Software Architecture

Michel R. V. Chaudron
LIACS & TU Eindhoven

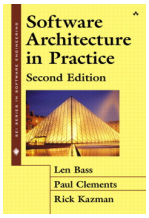
Leiden Institute of Advanced Computer Science

Lecture Outline

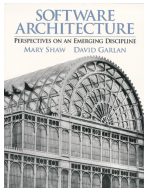
- What, Why, When, Where, Who SWARCH
- Describing
- Designing (start of)

} How

Software Architecture Books



- Software Architecture in Practice, **Second Edition**, L. Bass, P. Clements, R. Kazman, SEI Series in Software Engineering, Addison-Wesley, 2003

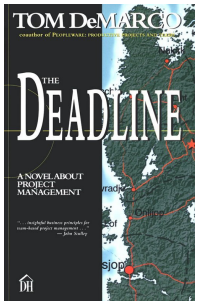


- Software Architecture: Perspectives on an Emerging Discipline, Mary Shaw, David Garlan, 242 pages, 1996, Prentice Hall

- Recommended Practice for Architectural Description, IEEE STD 1471-2000, 23 pages

Software Project Management

The Deadline
by Tom DeMarco

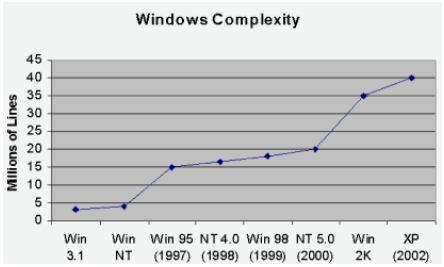


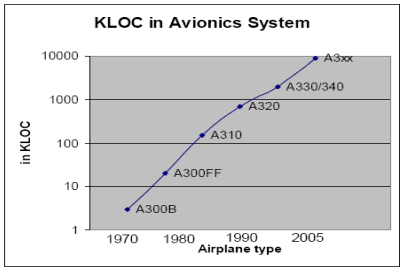
• easy & fun reading
• lots of lessons from practical experience

Also very though provoking:

Peopleware
by DeMarco & Lister,
Dorset House Publ., 2nd ed, 1999

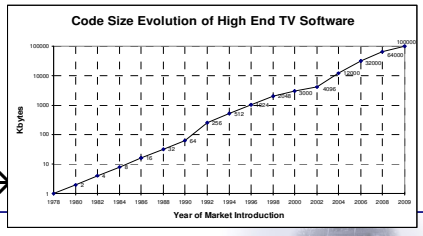
Increasing amount of software in systems





The amount of software increases 10 fold every 10 years.
Abstractions are needed.

Nb: logarithmic scale →



What is Software Architecture?

Classic Definitions 1

An architecture is the set of significant decisions about
- the organization of a software system,
- the selection of the structural elements and their interfaces by which the system is composed, together with their behaviour as specified in the collaborations among those elements,
- the composition of these structural and behavioural elements into progressively larger subsystems,
- the architectural style that guides this organization

The UML Modeling Language User Guide, Addison-Wesley, 1999
Booch, Rumbaugh, and Jacobson

What is Software Architecture?

Classic Definitions 2

The structure of the components of a program/system, their interrelationships, and principles and guidelines governing their design and evolution over time.

David Garlan and Dewayne Perry
April 1995 IEEE Transactions on Software Engineering

Example: FEI electron microscopes

Full Architecture Description

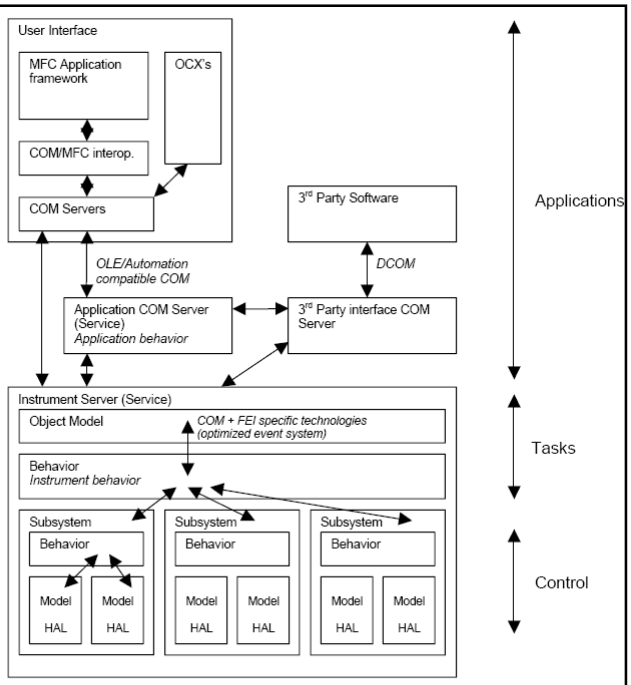


Figure 4: Execution model of the SDB software, in as far as this runs on the PC.

