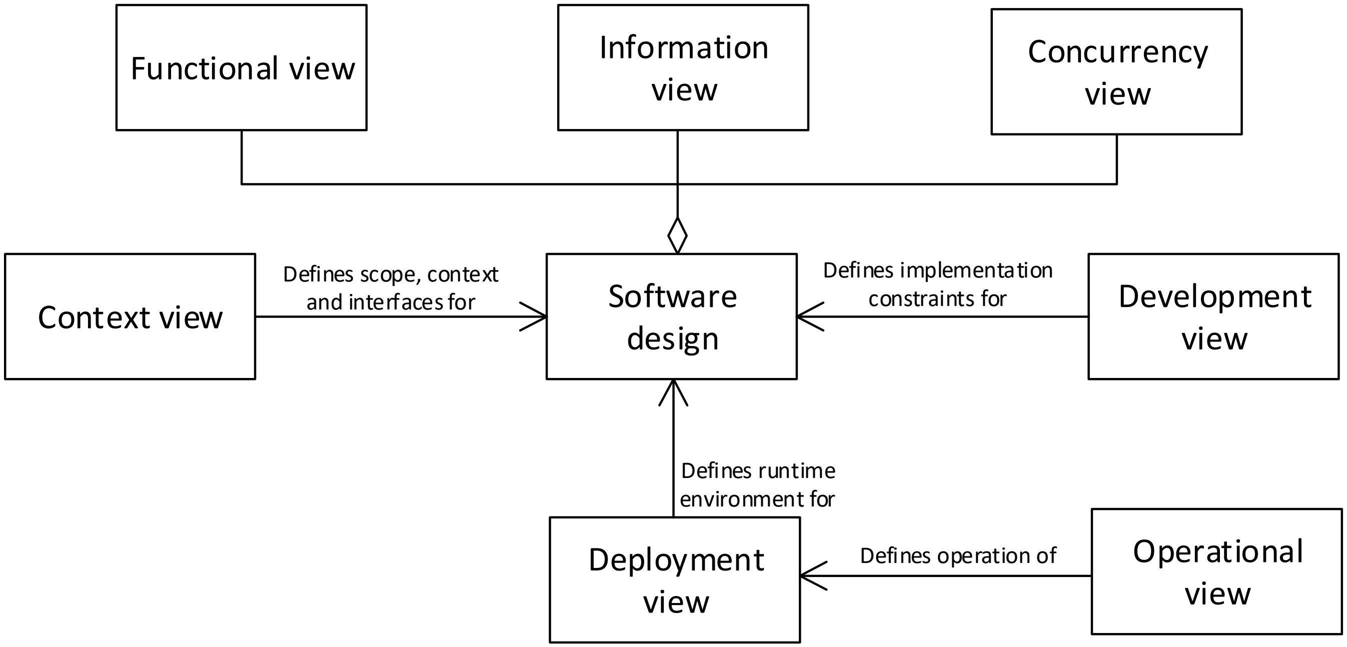
System design



New customer:

- European parliament wants to use it for elections and referenda in the whole of the EU.
- Extra concern: Handle 150 Million voters in one day

