



Best Practice Tour 2023

Stockholm
Brussels
Munich
Zurich
Frankfurt

Paris
Warsaw
Berlin
Budapest
Gothenburg



The past, present, and future of architecture

Bob Hruska

Principal Consultant

Workshop 1 - Best Practice Tour 2023

March 23, 2023



Architects love to hate documenting architecture...

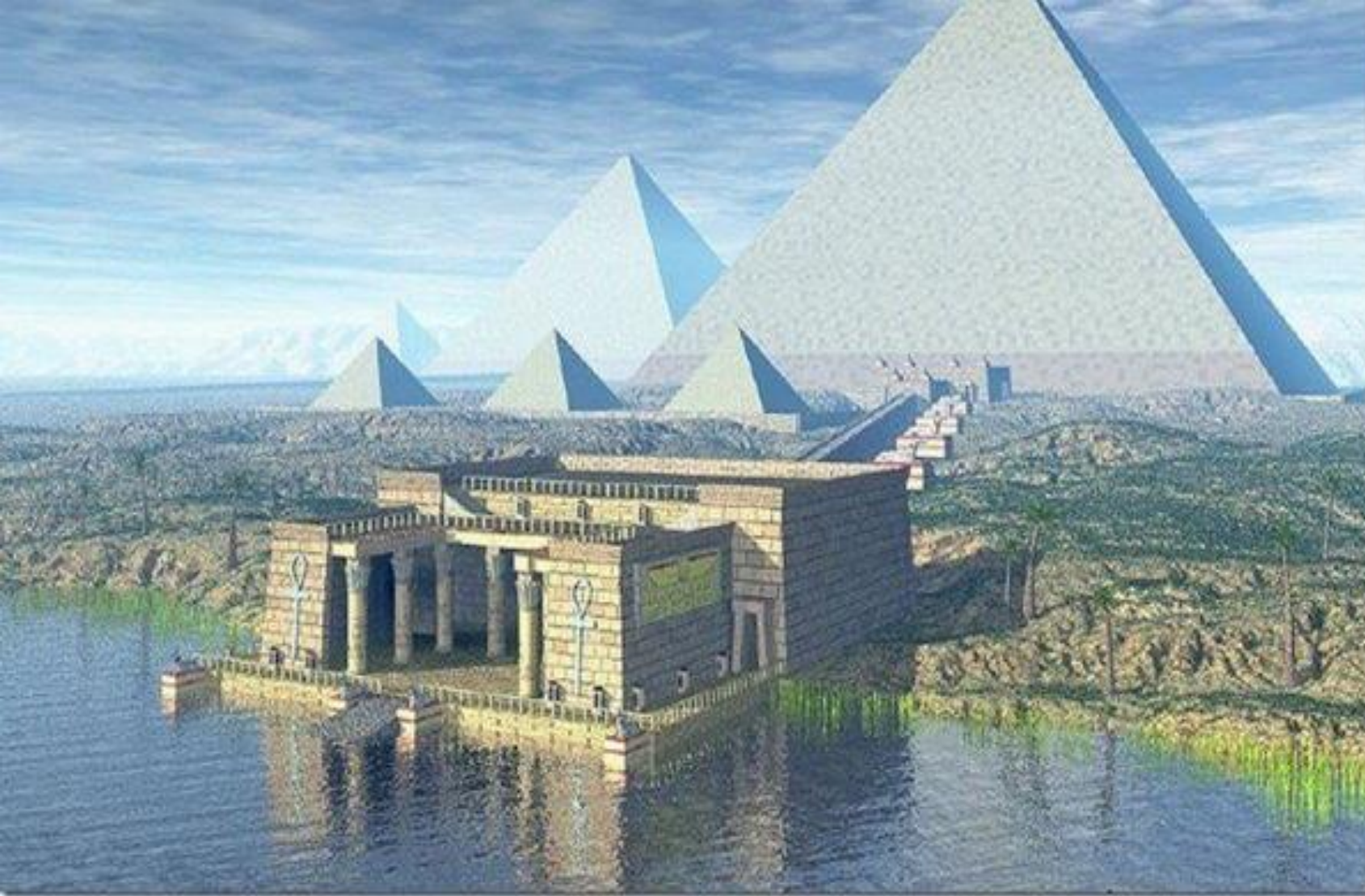
... they look at it as a necessary evil.

Bob Hruska

- OMG Certified UML® Professional™
- 20+ years' experience in software and systems engineering in several industries
- Experienced in the Capability Maturity Model Integration (CMMI) appraisal journey and with development of the New Product Introduction (NPI) process.
- Contributing to an institutionalization of cybersecurity as a part of a system development lifecycle.

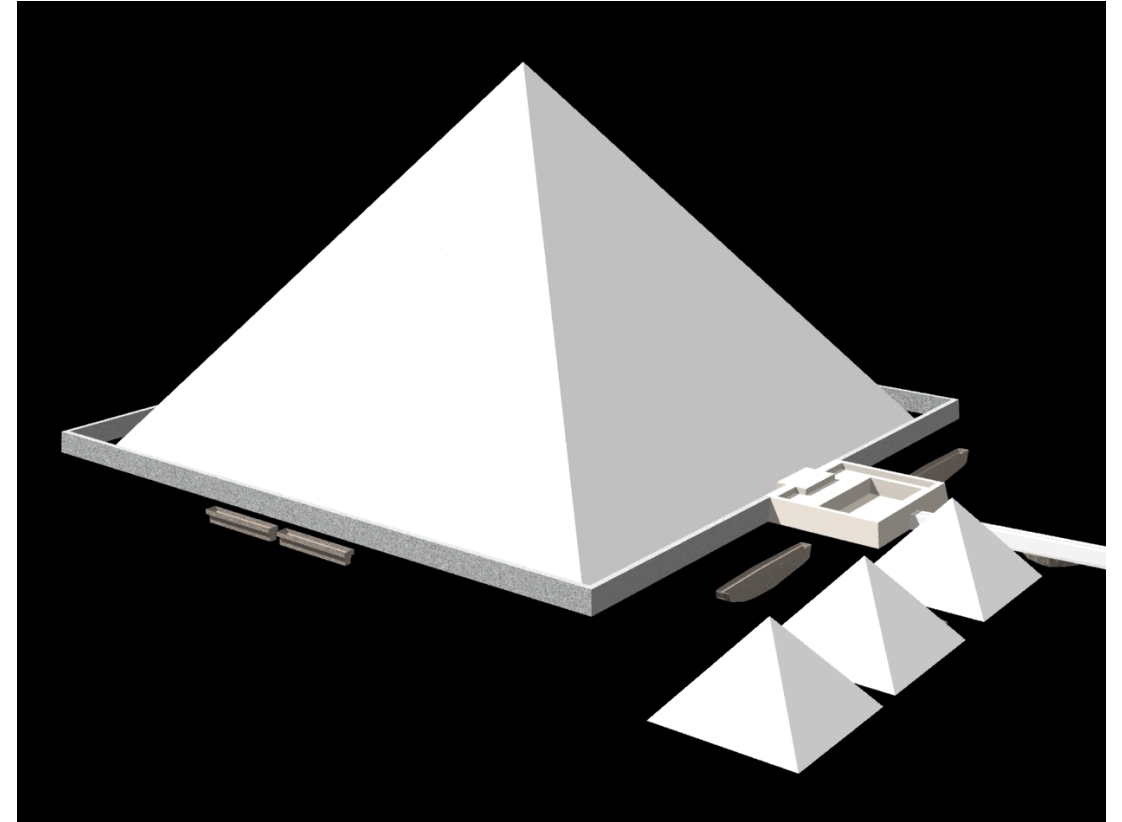
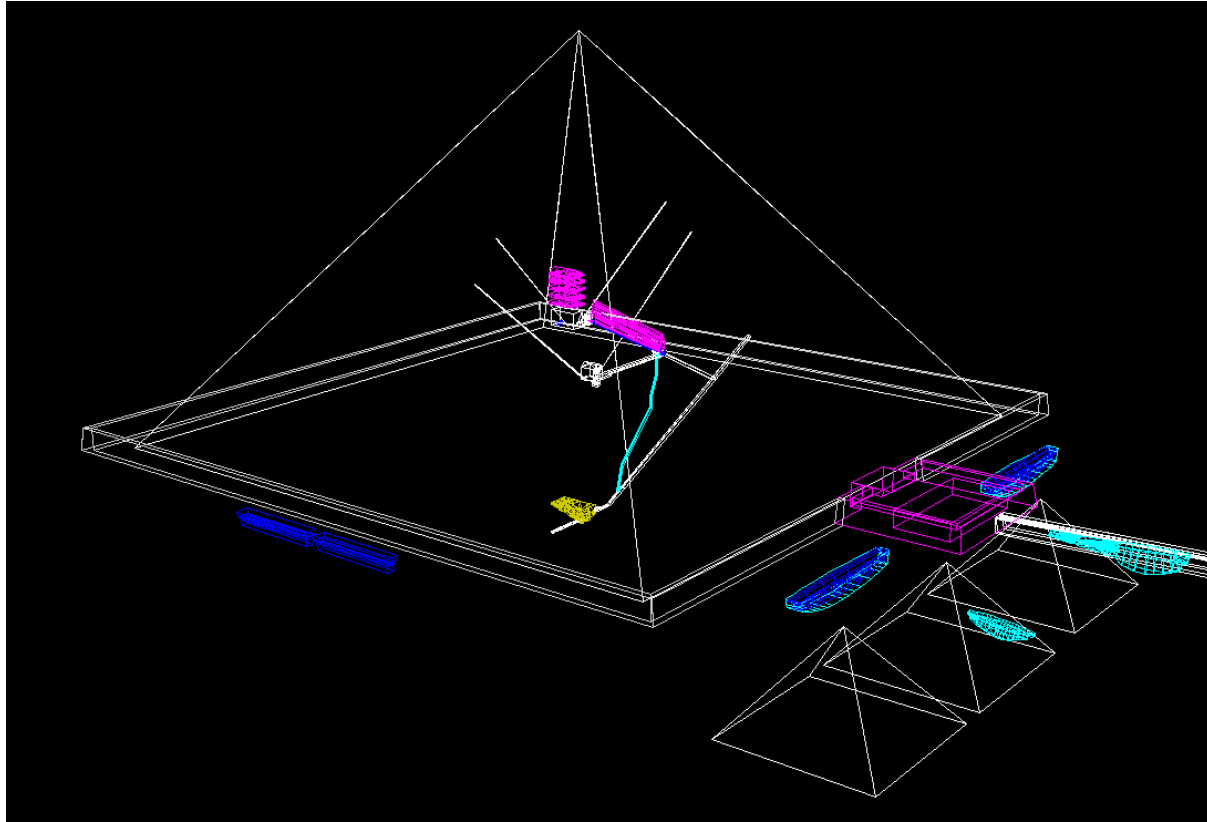
Let's address the questions:

- Why should we care about architecture at all?
- Why is documenting software architecture so difficult?
- What are the characteristics of good architecture?
- What role does architectural documentation play in a project's lifecycle?
- What are the architectural drivers?
- What are the benefits of the model beyond the modeling itself?



The Great Pyramid of Giza

There is no evidence, no ancient plans...



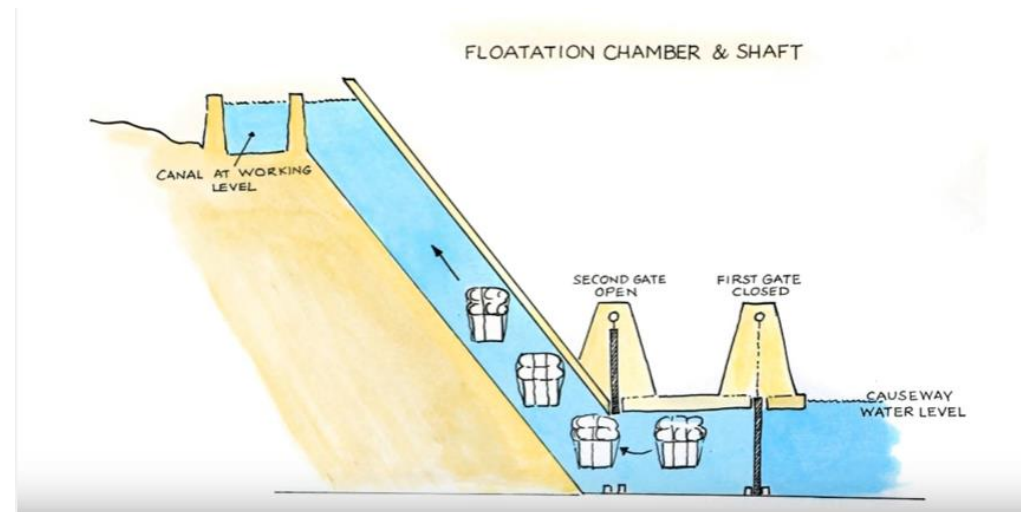
Giza Plateau Computer Model - University of Chicago

How were the Pyramids built then?

- The Ramp Theory



- The Water Shaft Theory



Source: <https://www.contiki.com/six-two/how-were-the-egyptian-pyramids-built/>

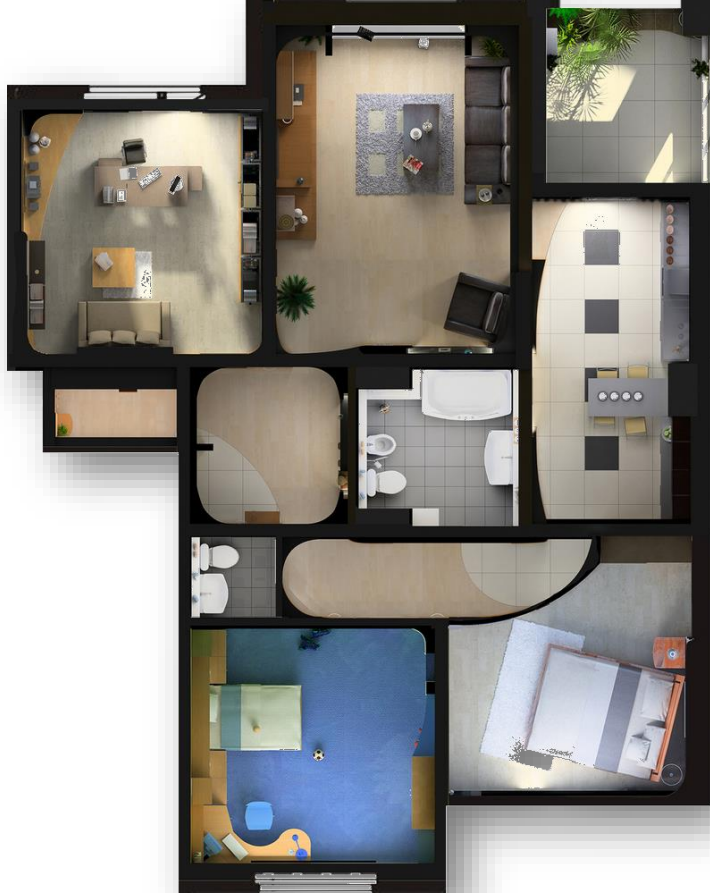
The oldest architectural plan



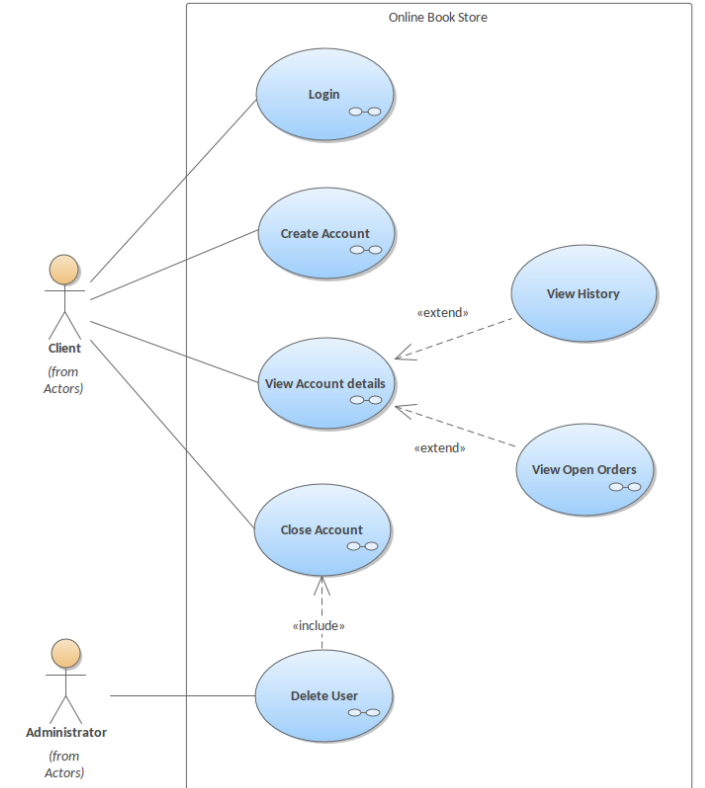
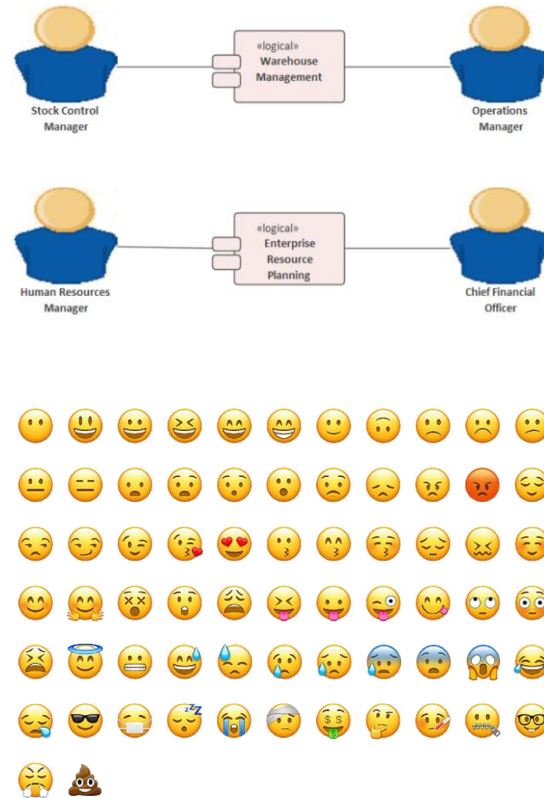
- Discovered in Iraq and dating back to the Mesopotamia civilization (8000-2000 B.C.)