Example: FEI electron microscopes

Full Architecture Description

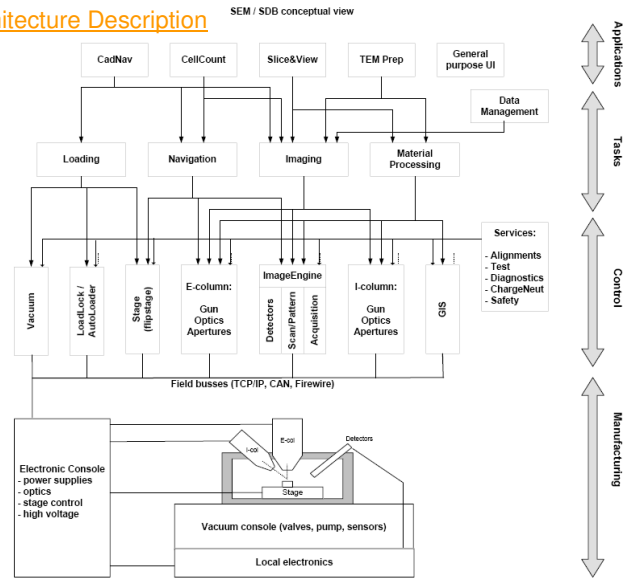


Figure 1: the SDB family system decomposition and interrelationships on the different levels of abstraction.

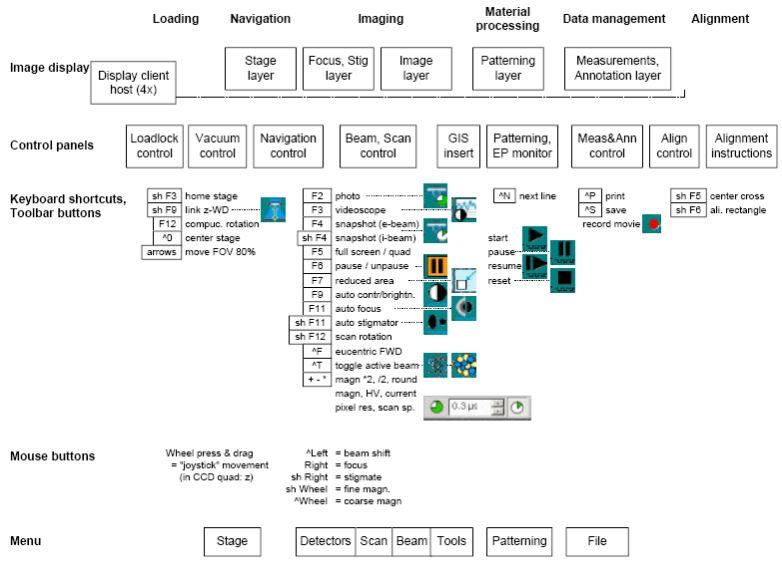
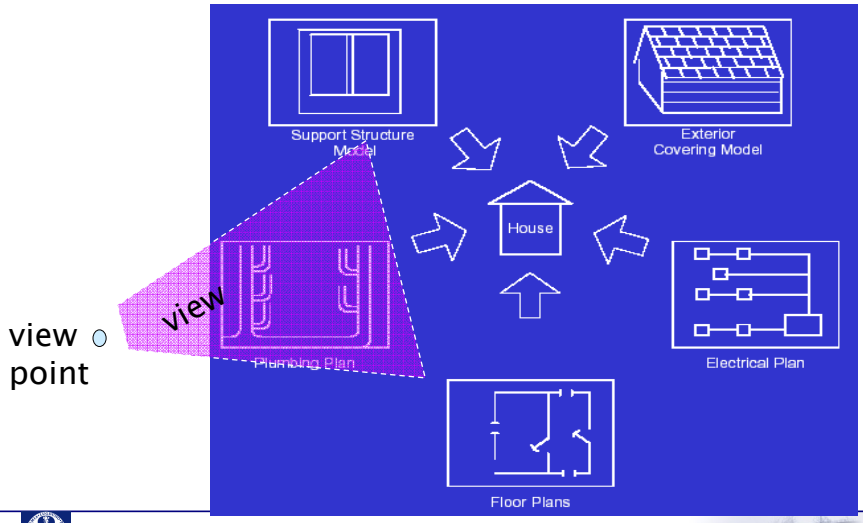


Figure 3: Functions offered through the user interface, arranged according to task and control means

Contents of a good architectural model

- A system's architecture will often be expressed in terms of several different *views*
  - The logical breakdown into subsystems
    - conceptual abstract view
    - functional decomposition
    - responsibility distribution
    - The interfaces among the subsystems
  - The dynamics of the interaction among components
  - The data that will be shared among the components
  - The components that will exist at run time, and the machines or devices on which they will be located

Viewpoints & views



Slide 11

MC2 hierarchy
- itself a view
- may apply to different views

Michel Chaudron; 27-2-2008

Architectural view

- An architectural view is a simplified description (an abstraction) of a system from a particular perspective/view point, covering particular concerns, and omitting entities that are not relevant to this perspective

Elements of Architectural Design

- Structure
  - decomposition, hierarchy
  - interfaces
- Behaviour
  - within and between components
- Data
- Design Decisions / Rationale

At different levels of abstraction:
- conceptual
- development
- run-time/ physical

Making design decisions

- To make each design decision, the software engineer uses:
  - Knowledge of
    - the application domain
    - the requirements
    - the design as created so far
    - the technology available
    - software design principles and 'best practices'
    - what has worked well in the past

Combination of top-down and bottom-up

Positioning Architecture

The question:	***The answer:***	***Implementation:***	***Deployment:***
Require-ments	Architecture	Design	Executable
• Features • Use cases • Dependability Timing Reliability Security • Quality • Standards • Etc.	• HL-Design Components Interfaces Interactions • Styles • Constraints • Guidelines • Reuse • Etc.	• Decomposition • Algorithms • Data structures • Distribution • Scheduling • Recovery • Language • Encryption • Etc.	• Memory allocation • Dynamic Instantiation • Call stacks • Garbage collection • Machine code • Etc.

