Lecture 4: Quality Attributes II

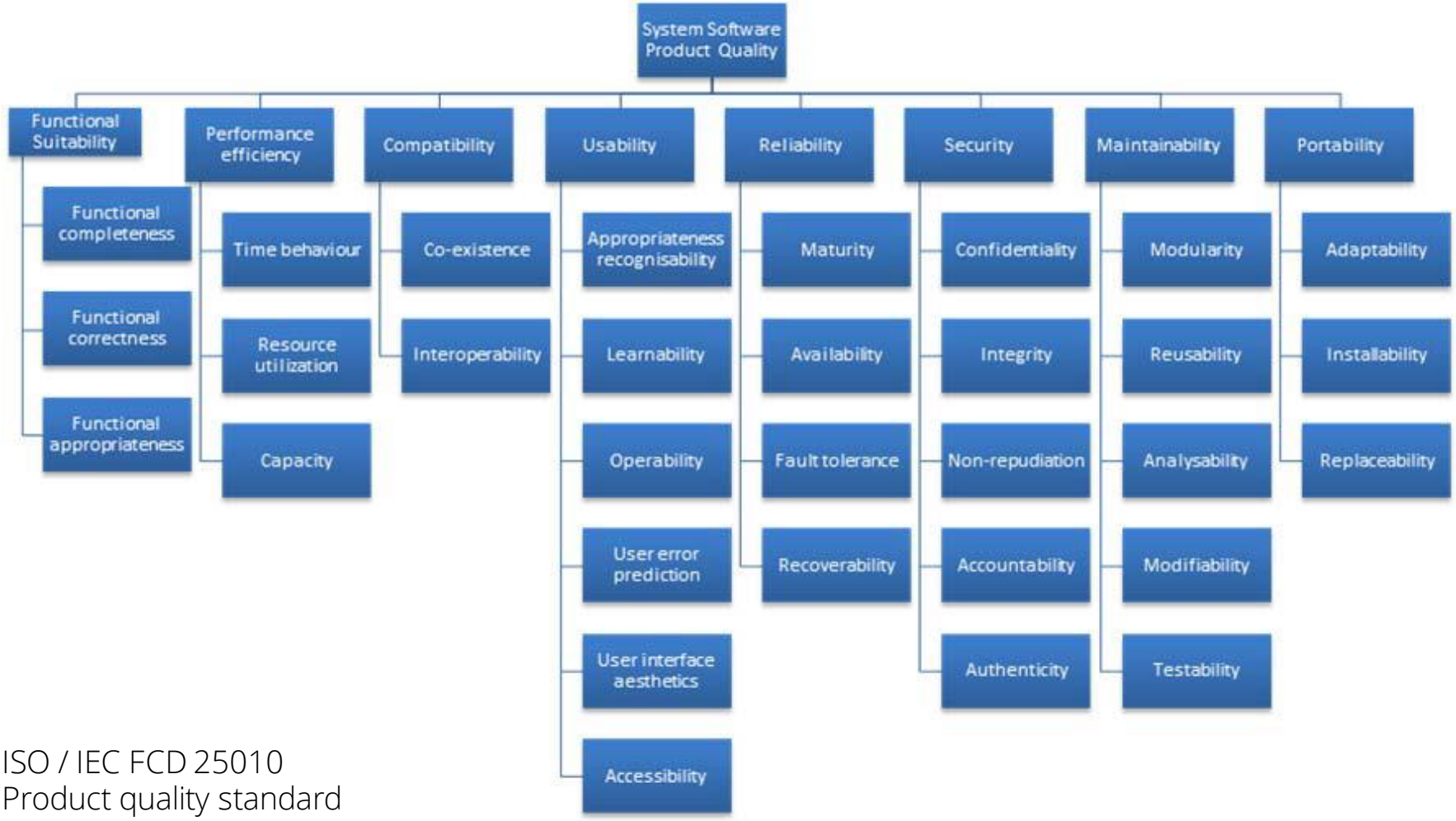
Utrecht University

For today

- 13:15 – 13:45: Quality attributes & tactics
- 13:45 – 14:45: Defining QAs & scenarios in your assignment
- 15:00 – 15:45: Analysis of Quality attributes
- 15:45 – 16:50: Work on your assignment
- 16:50 – 17:00: Wrap up

Utrecht University

Many different qualities to look into...



ISO / IEC FCD 25010
Product quality standard

Utrecht University

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?

	Availability	Performance	…
Context			
Functional			
Information			
Concurrency			
Development			
Deployment			
Operational			

Utrecht University

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?

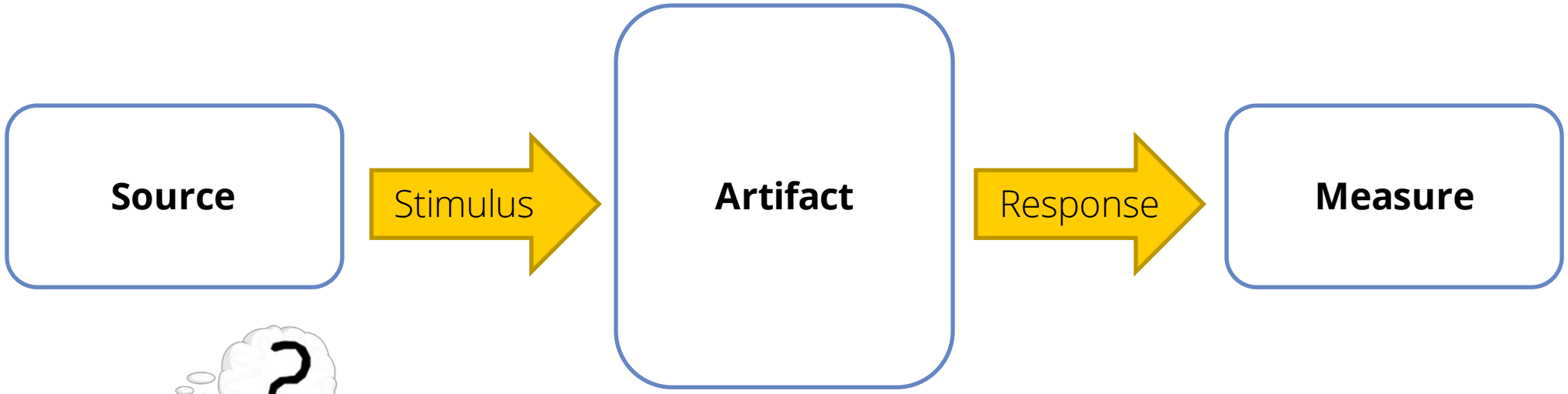
	Availability	Performance	…
Context			
Functional			
Information			
Concurrency			
Development			
Deployment			
Operational			

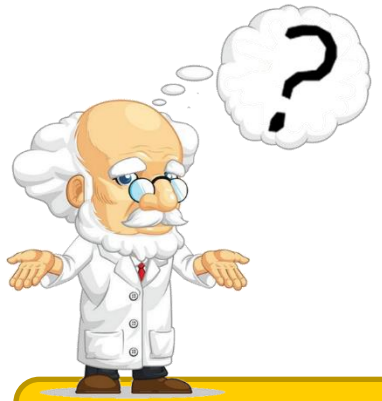
How to analyse this systematically?

Utrecht University

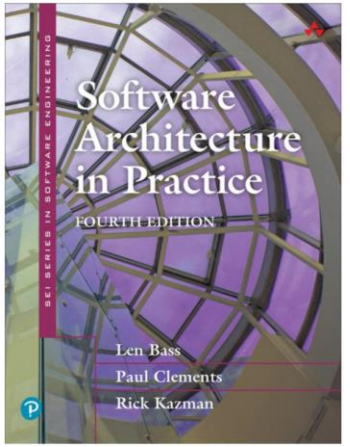
Addressing concerns with quality scenarios

Environment





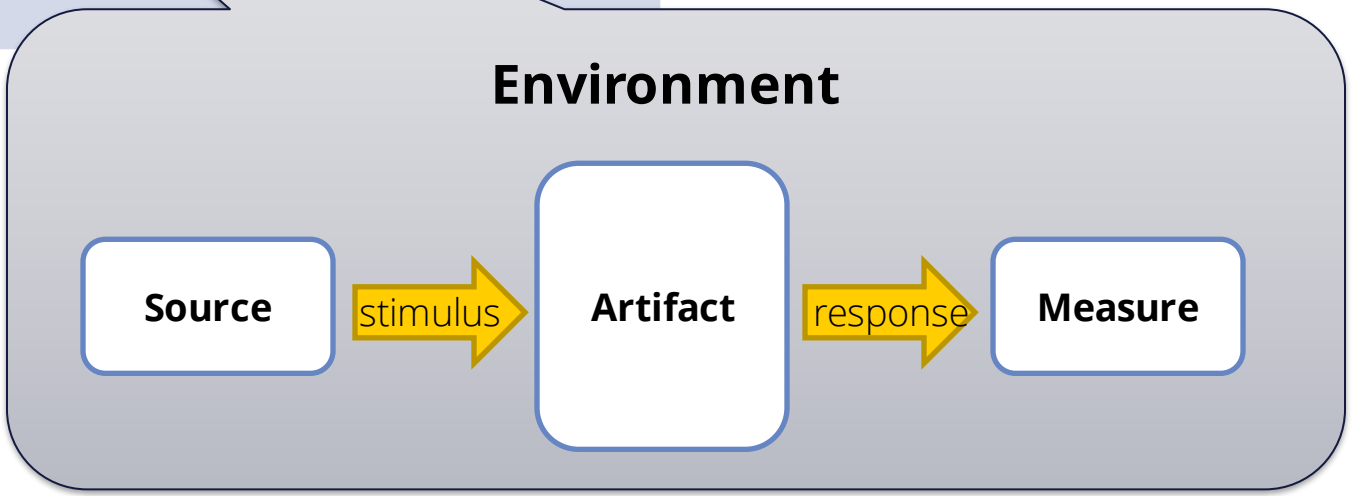
Is this a view?



Utrecht University

Viewpoints and Quality Attributes

	Availability	Performance	…
Context			
Functional			
Information			
Concurrency			
Development			
Deployment			
Operational			



Utrecht University

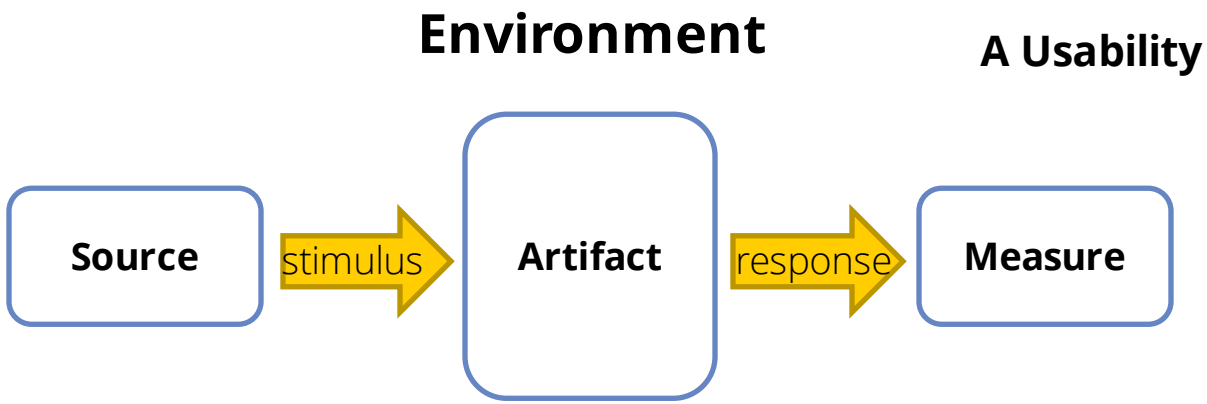
An example!

- When the user presses the green button, the Options dialog should appear

A performance QA: the dialog appears within 500ms

An availability QA: It may only fail in 1 out of 1000 times.

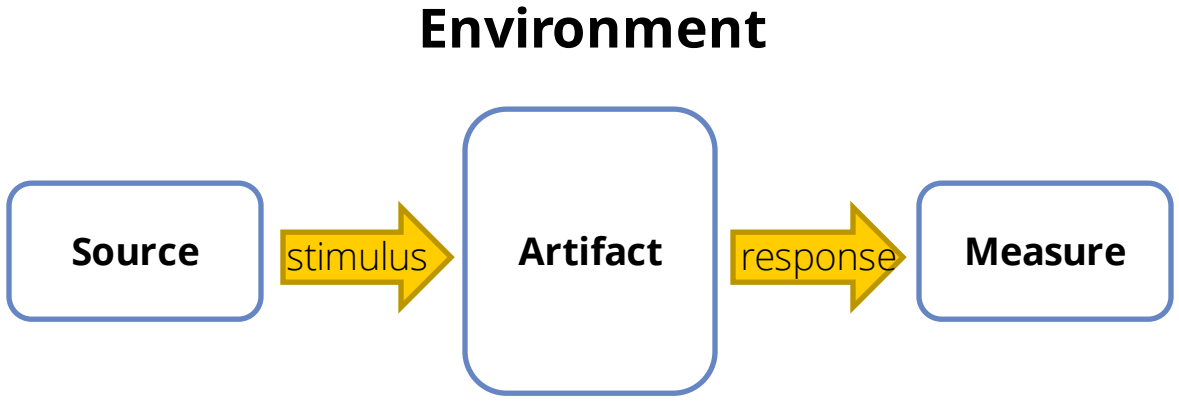
A Usability QA: The green button should be easy to be found



Utrecht University

Addressing quality properties

- Create general scenario as a context view:



- Address in each view how the QA is addressed
  - **Overlays on existing views**
  - **New views**

Tactics!

- Quality attributes can **and will** change views!!

Utrecht University

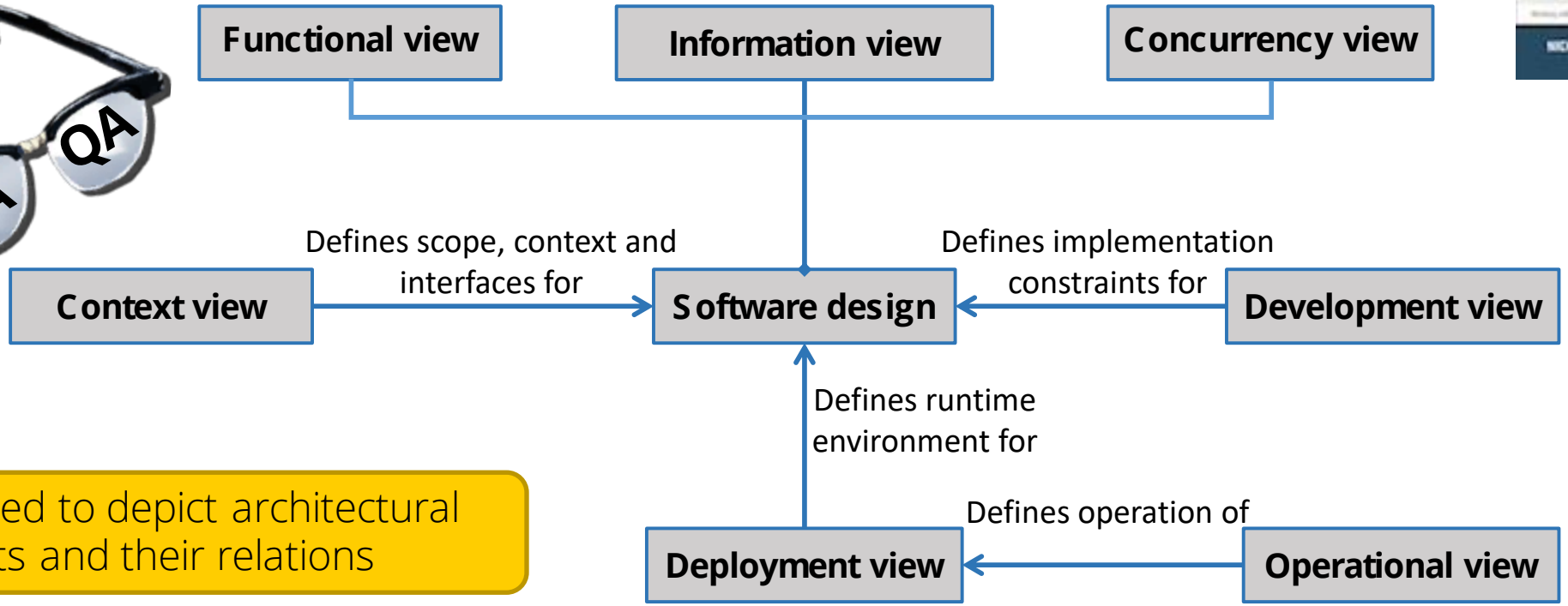
Addressing quality attributes

Utrecht University

E. Woods and Rozanski paper (2005): What is the main message?

Utrecht University

Quality Attributes: Viewpoints and perspectives



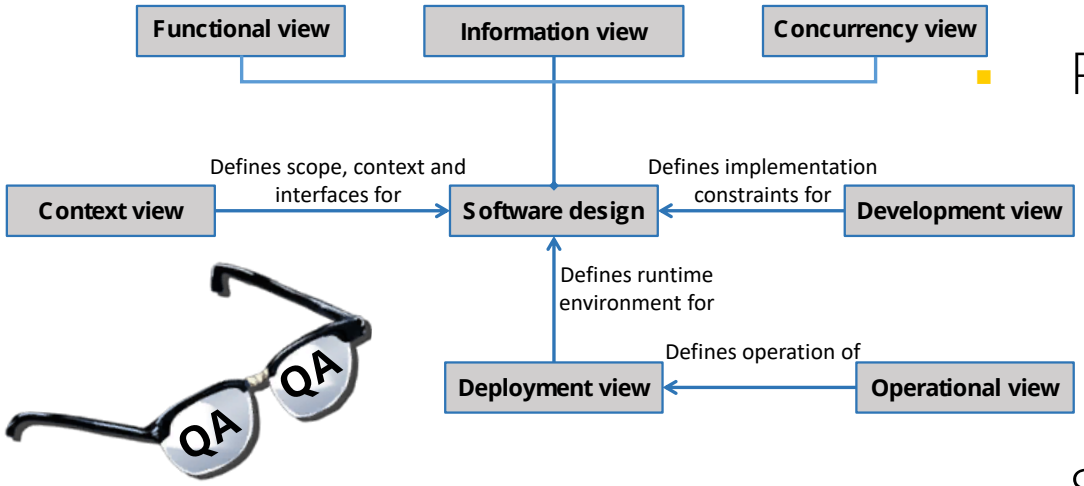
Views are used to depict architectural elements and their relations

QAs require analysis and evaluation!

QAs determine how you look at the views!

Utrecht University

Quality Attributes: Viewpoints and perspectives



- Perspective:

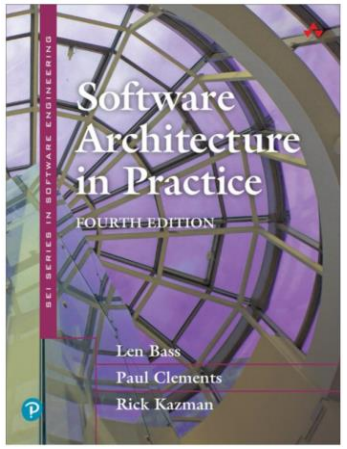
  a collection of activities, checklists, tactics and guidelines to guide the process of ensuring that a system exhibits a particular set of closely related quality properties that require consideration across a number of the system's architectural views.