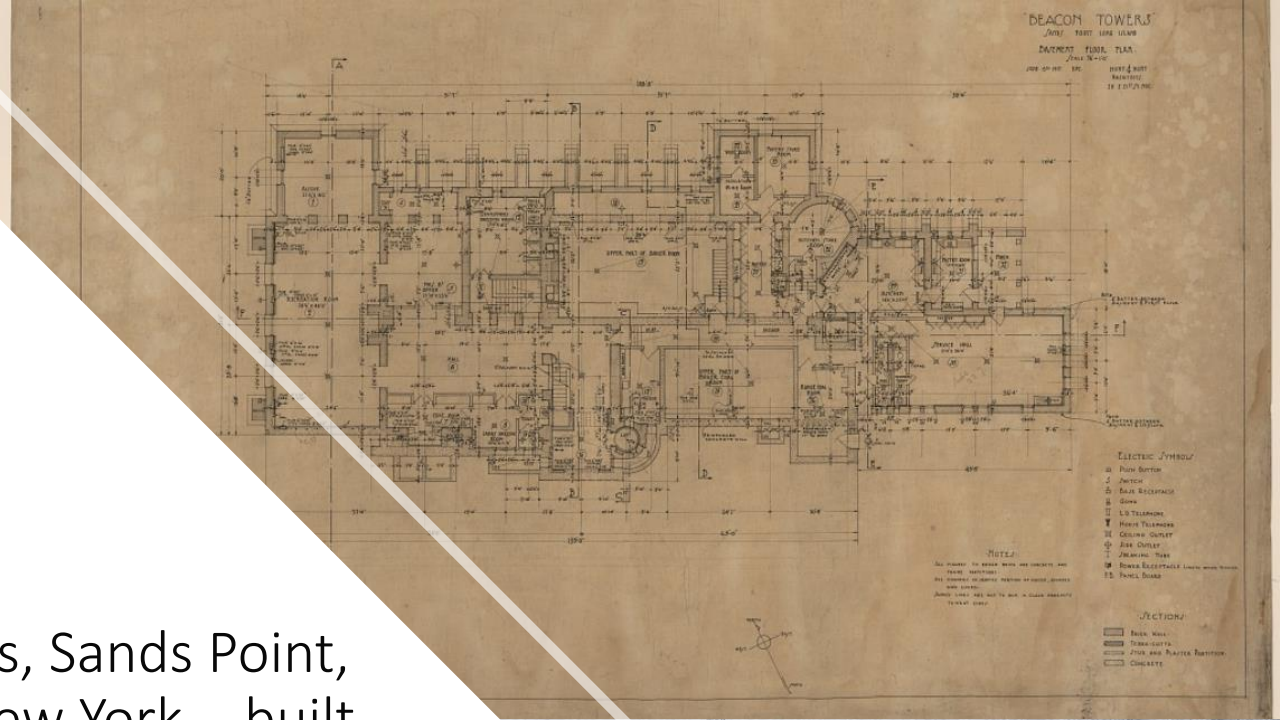
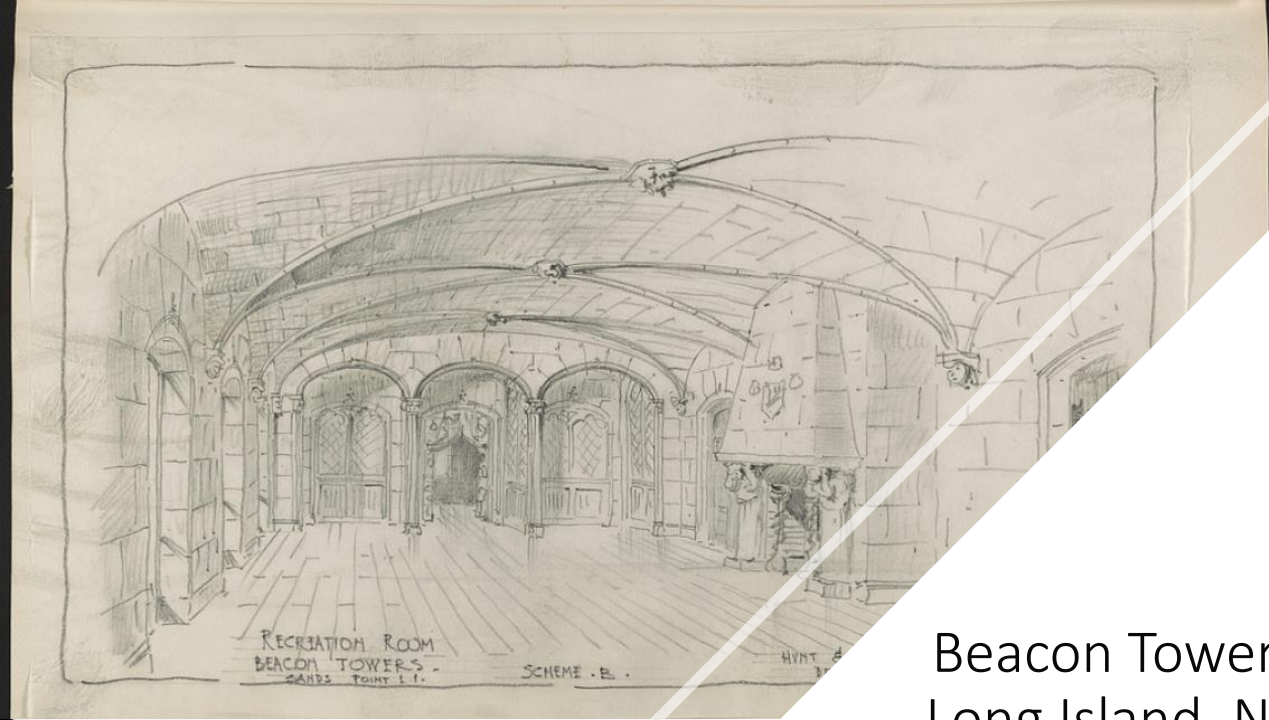
Thousands of years later

modern houses floor plan



4000 years and we're back to the same language 😊

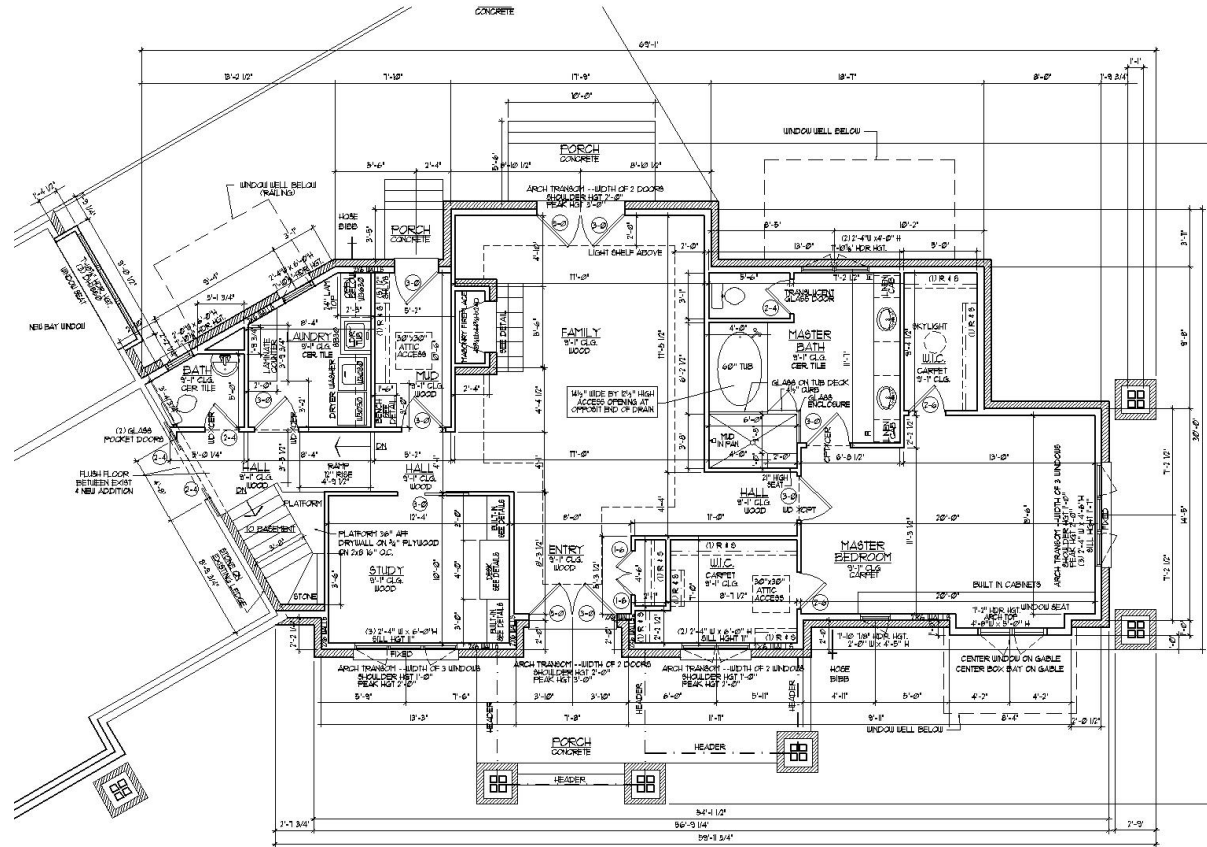




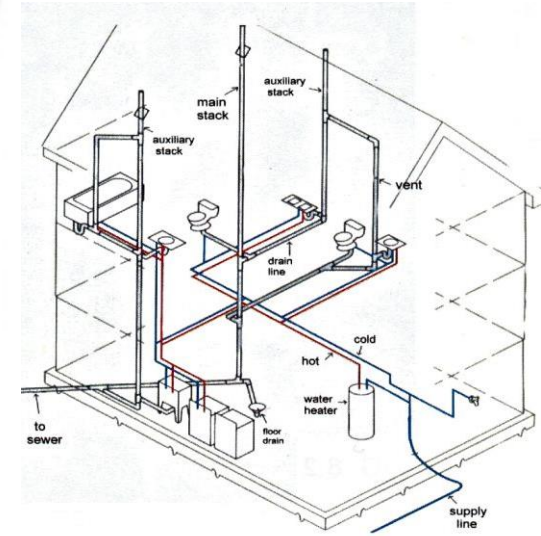
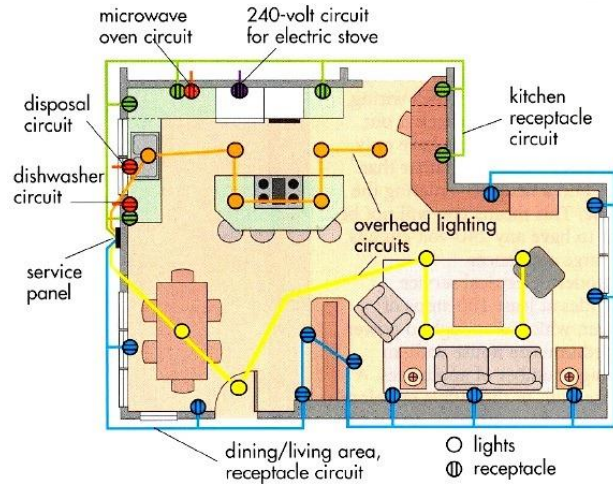
Beacon Towers, Sands Point, Long Island, New York. - built from 1917 to 1918

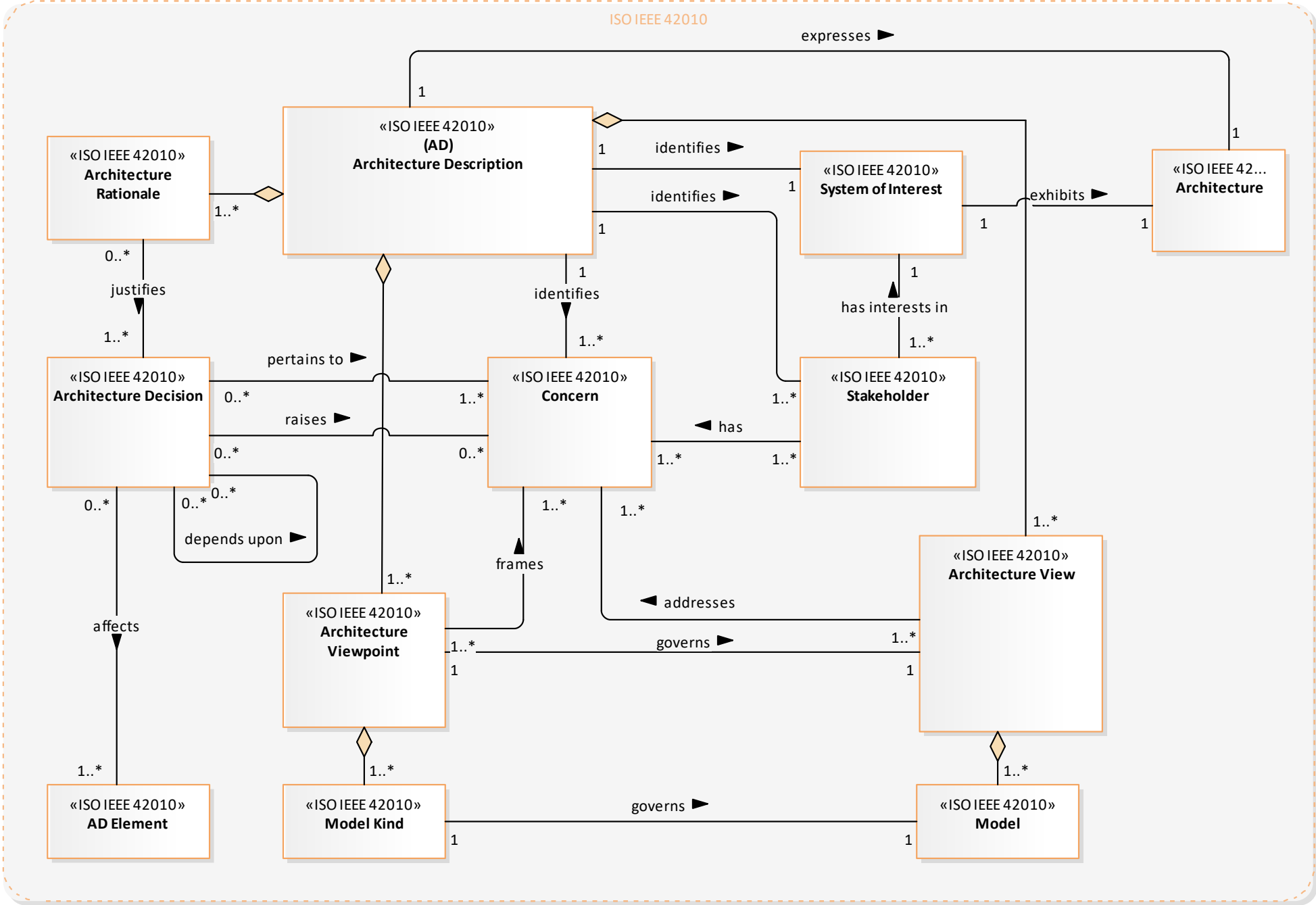


The point is to address different concerns





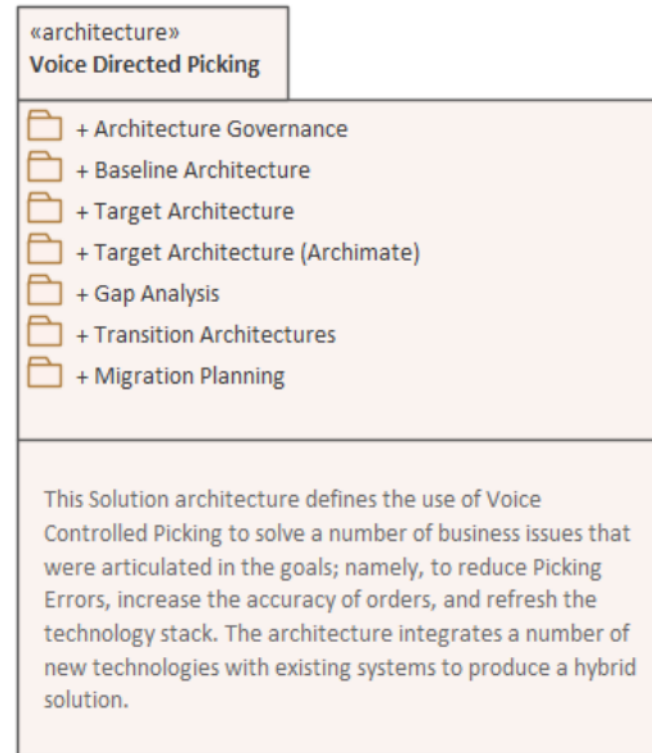




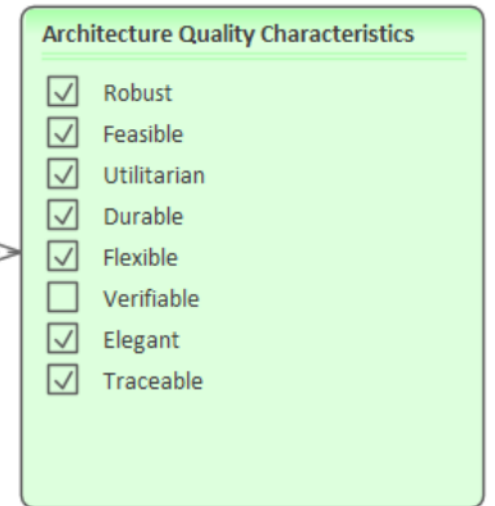
Characteristics of Good Architecture

- Durability (Firmatis)
- Utility (Utilitas)
- Beauty (Venustatis)

*The Roman architect Vitruvius defined three characteristics of good architecture in his treatise **De Architectura** more than 2,000 years ago.*