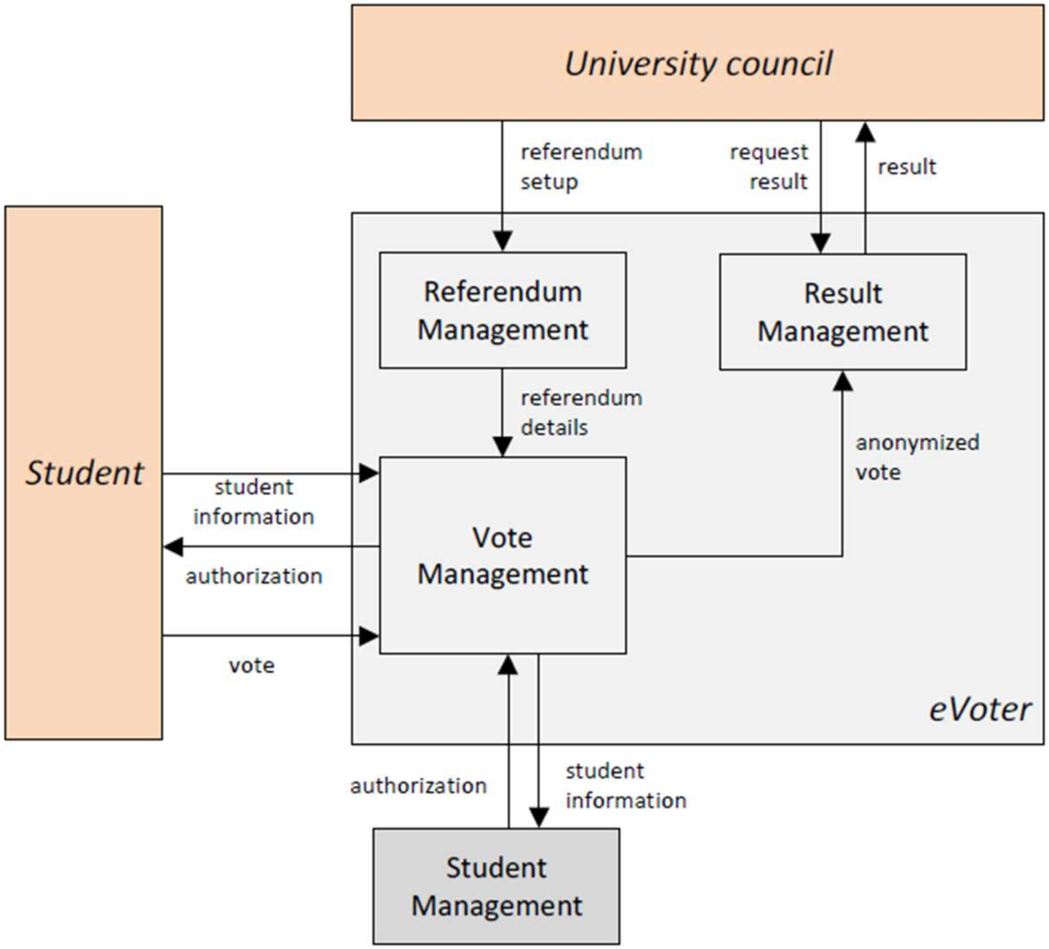
Viewpoint catalog



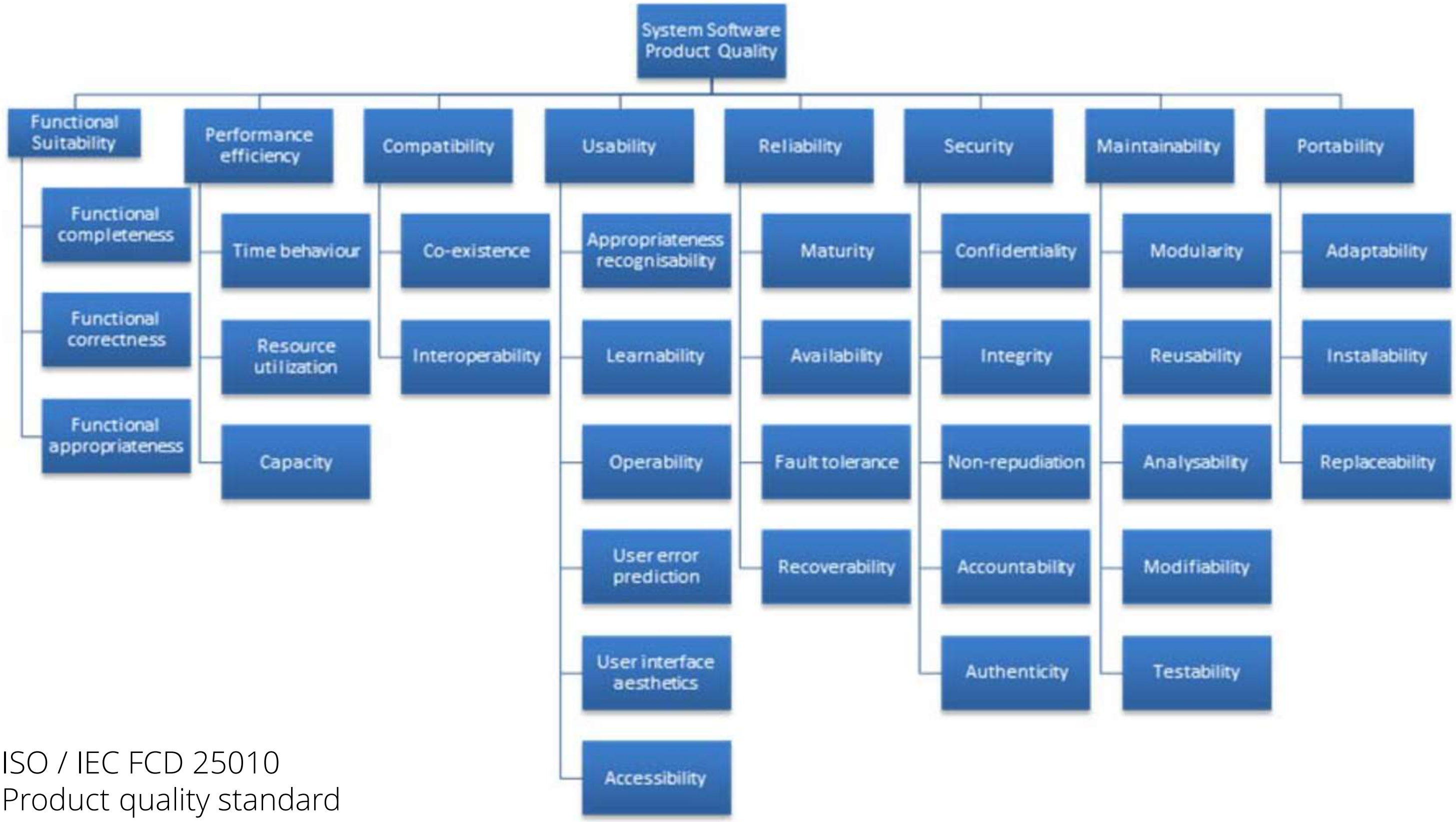
Where do we start?