Architecturally Significant Elements

An architecturally significant element has a significant impact on the **structure, performance, robustness, scalability, maintainability** and **evolvability** of the system.

Business Objectives of Software Architecture

Reduce development cost

- improved communication between developers, and
- earlier assessment of design alternatives and assessment of system risks

Reduce time-to-market

- allowing concurrent development of different subsystems
- enabling reuse

Business Objectives of Software Architecture

Reduce maintenance cost

- Design should plan for incorporation of foreseeable changes and extensions

Improve product quality

Increase fitness for use through stakeholder involvement;

reduce errors through enforcement of conceptual integrity

SA Objectives for Development 1/2

Management of Complexity

Define a model of a system that is intellectually manageable – better understanding

Answering of *what-if* questions

Allows stakeholders to evaluate different architectural solutions and their consequences

SA Objectives for Development 2/2

Feasibility study & risk analysis

Analysis of various (non-)functional features of the future product; identification of possible problems during development, production & operation

Project estimation, planning & organization

Allocation of components to concurrent teams

For Whom ?

An architecture is a (common) means of understanding of a system

- Customers, Users, Domain Experts
- Engineers:
  - analysts, architects
  - programmers
    - maintenance, development,
    - new members development team
- Marketing, Sales
- Management ...

Different types of architectures?

Stakeholders & their Concerns 1/2

Stakeholder	*Concern (Examples)*
Customer	Business goals
	Schedule & budget estimation
	Feasibility and risk assessment
	Requirements traceability & progress tracking
	Product-line compatibility
User	Consistency with requirements & use cases
	Future requirements growth accommodation
	Support of dependability & other X-abilities
Service manager	Reliability, availability and maintainability

Stakeholders & their Concerns 2/2

Stakeholders	*Concern (Examples)*
System engineer	Requirements traceability Support of tradeoff analyses Completeness of architecture Consistency of architecture with requirements
Developer	Sufficient detail for design and development Workable framework for system construction, e.g. selection/assembly of components & technologies Resolution of development risks
Maintainer	Guidance on software modification Guidance on architecture evolution Interoperability with existent systems

Multiple Purposes of Architecture

Understanding
+ Analyzing + Communicating + Constructing

Picture from Gerrit Muller, How to Create a Managable Platform Architecture

When Architecting?

- **When developing a new system**
- **When changing a system**
  - if an architecture description is not available, or insufficient, as a basis for change
  - adapt the architecture documentation to changes
- **When integrating existing systems**
- **For special communication needs**
  to provide a common ground for understanding

… giving people the appropriate tools to frame and structure their discussion and decision making is an enormous benefit to the disciplined development of complex systems.

Software Architecture in Practice 2nd ed.
Bass, Clements, Kazman

What is Software Architecture?

Definition 3

Architecture of software is a collection of design decisions that are expensive to change.

Alexander Ran, Nokia Research
September 2001 European Conference on Software Engineering

"The things that are fixed"

Describing Architectures

Leiden Institute of Advanced Computer Science

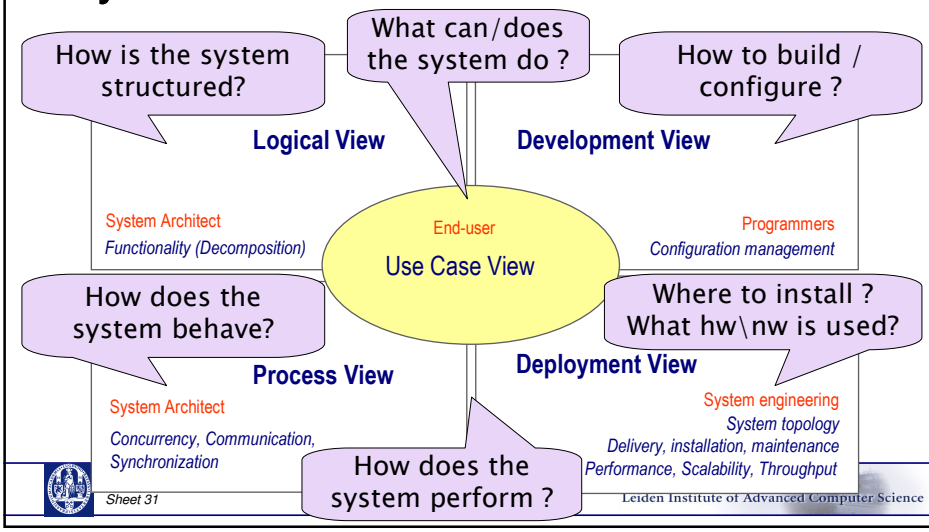
Philippe Kruchten's Definition

Software architecture is not only concerned with structure and behaviour, but also with