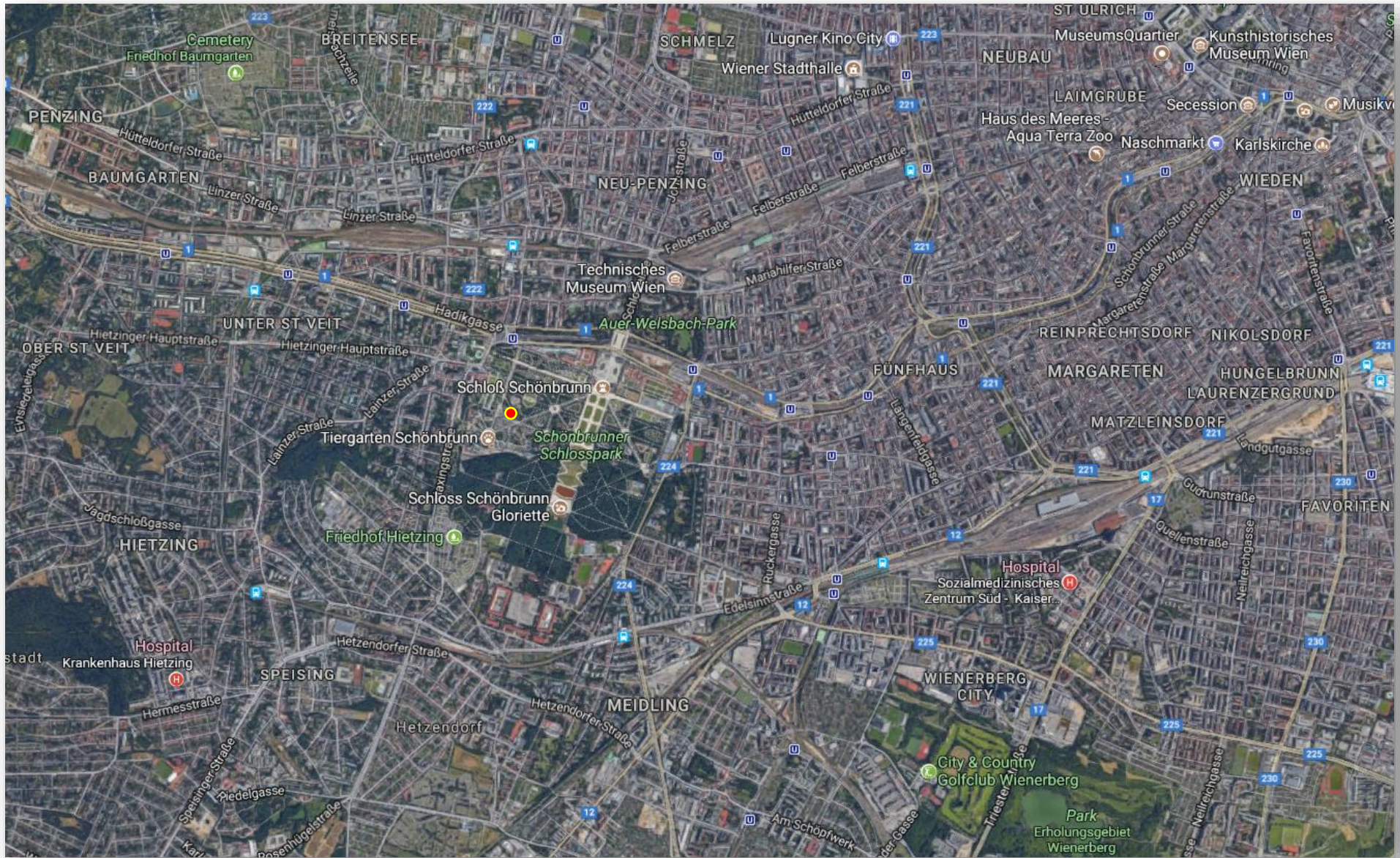


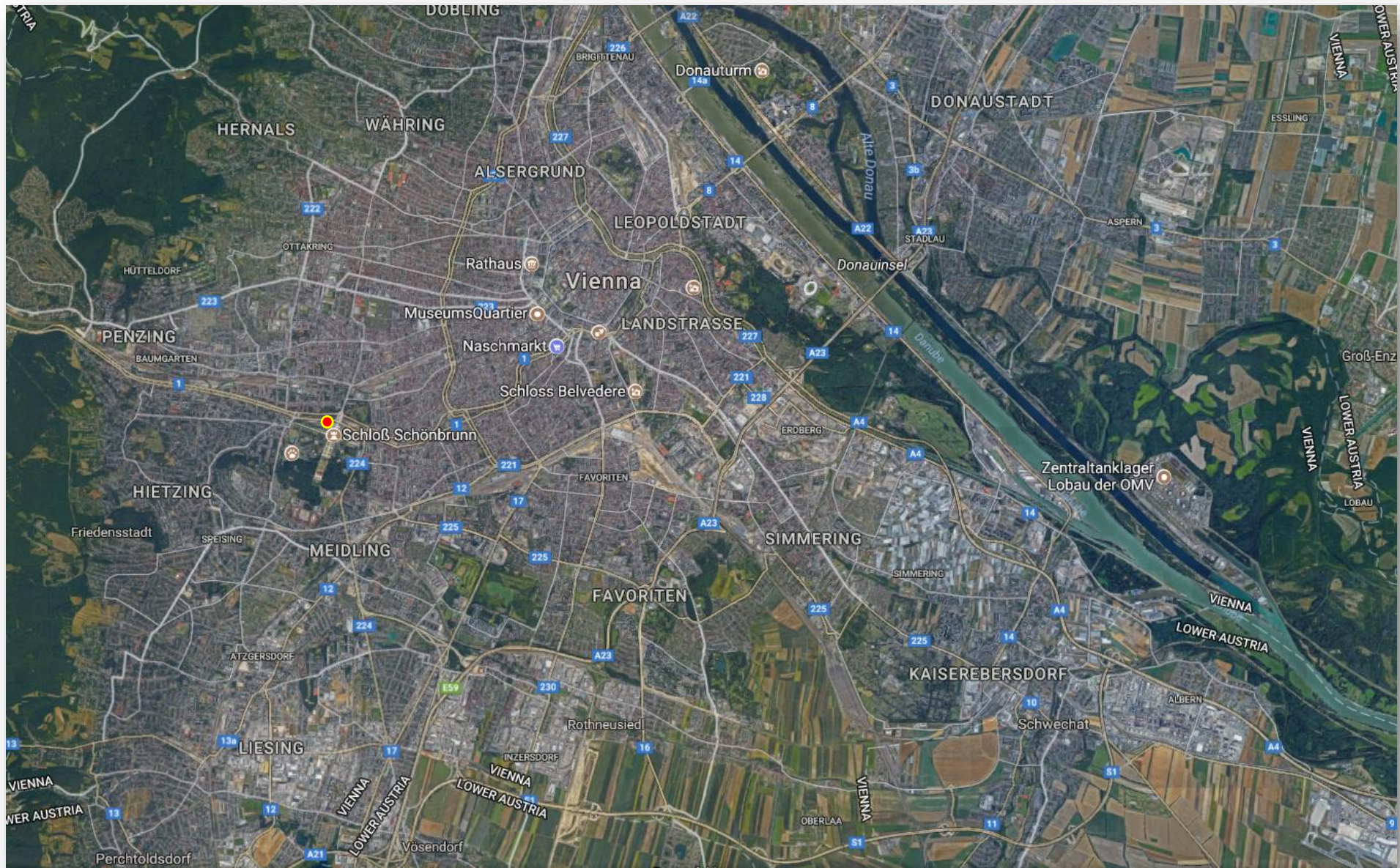
(from Solution Architectures)

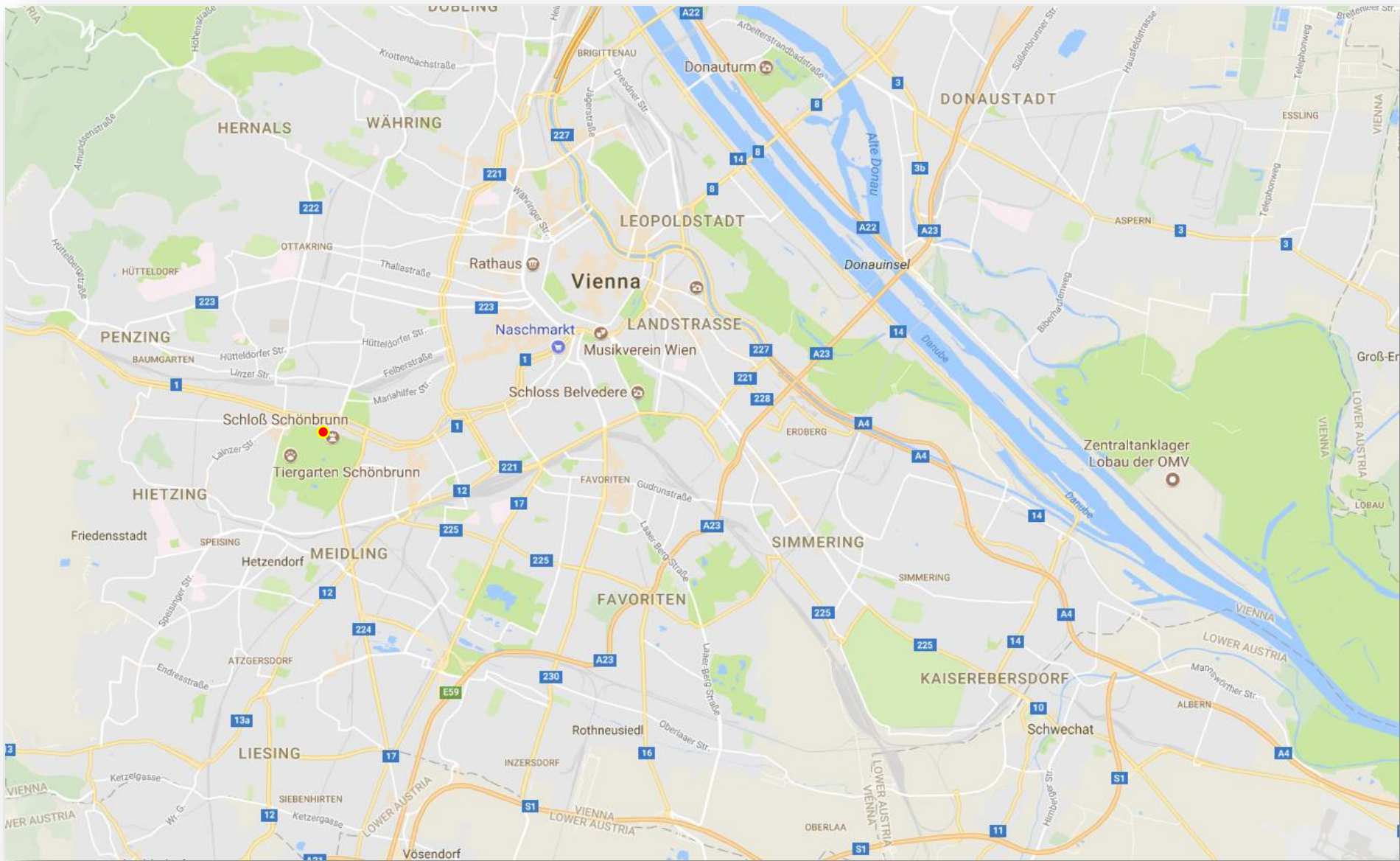


This diagram demonstrates an editable checklist that can be used to give an Architecture Health Check at a glance using a number of checkboxes that can be set to indicate whether the Architecture has the particular quality.