Utrecht University

System design



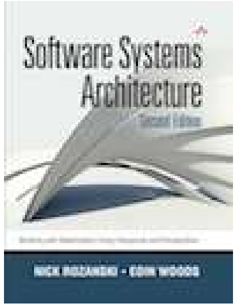
Many different qualities to look into…



- System Software Product Quality
  - Functional Suitability
    - Functional completeness
    - Functional correctness
    - Functional appropriateness
  - Performance efficiency
    - Time behaviour
    - Resource utilization
    - Capacity
  - Compatibility
    - Co-existence
    - Interoperability
  - Usability
    - Appropriateness recognisability
    - Learnability
    - Operability
    - User error prediction
    - User interface aesthetics
    - Accessibility
  - Reliability
    - Maturity
    - Availability
    - Fault tolerance
    - Recoverability
  - Security
    - Confidentiality
    - Integrity
    - Non-repudiation
    - Accountability
    - Authenticity
  - Maintainability
    - Modularity
    - Reusability
    - Analysability
    - Modifiability
    - Testability
  - Portability
    - Adaptability
    - Installability
    - Replaceability

ISO / IEC FCD 25010
Product quality standard

Viewpoints and Quality Attributes



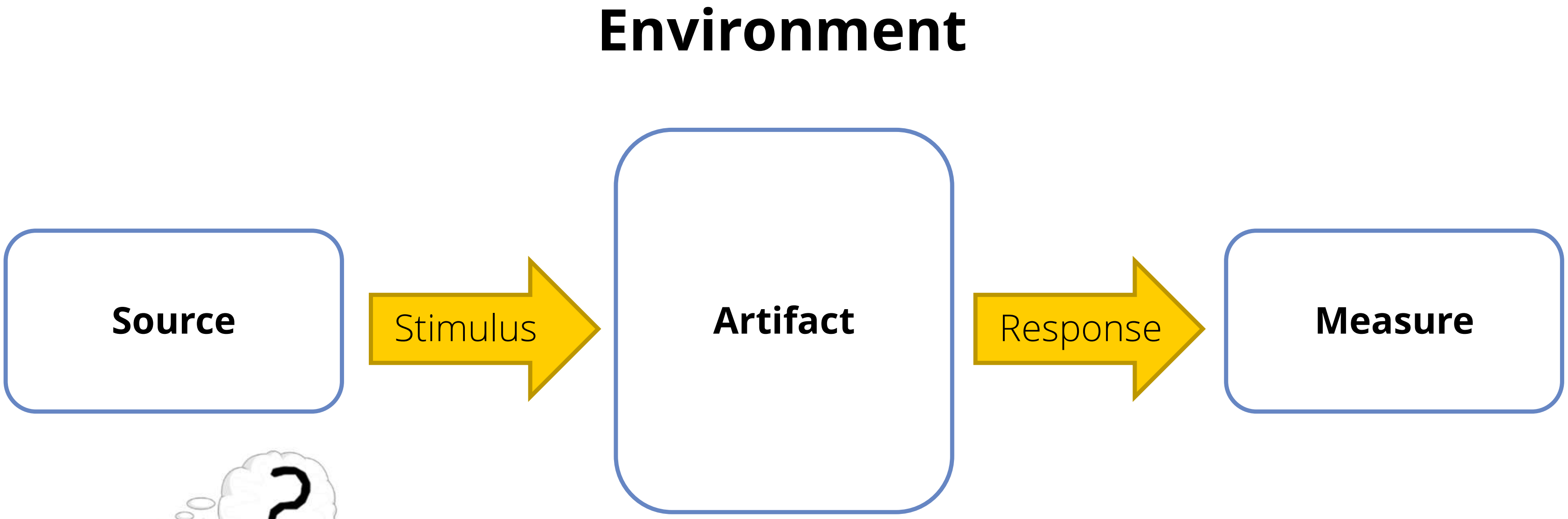
- Put the important QAs on the columns
- For each cell:

What is the importance of the QA on the view?
How influence the view and QA each other?

	... Security	... Availability	... Scalability	Reliability
Context	A	A, H,		
Functional	A BCD	A, H		
Information	E, F			
Concurrency	G .	G.		
Development				
Deployment				
Operational				

Utrecht University

Addressing concerns with quality scenarios



Environment

Source

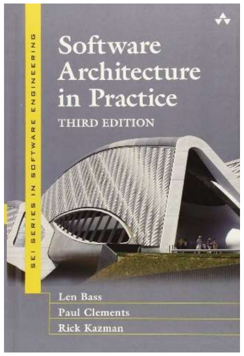
Stimulus

Artifact

Response

Measure

Is this a view?



SEI SERIES IN SOFTWARE ENGINEERING

Software Architecture in Practice

THIRD EDITION

Len Bass

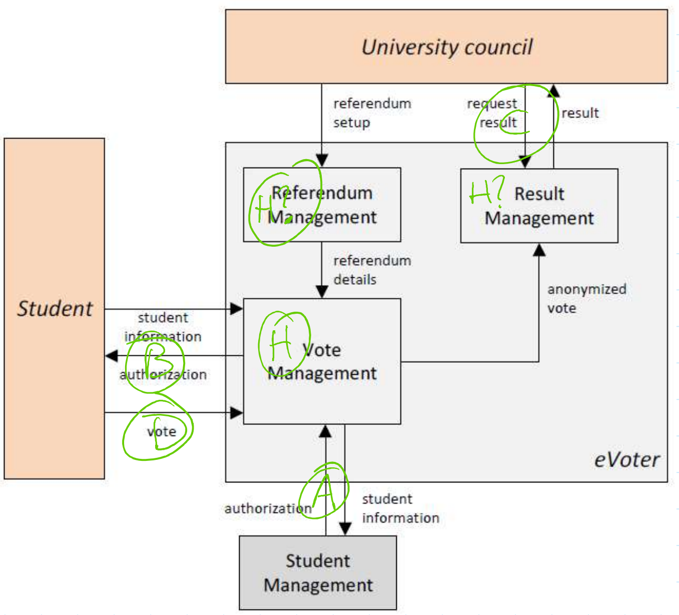
Paul Clements

Rick Kazman

maandag 11 maart 2024 09:28

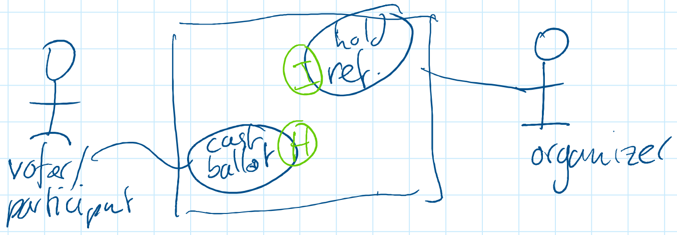
Functional Architecture.

Annotations: "points of contact" with the different perspectives.