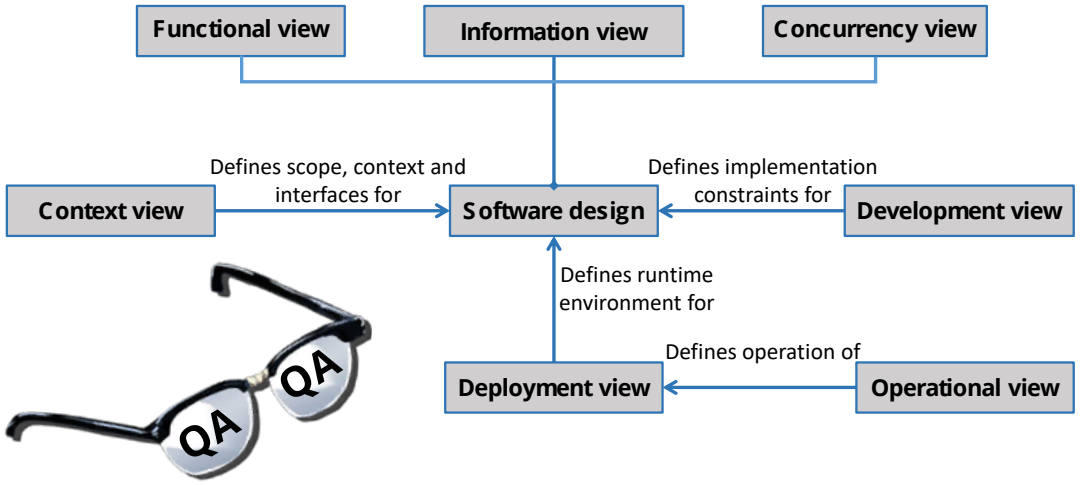
- Systematic approach:

  1. **Which concerns are related to the QA?**
  2. **Which views are applicable to the QA?**
  3. **Which activities do you need to satisfy the QA?**
  4. **Which tactics can you apply to satisfy the QA?**
  5. **What are problems & pitfalls?**
  6. **Checklist: did I forget anything?**

Utrecht University

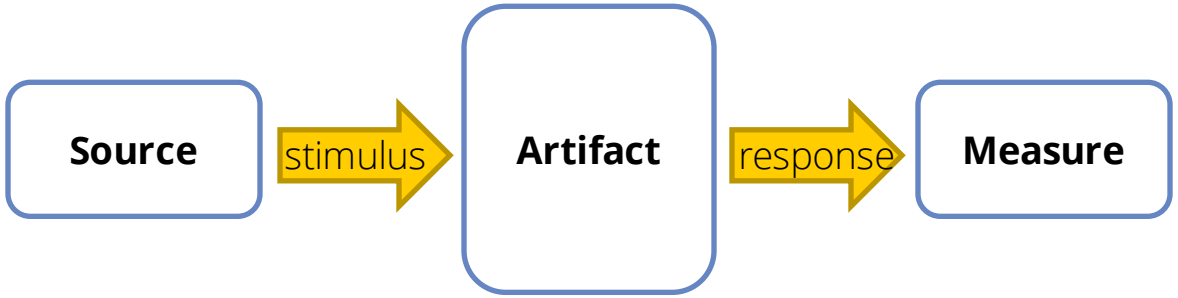
Quality attributes in SAiP





- Ch 4: Availability
- Ch 5: Deployability
- Ch 6: Energy Efficiency
- Ch 7: Integrability
- Ch 8: Modifiability
- Ch 9: Performance
- Ch 10: Safety
- Ch 11: Security
- Ch 10: Testability
- Ch 11: Usability

Environment

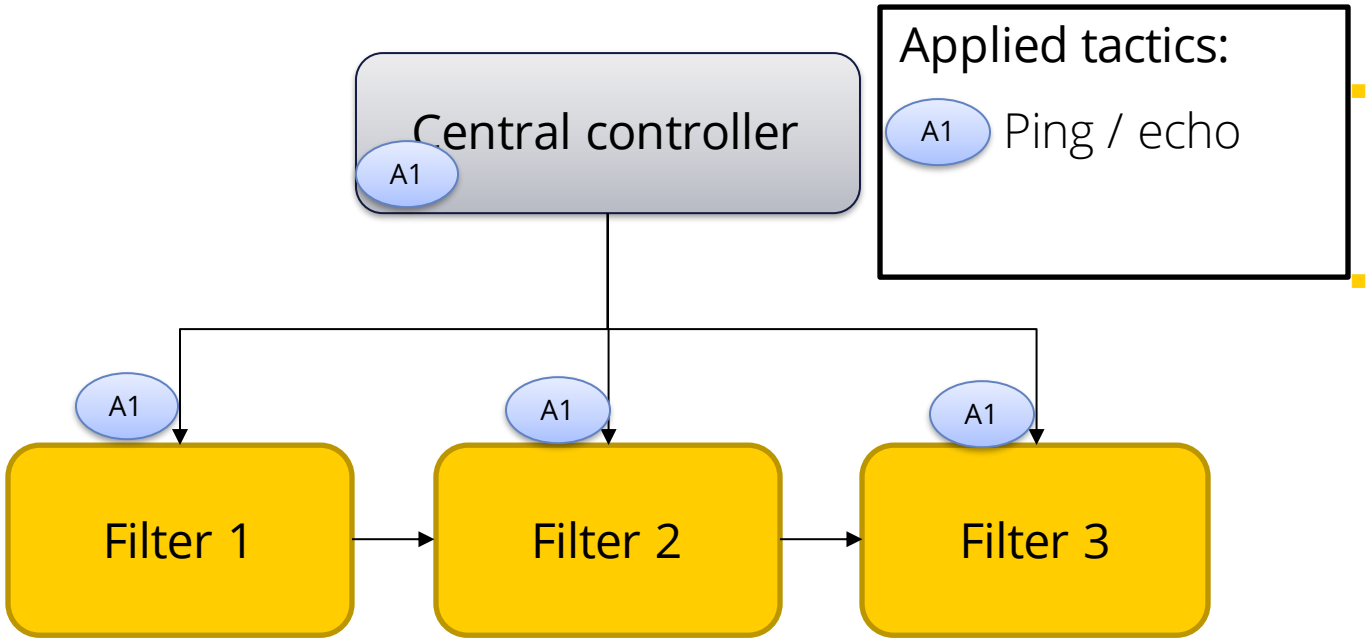


For each quality attribute:

- **A list of example quality scenarios**
- **A list of tactics that can be applied**

Utrecht University

Tactics to satisfy a quality attribute



Problem:

Detect availability subsystems

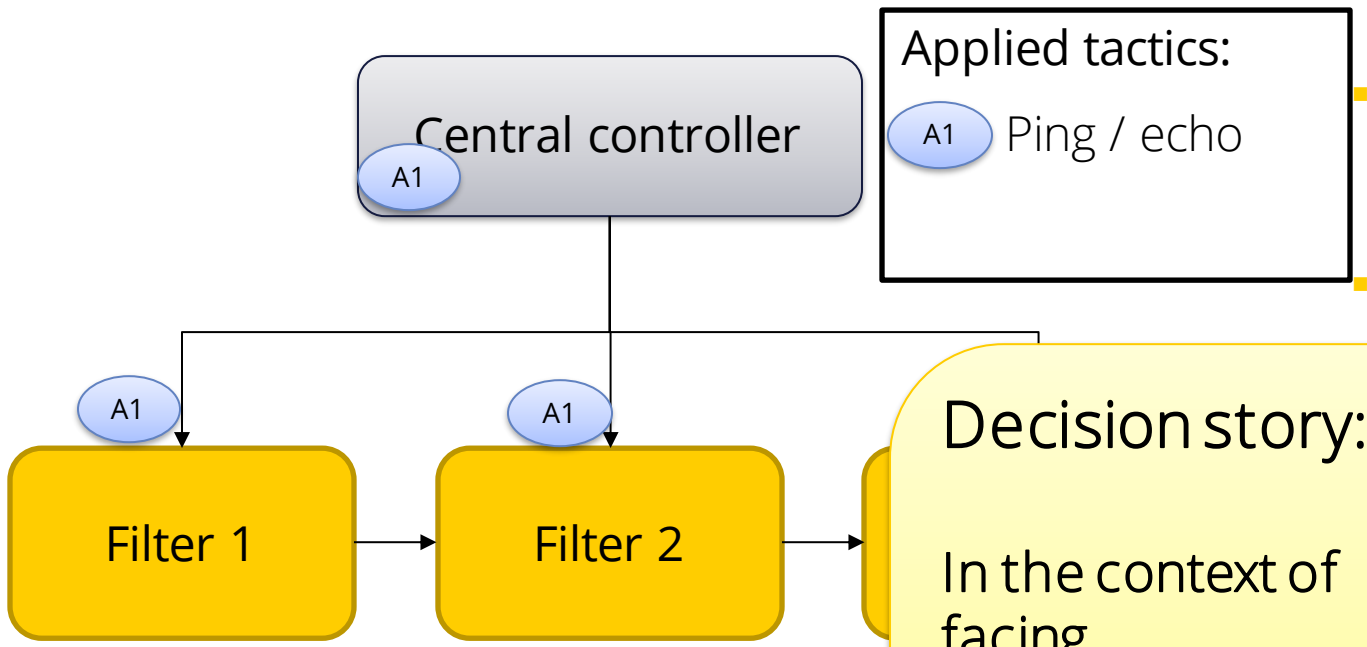
Available tactics:

- **Ping / echo**
- **Monitor**
- **Timestamp**
- **Sanity checking**
- **Condition monitoring**
- **Voting**
- **Exception detecting**
- **Self test**

N. Harrison, P. Avgeriou (2010). How do architecture patterns and tactics interact? A model and annotation.
In: Journal of Systems and Software 83(10), pp 1735-1758

Utrecht University

Tactics to satisfy a quality attribute



Decision story: Detect availability subsystems

In the context of	filter-based processing,
facing	availability,
We considered	ping / echo
	self tests
And decided for	ping / echo
To achieve	fast fault recovery
Accepting	the addition of a new sub system

N. Harrison, P. Avgeriou (2010). How do architecture p
In: Journal of Systems and Software 83(10), pp 1735-1758

Utrecht University

For now:

- Continue : Which views are required to assess?
  Define a table that assesses severity for each QA
- What are quality scenarios in your assignment?
  Define for each cell a set of Quality scenarios

	Availability	Performance	…
Context			
Functional			
Information			
Concurrency			
Development			
Deployment			
Operational			

Utrecht University

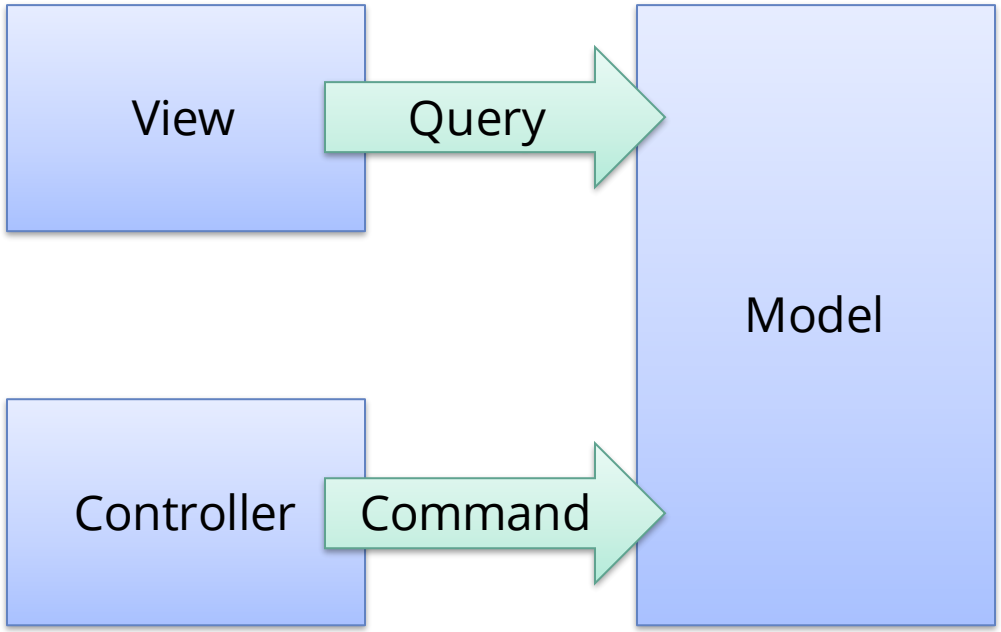
For today

- 13:15 – 13:45: Quality attributes & tactics
- 13:45 – 14:45: Defining QAs & scenarios in your assignment
- 15:00 – 15:45: Analysis of Quality attributes
- 15:45 – 16:50: Work on your assignment
- 16:50 – 17:00: Wrap up

Analysis of quality attributes

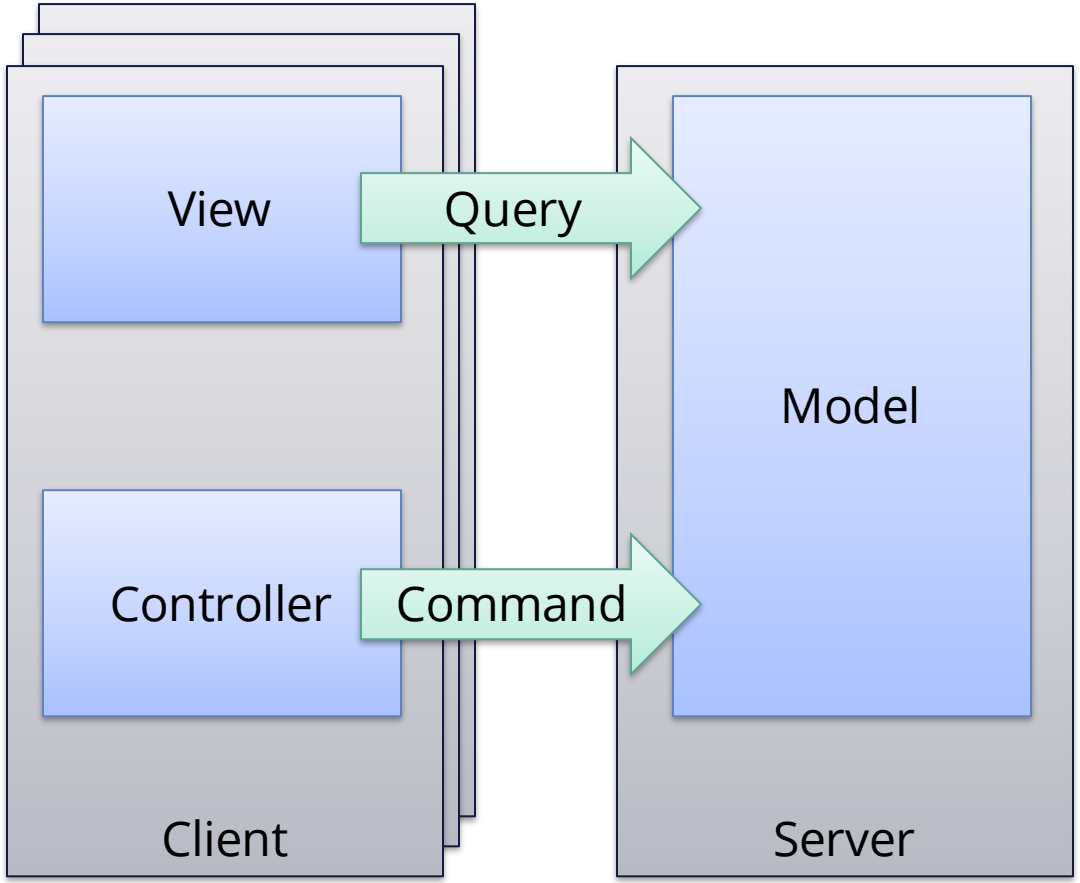
Utrecht University

Let's have a look at Model-View-Controller



Utrecht University

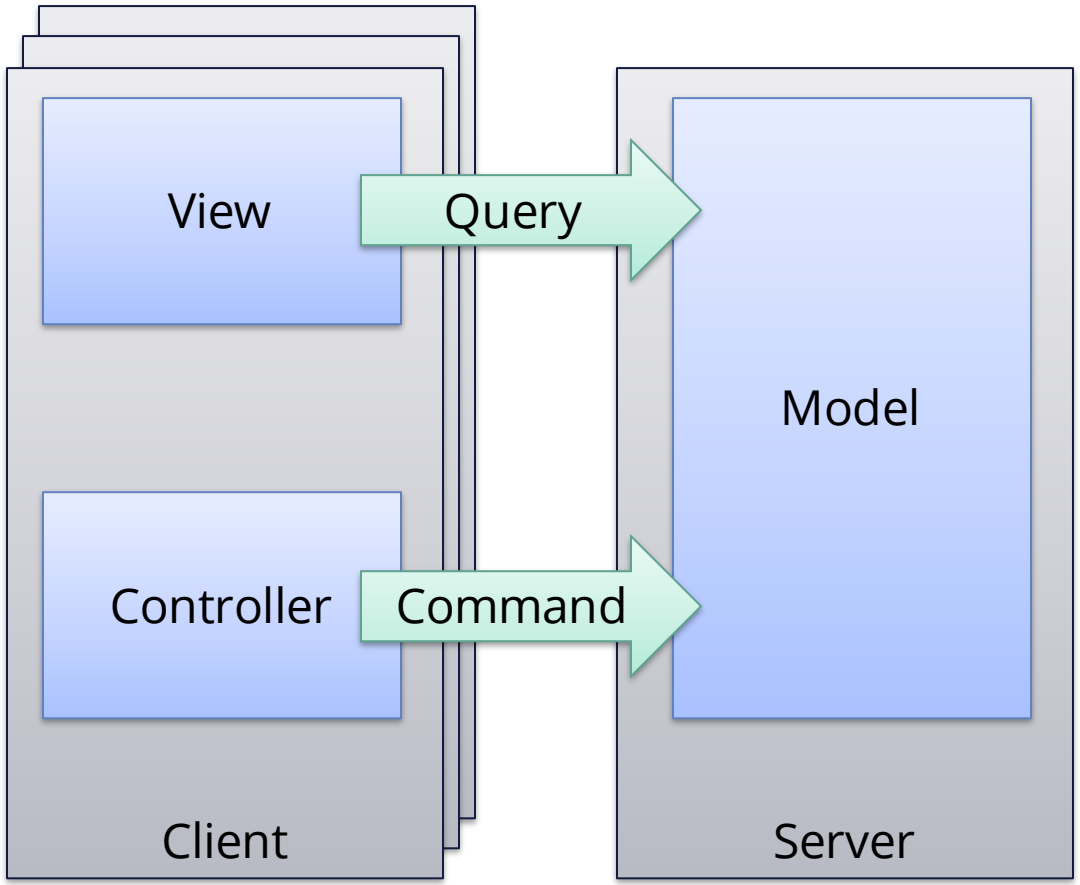
Let's have a look at Model-View-Controller



What if there are 10.000 clients running concurrently?

Utrecht University

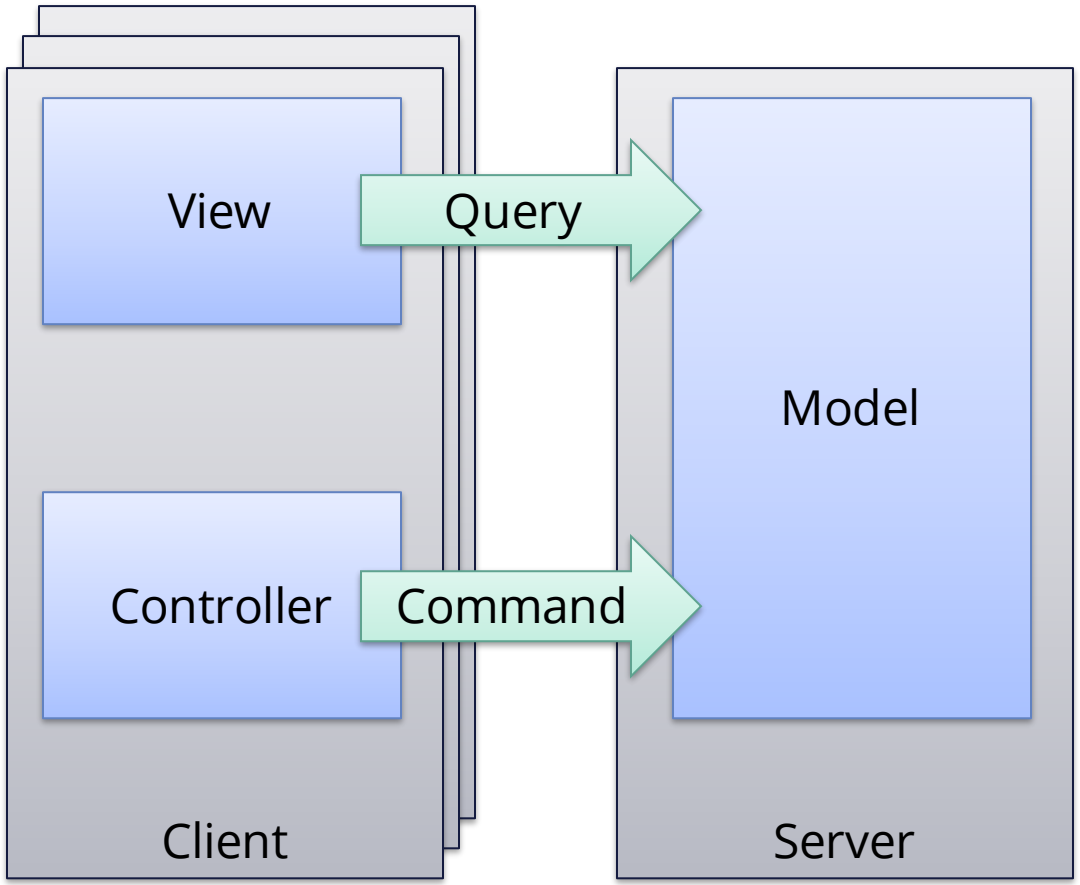
Modeling for quality property analysis



- Some quality properties have **well-understood, time-tested analytic** models
  - **Performance**
  - **Availability**
  - **…**
- **Analytical model**: supports quantitative analysis
- **Parameters**: variables to configure a model