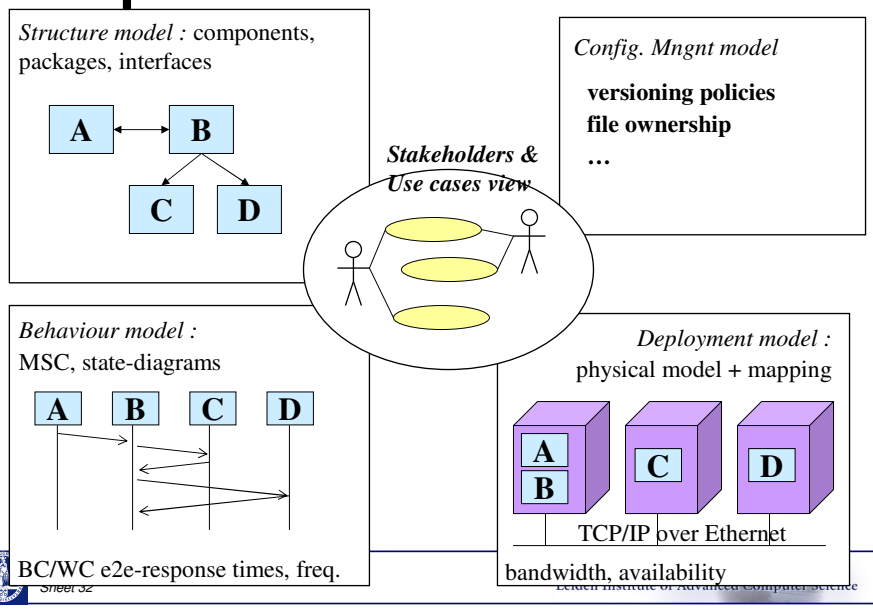
- usage
- functionality
- performance
- resilience
- reuse
- comprehensibility
- economic and technological constraints and tradeoffs
- aesthetics

The Rational Unified Process -- An Introduction, Addison-Wesley, 1999.

4+1 Views Representation of System Architecture



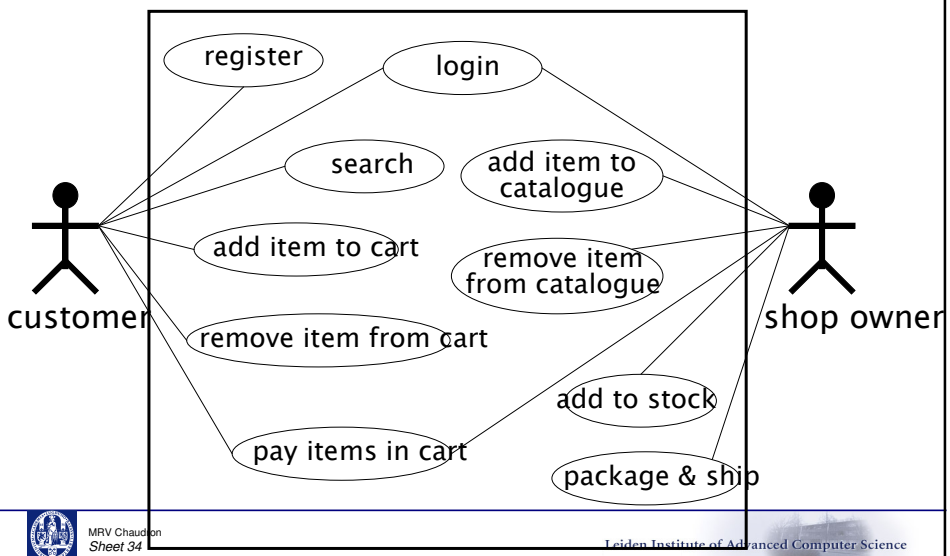
Example 4+1 model



Use Case Diagram

- Captures system functionality as seen by users
- Built in early stages of development
- Purpose
  - Specify the context of a system
  - Capture the requirements of a system
  - Validate an architecture's completeness
  - Drive implementation and generate test cases
- Developed by analysts and domain experts