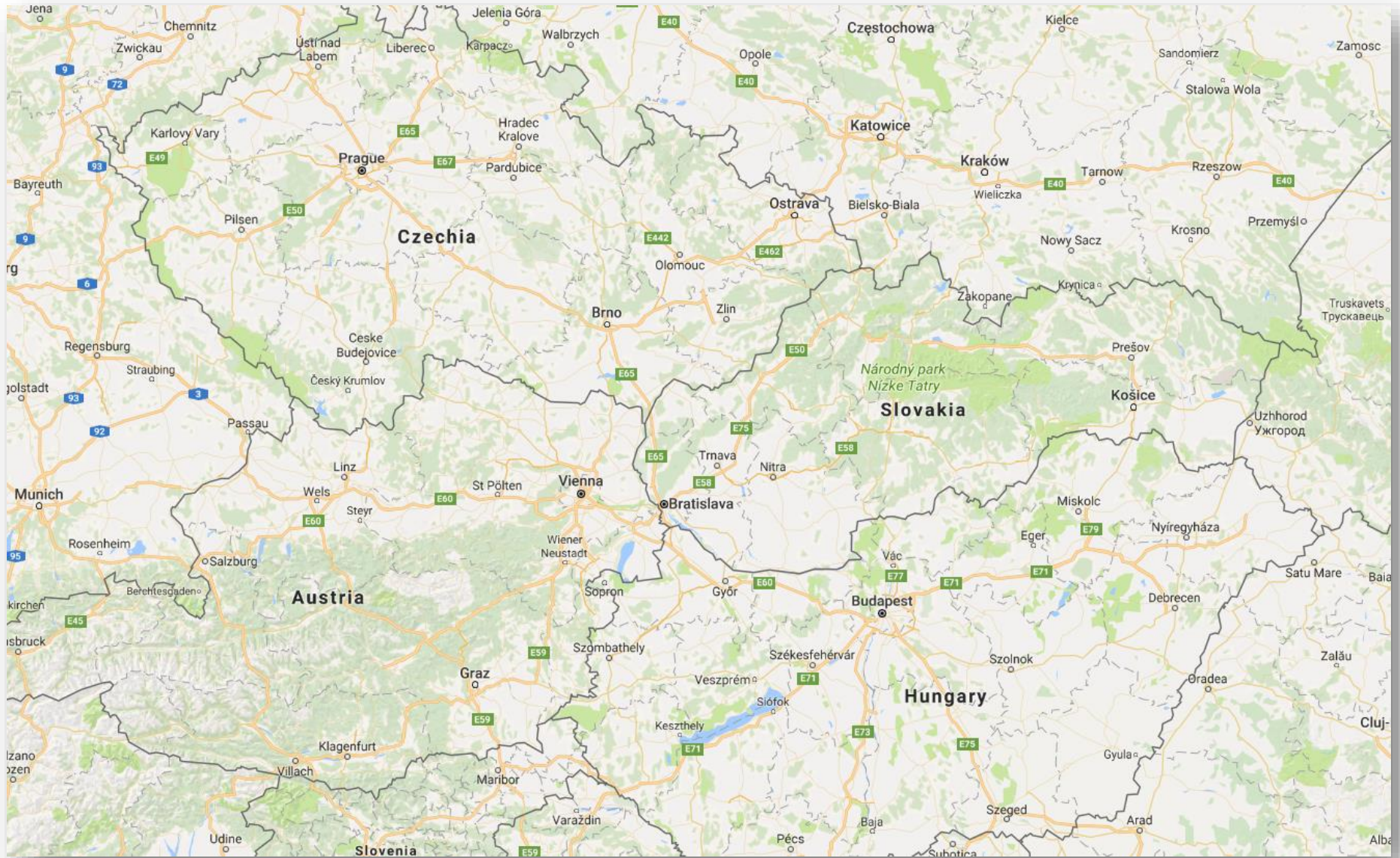


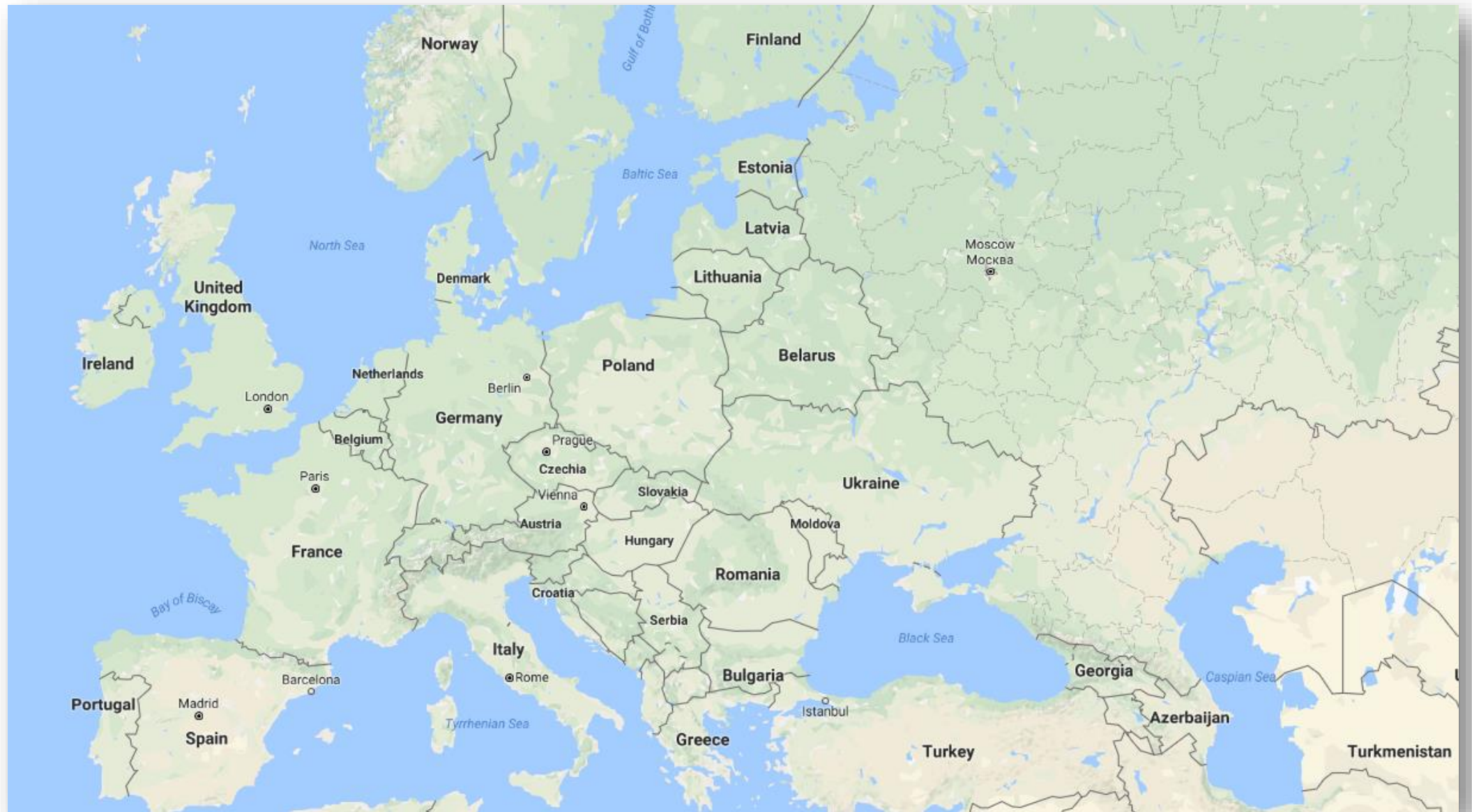






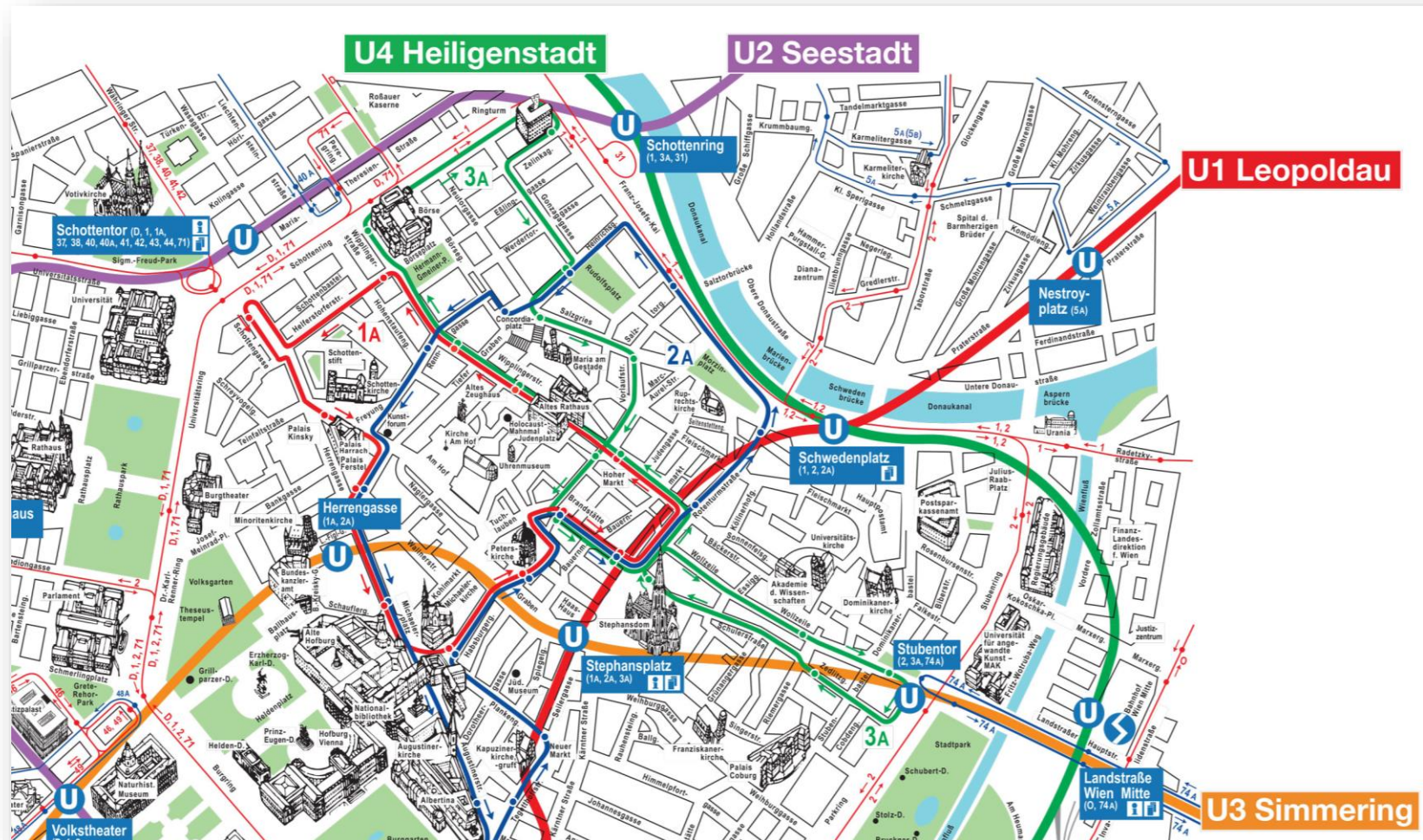
Reality can be modeled in ways
that communicate spatial
information effectively



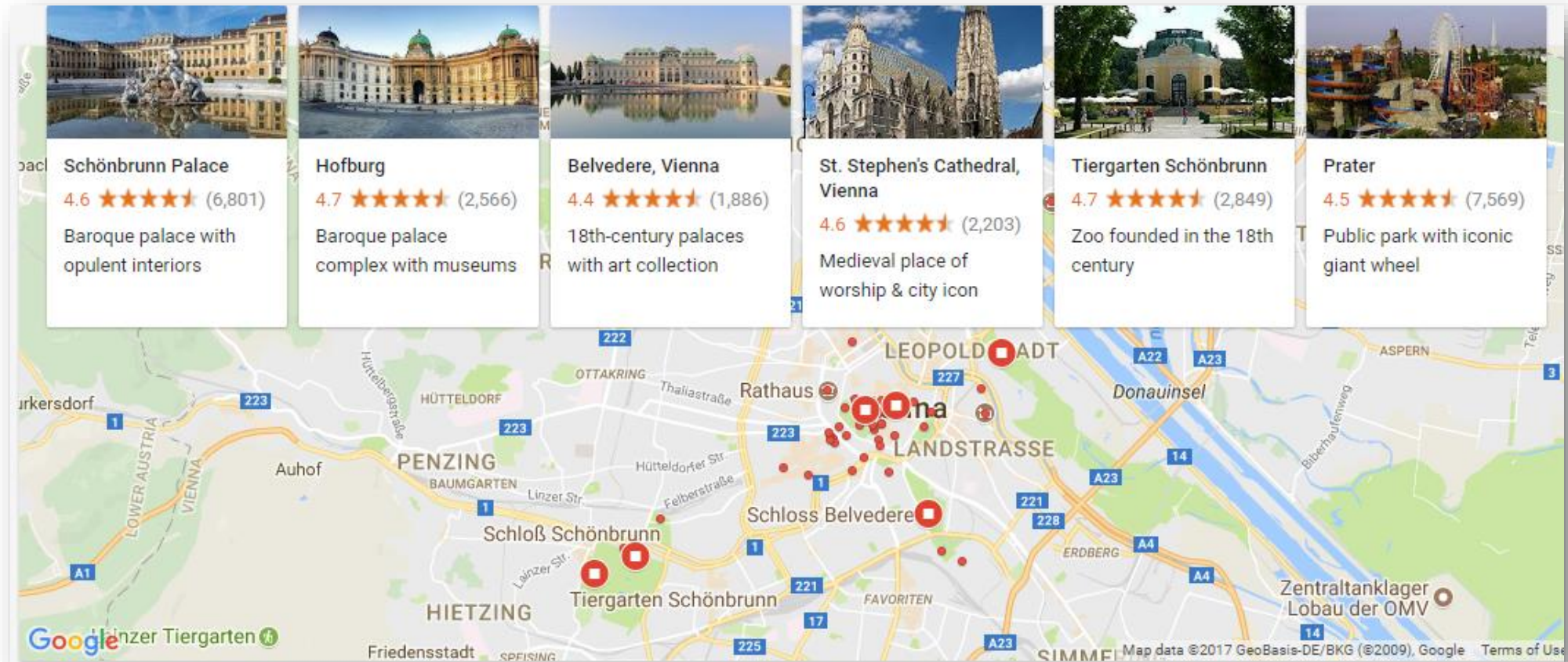


Enough detail to start
exploring

Very detailed and precise (public transportation, buildings, etc.)



Travel Guidebook (maps, POI, itineraries, etc.)



Architecture diagrams should be like maps...that help software developers navigate a large and/or complex codebase...

What do you see as the future of software architecture documentation?

Eoin: I hope that in the future we'll need very little software architecture documentation because we'll be able to see the architecture in the code and the running system! One of the reasons we need much of our architecture documentation today is because there's no way of representing architectural structures directly using the technologies we have at our disposal. I'd love to see our architectural constructs as first class implementation structures and then architecture documentation can evolve to capture decisions, rationale and analysis, rather than just capturing structures. On the way to this nirvana, I hope that work going on in the areas of DSLs and ADLs (architecture description languages) point the more immediate way forward, as we improve our description languages, on the way to working out how to embed the information right in the running system.

Grady: There is a lot of energy being applied today with regard to architectural frameworks and methods: TOGAF, NEA, DoDAF, MoDAF, FSAM, Zachman, and so on. The good news is that there is a vibrant dialog going on with regard to these frameworks and methods - but I expect there will be a shakeout/simplification over time.

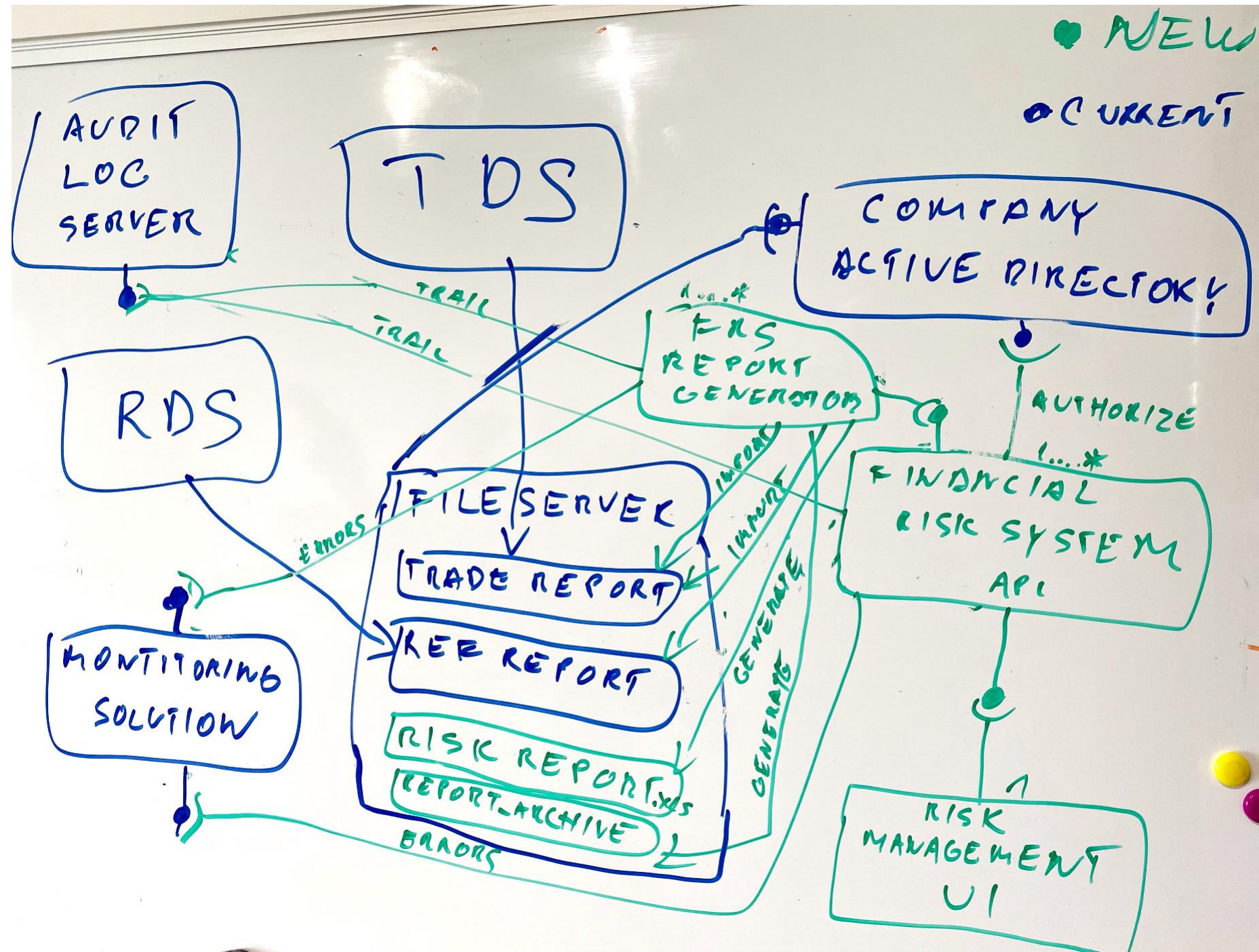
Paulo: The software architecture discipline is fairly new. There is a long path ahead until we get to a point where an architect creates architecture documentation that is readily understood by a developer who has never worked with that architect. The way to get there is to let new architects learn software architecture at school rather than try-and-error in the battlefield. This education includes proper ways to represent the software architecture for other people's consumption. Important initiatives in the direction of good software architecture education are: the work of Grady Booch on the handbook of software architecture and the publications and curriculum developed at the SEI.

Len: The ideal development environment is one for which the documentation is available for essentially free with the push of a button. This will require an integrated development, requirements management, and project management environment. Although this will be a long time coming, it provides a worthy goal to strive for.

...14 Years Later

NEW

CURRENT



UNIX BOX

RS
TRANSPORT + LOGIC

JBOSS INSTANCE

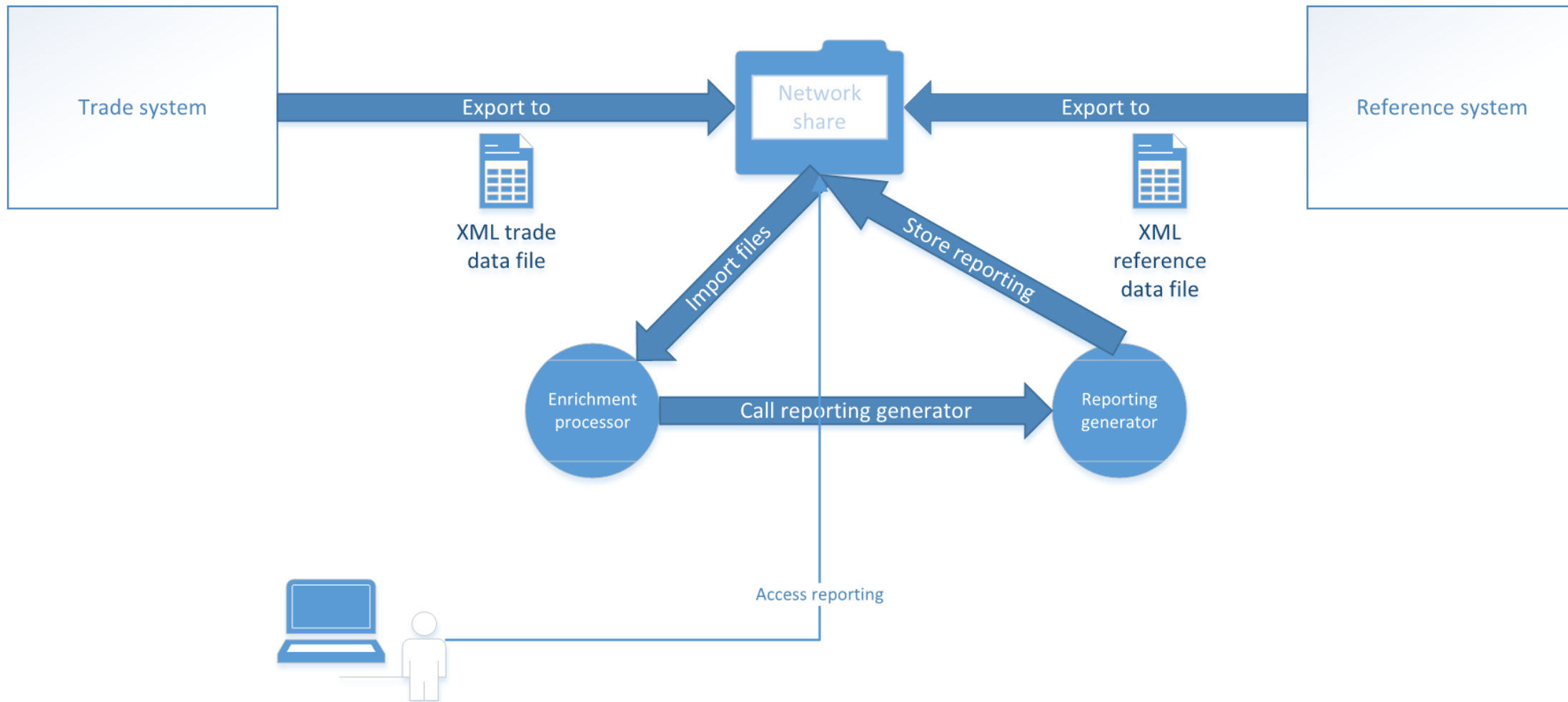
ERROR

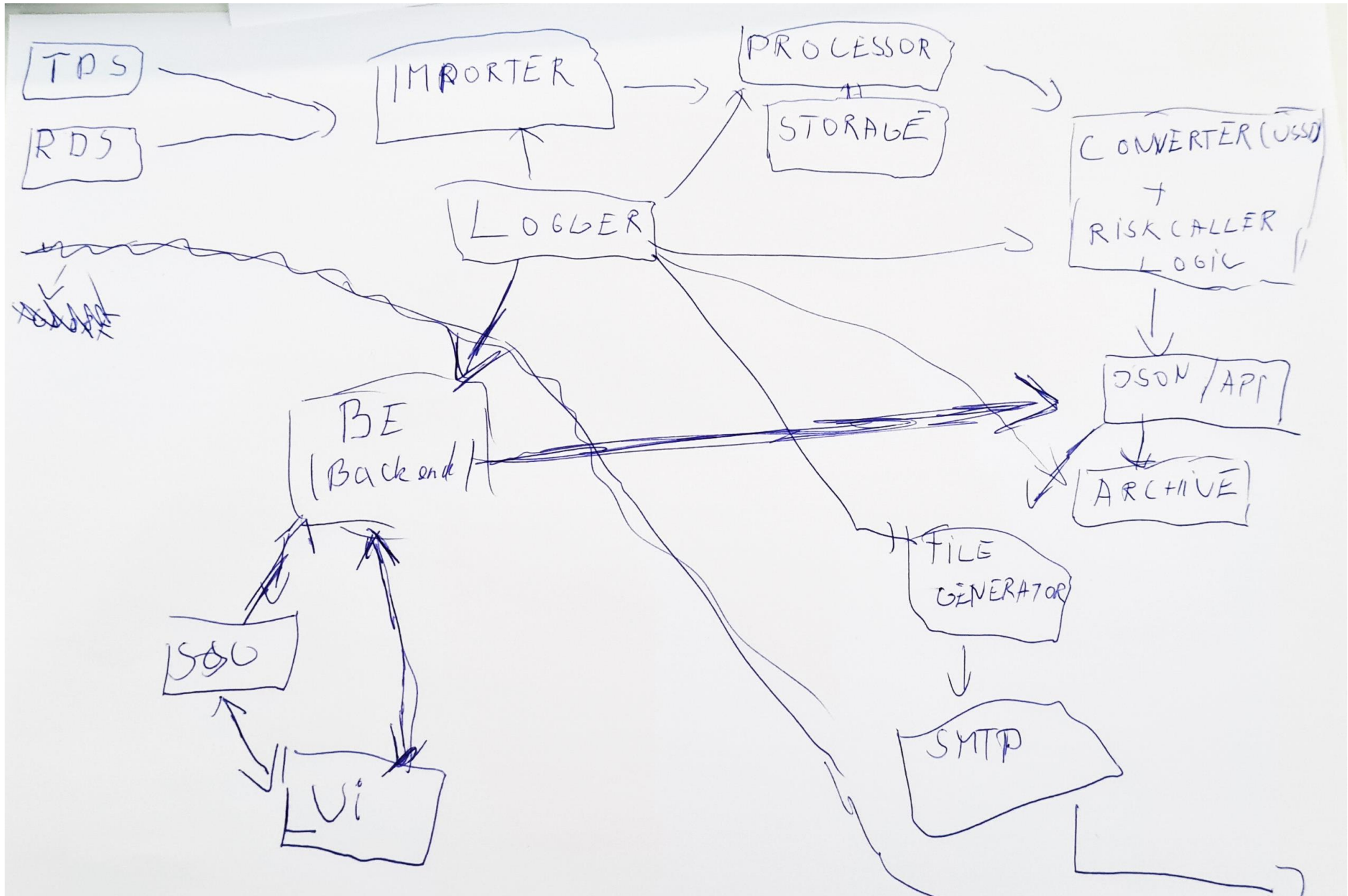
JBOSS INSTANCE (WEB CONTAINER ONLY)