Utrecht University

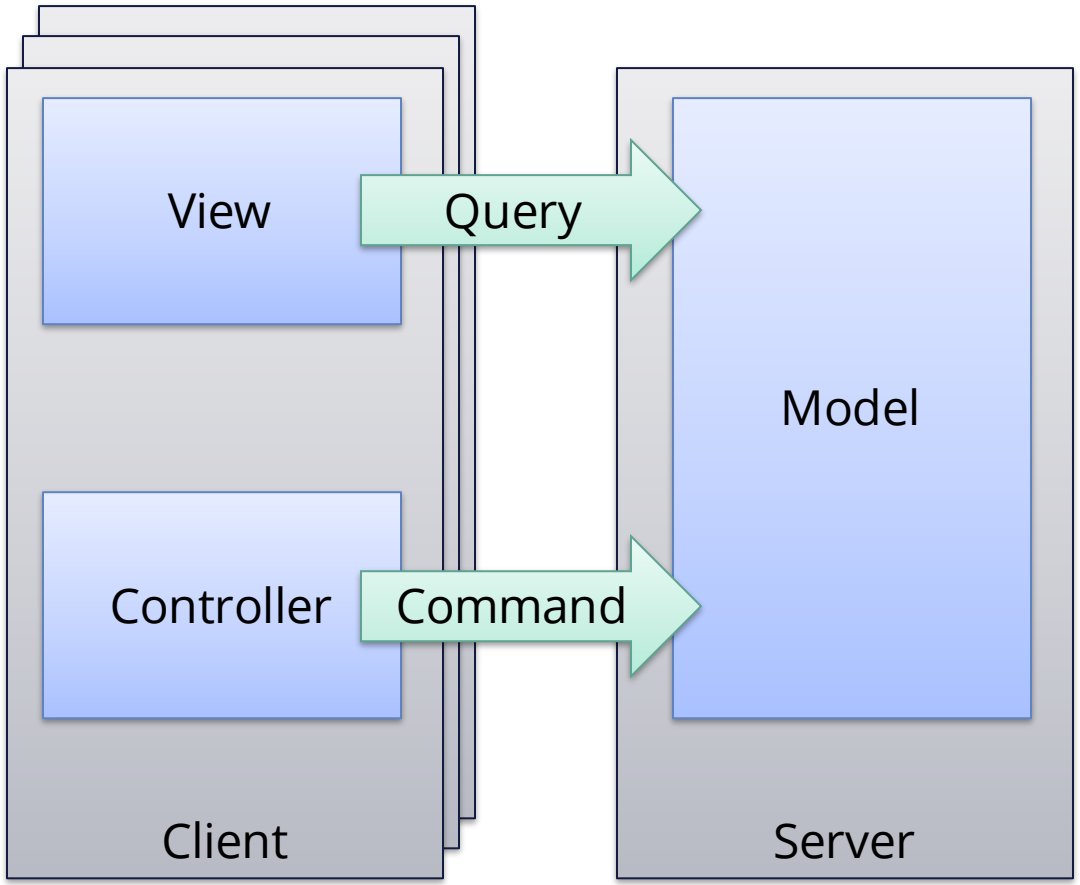
Architecture Analysis & Evaluation



- Analysis
  - **Formal analysis techniques to ensure "correctness"**
  - **Thought experiments**
  - **Back-of-the-envelope analysis**
  - **Experiments, simulations and prototypes**
- Evaluation
  - **Formal review and structured walkthrough**
  - **(Functional) scenarios**
  - **Presentation**
  - **Review sessions**

Utrecht University

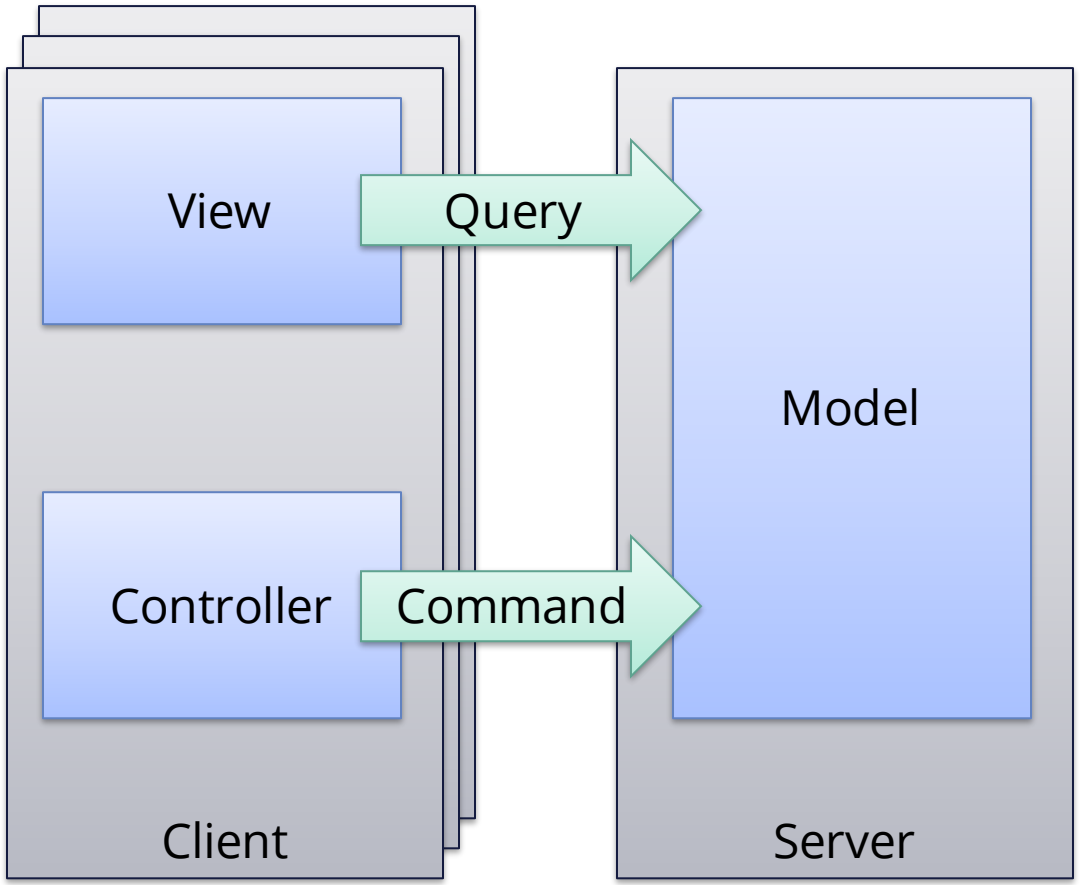
For all techniques



- Critical assessment
  - **NOT: making severe or negative judgments**
  - **BUT: making careful and analytical evaluations to come to a skillful judgment**
- What are the trade-offs?
  - **All alternatives covered?**
  - **Is this the best candidate?**
- What are the risks?
  - **All risks identified and covered?**
- What are the constraints and assumptions?
  - **Are these valid?**

Utrecht University

For all techniques



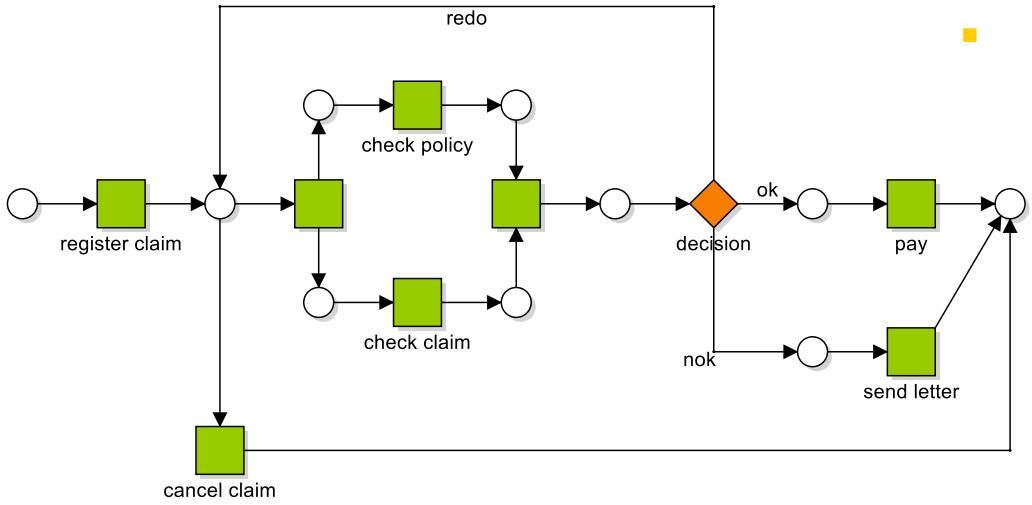
- Critical assessment
  - **NOT: making severe or negative judgments**
  - **BUT: making careful and analytical evaluations to come to a skillful judgment**
- What are the trade-offs?
  - **All alternatives covered?**
  - **Is this the best candidate?**
- What are the risks?
  - **All risks identified and covered?**
- What are the constraints and assumptions?
  - **Are these valid?**

Utrecht University

Architecture analysis

Utrecht University

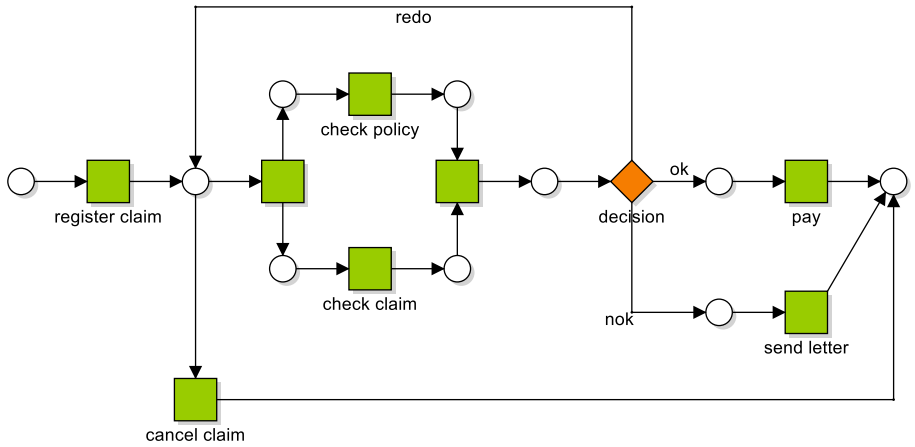
Performance model: Petri nets



- Parameters:
  - **Delays**
  - **Serving times**
  - **Decisions paths**
  - **Resource availability**
  - **Channel behaviour**
  - **…**

Utrecht University

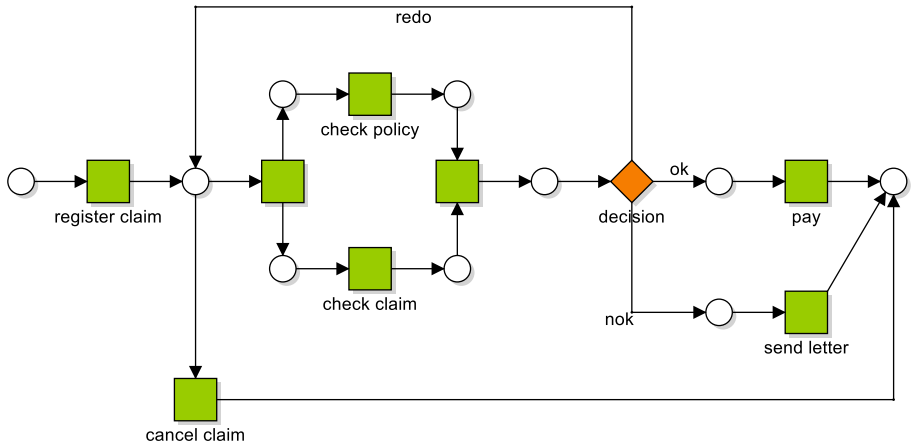
Performance model: Petri nets

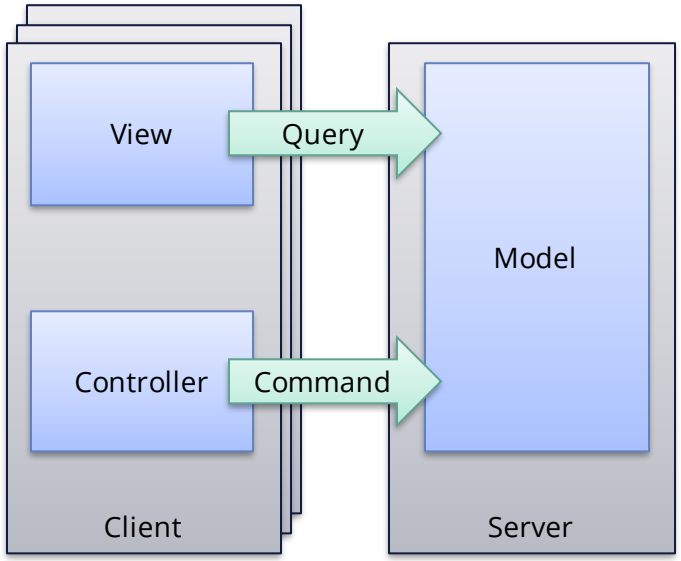


- Parameters:
  - **Delays**
  - **Serving times**
  - **Decisions paths**
  - **Resource availability**
  - **Channel behaviour**
  - **…**
- Models:
  - **Time Petri nets (time on tokens)**
  - **Timed Petri nets (time on transition duration)**
  - **Stochastic nets**
  - **Queueing nets (FIFO behaviour on places)**
  - **…**

Utrecht University

Performance model: Petri nets

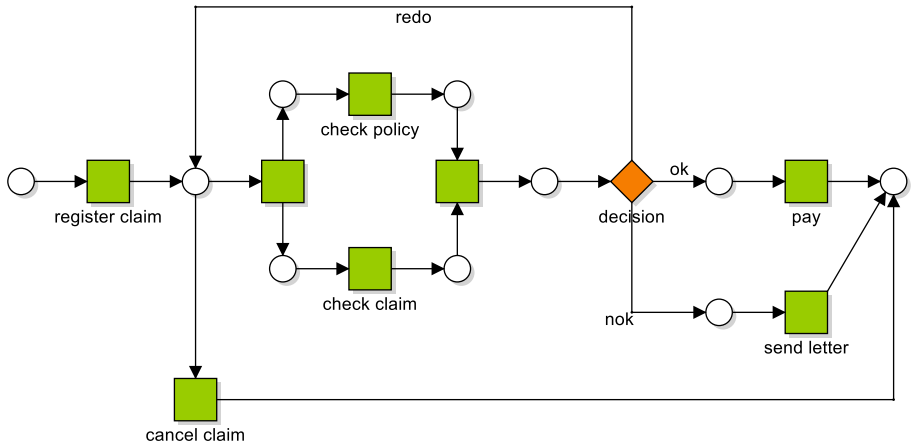


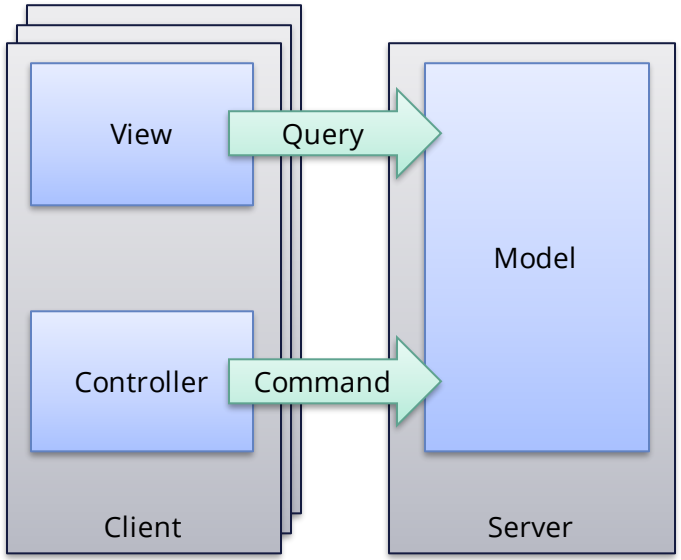


- Parameters:
  - **Delays**
  - **Serving times**
  - **Decisions paths**
  - **Resource availability**
  - **Channel behaviour**
  - **…**
- Models:
  - **Time Petri nets (time on tokens)**
  - **Timed Petri nets (time on transition duration)**
  - **Stochastic nets**
  - **Queueing nets (FIFO behaviour on places)**
  - **…**

Utrecht University

Performance model: Petri nets





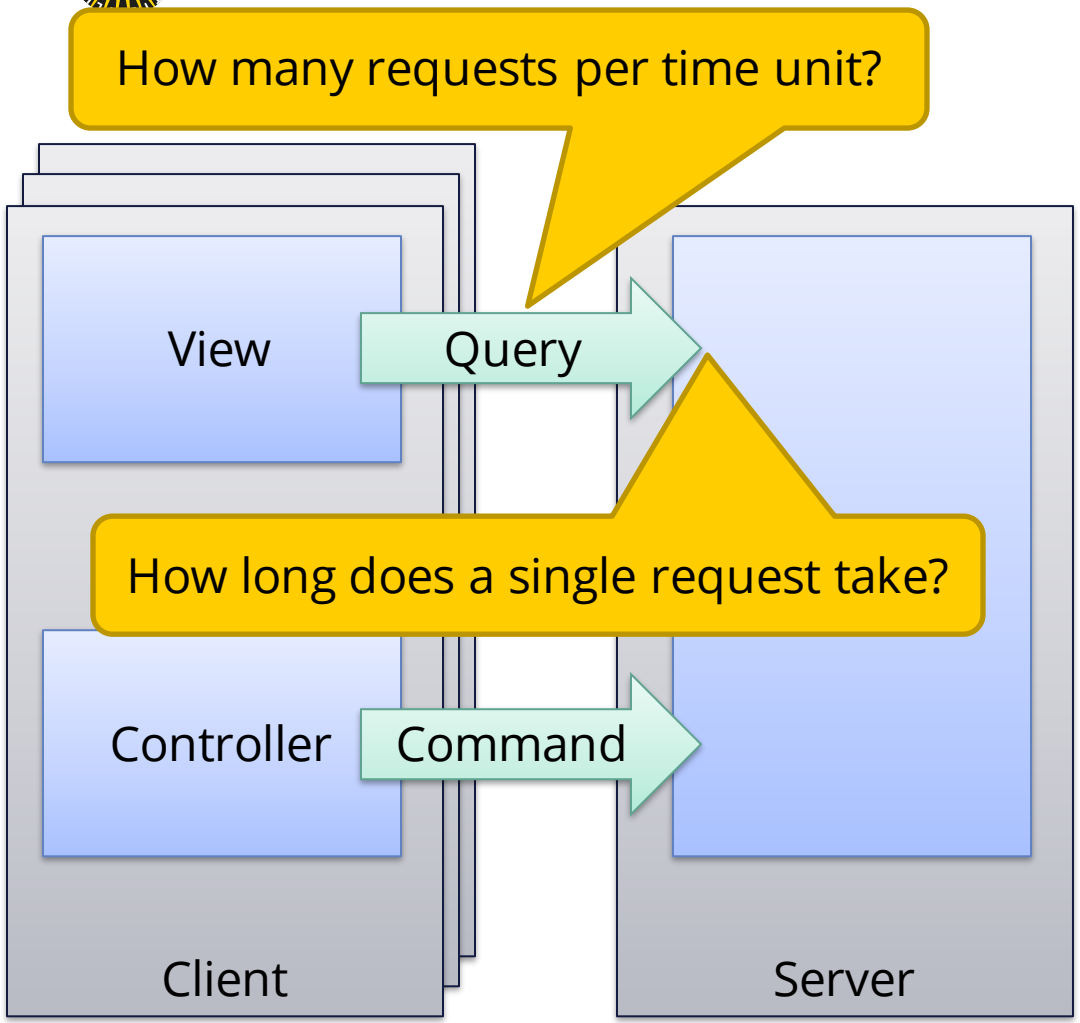
- Parameters:
  - **Delays**
  - **Serving times**
  - **Decisions paths**
  - **Resource availability**
  - **Channel behaviour**
  - **…**

- Models:
  - **Time Petri nets (time on tokens)**
  - **Queueing nets (FIFO behaviour on places)**
  - **…**

Nice, but how do you use these in practice?