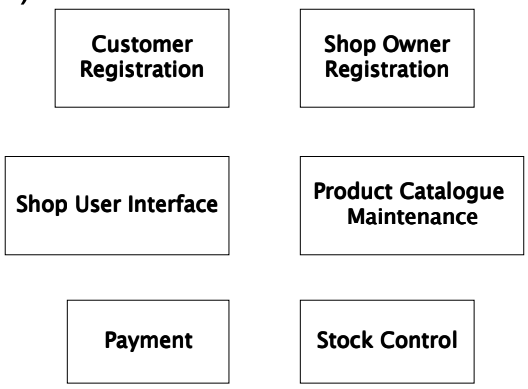
Case: Web shop



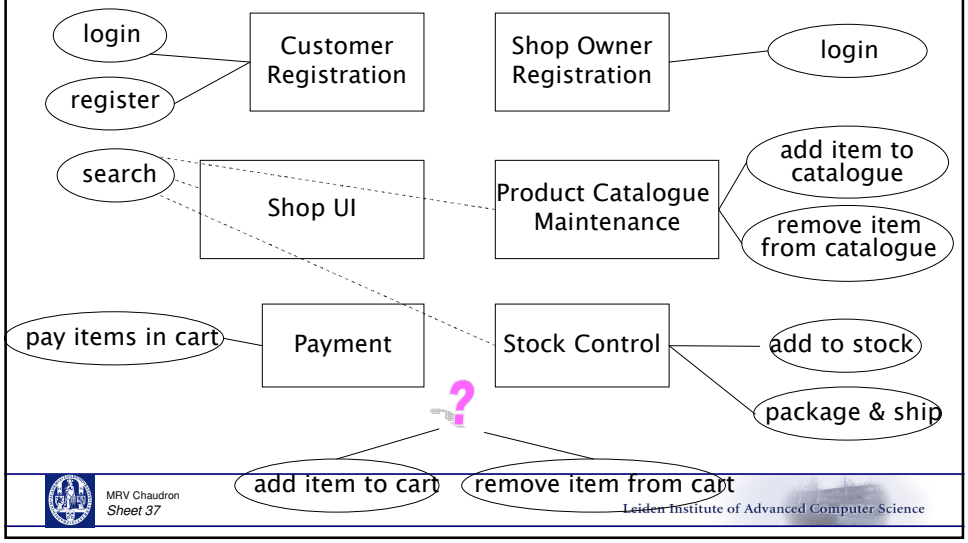
Structure Diagram

- Defines subsystems of functionality
- Purpose
  - Define decomposition into subsystems
  - Provide support for use-cases
- Use Component diagram
  - May use Class, but this suggests OO implem.

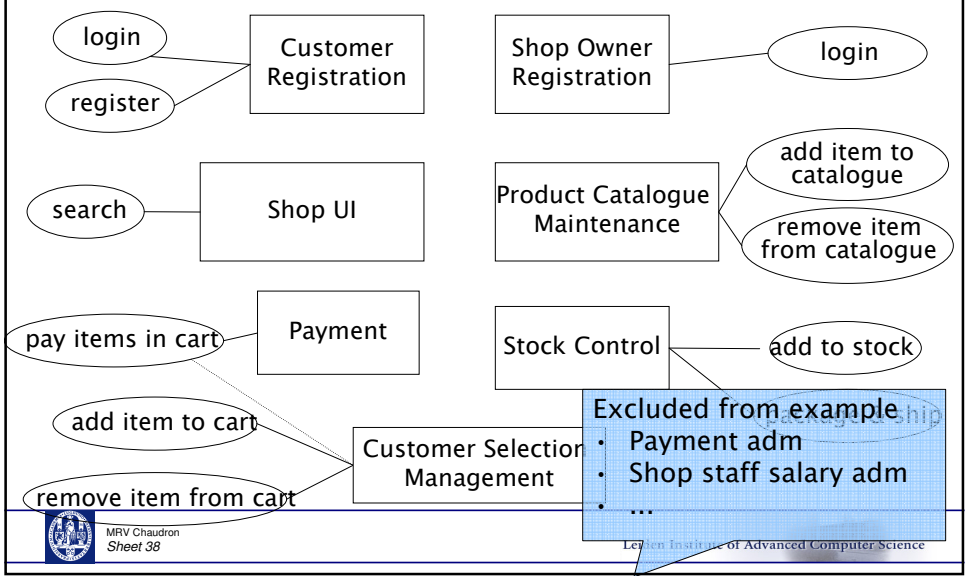
Web Shop: Functional Areas (V0.1)



Check Use Cases Against Functional Areas



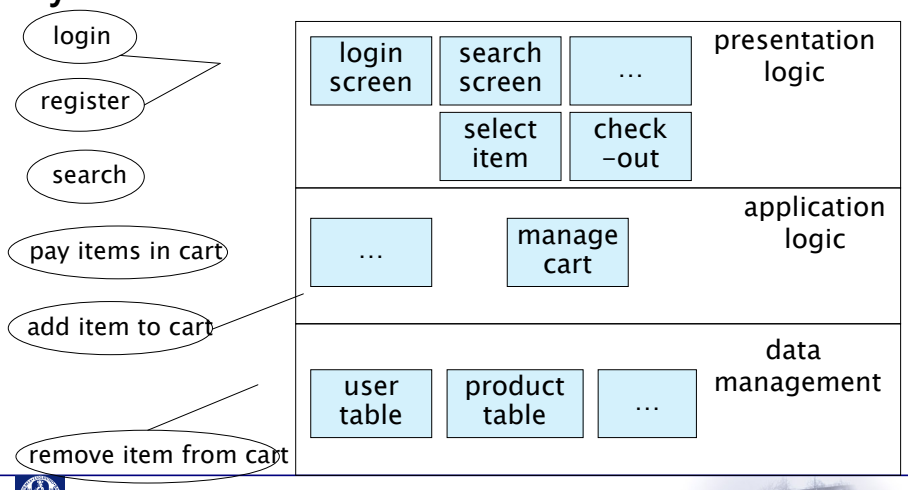
Web Shop: Functional Areas (V0.2)