WINDOWS BOX

MS
SQL
SERVER

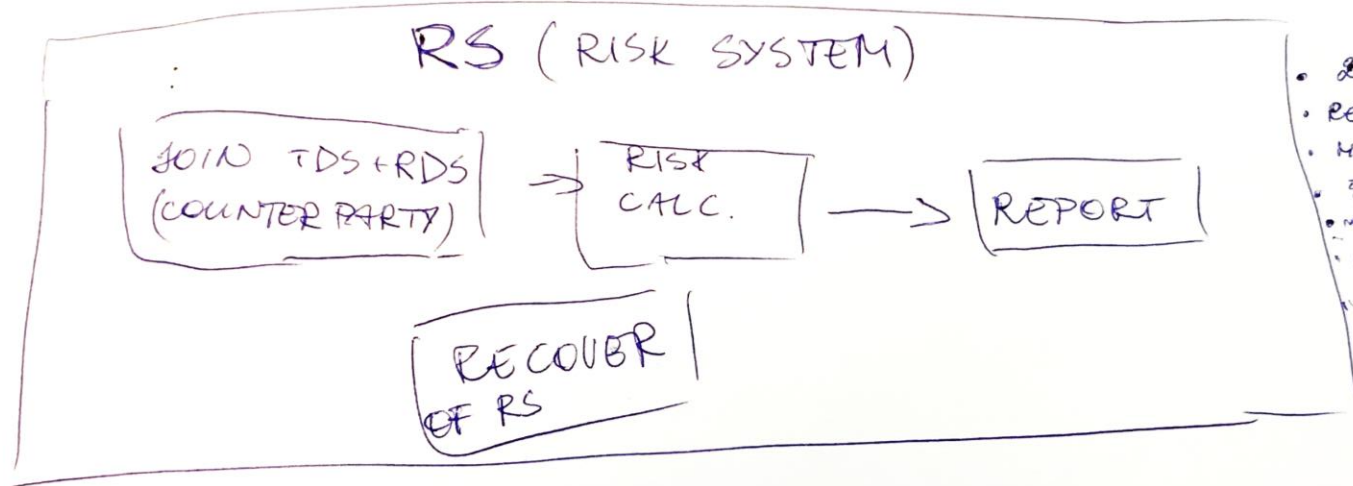
MS
REPORTING SERVICE



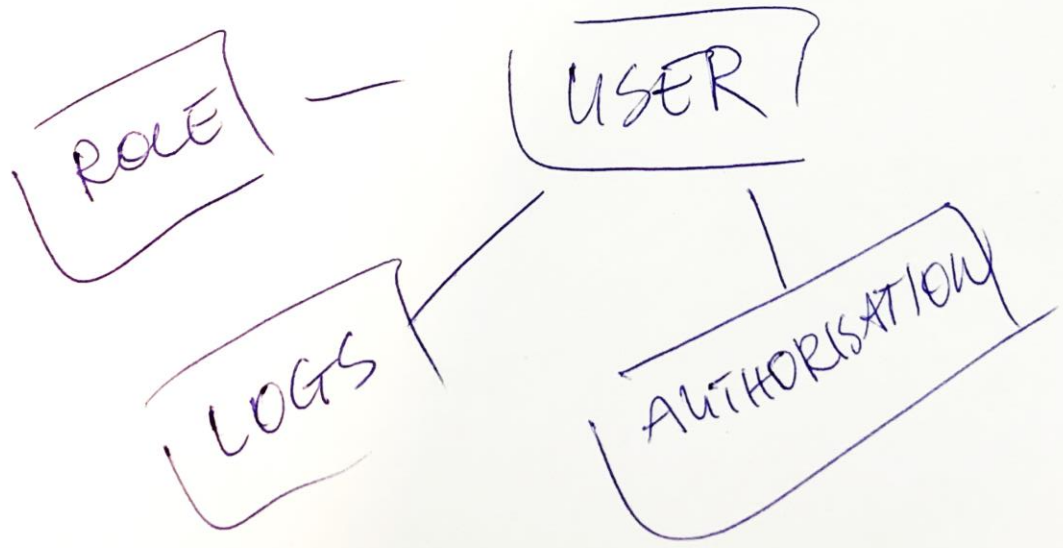


TDS

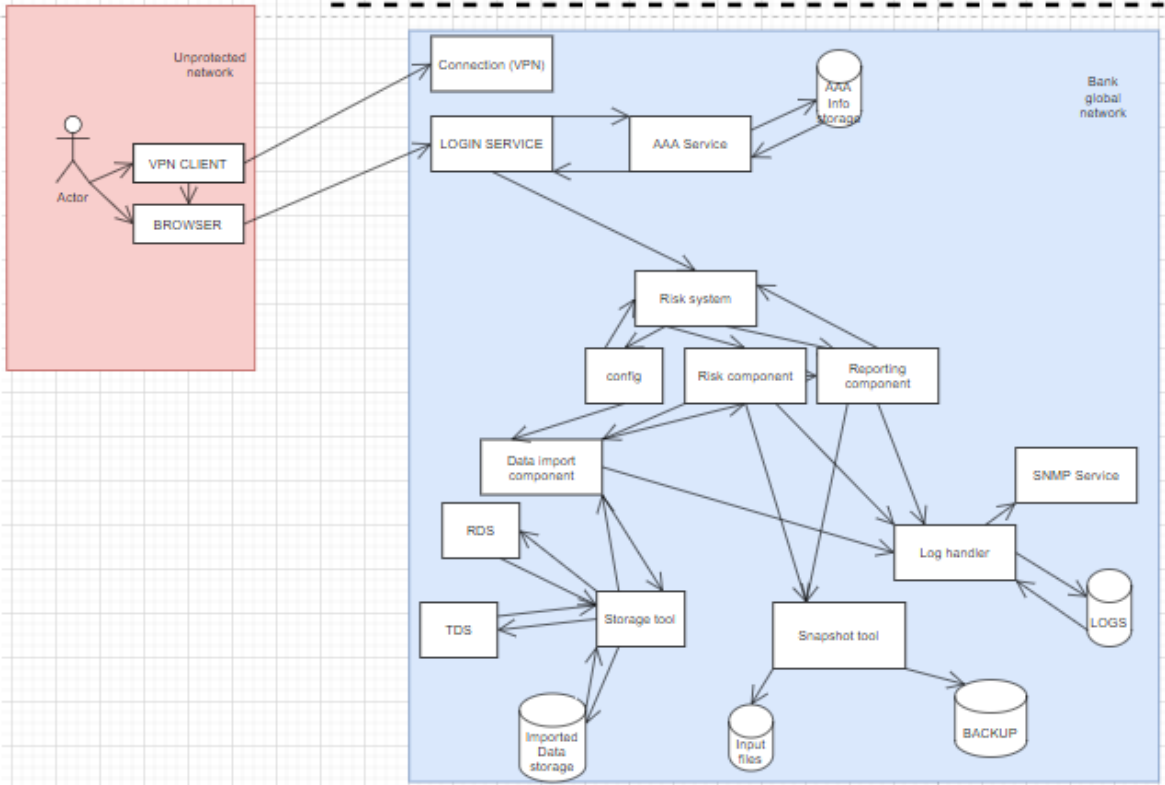
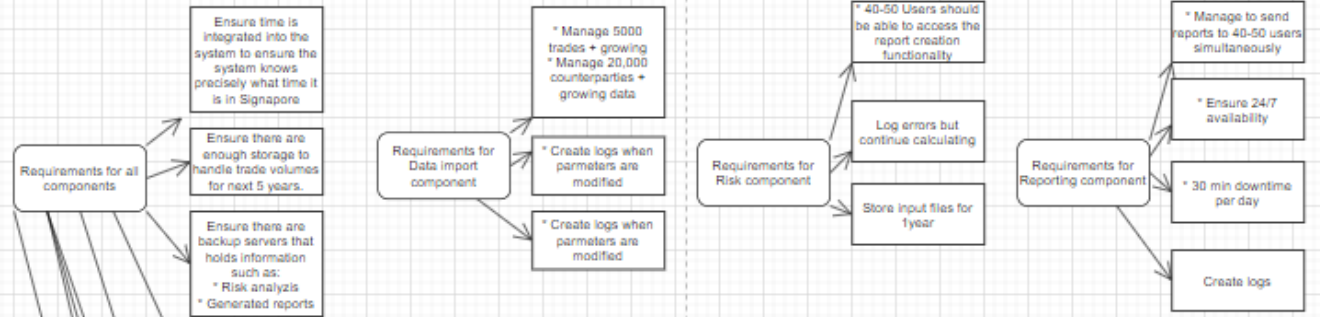
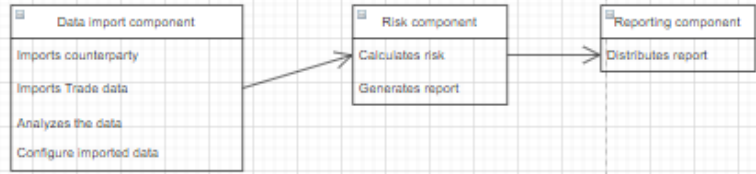
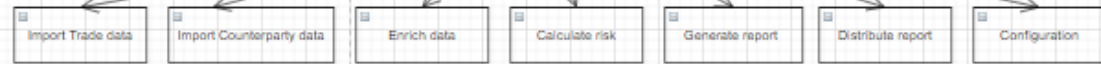
RDS

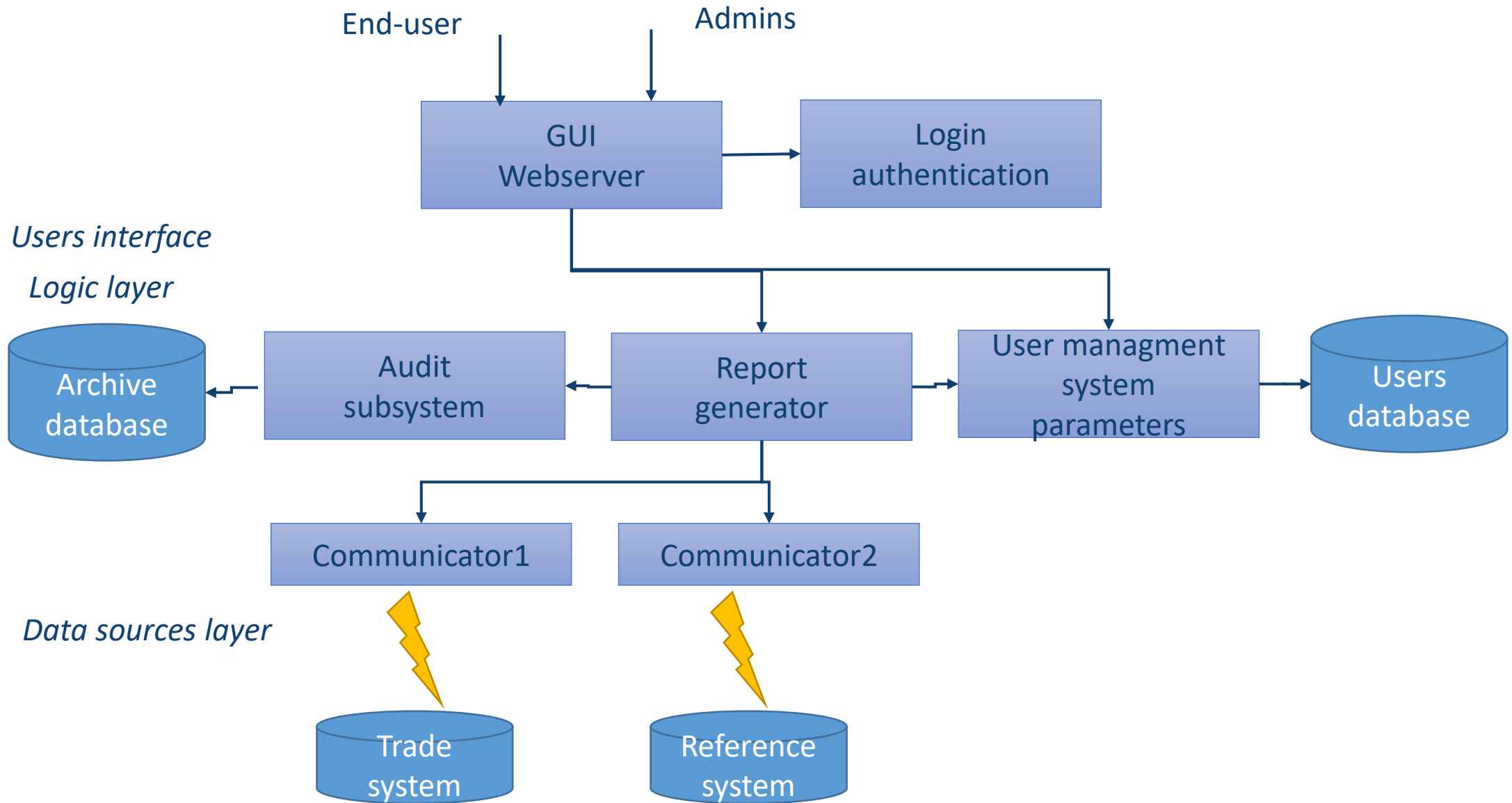


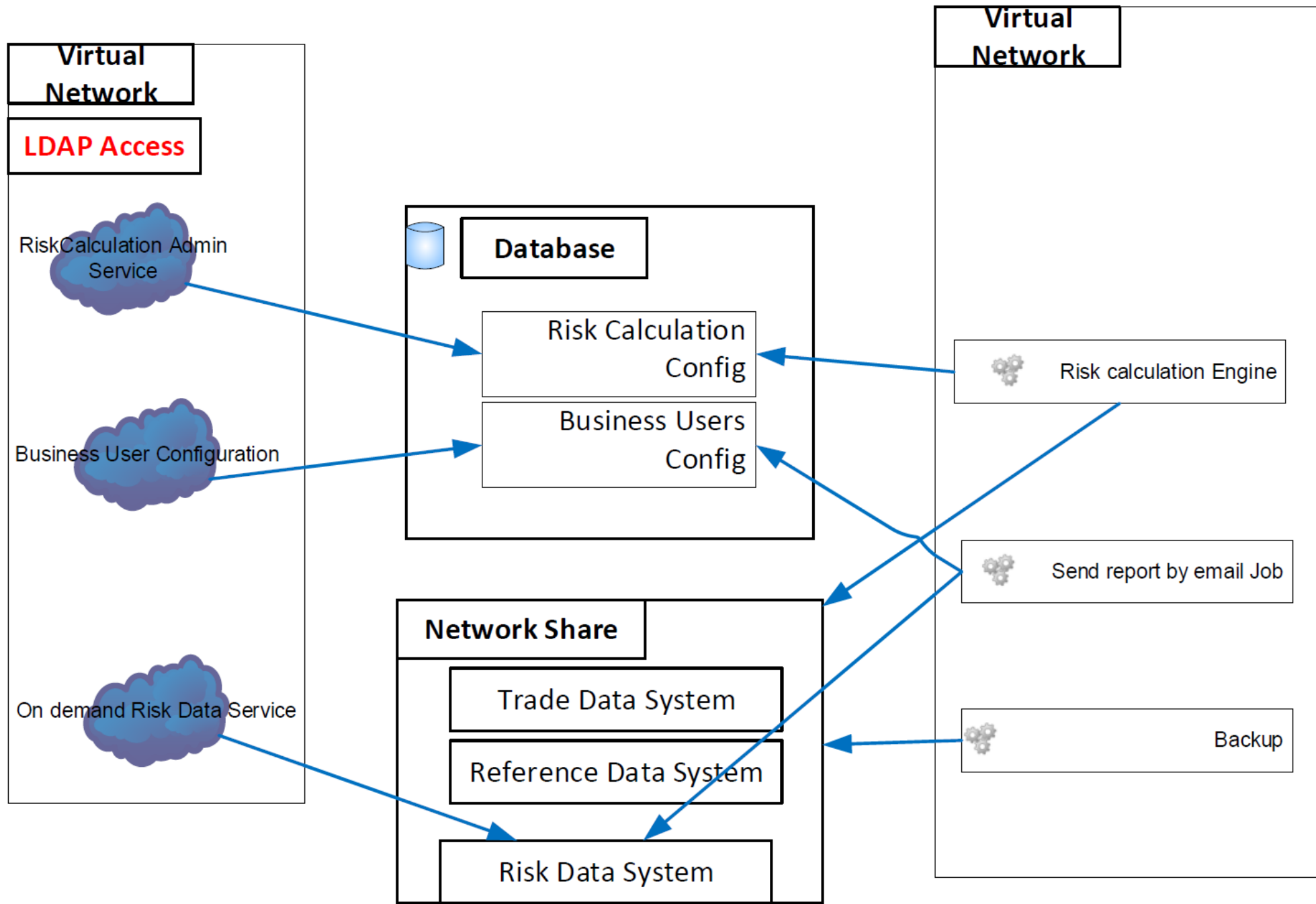
- 24/7
- READY BEFORE 9 AM (SING)
- MAX DOWNTIME . . .
- 5 YEARS TRADE VOLUMES
- 5000 TRADES
- ~20000 COUNTERPARTIES