Utrecht University

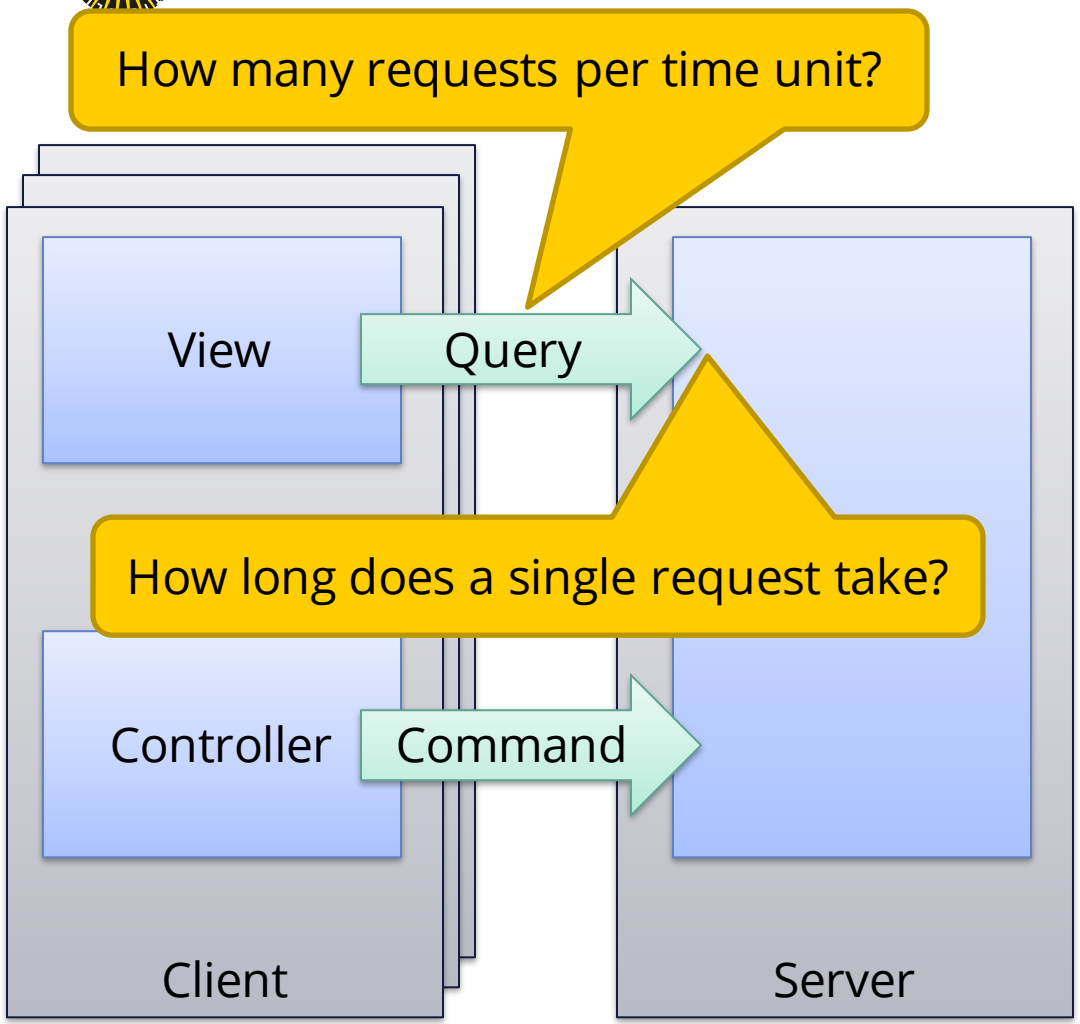
Modeling performance and capacity



- Step 1: Determine parameters

Utrecht University

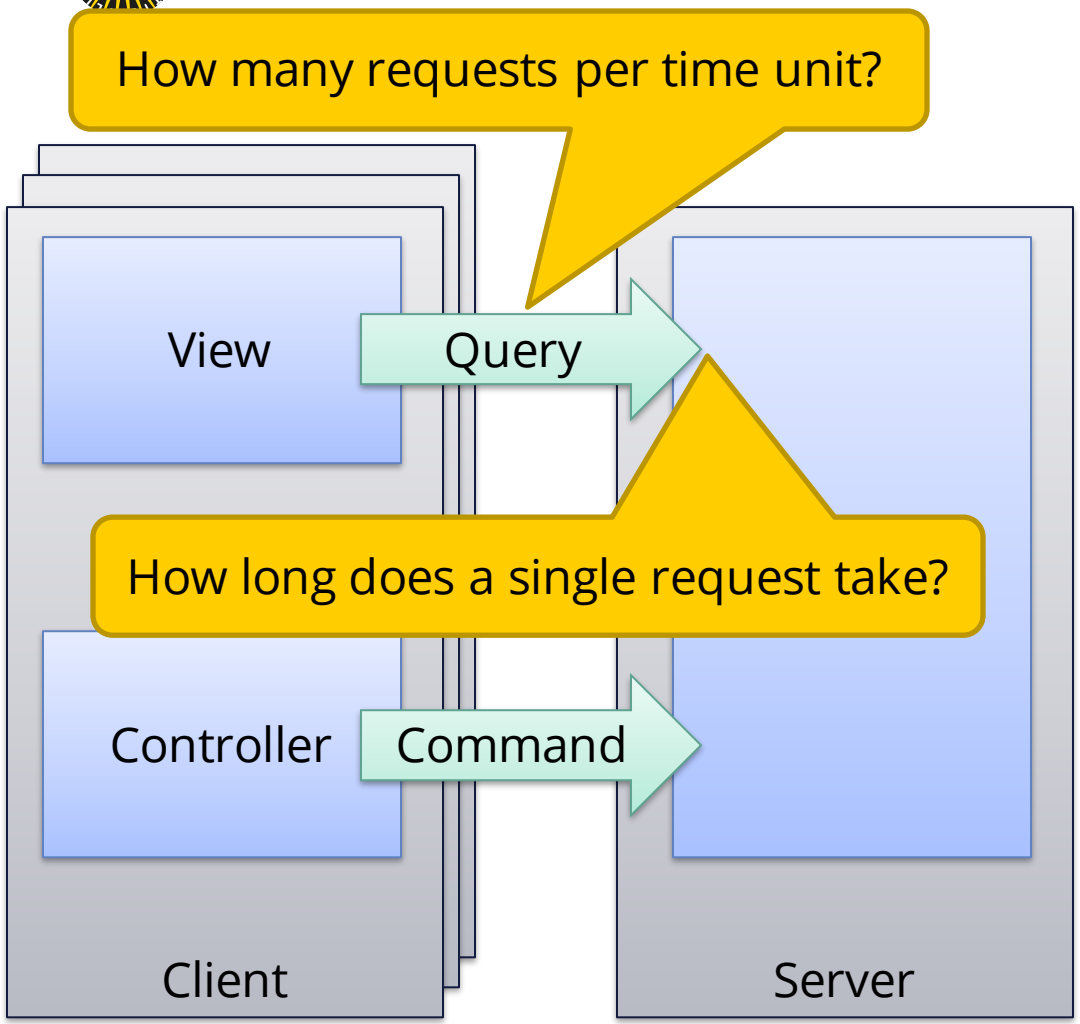
Modeling performance and capacity



- Step 1: Determine parameters
  - **Requests per time unit:** 60 per hour per client
  - **Time units to handle a request:** 3s per request
- Step 2: Back-of-the-envelope analysis

Utrecht University

Modeling performance and capacity



- Step 1: Determine parameters

  Requests per time unit: 60 per hour per client

  Time units to handle a request: 3s per request

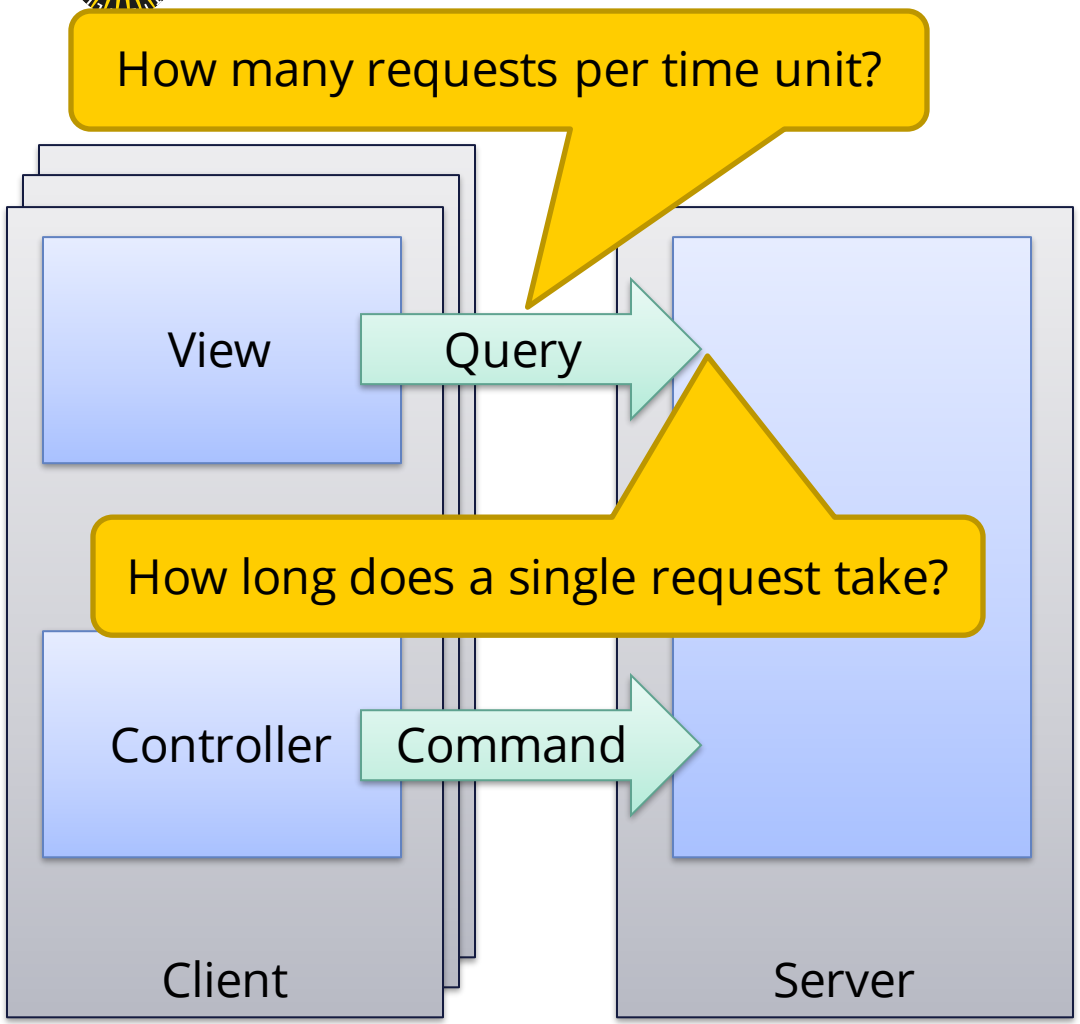
- Step 2: Back-of-the-envelope analysis

  Handling requests:

  3s per request ➔ 20 per minute ➔ 1200 per hour

Utrecht University

Modeling performance and capacity



- Step 1: Determine parameters

  Requests per time unit: 60 per hour per client

  Time units to handle a request: 3s per request

- Step 2: Back-of-the-envelope analysis

  Handling requests:

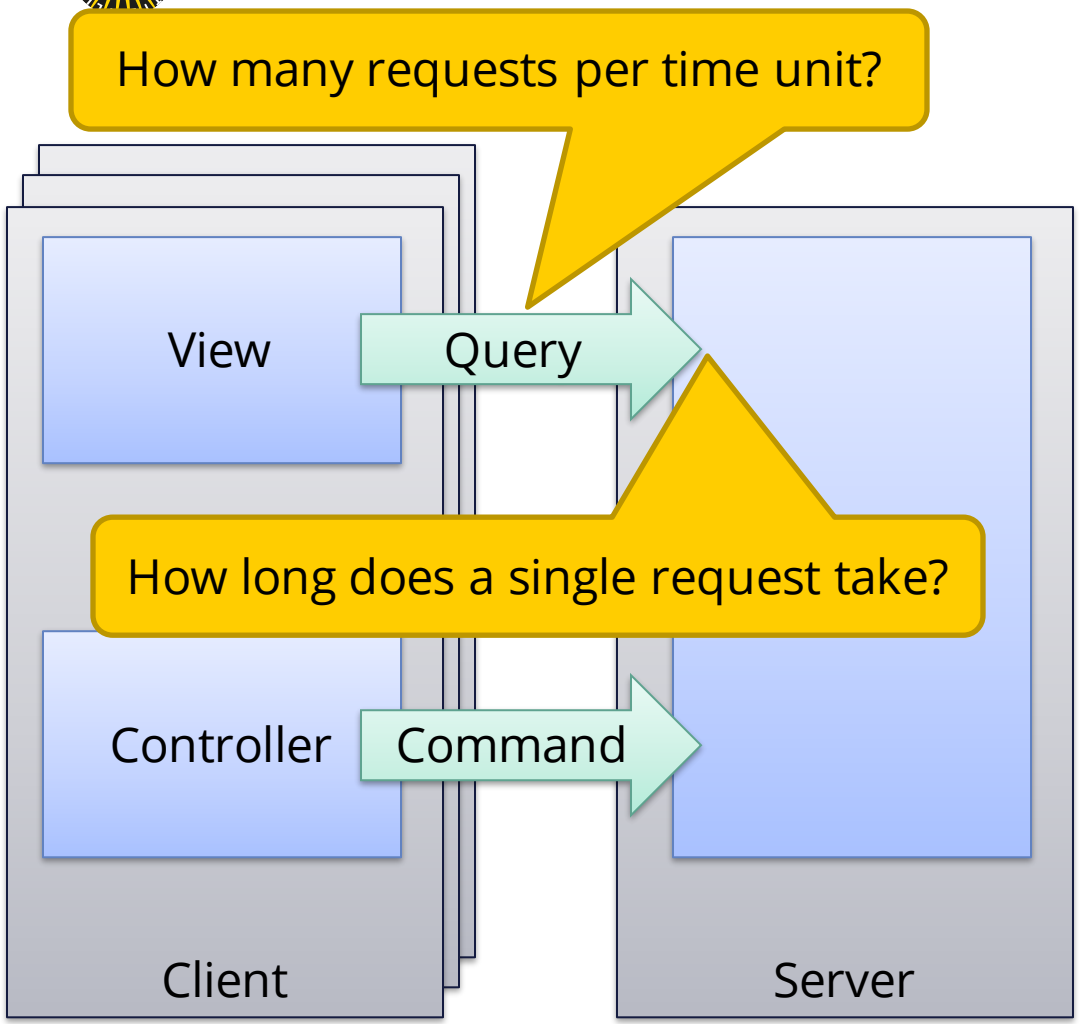
  3s per request ➔ 20 per minute ➔ 1200 per hour

  Requests required: (10.000 clients)

  60 ph per client ➔ 600,000 per hour

Utrecht University

Modeling performance and capacity



- Step 1: Determine parameters

  Requests per time unit: 60 per hour per client

  Time units to handle a request: 3s per request

- Step 2: Back-of-the-envelope analysis

  Handling requests:

  3s per request ➔ 20 per minute ➔ 1200 per hour

  Requests required:

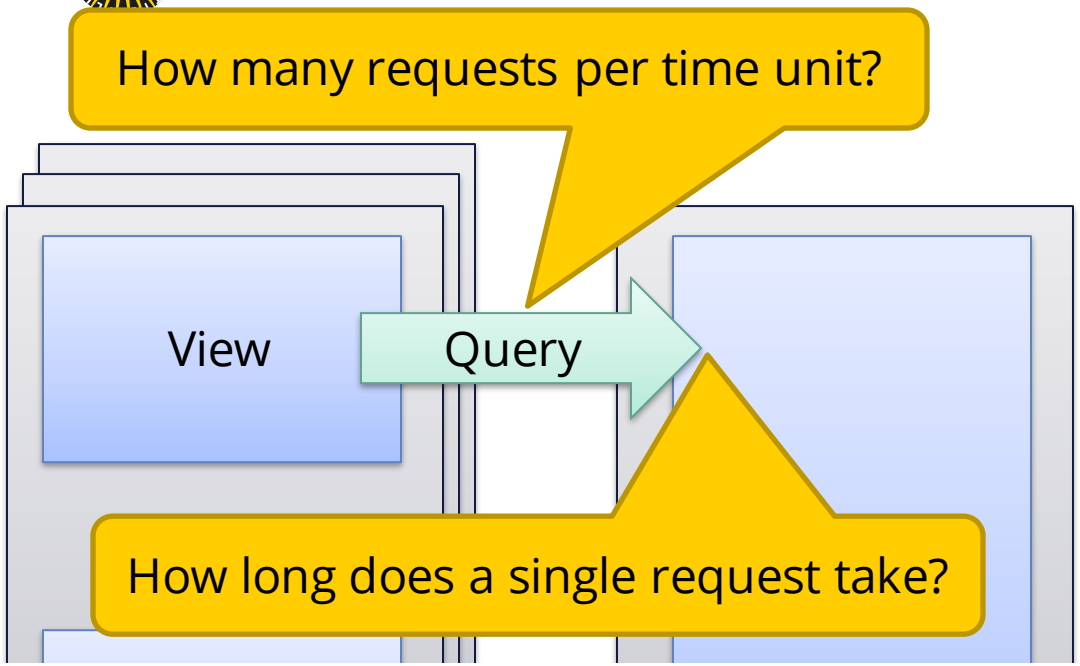
  60 ph per client ➔ 600,000 per hour

  Concurrent query interfaces:

  600,000/1200: 500 INSTANCES !

Utrecht University

Modeling performance and capacity



- Step 1: Determine parameters

  Requests per time unit: 60 per hour per client

  Time units to handle a request: 3s per request

- Step 2: Back-of-the-envelope analysis

  Handling requests:

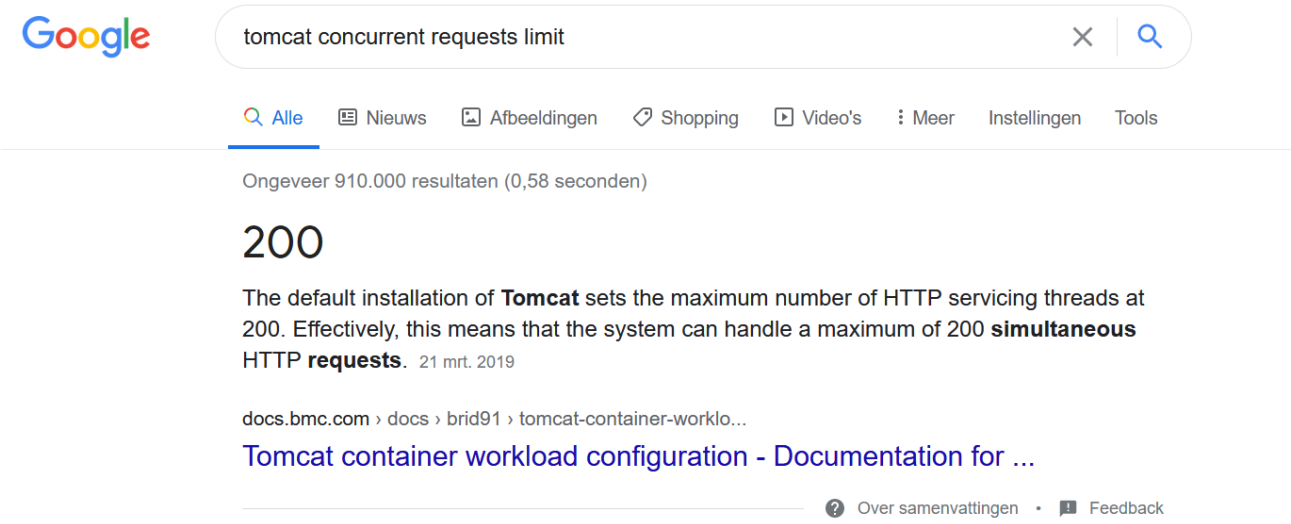
  3s per request ➔ 20 per minute ➔ 1200 per hour

  Requests required:

  60 ph per client ➔ 600,000 per hour

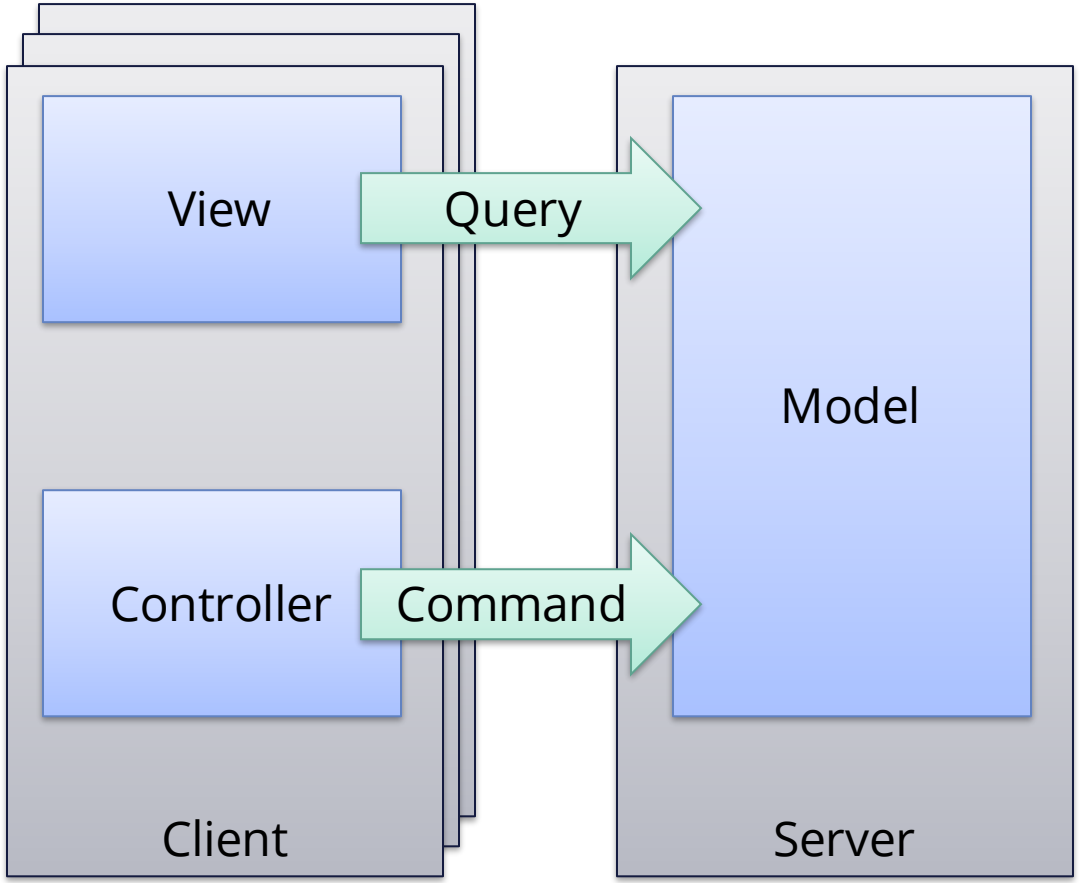
  Concurrent query interfaces:

  600,000/1200: 500 INSTANCES !