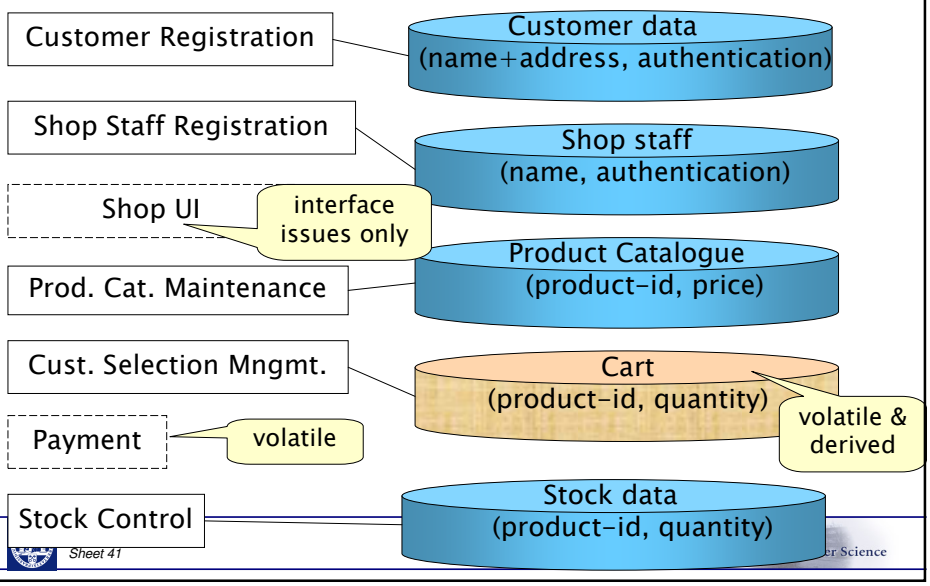
Web Shop: Responsabilities

Customer Registration	Entry, storage & retrieval of customers
Shop Owner Registration	Entry, storage & retrieval of shop staff
Shop UI	Provide customers access to product data
Prod. Cat. Maintenance	Entry, storage & retrieval of product data
Cust. Selection Mngmt.	Register customer product selection
Payment	Handle transaction between customer & shop
Stock Control	Register available products in stock

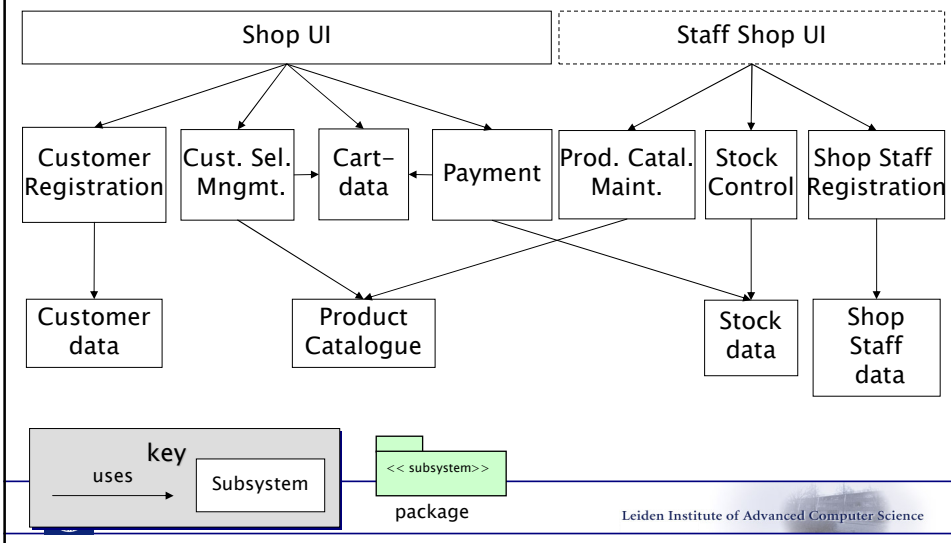
Identify support for Use Cases at different layers in the architecture



Identification of Data Domains



Identification of Dependencies



View: Definition (from IEEE 1471)

3.4 Architectural Description (AD): A collection of products to document an architecture.

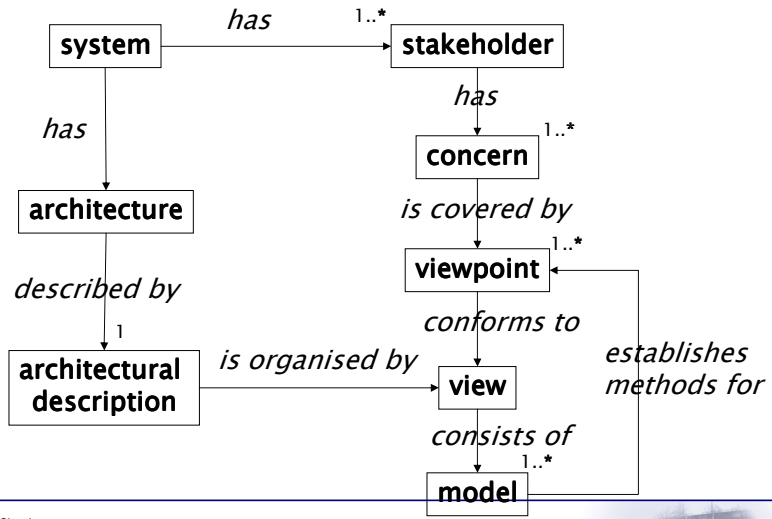
3.9 View: A representation of a whole system from the perspective of a related set of concerns.

A view may consist of one or more *architectural models*

Each such architectural model is developed using the methods established by its associated architectural viewpoint.