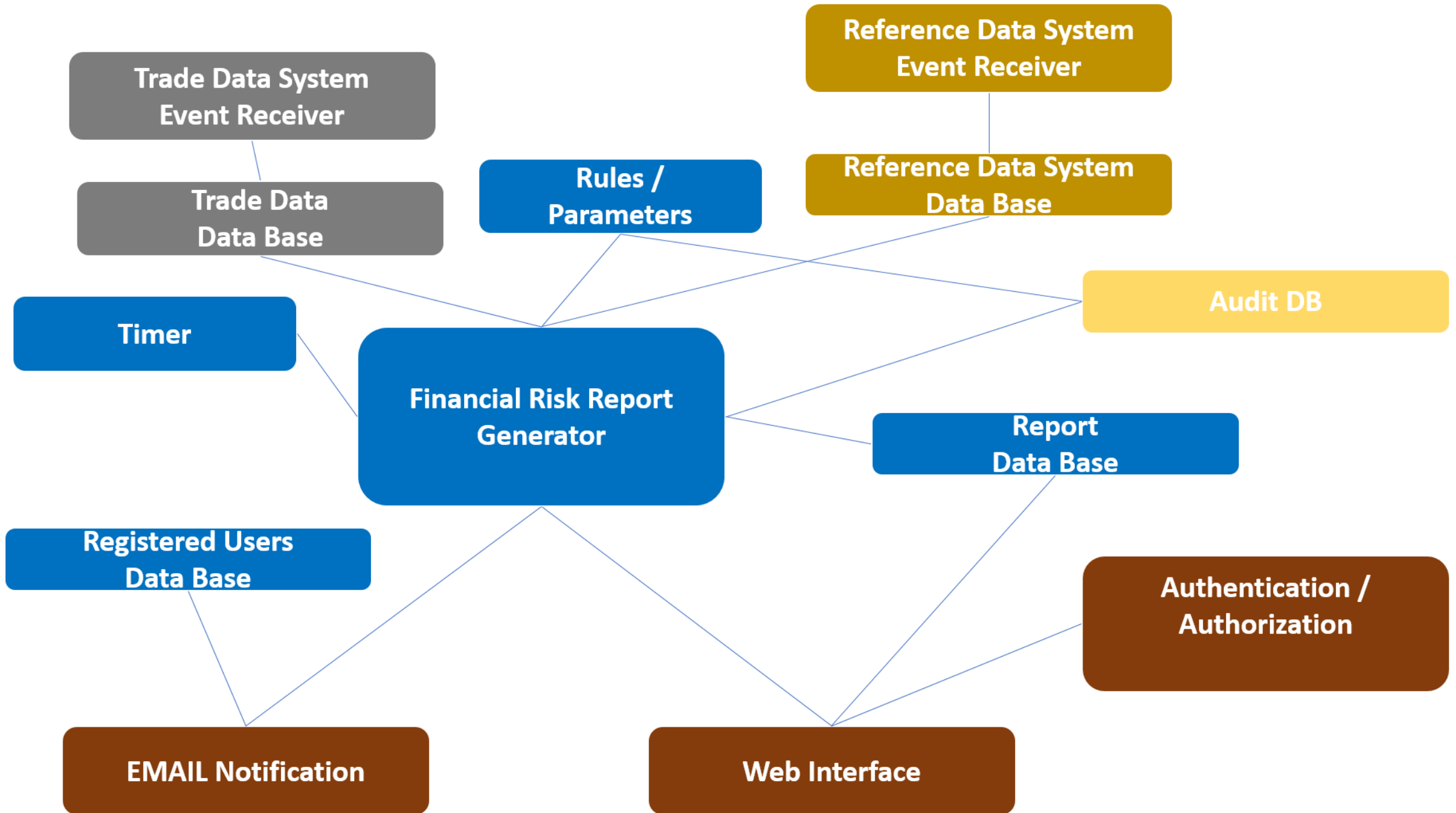
Risk System







Financial Risk System – Software Components



Params

Calcs

~~Params~~

~~ret - calc~~

~~Calc - bus~~

~~Calc - Risk outputs~~

~~Flow App Log~~

EH?

App
~~Flow Log~~
Date
- Risk Cell
- calcs
- par

Params - Riskolit

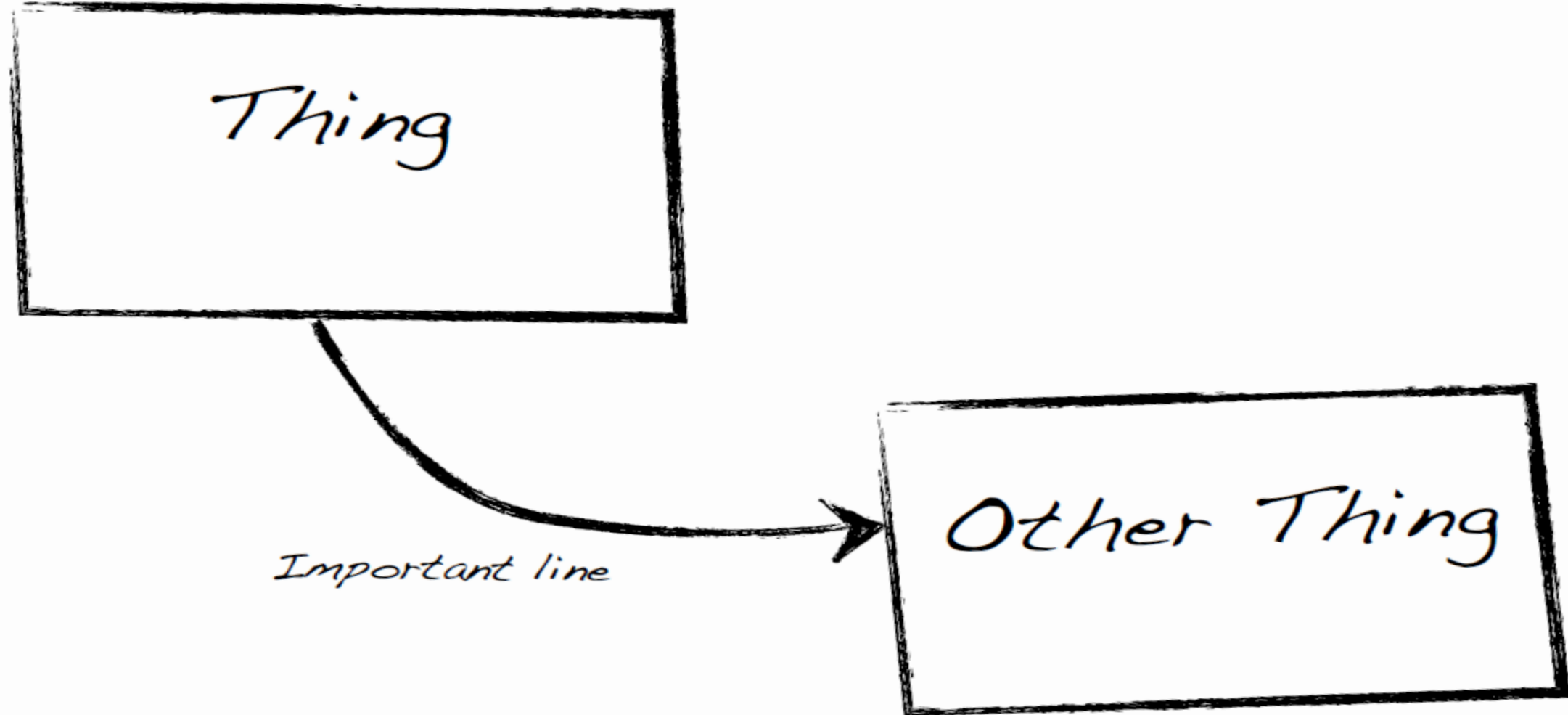
Batch

~~Batch~~
B-id:
custid

~~Batch~~



Boxes & lines



- What is this shape/symbol?
- What is this line/arrow?
- What do the colors mean?
- What level of abstraction is shown?
- Which diagram do we read first?

Information is likely
still stuck in your heads

Software architects
struggle to communicate
architecture

Titles

Short and meaningful, numbered if diagram order is important

Labels

Be wary of using acronyms, especially those related to the business/domain that you work in

Lines

Favor unidirectional arrows, add descriptive text to provide additional information

Layout

Sticky notes and index cards make a great substitute for drawn boxes, especially early on

Orientation

Most important thing in the middle; be consistent across diagrams

Color

Ensure that color coding is made explicit; watch out for color-blindness and monochromatic printers

Shapes

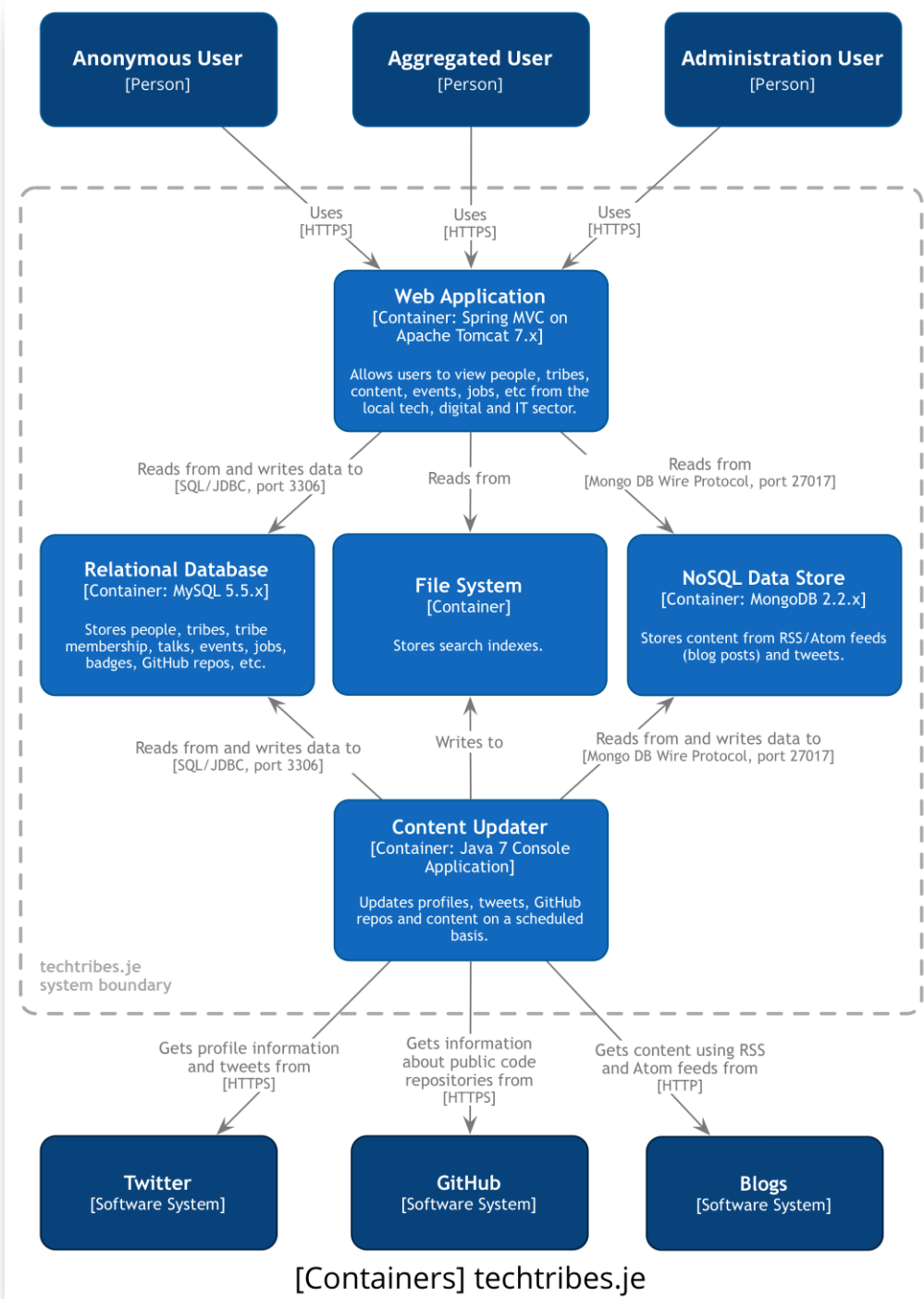
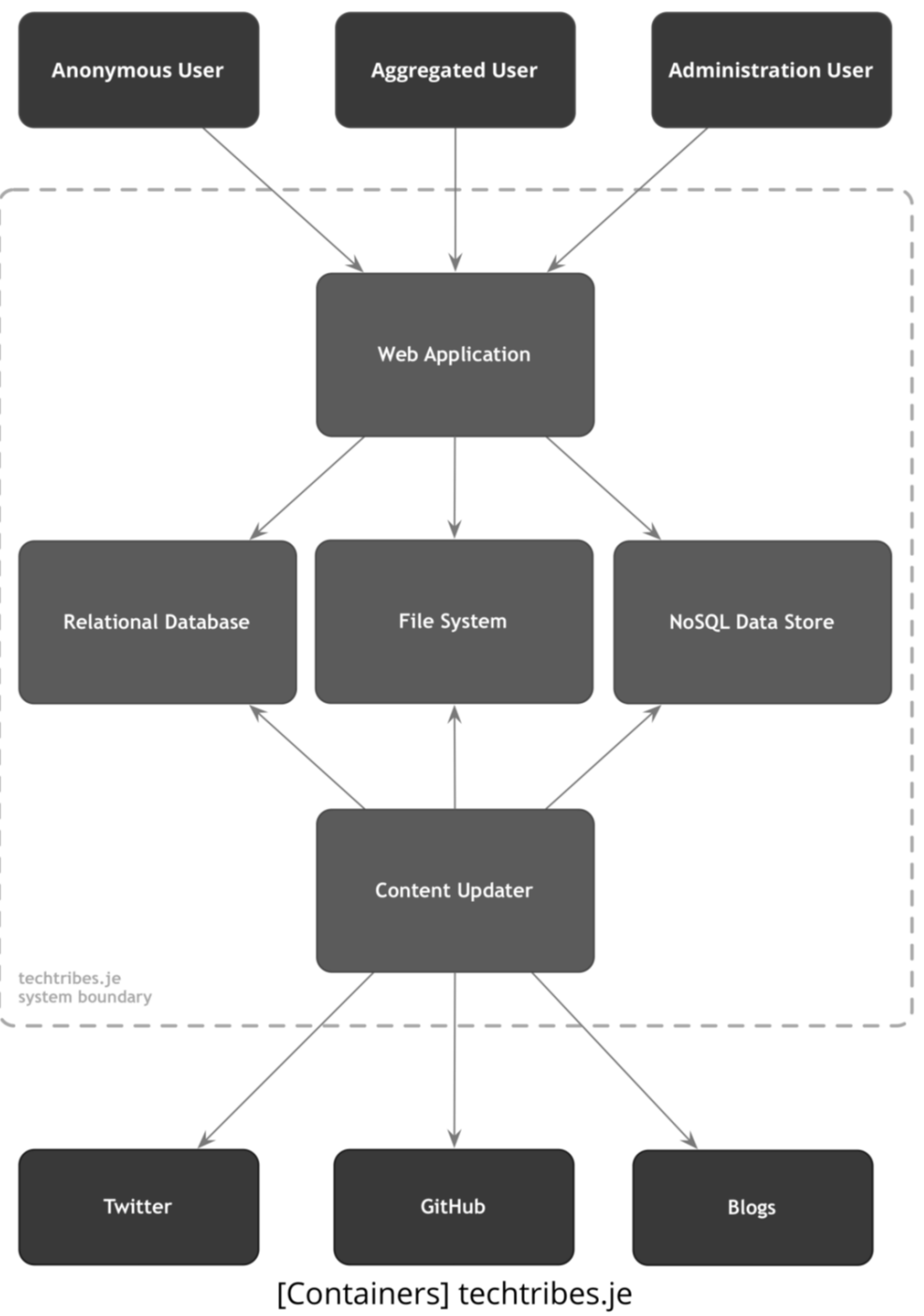
Don't assume that people will understand what different shapes are being used for

Responsibilities

Adding responsibilities to boxes can provide a nice "at a glance" view (Miller's Law; 7 ± 2)

Keys

Explain shapes, lines, colors, borders, acronyms, etc.



Well-defined structure is when you...

- Can see the solution from multiple levels of abstraction
- Understand the big picture (context)
- Understand the logical containers
- Understand the major components used to satisfy the important user stories/features
- Understand the notation, color coding, etc. used on the diagrams
- Can see the traceability between diagrams (views)

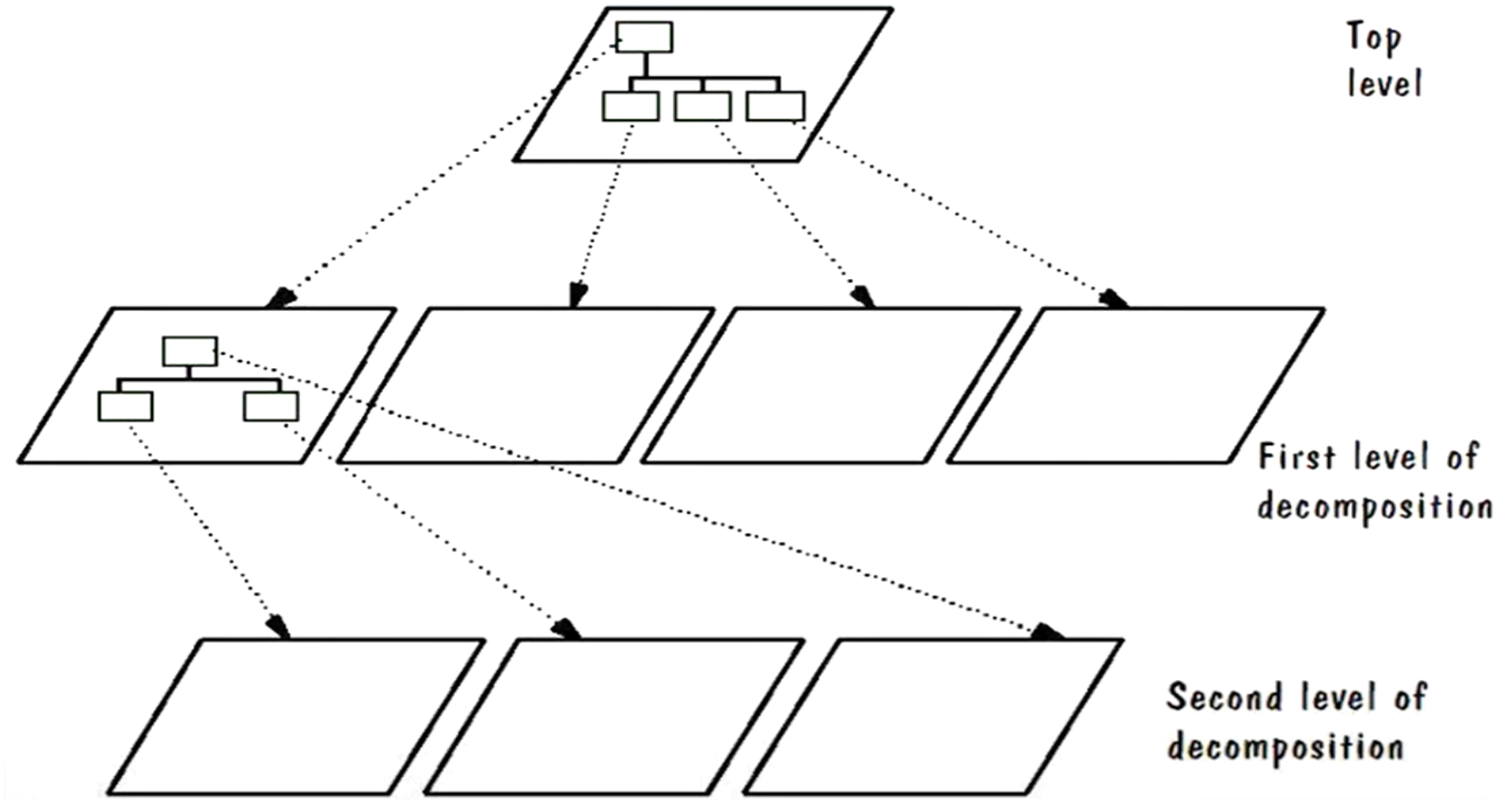


Understand how the significant elements fit together.