Utrecht University

For all techniques



- Critical assessment
  - **NOT: making severe or negative judgments**
  - **BUT: making careful and analytical evaluations to come to a skillful judgment**
- What are the trade-offs?
  - **All alternatives covered?**
  - **Is this the best candidate?**
- What are the risks?
  - **All risks identified and covered?**
- What are the constraints and assu[illegible]ns?
  - **Are these valid?**

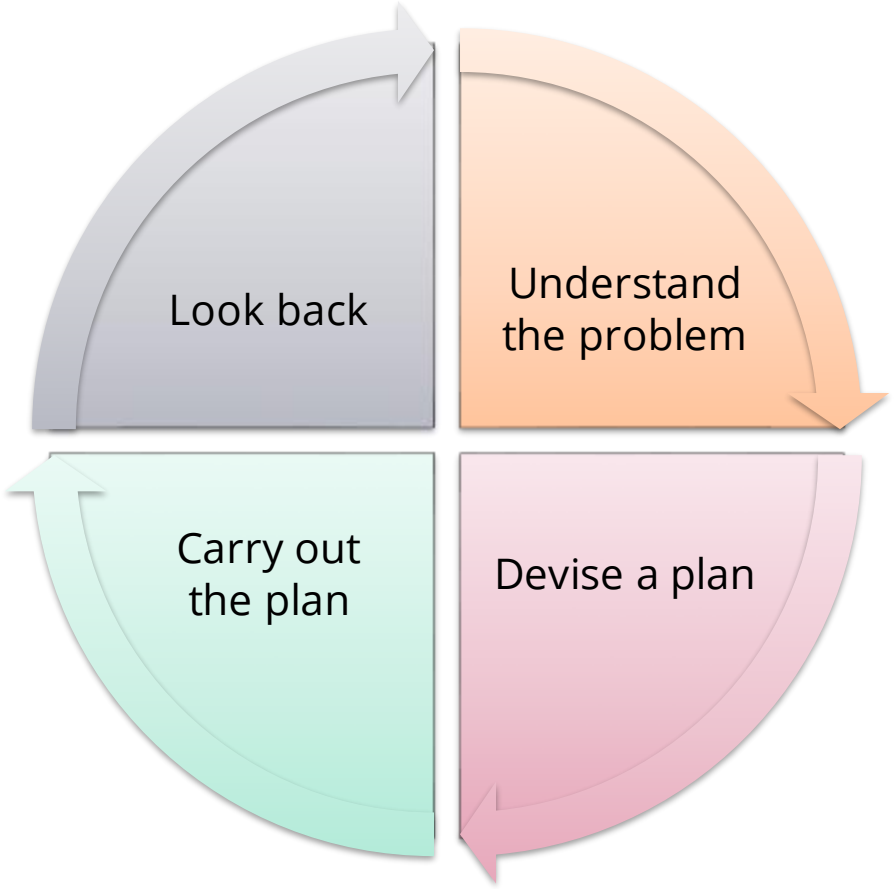
But all depends on “critical thinking”!!!

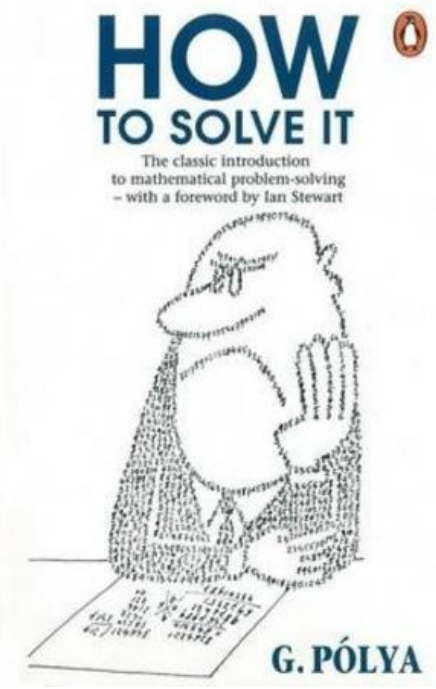
Utrecht University

Critical thinking

Utrecht University

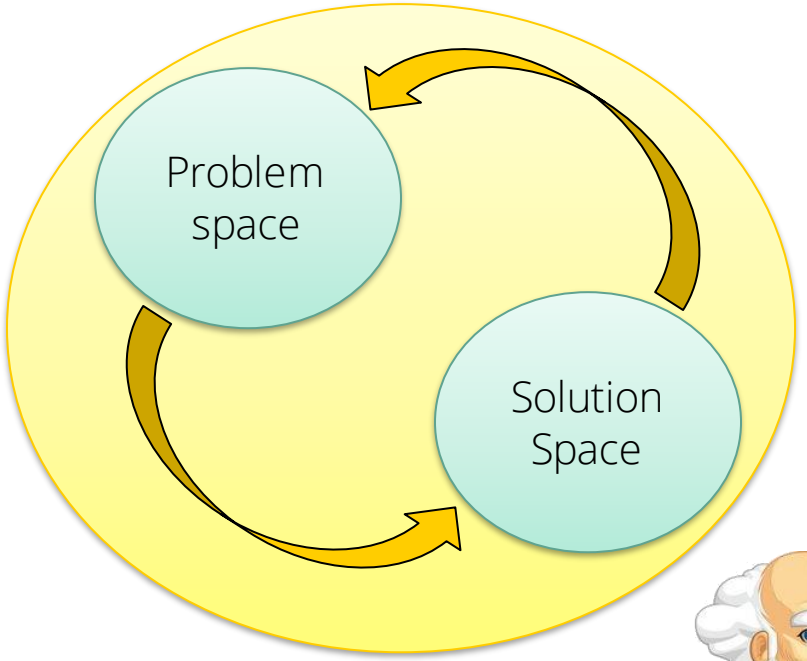
4 stages from “how to solve it”





- Understand the problem
  - **What are the unknowns?**
  - **What are the data?**
  - **What are the conditions?**
  - **Is it possible to satisfy the condition?**
- Devise a plan
  - **Have you seen a similar problem?**
  - **Do you know of related problems?**
  - **Divide and conquer:**
    - **is there a part for which you know a solution?**
- Carry out the plan
  - **Is each step performed correctly? What evidence do you have?**
- Look back
  - **Can you check the results? How about the arguments?**
  - **Can you obtain the results in a different way?**
  - **Can you apply the solution elsewhere?**

The art of posing questions



- IDEALS Approach:

Identify the problem	What is the real question at hand?
Define the context	What facts frame this problem?
Enumerate choices	What are plausible options?
Analyze options	What is the best course of action?
List reasons explicitly	Why is it the best course of action?
Self-correct	Look again… what did we miss?

- For each view, perspective concern, … we repeat this process

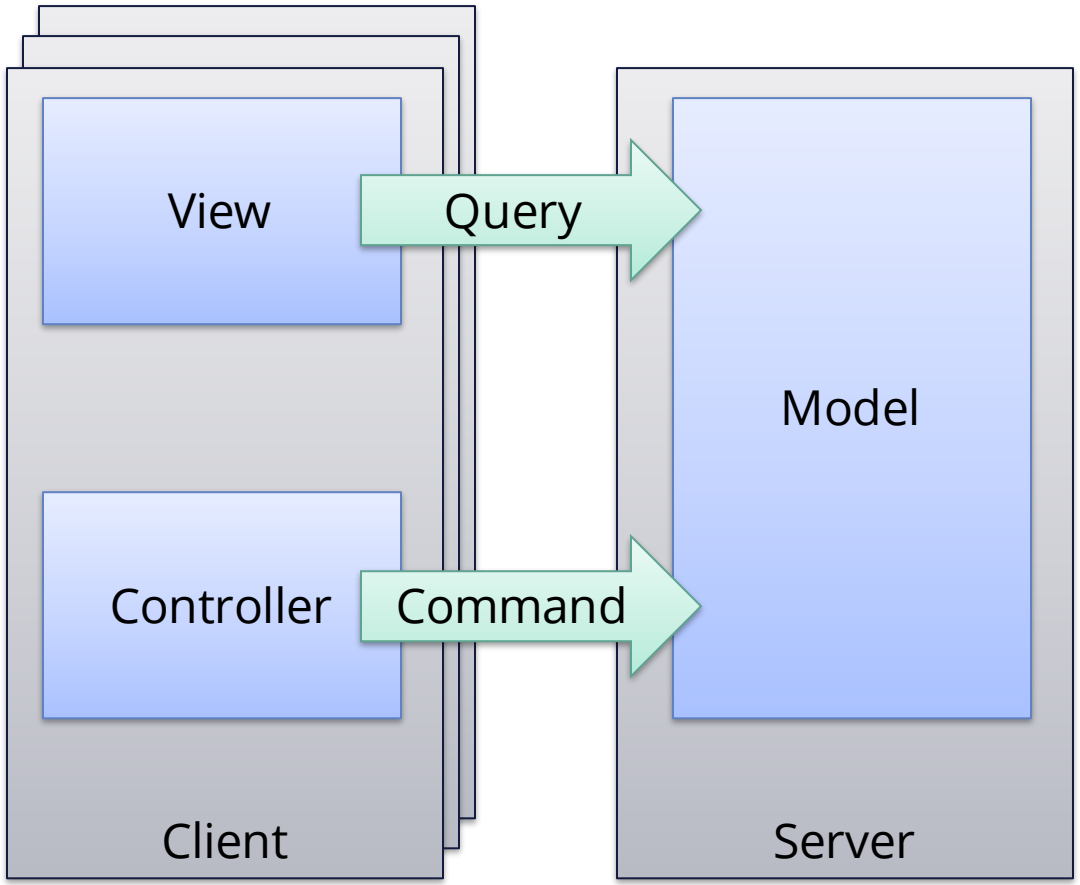
Sounds simpler than it is…

Utrecht University

Back to software architecture

Utrecht University

Can we solve this in a better way?



- Step 1: Determine parameters

  Requests per time unit: 60 per hour per client

  Time units to handle a request: 3s per request

- Step 2: Back-of-the-envelope analysis

  Handling requests:

  3s per request ➔ 20 per minute ➔ 1200 per hour

  Requests required:

  60 ph per client ➔ 600,000 per hour

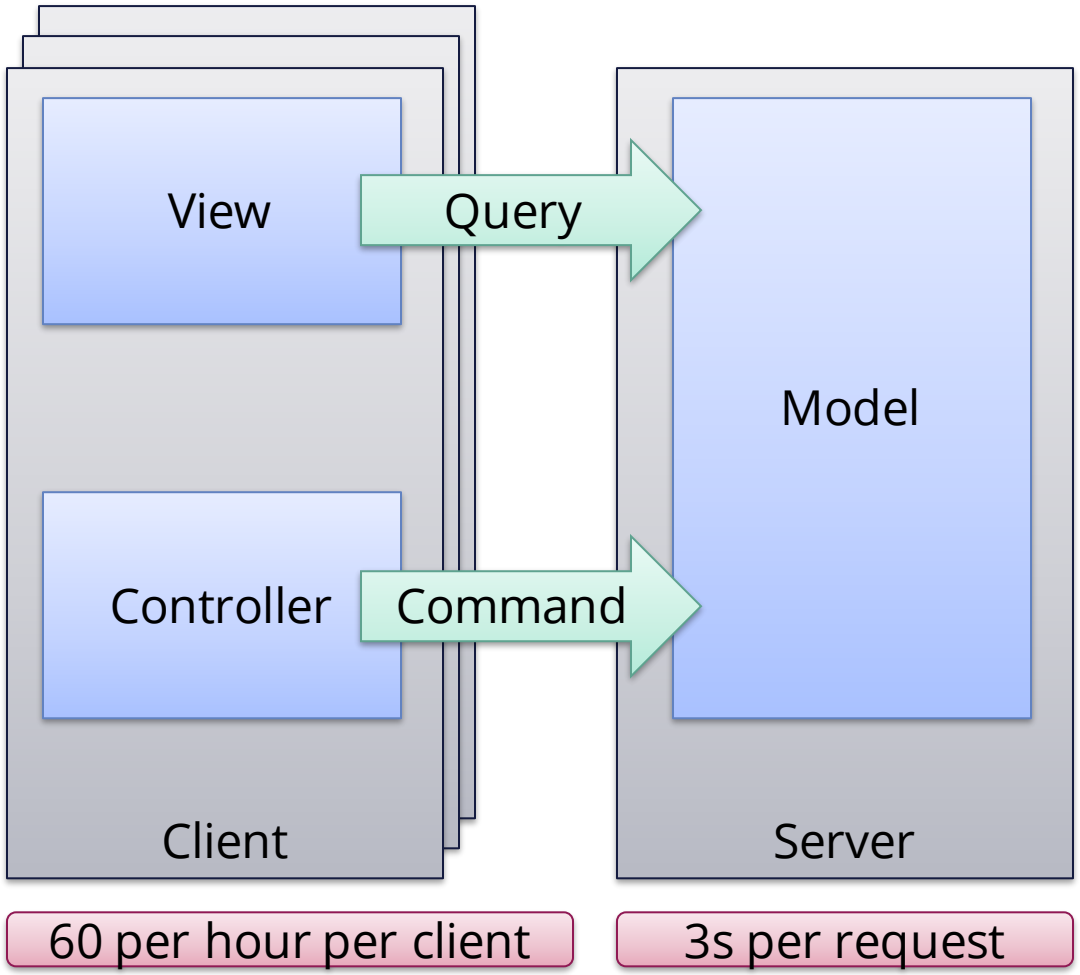
  Concurrent query interfaces:

  600,000/1200: 500 INSTANCES !

What questions would you pose?