Context

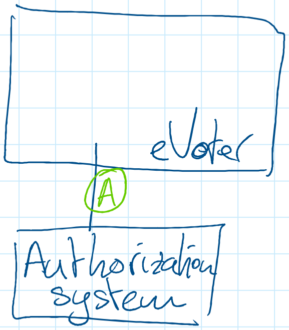
- Use case diagram.



Only two real use cases:

- Holding a referendum,
- Casting your ballot for a referendum.

- External Systems Diagram



Only external sytem: authorization system, like dutch DIGID

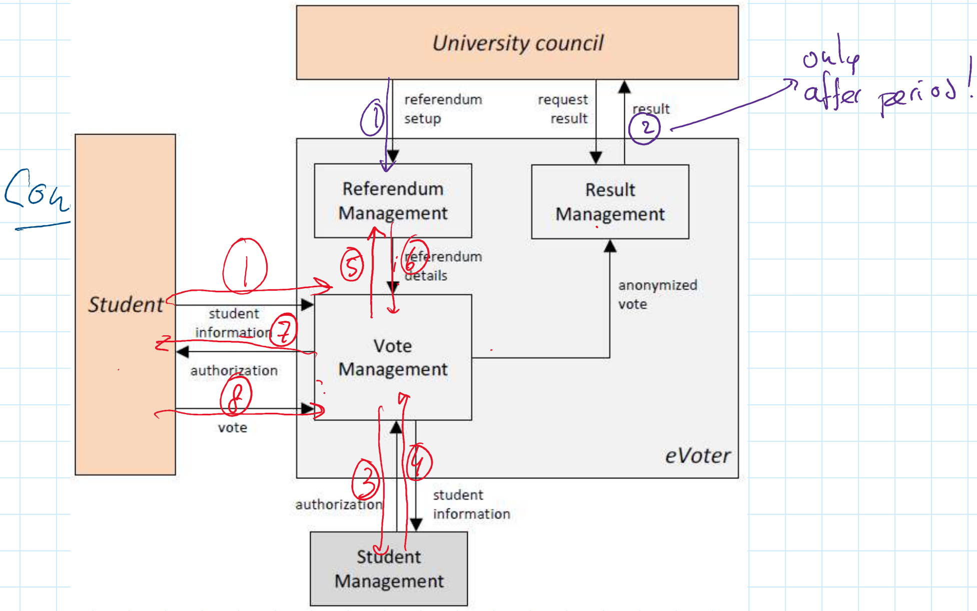
A: 150 M voters in 10h is 15M voters per hour is 4,16k voters per second.

Thus, we need an SLA with the Auth service for >4,16k connections per second.

Information views

Scenarios

As overlays on the functional Architecture



A: Casting a ballot:

A: Casting a ballot:

log in, if succeeded, send referendum details directly to user. User then casts vote.

Alternative path:

After (4) the log in failed, messaged to student.

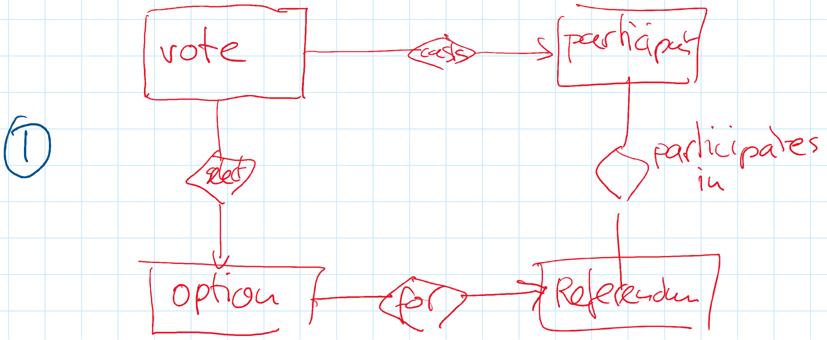
B Holding a referendum

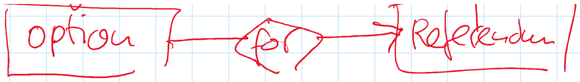
Result can only be retrieved After referendum is over!

Decision: The actual ballots are stored inside VoteManagement, to ensure security.