An architectural model may participate in more than one view.

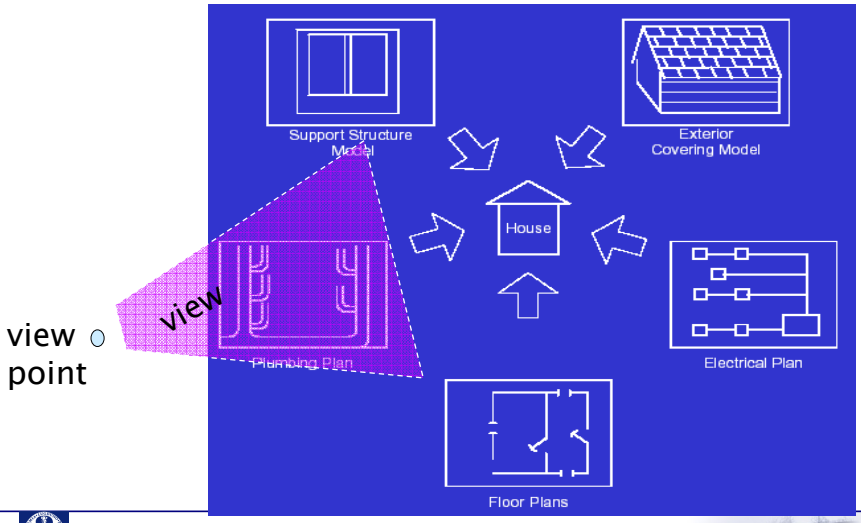
Overview (According to IEEE 1471)



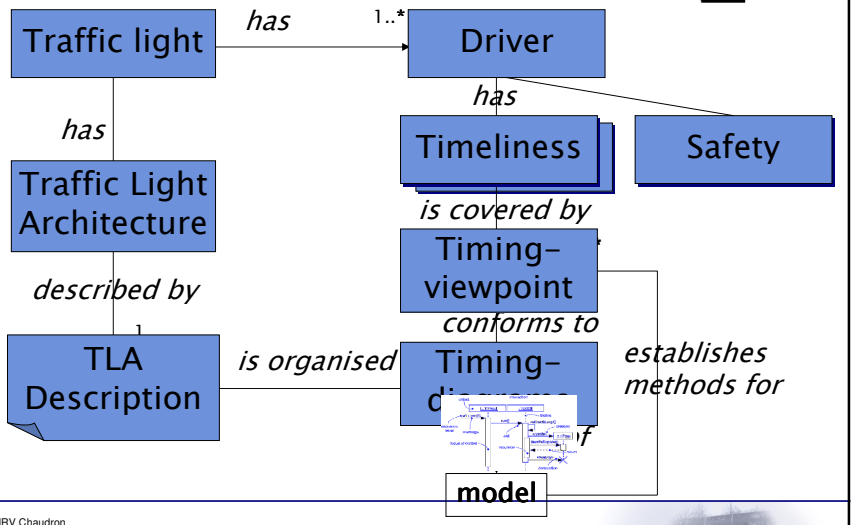
Architectural view

- An architectural view is a simplified description (an abstraction) of a system from a particular perspective/view point, covering particular concerns, and omitting entities that are not relevant to this perspective

Viewpoints & views



Overview - example



Recommendations for Architecture Description

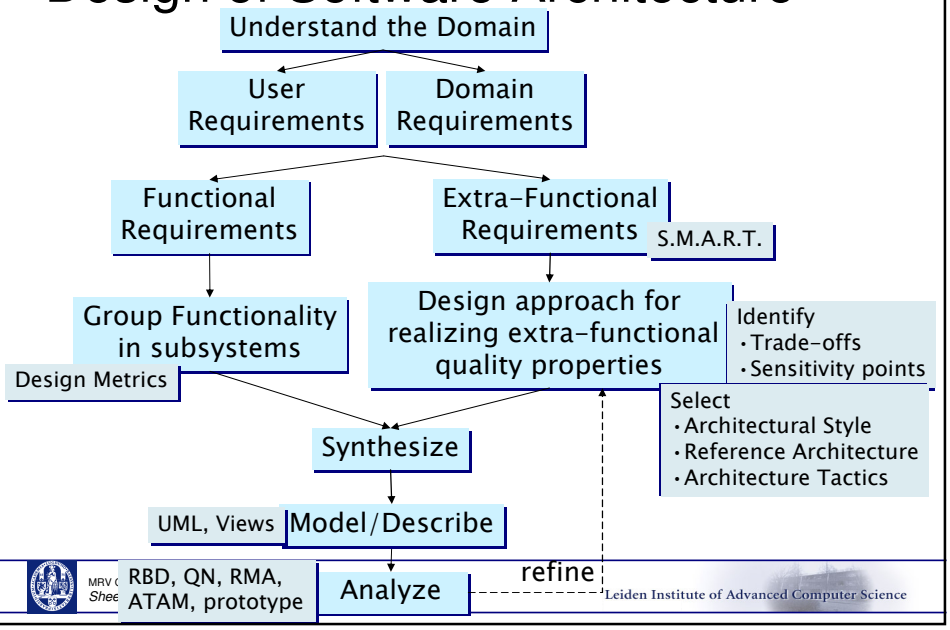
- describe the system **goals** & the **assumptions on the environment**
- describe the design **principles, decisions, guidelines**
  - and their **rationale**
- describe **several views** that can be combined in a consistent model
  at least the following views should be given:
  - **functional/structural (decomposition) view**
    - include a description of the interfaces between (sub)systems
  - **process/dynamical/behaviour view**
  - **deployment view**
- prevent mixing of views
- address **non-functional** (*ilities) aspects
- use a well-defined notation and include its **key/legend**
  - this aids systematic use of notation/avoids inconsistent use
  - improves common understanding
  - prevents mixing of different levels of abstraction
- add explanation in **natural language**