Abstraction

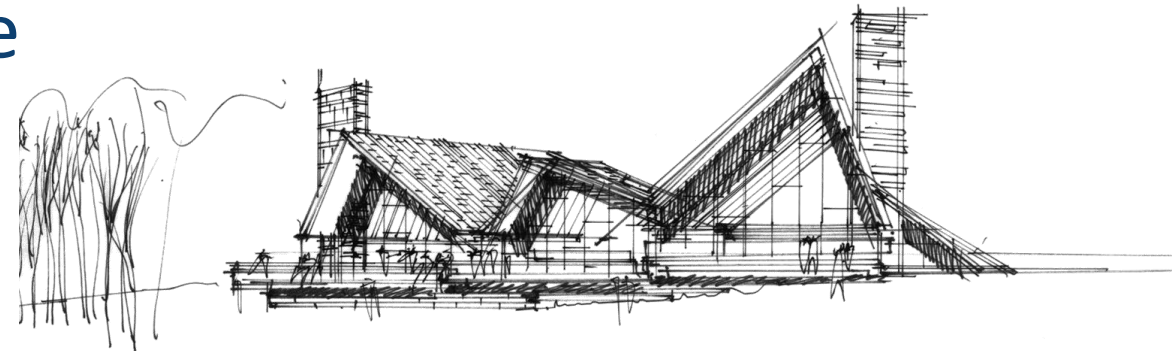


Details



Well-defined vision is when you...

- Understand the major **technology decisions**
- Understand the **implementation strategy** (frameworks, libraries, APIs, etc.)
- Can **visualize** the code structure



If this works for others than why not for SW?

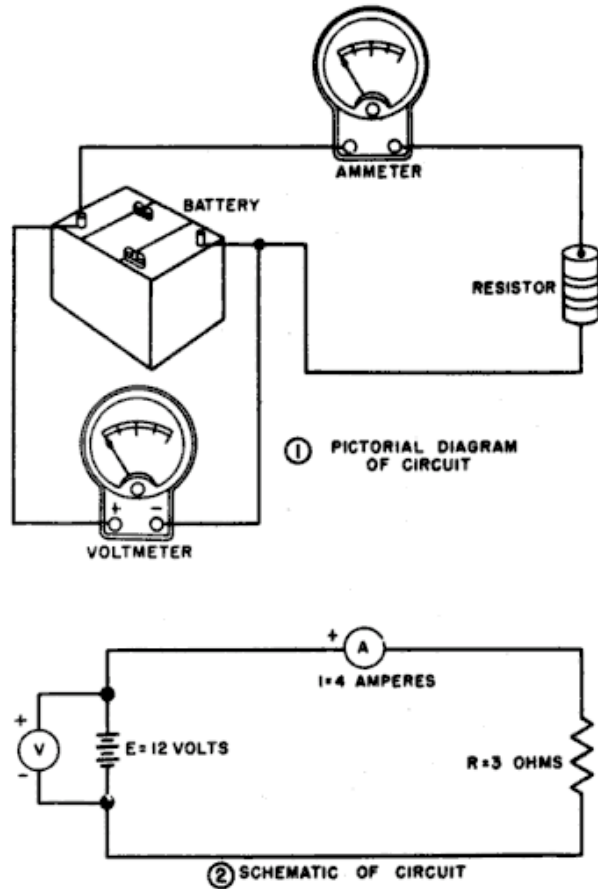
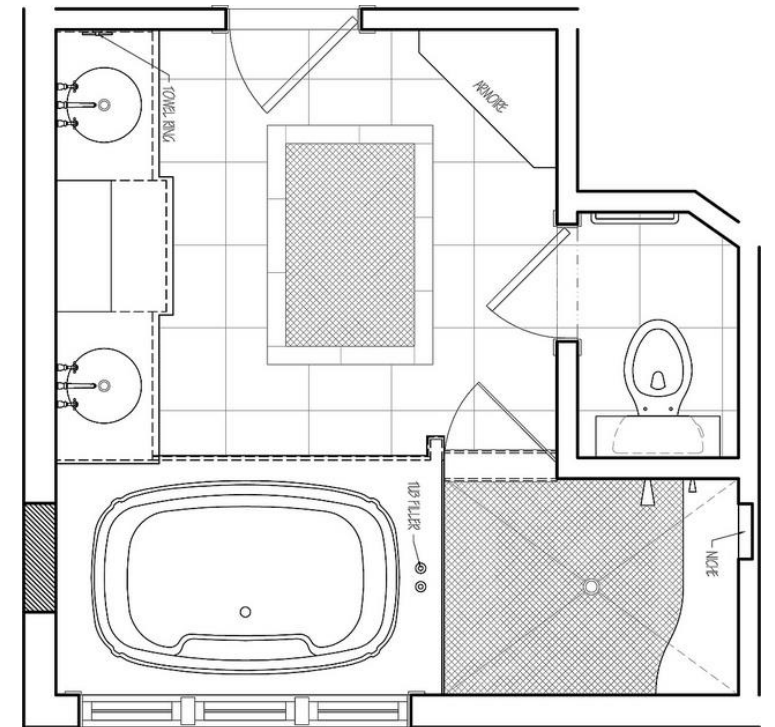


Figure 48. Diagram of a basic circuit.





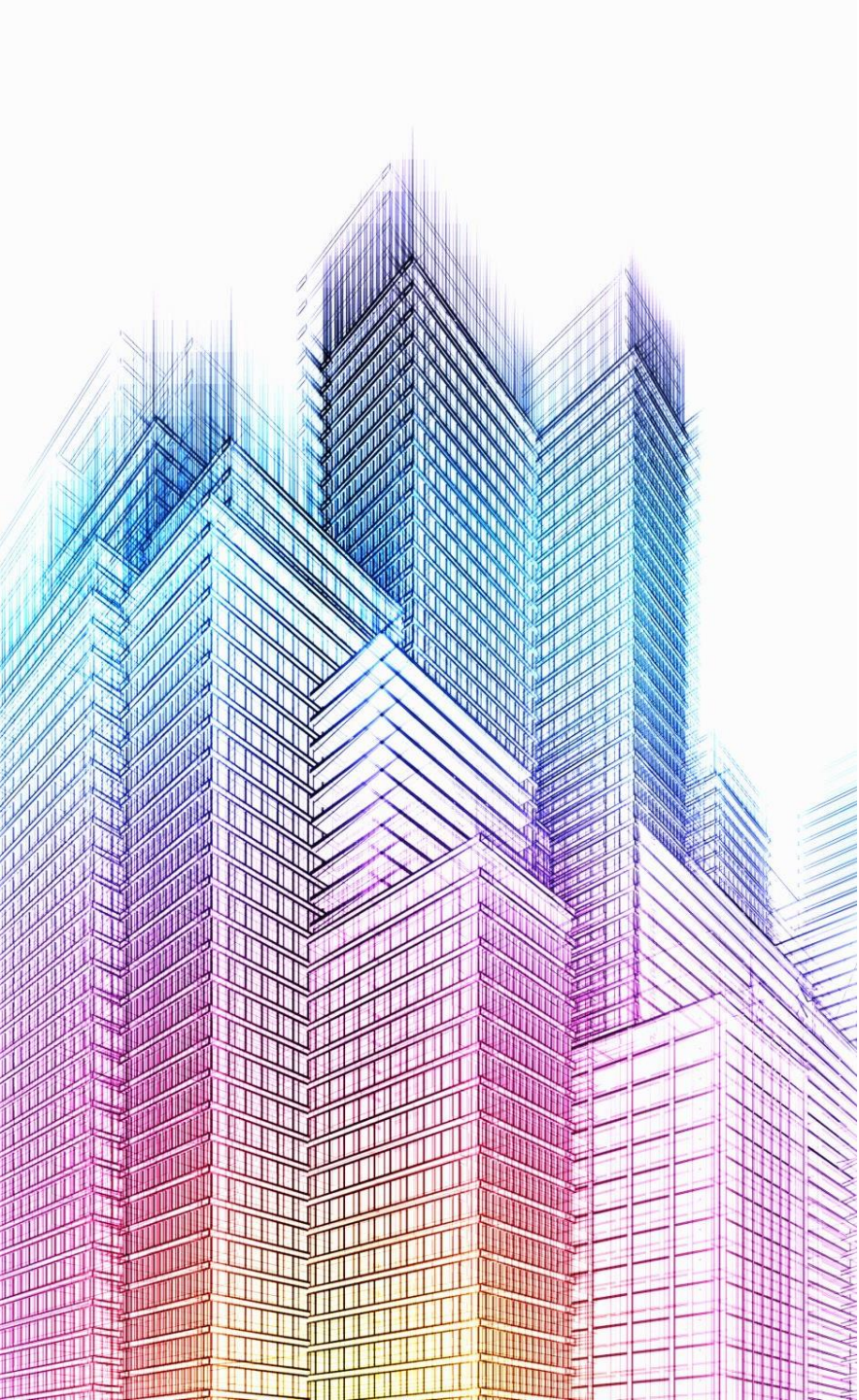
Is this the
reason 😊?

"Balance these 6 nails without letting them touch the wood"










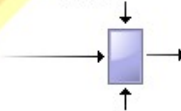

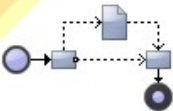

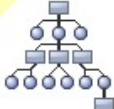
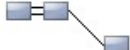
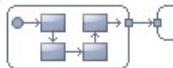
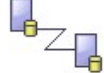
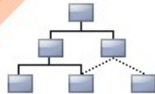

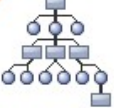


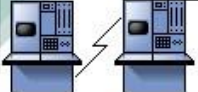


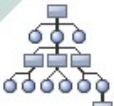






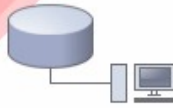





Software Is All Around...

- Wide range of industries ranging from **aerospace** and **automotive** engineering to **finance, defense, government, entertainment, telecommunications** etc.
- Different engineering domains
 - specific **technologies** or **coding languages**
- Vast of **modeling languages** and **frameworks**
 - difficult to identify the right tool or combination of tools to meet your exact modeling requirement or scenario



Types of Architecture

- Sub-architectures
 - Business architecture
 - Information architecture
 - Application architecture
 - Technology architecture
- Architectural views
 - Security architecture
 - Geospatial architecture
 - Social architecture

The Zachman Framework	DATA What	FUNCTION How	NETWORK Where	PEOPLE Who	TIME When	MOTIVATION Why
SCOPE (Contextual) Planner	Things Important to the Business 	Processes the Business Performs 	Locations in which the Business Operates 	Organizations Important to the Business 	Events/Cycles Significant to the Business 	Business Goals/Strategies 
BUSINESS MODEL (Conceptual) Owner	Conceptual Data Model 	Business Process Model 	Business Logistics 	Work Flow Model 	Master Schedule 	Business Plan 
SYSTEM MODEL (Logical) Designer	Logical Data Model 	Application Architecture 	Distributed System Architecture 	Human Interface Architecture 	Processing Structure 	Business Rule Model 
TECHNOLOGY MODEL (Physical) Builder	Physical Data Model 	System Design 	Technology Architecture 	Presentation Architecture 	Control Structure 	Rule Design 
DETAILED REPRESENTATIONS Sub-Contractor	Data Definition 	Program 	Network Architecture 	Security Architecture 	Timing Definition 	Rule Specification 
FUNCTIONING ENTERPRISE	Data 	Function 	Network 	Organization Units 	Schedule 	Strategy 

Language vs. Methodology

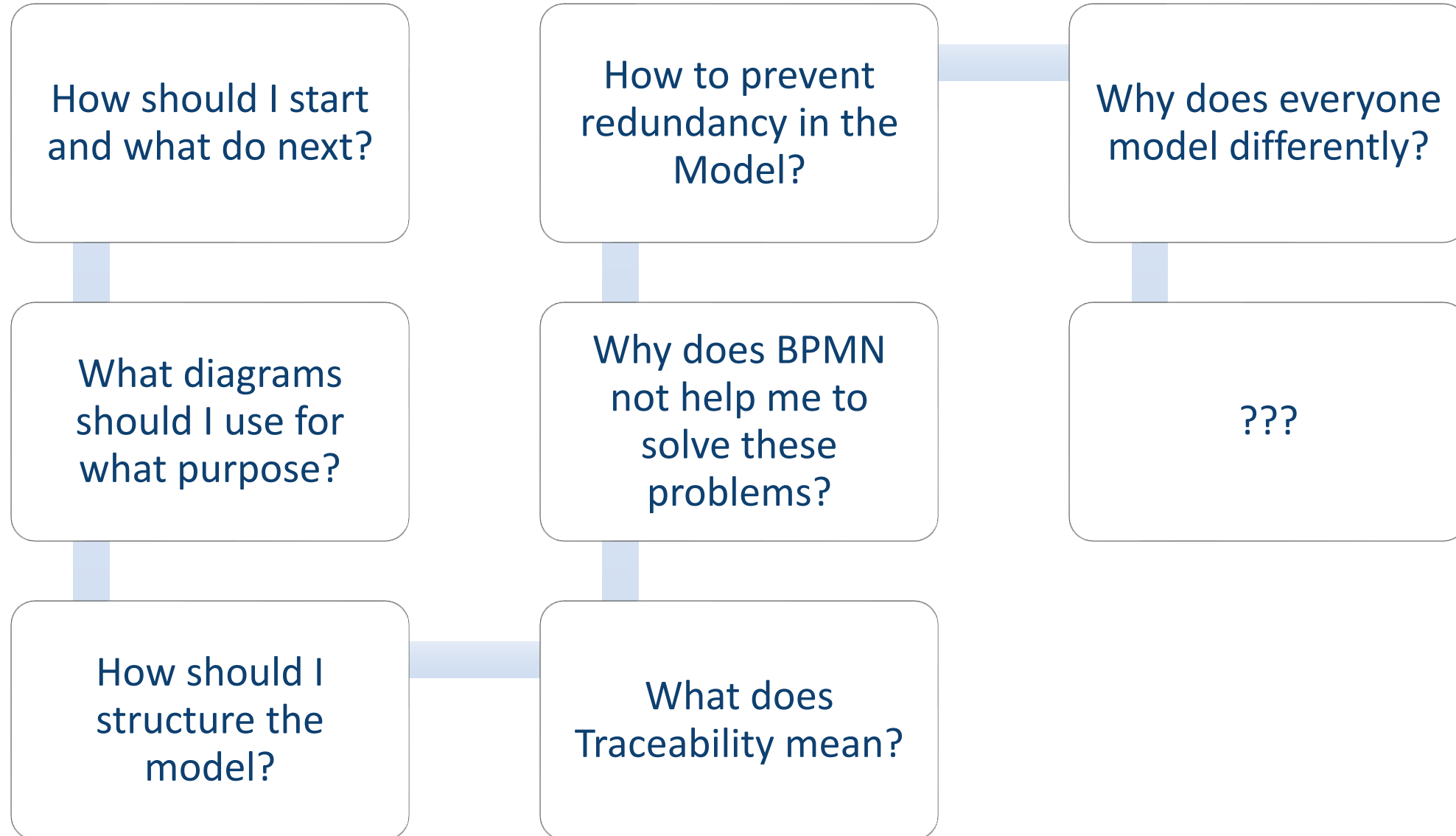
- **Modeling Language**

- Defines elements and their relationship
- Defines syntax and semantics
- *What type of elements can be used during modeling?*
- E.g. Boxes and lines 😊, UML, SysML, BPMN, ArchiMate ...

- **Development Methodology**

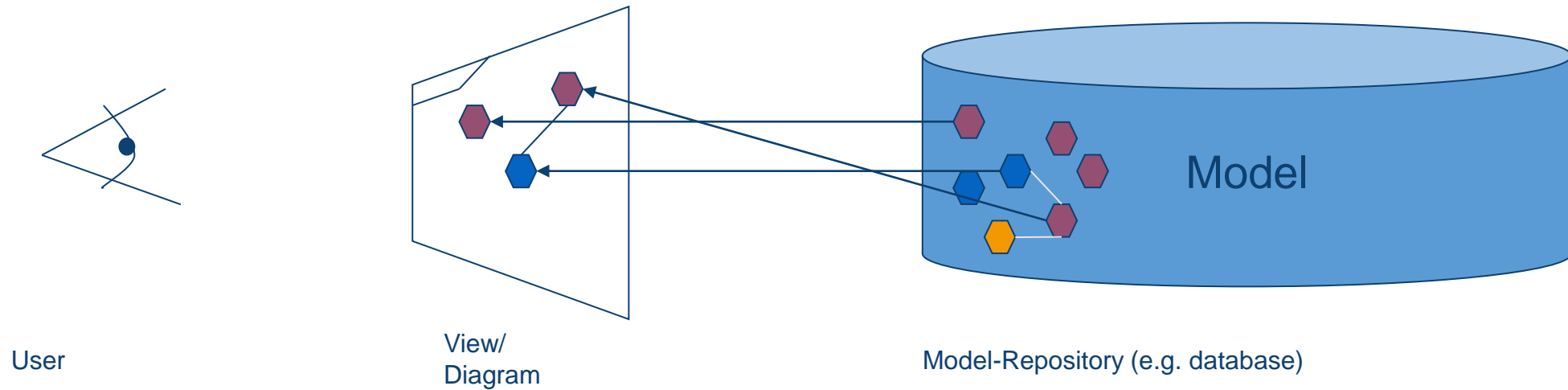
- Defines the steps of the Architecture process
- Defines the usage of the model elements and diagrams
- *How shall the model be built?*
- E.g. Zachman Framework, FEAF, DoDAF and TOGAF, ...