In the context of:	storing ballot information
facing	security
We decided to	store the ballot information inside VoteManagement
And not	store the ballot information inside ResultManagement
To achieve	reliable referenda, where it is impossible to view results while the referenda is running
Accepting	The overhead of calculating the results after the referendum is over

Data view

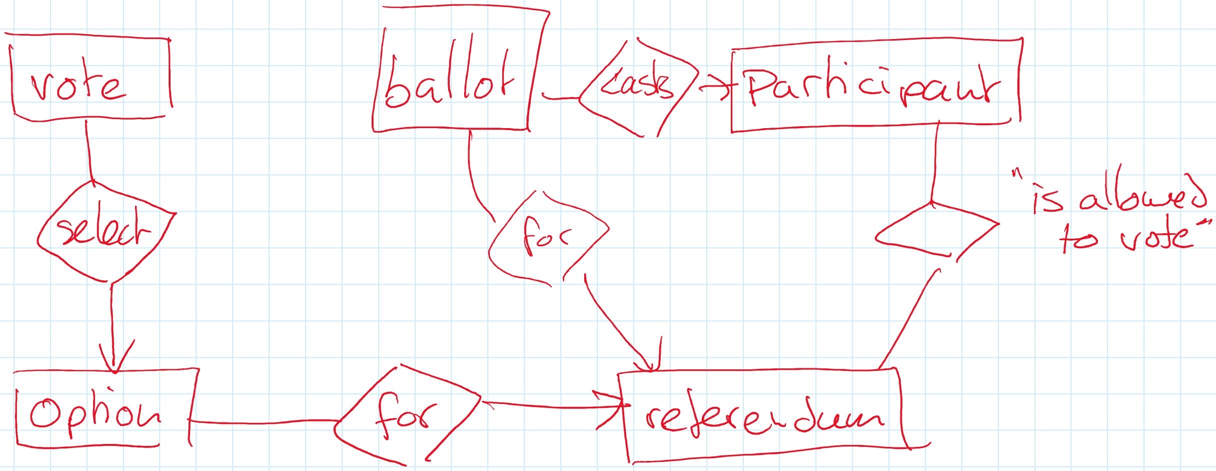




Constraint: Each participant can vote
at most once for a referendum.

However this model is potentially unsafe, as relation "casts" stores privacy-sensitive data.

②



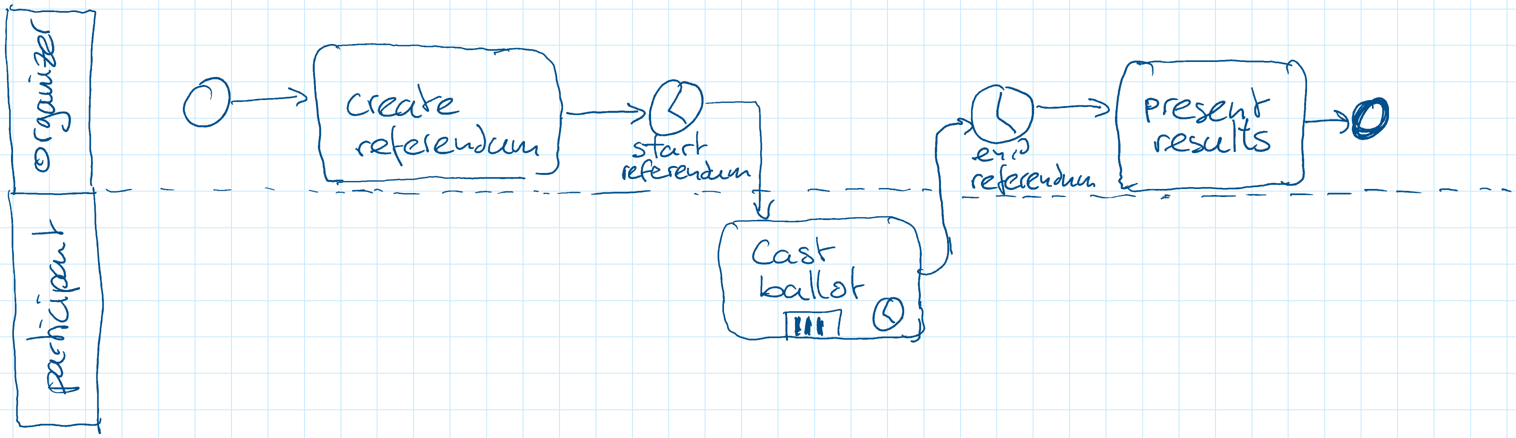
Now, we know when somebody voted ("ballot"), but we do not have a connection with the actual value.