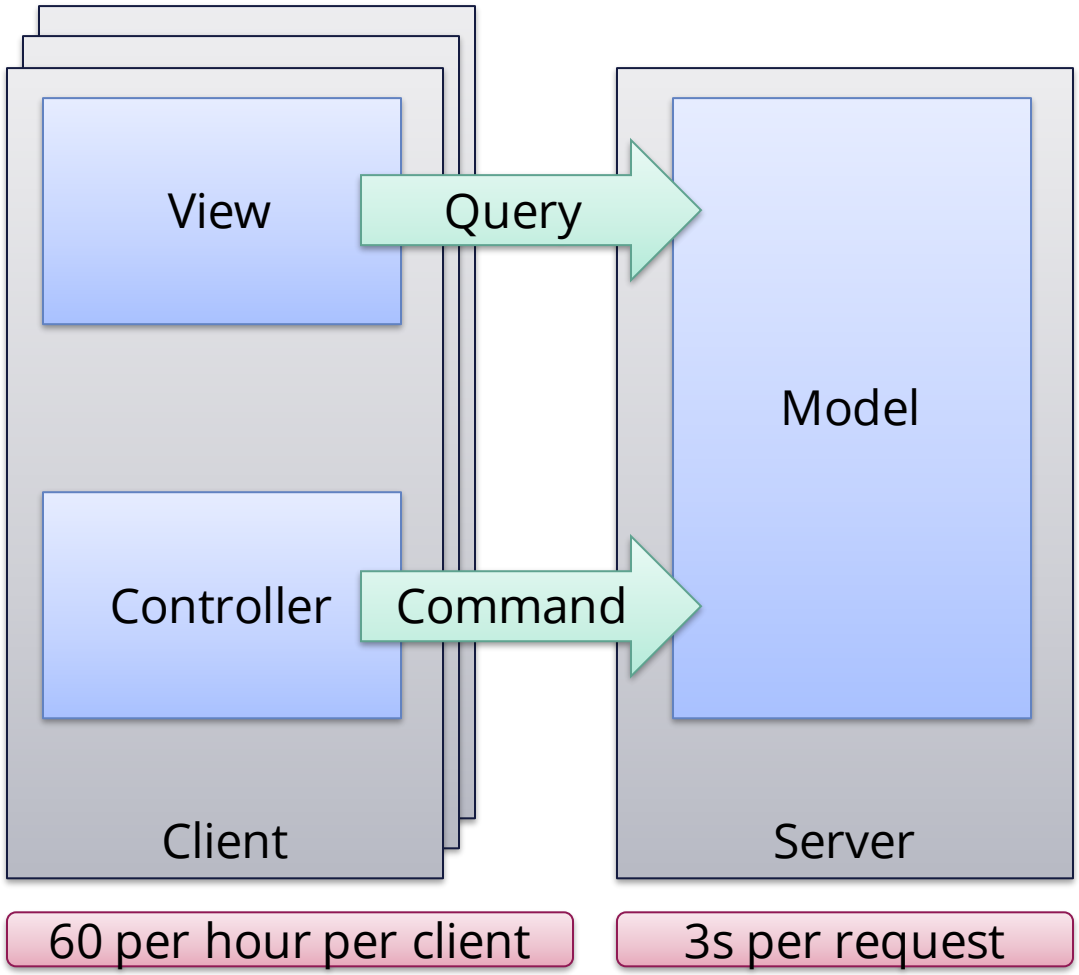
Utrecht University

Step 1: create quality scenarios

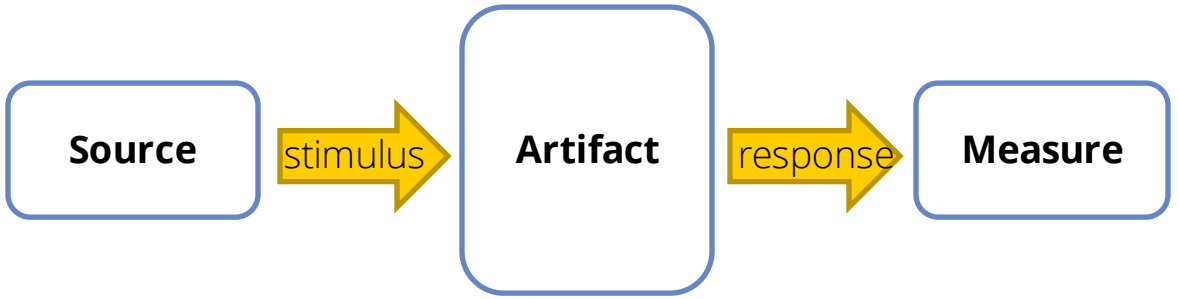


Utrecht University

Step 1: create quality scenarios

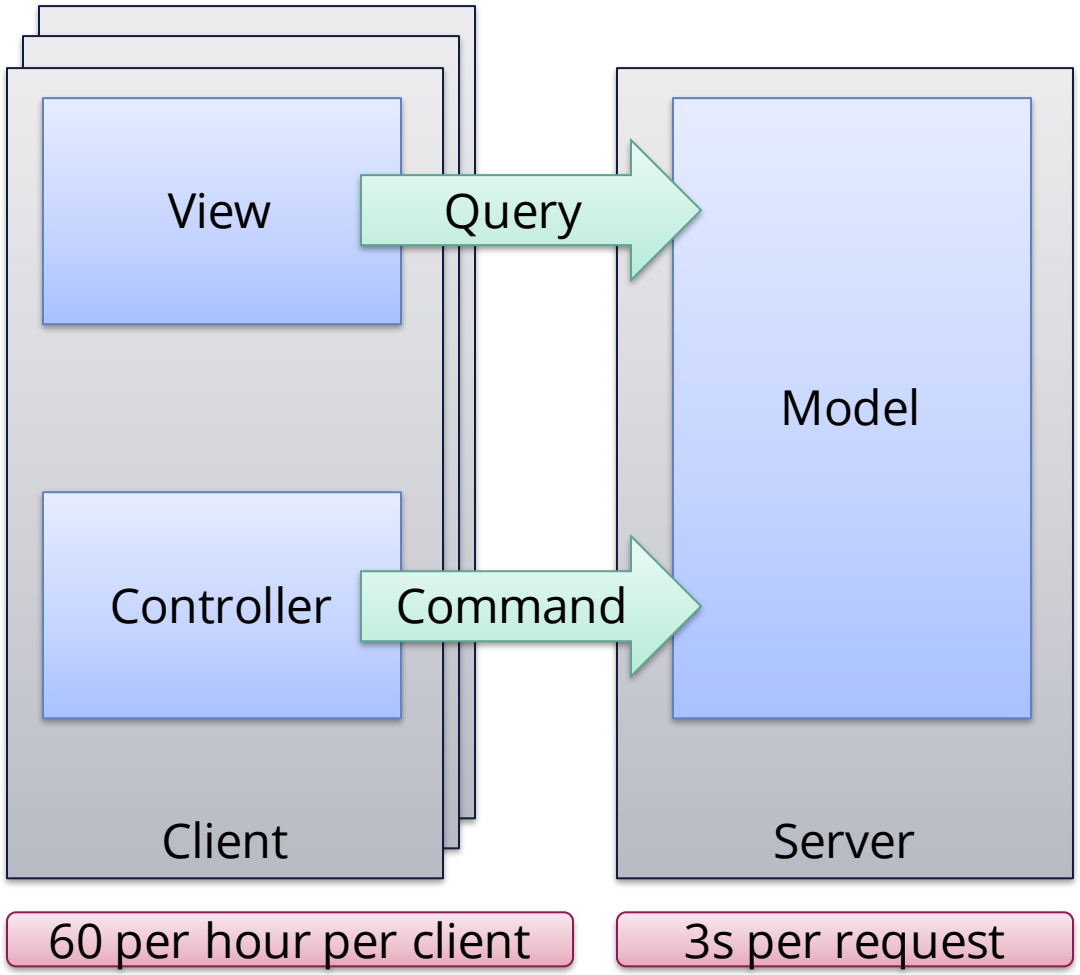


Environment

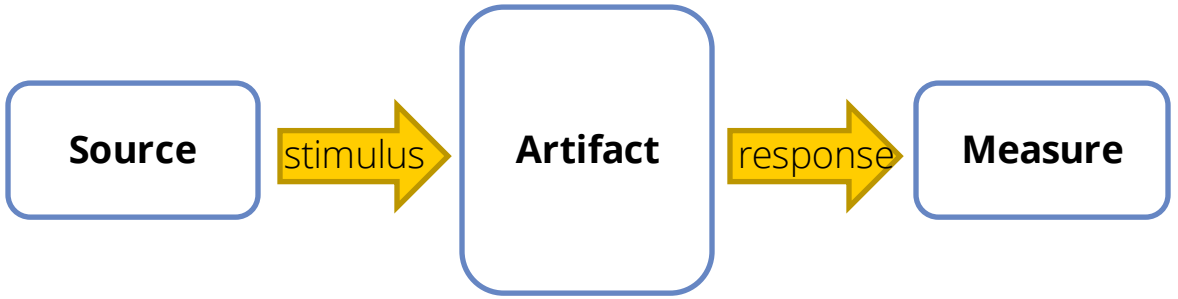


Utrecht University

Step 1: create quality scenarios



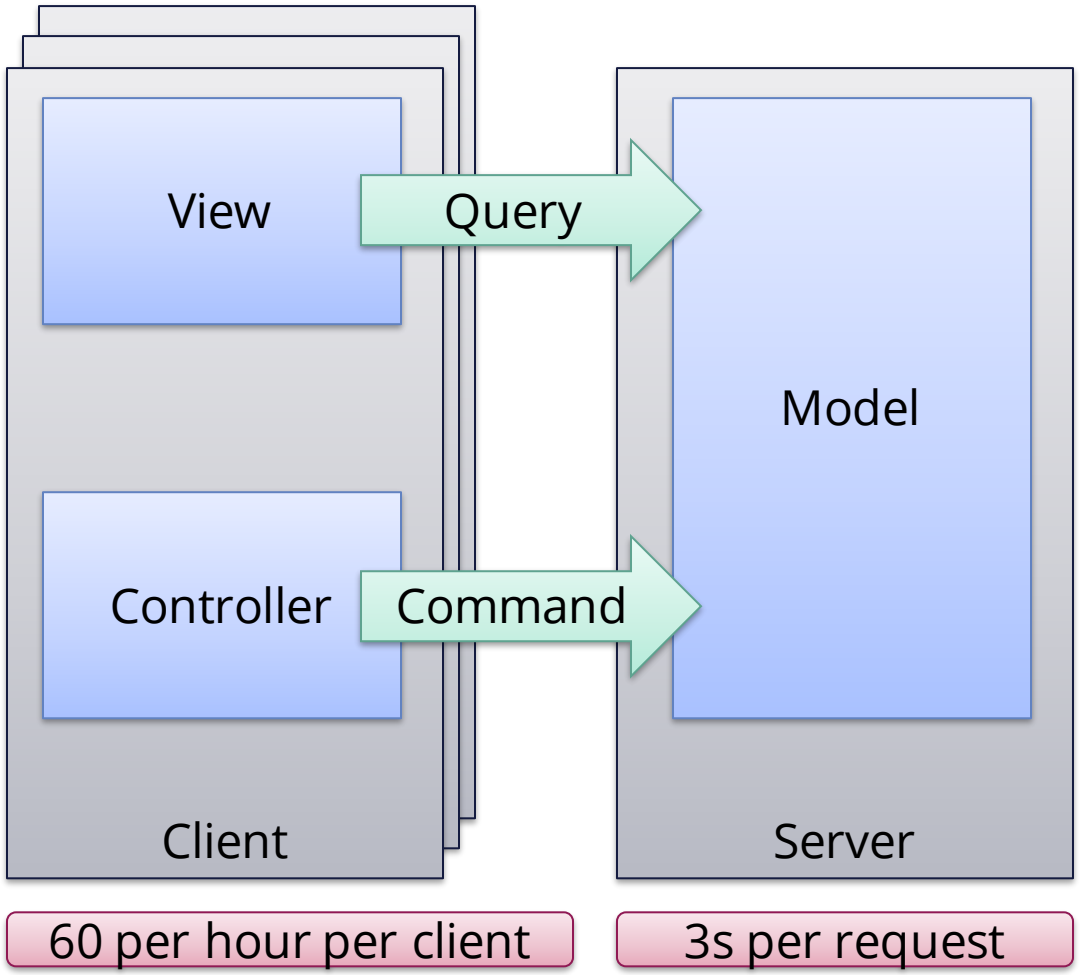
Environment



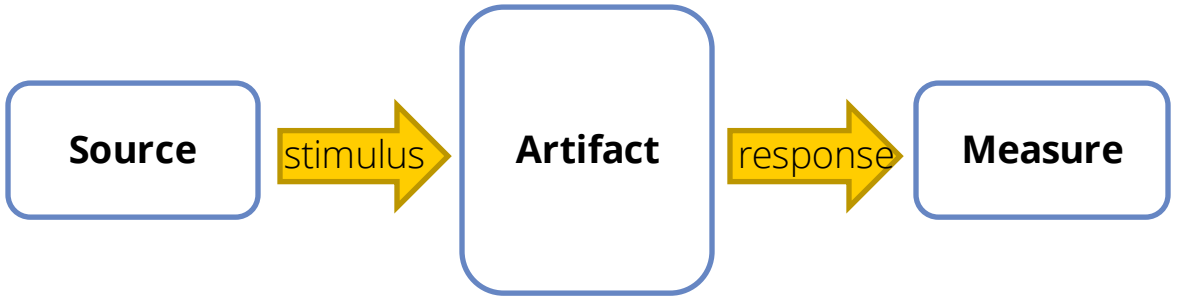
	QS 1	QS 2
Environment	Running system	Running system
Source	User	User
Stimulus	Initiates a query	Initiates a command
Artifact	System	System
Response	Gives result	Command processed
Measure	Latency	Latency

Utrecht University

Step 1: create quality scenarios



Environment

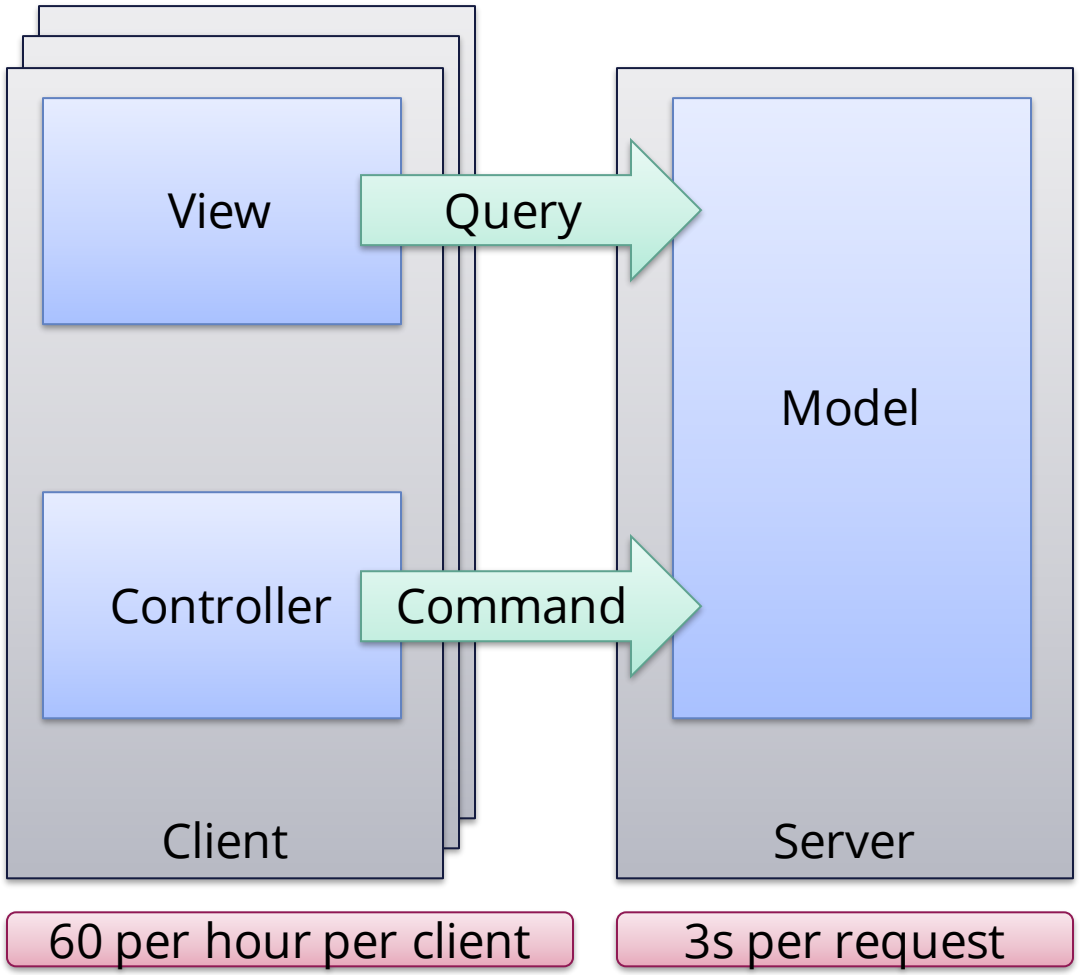


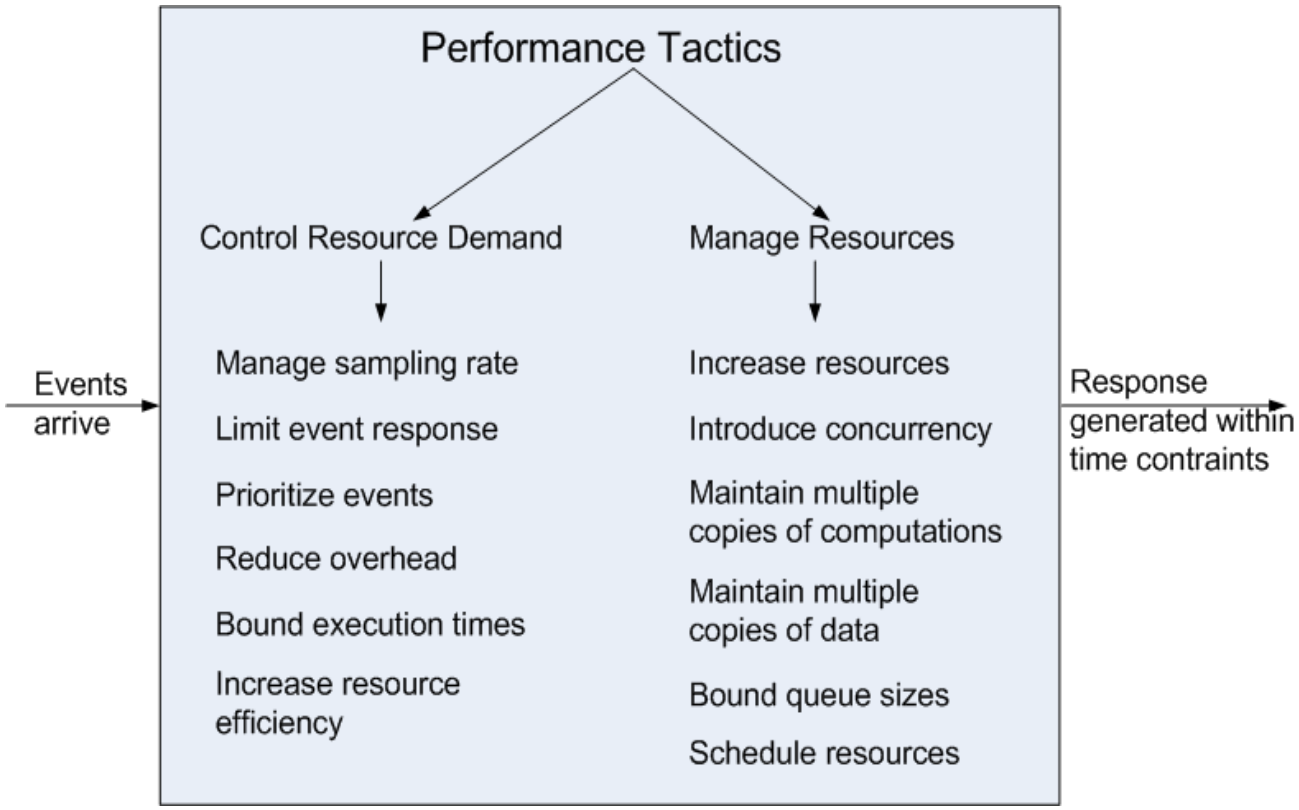
	QS 1	QS 2
Environment	Running system	Running system
Source	User	User
Stimulus	Initiates a query	Initiates a command
Artifact	System	System
Response	Gives result	Command processed
Measure	Latency	Latency

Current solution requires 5.000 query interfaces!

Utrecht University

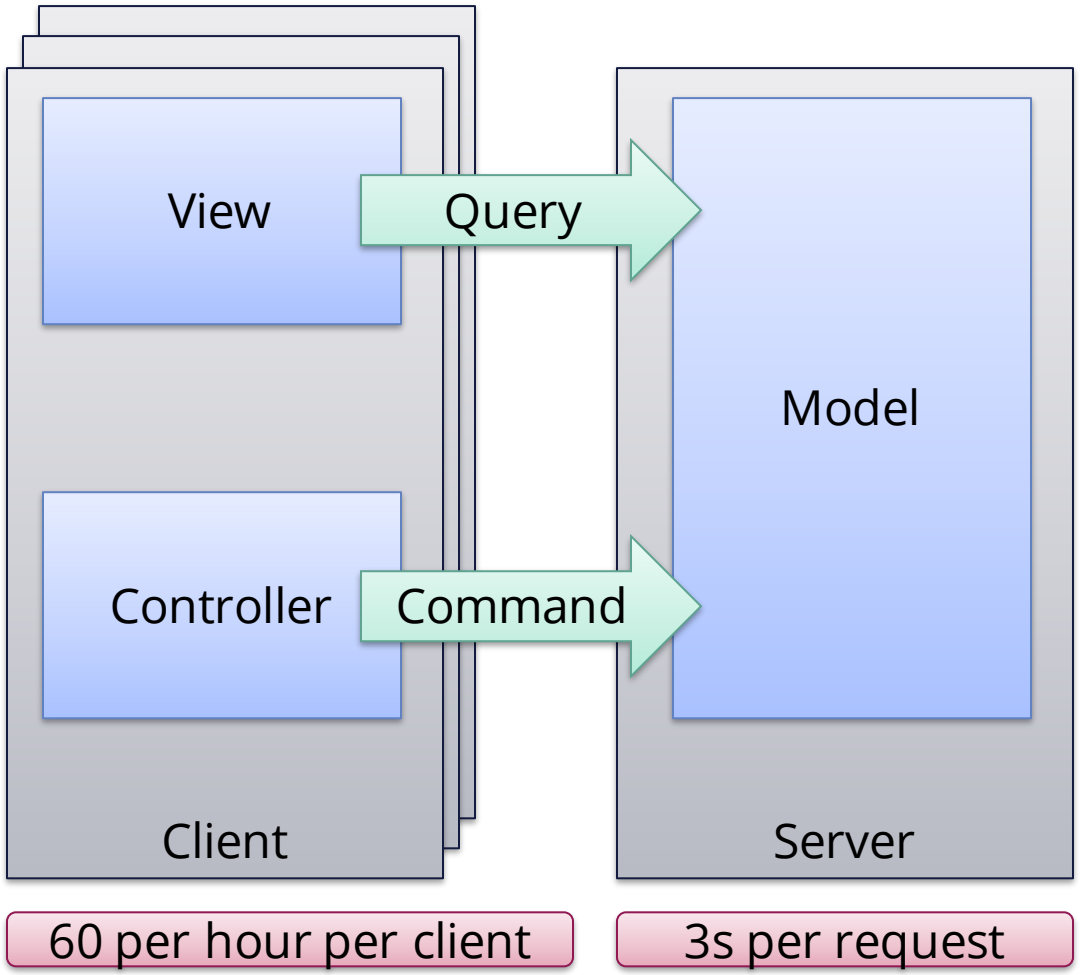
Step 2: can we use tactics?

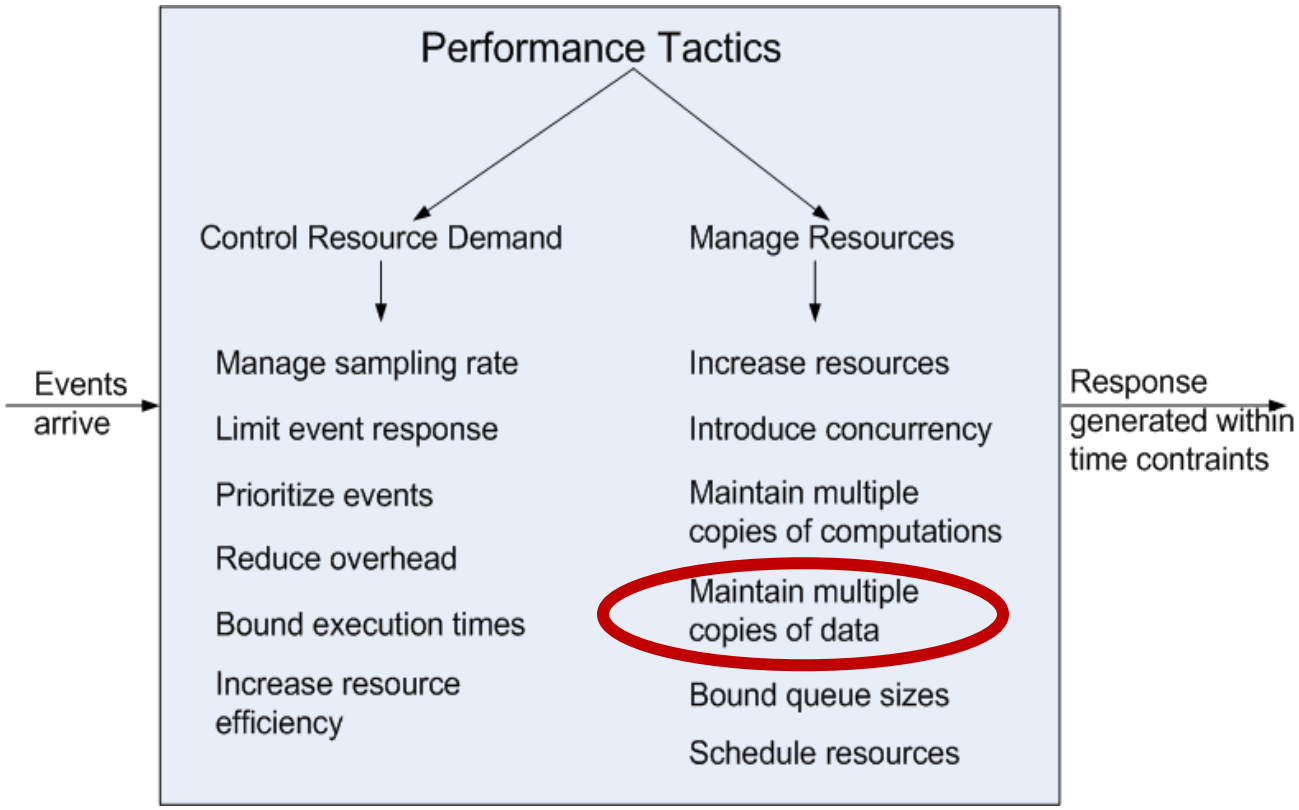




Utrecht University

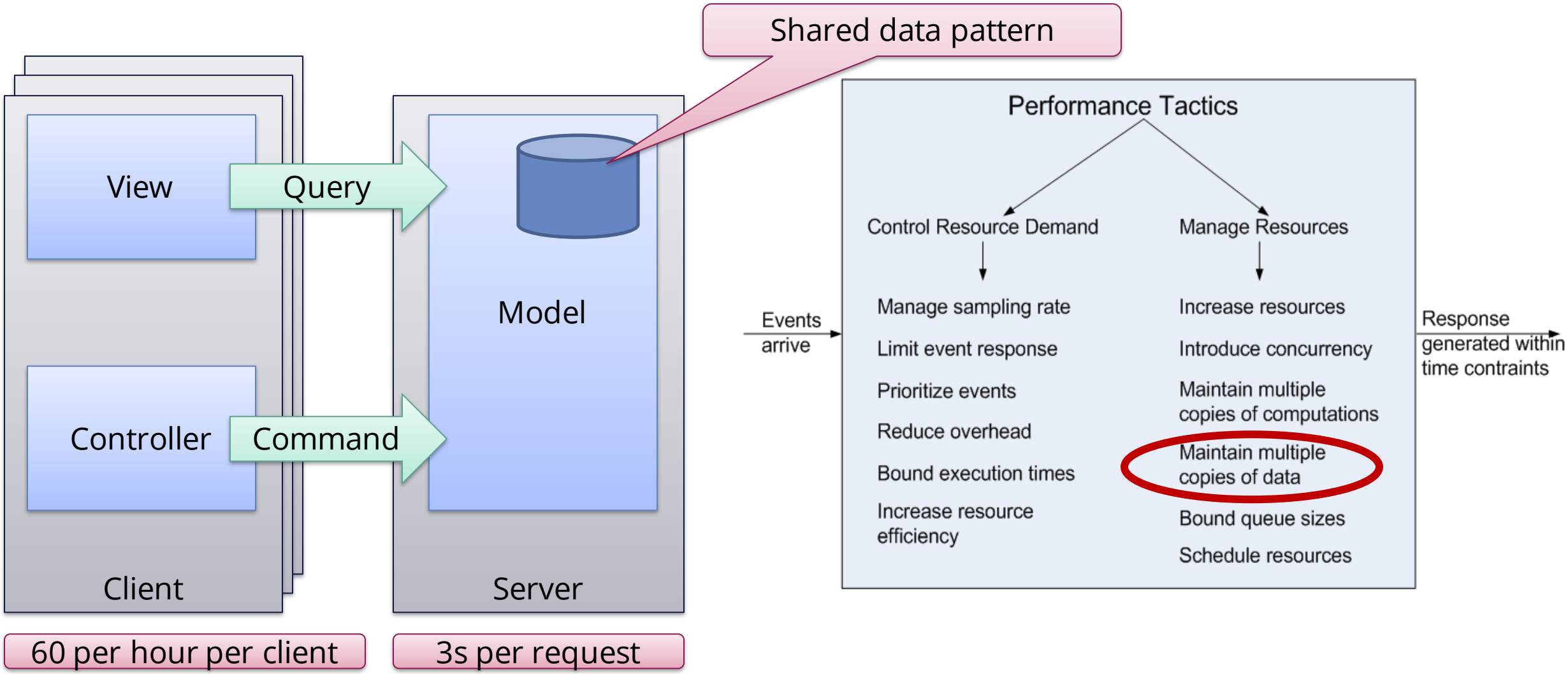
Step 2: can we use tactics?