Modeling Methodology gives the Answers



What is a model?

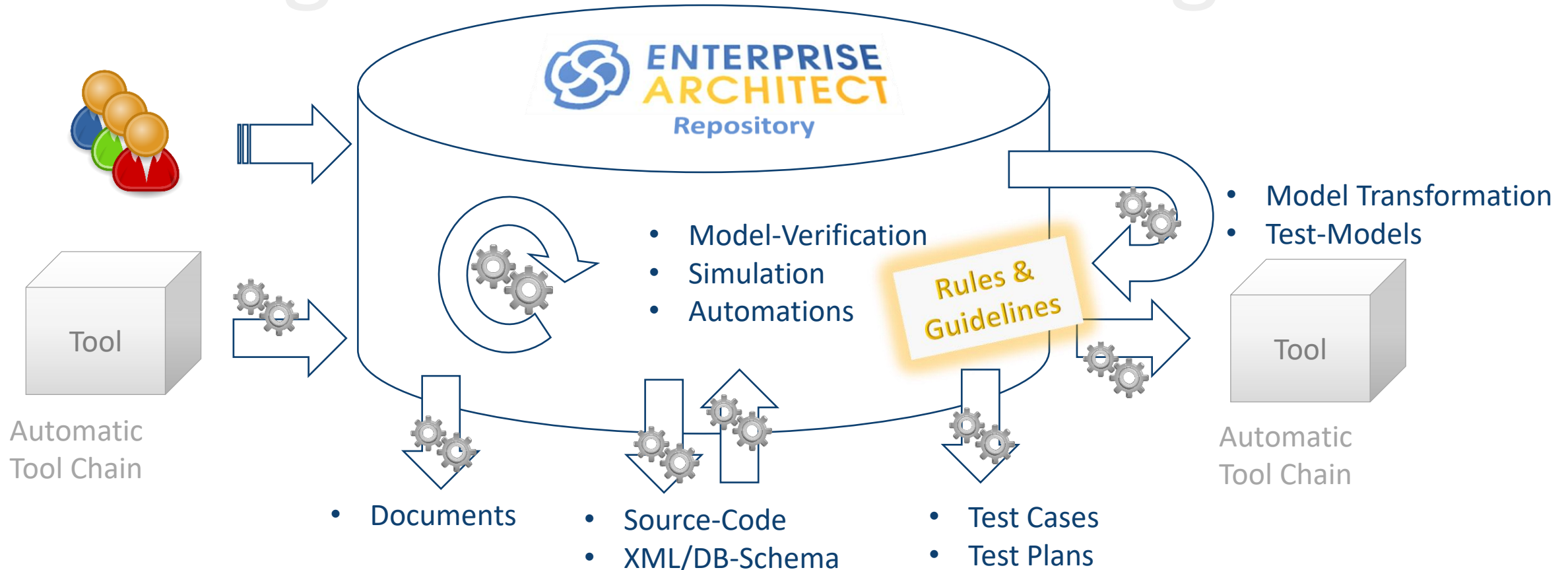
- **Model** is an **abstraction** of the **reality**
 - To provide information in an understandable way
 - To show essential system aspects
 - For communication purpose (project member, customer)
 - To represent complex architectures
- **Diagram** is a **graphical representation** of the **model**



Modeling notation

What is the advantage of using modeling tools?

Garbage in → Garbage out

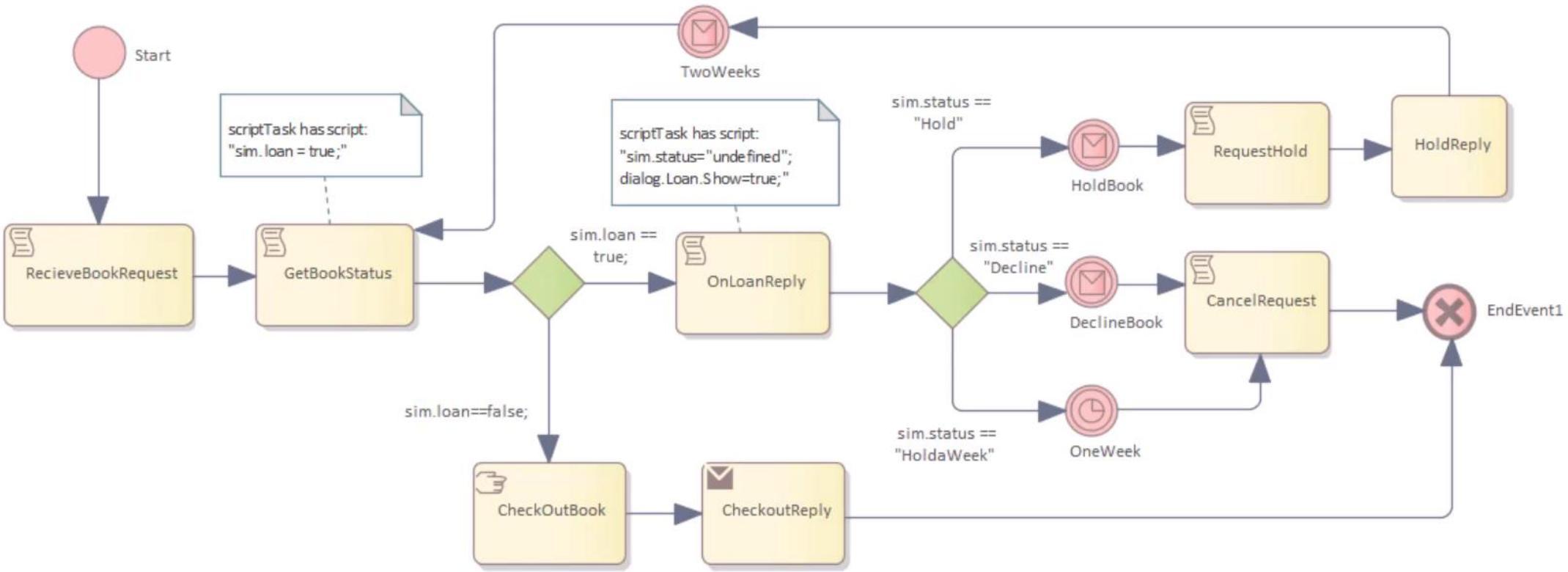


If everything is made “right“, you can automate it!

Business Process Diagram

Book Lending Example

To run the simulation: from the 'Simulate' ribbon, select the 'Start' button, or from the diagram's context menu, choose 'Execute Simulation | Interpreted Simulation'.





What went wrong?



We didn't pay enough attention to the right
architecture drivers!



(especially quality attributes)

Quality Attributes are Architectural Drivers

- Benchmarks that describe a system's intended behavior within the environment in which it was built.
- Requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors.




- **Availability** - Is it available when and where I need to use it?
- **Installability** - How easy is it to correctly install the product?
- **Integrity** - Does it protect against unauthorized access and data loss?
- **Interoperability** - How easily does it interconnect with other systems?
- **Performance** - How fast does it respond or execute?
- **Reliability** - How long does it run before experiencing a failure?
- **Recoverability** - How quickly can the user recover it from a failure?
- **Robustness** - How well does it respond to unexpected operating conditions?
- **Safety** - How well does it protect against injury or damage?
- **Usability** - How easy is it for people to learn and use?

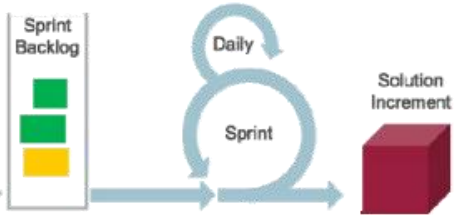


- **Efficiency** — how well does it utilize processor capacity, disk space, memory, bandwidth, and other resources?
- **Flexibility** — How easy can it be updated with new functionality?
- **Maintainability** — How easy is it to correct defects or make changes?
- **Portability** — How easily can it be made to work on other platform?
- **Reusability** — How easily can we use components in other systems?
- **Scalability** — How easily can I add more users, servers, or other extensions?
- **Supportability** — How easy will it be to support after installation?
- **Testability** — Can I verify that it was implemented correctly?

The ISO/IEC
42010
standard
defines
architecture
as:

'The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and **evolution**'.



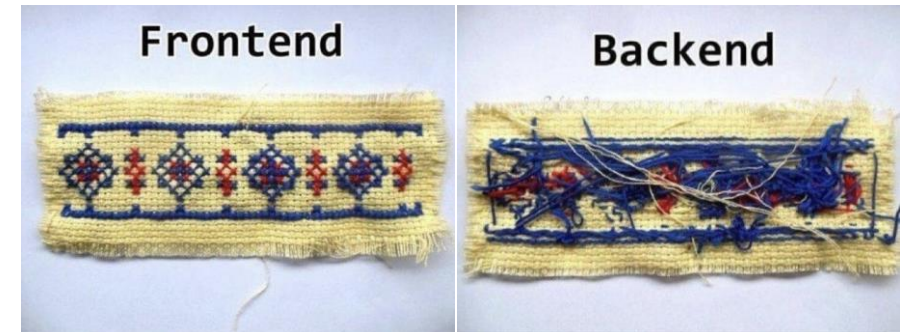
Continuous stream of architectural decisions

	Traditional	Agile
Development	 <p>Big Bang</p>	 <p>Continuous stream of improvements</p>
Architecture	 <p>Big Up-Front Design</p>	 <p>Continuous stream of Architectural Decisions</p>

Balance your backlog

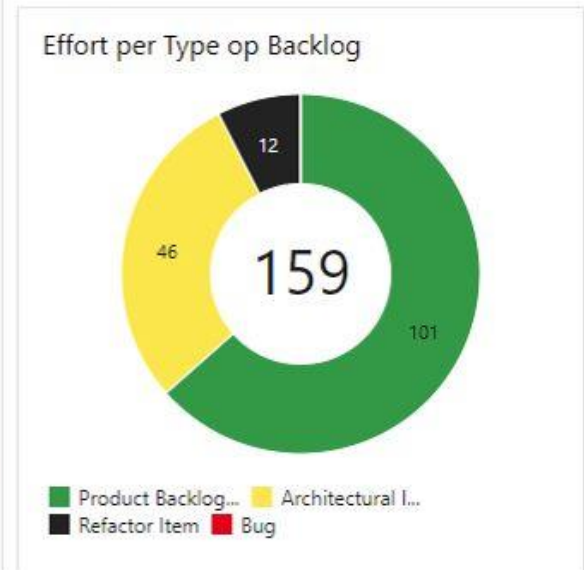
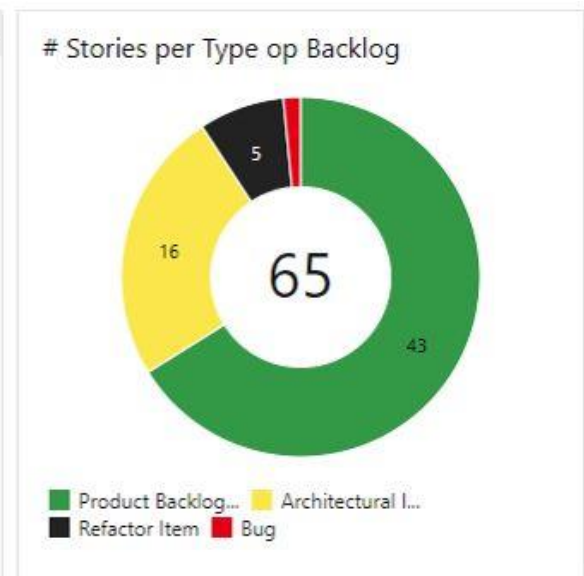
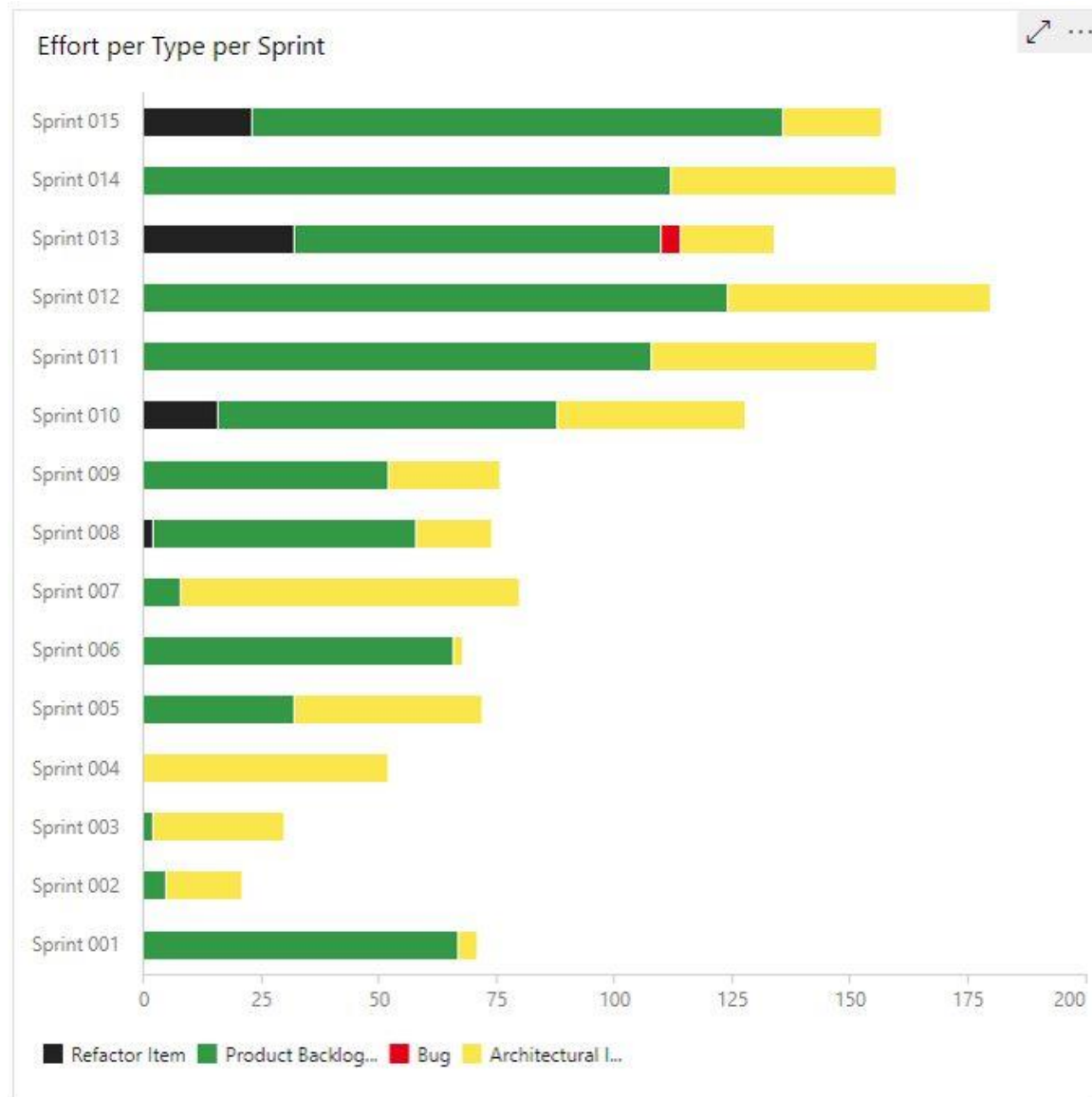
- Green – Features to be delivered, the functional user stories
- Yellow – Architectural infrastructure that support the quality requirements
- Red – Defects that are identified and need to be addressed
- Black – Technical debt that builds up as the product is built and key decisions are deferred or poor work done

	Visible	Invisible
Positive Value	New features Added functionality	Architectural, Structural features
Negative Value	Defects	Technical Debt



Source: <https://philippe.kruchten.com/2013/12/11/the-missing-value-of-software-architecture/>

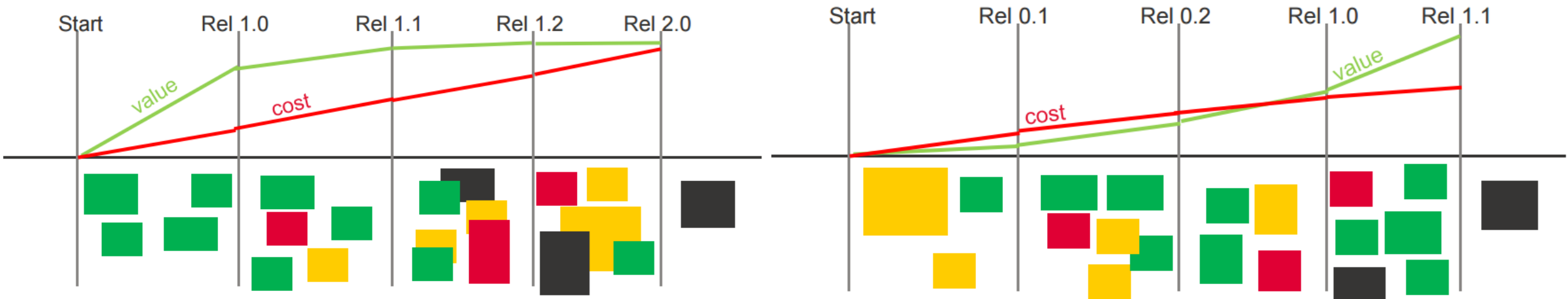
Backlog coloring scheme example



Architecture Roadmapping strategies

Release strategy 1: value-first

Release strategy 2: architecture-first



Beware of “analysis paralysis”!

Why “describe” architecture?

- **To document your work**
 - Sooner or later, someone else will want to understand what you have done.
 - Carefully-selected architectural descriptions are an effective way of conveying understanding.
- **To surface “unknowns”**
 - Often, you don’t know what you (really) know until you try and describe (or explain) it
 - Some of these “unknowns” are genuine project risks



Focus on creating architectural descriptions where it counts.

If you think good
architecture is
expensive, try bad
architecture.

Brian Foote & Joseph Yo

OOP

SIGS DATACOM

www.sigsoptions.com

OOP
DATACOM





1

READY