Decision: more complex data model for privacy.

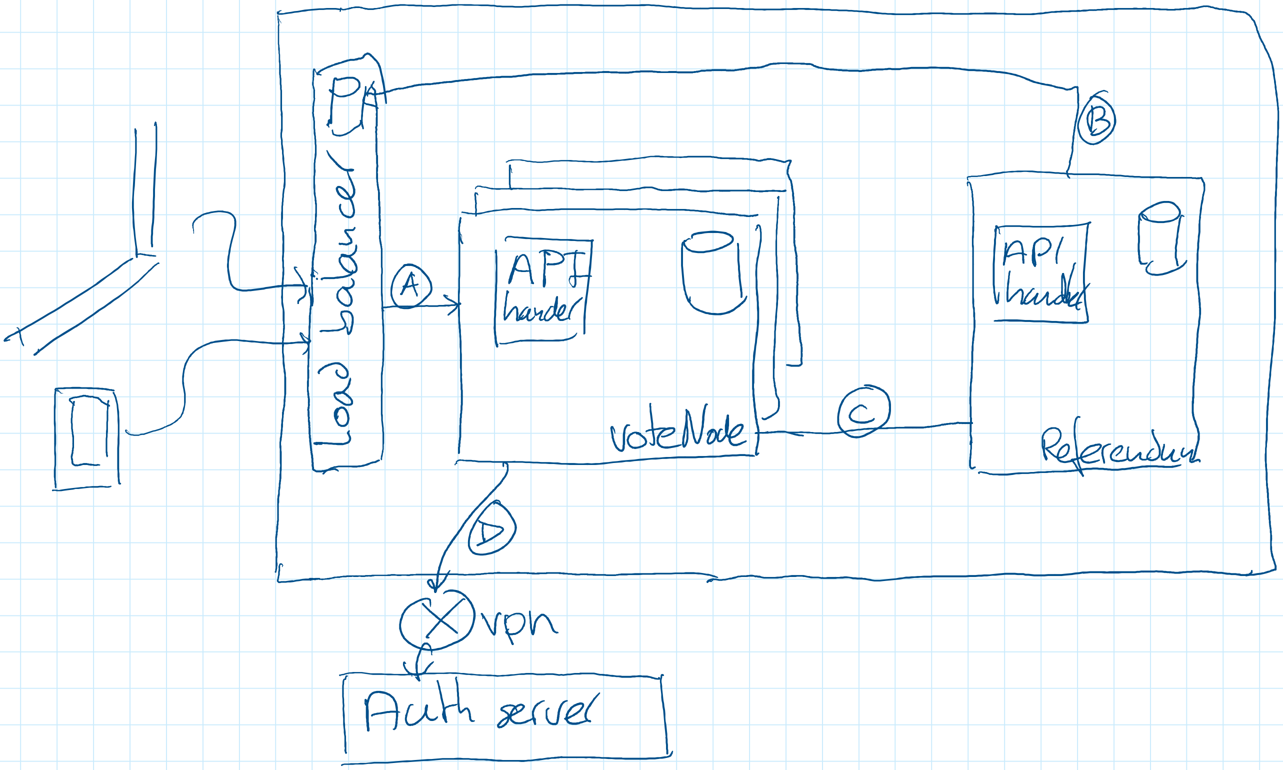
In the context of	storing data
facing	Privacy
We decided to	Use datamodel ①
And not	datamodel ②
To achieve	Anonymity of the voting results
Accepting	the complexity involved to ensure that the data remains consistent (e.g. that #ballot equals # votes.

Note that another consequence of this decision is restoring faults: what if
#ballots ≠ # votes

Process view.



Deployment view.



(A) Loadbalancer decides which node a participant is directed to. This node takes the complete Vote-process of that participant.

(B) Referendum populates the load balancer to be able to distribute participants over VoteNodes

(C) Referendum sends a notification which referendum is open

Ⓒ Referendum sends a notification which referendum is open

Only retrieves data after the deadline of the referendum

Ⓓ Unknown protocol, to investigate

Implicit decision

In the context of	the eVoter application
Facing	functionality
We decided to	Go for a distributed (virtual) server architecture
And not	- "Paper supporting technology - Blockchain - ...
Accepting	that the system cannot be used for elections, as the system does not have any means for trust and reliability (i.e., it does not allow recounts).