Utrecht University

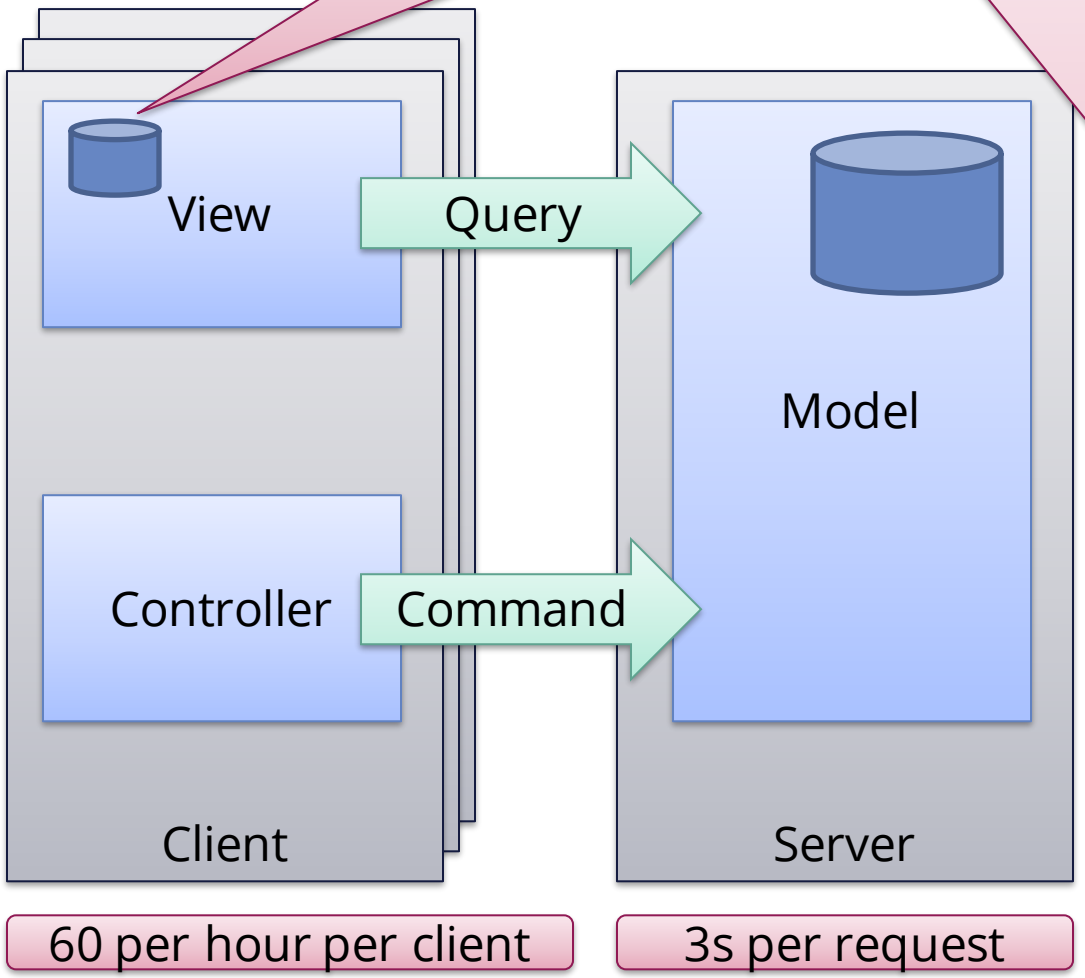
Step 2: can we use tactics?



Utrecht University

Step 2: can we use tactics?

Materialized view



60 per hour per client

3s per request

S. Koenig and R. Paige (1981). A transformational framework for the automatic control of derived data. In: Very Large Data Bases - Volume 7 (VLDB '81). pp, 306–318.

Efficiently Updating Materialized Views*

José A Blakeley, Per-Åke Larson, Frank Wm Tompa

Data Structuring Group,
Department of Computer Science,
University of Waterloo,
Waterloo, Ontario, N2L 3G1

Abstract

Query processing can be sped up by keeping frequently accessed users' views materialized However, the need to access base relations in response to queries can be avoided only if the materialized view is adequately maintained We propose a method in which all database updates to base relations are first

rived relation–or *view*–is defined by a relational expression (i e , a query evaluated over the base relations) A derived relation may be *virtual*, which corresponds to the traditional concept of a view, or *materialized*, which means that the resulting relation is actually stored As the database changes because of updates applied to the base relations, the materialized views may also require change A materi-

onse
rated within
contraints

Bound execution times

Increase resource efficiency

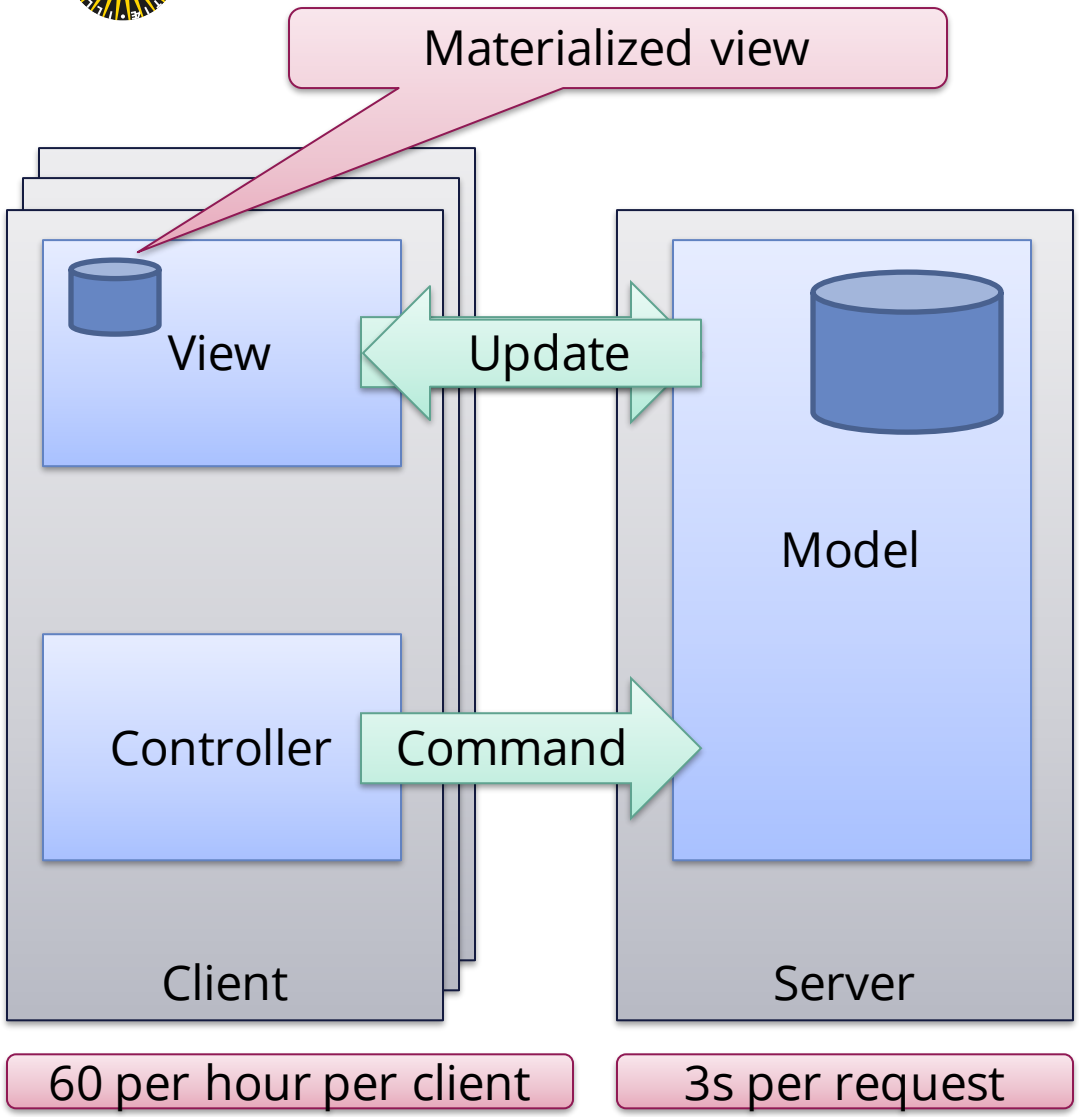
Maintain multiple copies of data

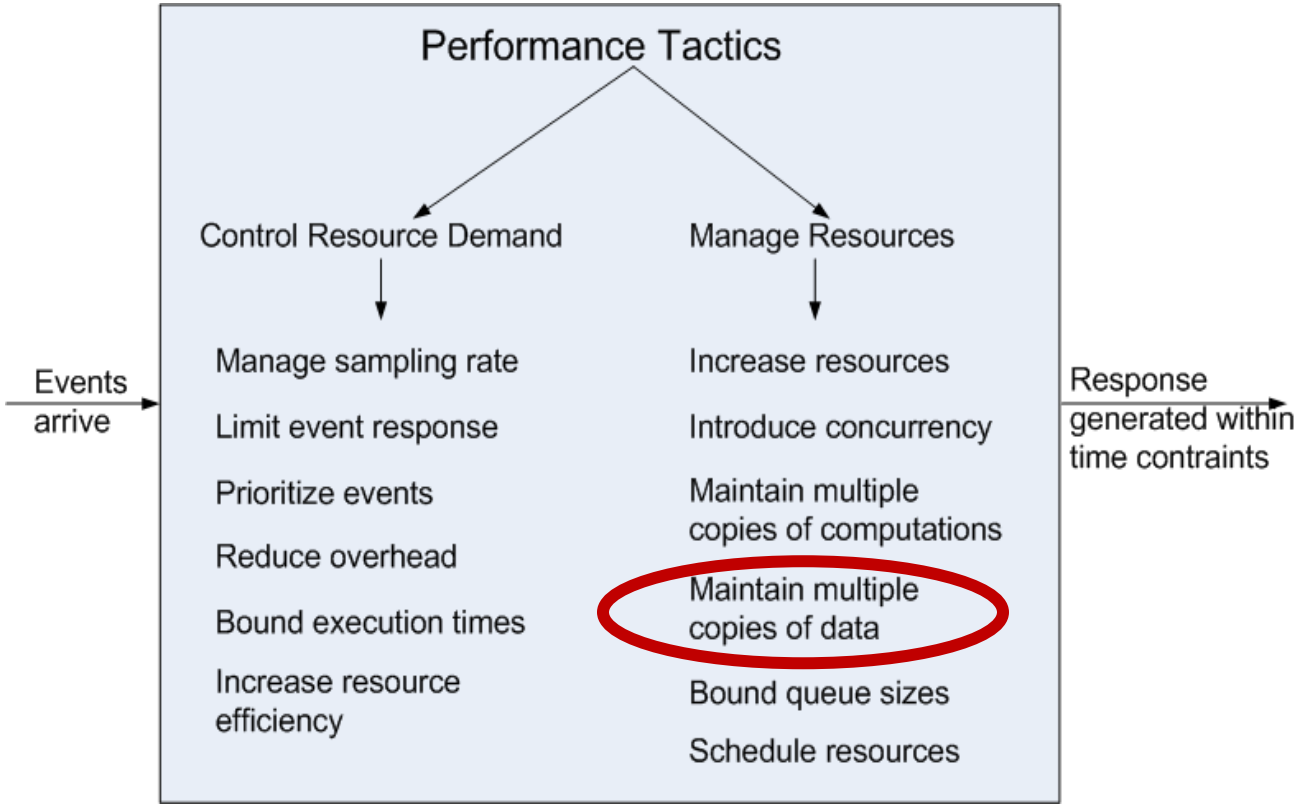
Bound queue sizes

Schedule resources

Utrecht University

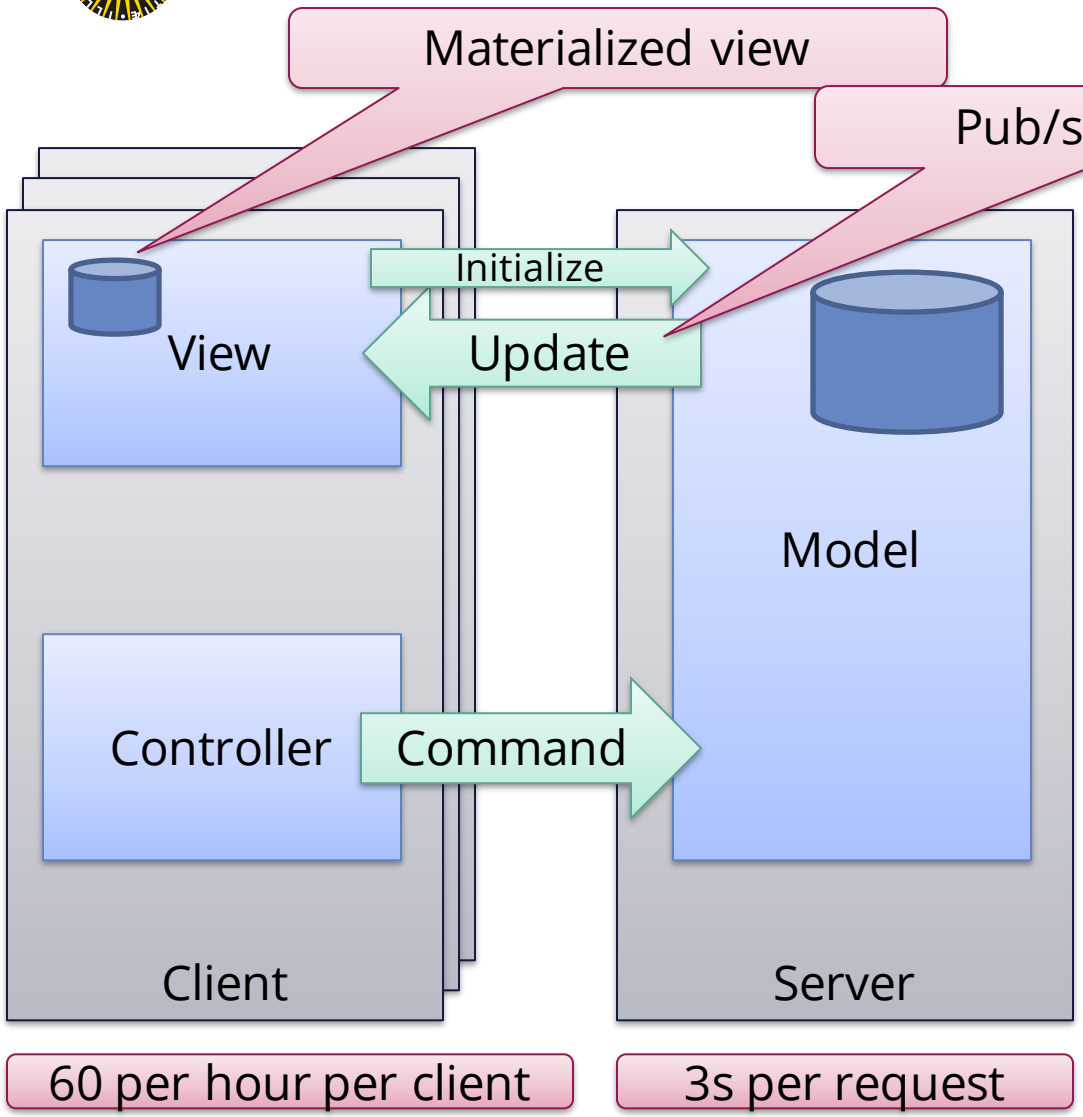
Step 3: architectural patterns?

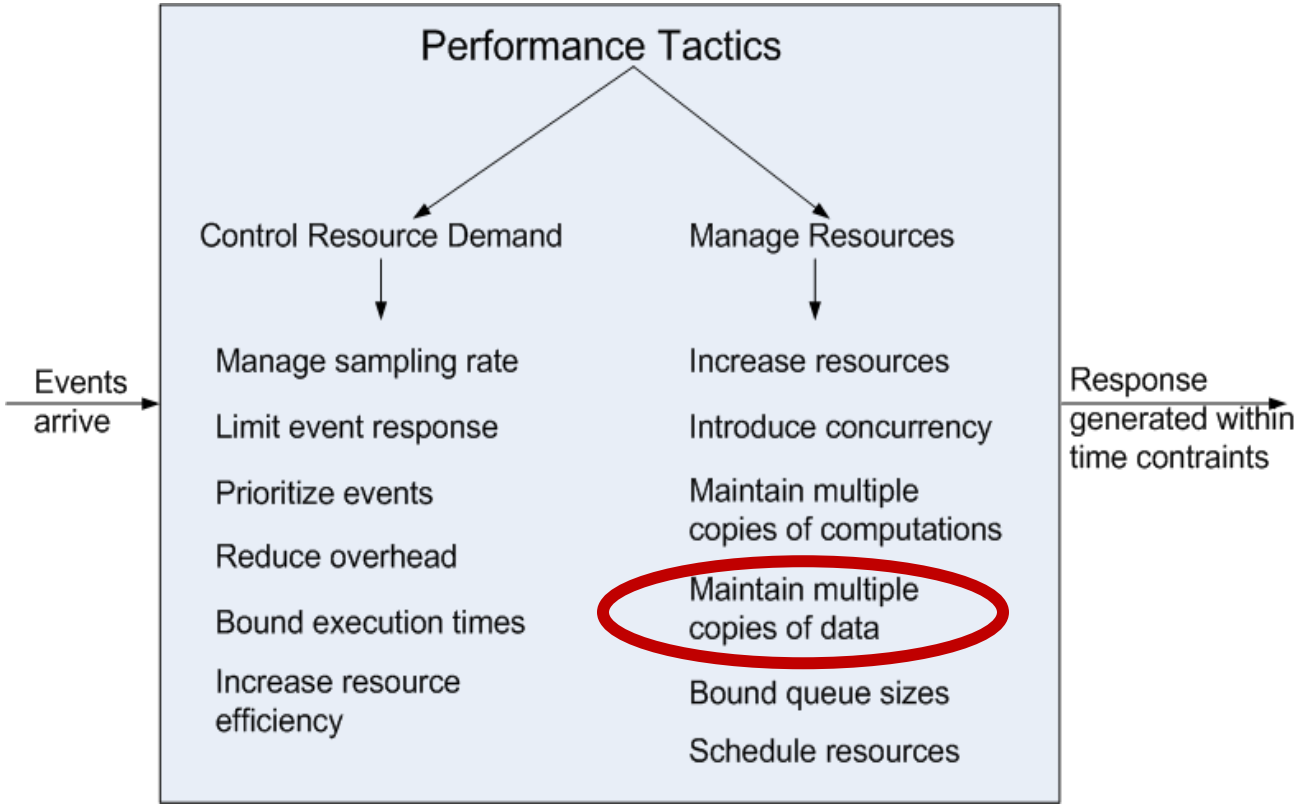




Utrecht University

Step 3: architectural patterns?



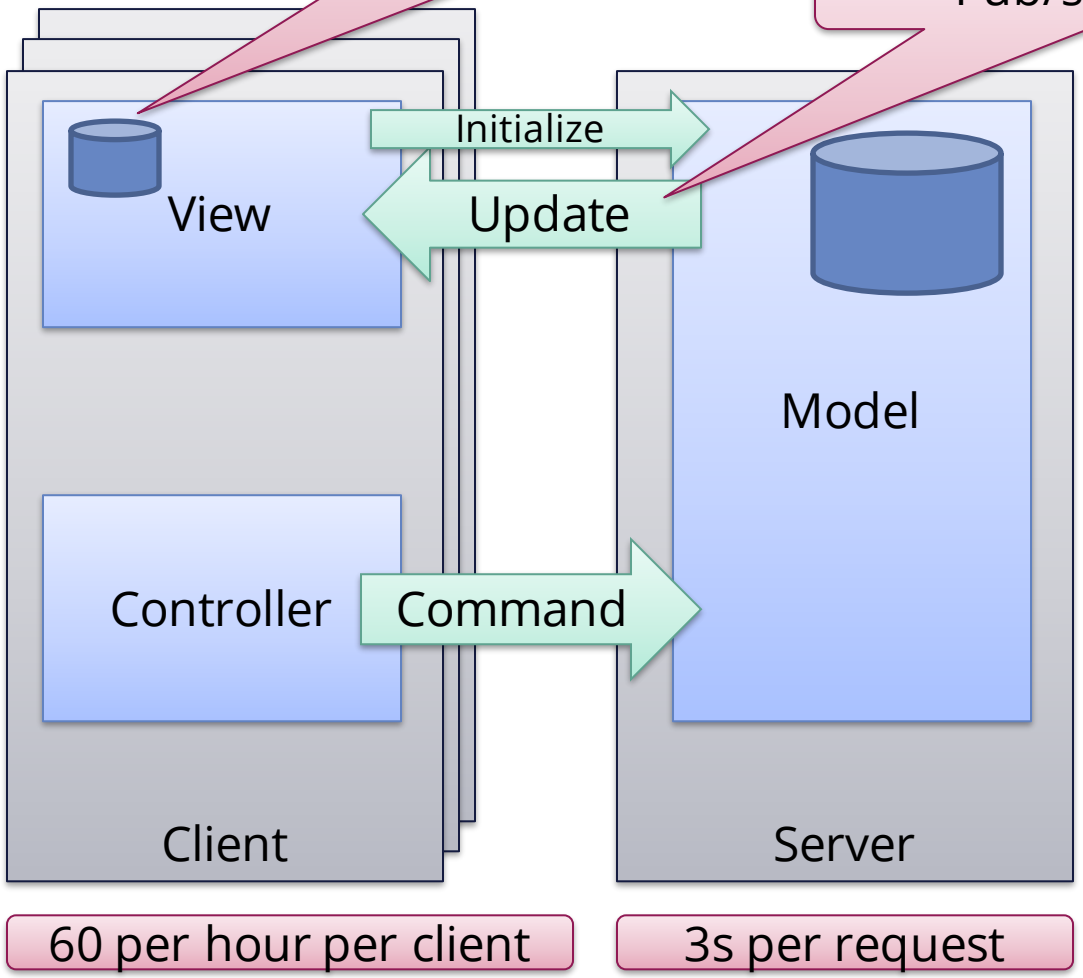


Utrecht University

Step 4: document decision!

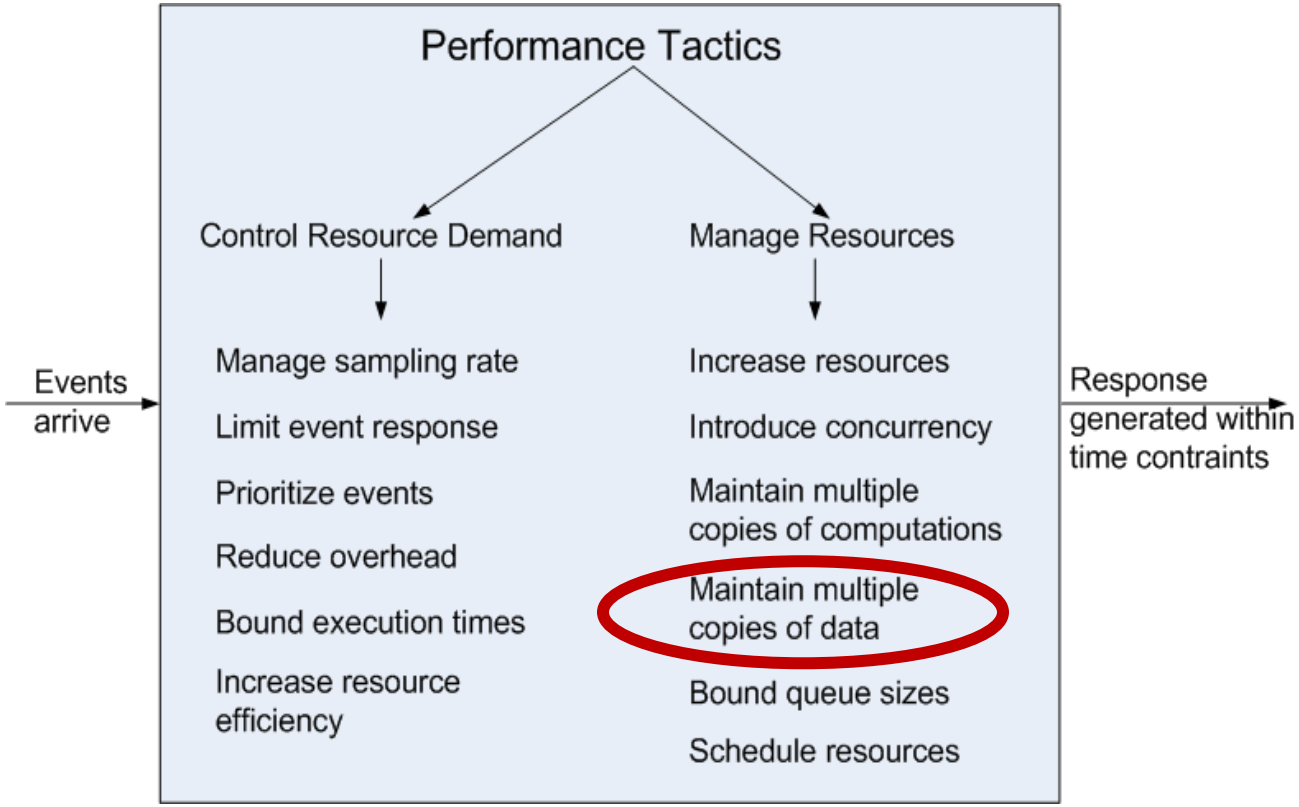
Materialized view

Pub/sub pattern!



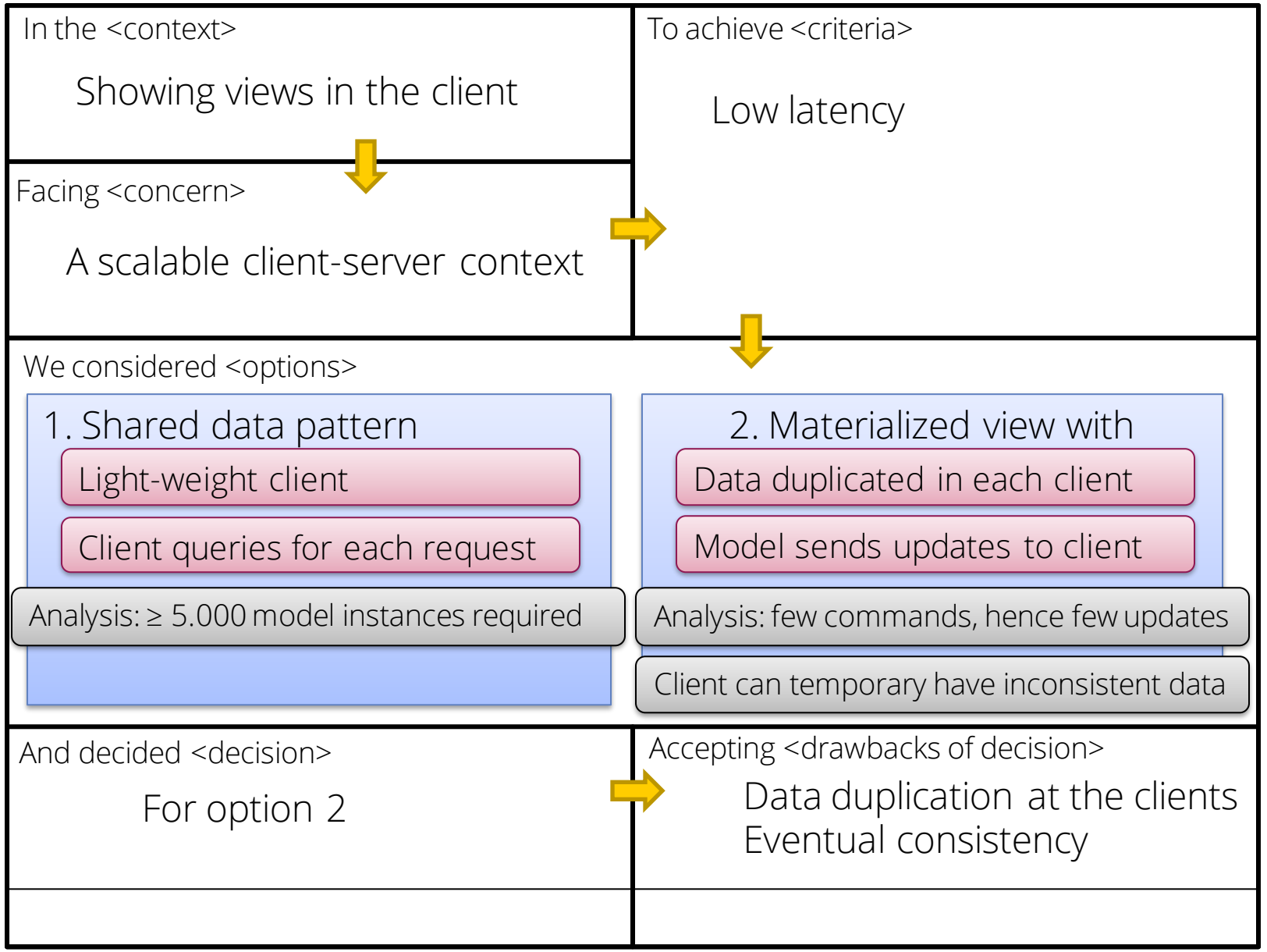
60 per hour per client

3s per request



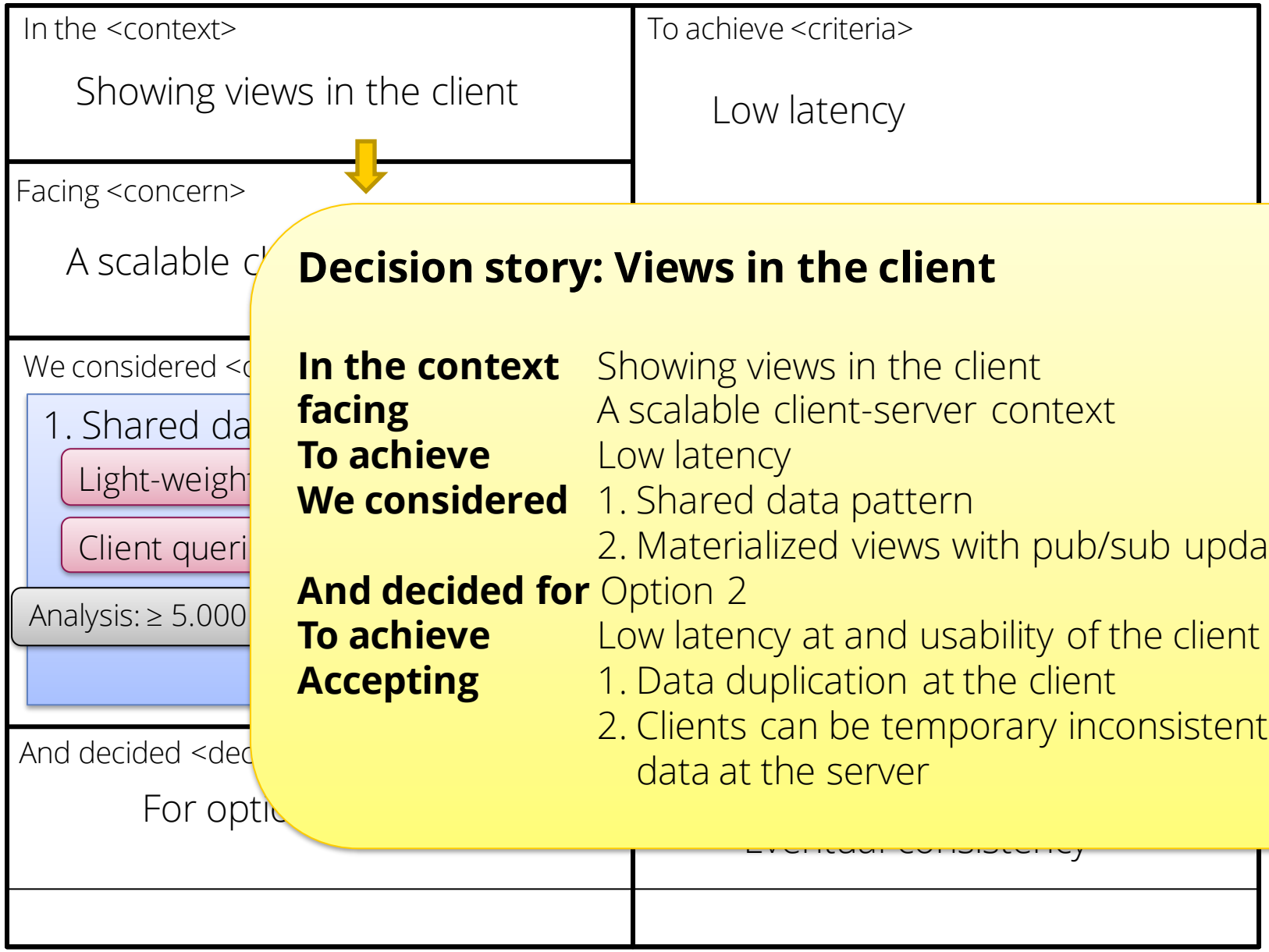
Utrecht University

Step 4: document decision!



Utrecht University

Step 4: document decision!



Decision story: Views in the client

In the context	Showing views in the client
facing	A scalable client-server context
To achieve	Low latency
We considered	1. Shared data pattern 2. Materialized views with pub/sub updates
And decided for	Option 2
To achieve	Low latency at and usability of the client
Accepting	1. Data duplication at the client 2. Clients can be temporary inconsistent with data at the server

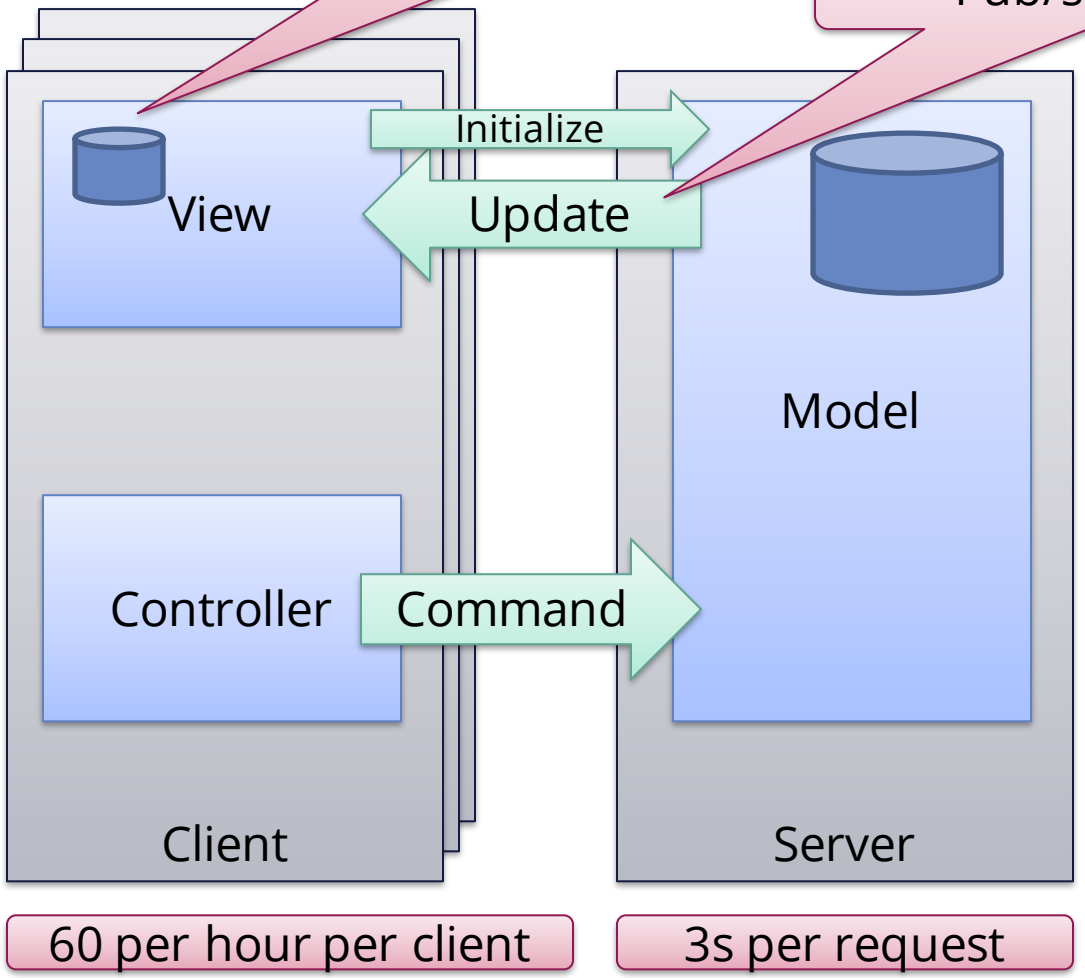
Utrecht University

Step 3: architectural patterns?

Materialized view

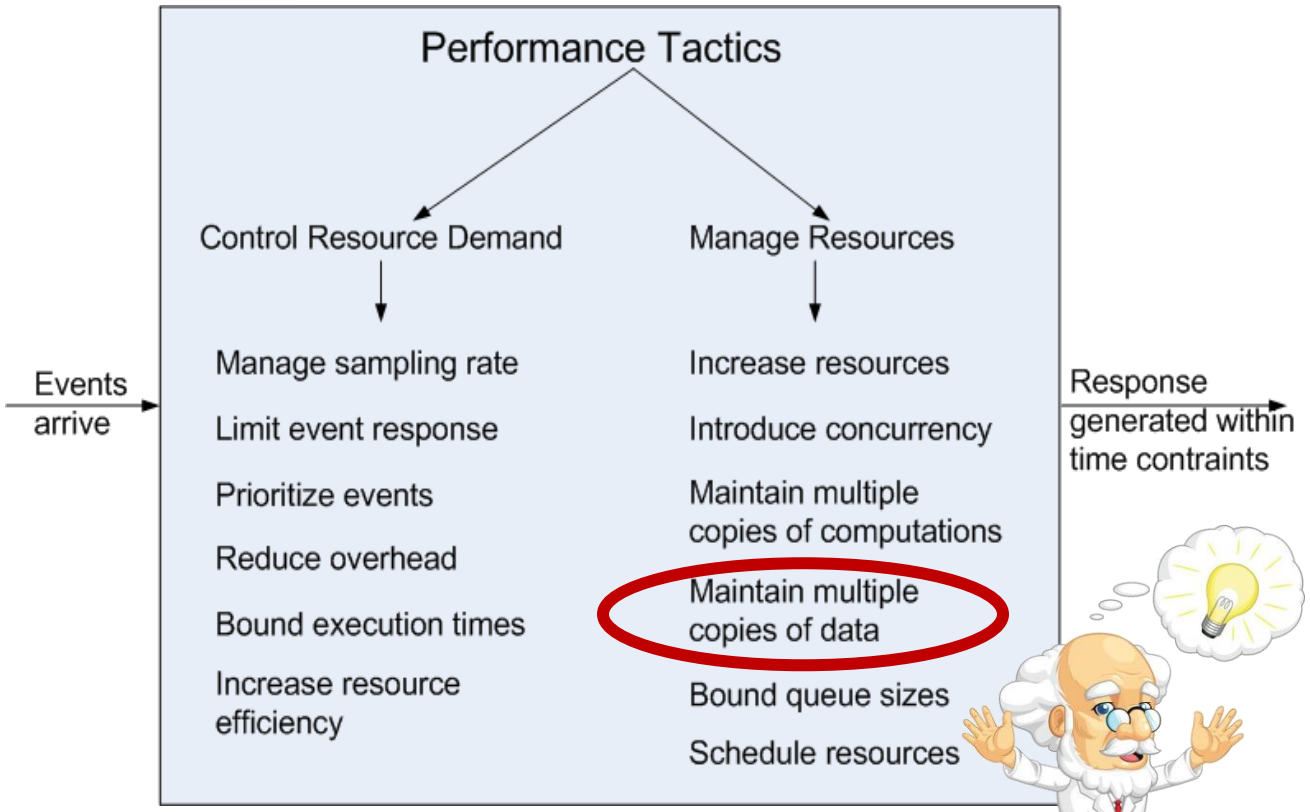
"Database per service" pattern

Pub/sub pattern!



60 per hour per client

3s per request



This solution is called:
CQRS: Command-Query-Responsibility Segregation

Utrecht University

To summarize

- Analysis
  - **Formal analysis techniques to ensure “correctness”**
  - **Thought experiments**
  - **Back-of-the-envelope analysis**
  - **Experiments, simulations and prototypes**
- Critical assessment of quality attributes
  - **Step 1: create quality scenarios (context views)**
  - **Step 2: solicit tactics**
  - **Step 3: apply tactics to create candidate solutions**
  - **Step 4: Choose a candidate & document your decision**

Utrecht University

For now: Assignment time

- Prioritize quality attributes and scenarios
- Document and Go through your decisions:
  What QAs did you analyse for the options?
  Revisit your decisions, and extend the analysis
- Which tactics you choose and why?

Utrecht University

For today

- 13:15 – 13:45: Quality attributes & tactics
- 13:45 – 14:45: Defining QAs & scenarios in your assignment
- 15:00 – 15:45: Analysis of Quality attributes
- 15:45 – 16:50: Work on your assignment
- 16:50 – 17:00: Wrap up

Utrecht University

Next Lecture: Thursday By Claudio

- Queuing networks
- Read CHP 14, 4-13
- Read Paper:

(M) S. Klock, J.M.E.M. van der Werf, J.P. Guelen and S. Jansen (2017). Workload-Based Clustering of Coherent Feature Sets in Microservice Architectures. In International Conference on Software Architecture, pp. 11-20.

Utrecht University

DISCLAIMER

The information in this presentation has been compiled with the utmost care, but no rights can be derived from its contents.