Concluding Remarks

Experience is the hardest kind of teacher.
It gives the test first and the lesson afterward.
Susan Ruth, 1993

- Software Architecture is a critical aspect in the design and development of software
- We discussed definitions and objectives of Sw.Arch.
- Good architectural design requires human creativity, hard work, and a critical attitude.
- Understanding of basic principles of architecture design, analysis, documentation, and process are necessary, but experience is hard to beat.

Design of Software Architecture



WWW References Software Architecture

- **Software architecture resources (Gert Florijn, Serc)**
  http://www.serc.nl/people/florijn/interests/arch.html
- **Software Architecture at the SEI**
  http://www.sei.cmu.edu/ata/ata_init.html
  inspired by practice; focus on architecture evaluation; lots of papers
- **Software Architecting Process / Success Factors & Pitfalls**
  http://www.bredemeyer.com/howto.htm
- **Architectural Blueprints: the 4+1 view model of Software Architecture** http://www.rational.com/media/whitepapers/Pbk4p1.pdf.
  The original paper by Kruchten: nice examples, but old (pre-UML) notation

Questions

Recommendations for Architecture Description

- describe the system **goals** & the **assumptions on the environment**
- describe the design **principles, decisions, guidelines**
  - and their **rationale**
- describe **several views** that can be combined in a consistent model
  at least the following views should be given:
  - **functional/structural (decomposition) view**
    - include a description of the interfaces between (sub)systems
  - **process/dynamical view**
  - **deployment view**
- prevent mixing of views
- address **non-functional** (*ilities) aspects
- use a well-defined notation and include its **key/legend**
  - this aids systematic use of notation/avoids inconsistent use
  - improves common understanding
  - prevents mixing of different levels of abstraction
- add explanation in **natural language**

Zachman Enterprise Architecture Framework

1987, extended: 1992

	DATA *What*	FUNCTION *How*	NETWORK *Where*	PEOPLE *Who*	TIME *When*	MOTIVATION *Why*	
SCOPE (CONTEXTUAL) *Planner*	List of Things Important to the Business ENTITY = Class of Business Thing	List of Processes the Business Performs Function = Class of Business Process	List of Locations in which the Business Operates Node = Major Business Location	List of Organizations Important to the Business People = Major Organizations	List of Events Significant to the Business Time = Major Business Event	List of Business Goals/Strat Ends/Means=Major Bus. Goal/ Critical Success Factor	SCOPE (CONTEXTUAL) *Planner*
ENTERPRISE MODEL (CONCEPTUAL) *Owner*	e.g. Semantic Model Ent = Business Entity Reln = Business Relationship	e.g. Business Process Model Proc. = Business Process I/O = Business Resources	e.g. Business Logistics System Node = Business Location Link = Business Linkage	e.g. Work Flow Model People = Organization Unit Work = Work Product	e.g. Master Schedule Time = Business Event Cycle = Business Cycle	e.g. Business Plan End = Business Objective Means = Business Strategy	ENTERPRISE MODEL (CONCEPTUAL) *Owner*
SYSTEM MODEL (LOGICAL) *Designer*	e.g. Logical Data Model Ent = Data Entity Reln = Data Relationship	e.g. Application Architecture Proc. = Application Function I/O = User Views	e.g. Distributed System Architecture Node = I/S Function (Processor, Storage, etc) Link = Line Characteristics	e.g. Human Interface Architecture People = Role Work = Deliverable	e.g. Processing Structure Time = System Event Cycle = Processing Cycle	e.g., Business Rule Model End = Structural Assertion Means = Action Assertion	SYSTEM MODEL (LOGICAL) *Designer*
TECHNOLOGY MODEL (PHYSICAL) *Builder*	e.g. Physical Data Model Ent = Segment/Table/etc. Reln = Pointer/Key/etc.	e.g. System Design Proc.= Computer Function I/O = Data Elements/Sets	e.g. Technology Architecture Node = Hardware/System Software Link = Line Specifications	e.g. Presentation Architecture People = User Work = Screen Format	e.g. Control Structure Time = Execute Cycle = Component Cycle	e.g. Rule Design End = Condition Means = Action	TECHNOLOGY MODEL (PHYSICAL) *Builder*
DETAILED REPRESEN-TATIONS (OUT-OF-CONTEXT) *Sub-Contractor*	e.g. Data Definition Ent = Field Reln = Address	e.g. Program Proc.= Language Stmt I/O = Control Block	e.g. Network Architecture Node = Addresses Link = Protocols	e.g. Security Architecture People = Identity Work = Job	e.g. Timing Definition Time = Interrupt Cycle = Machine Cycle	e.g. Rule Specification End = Sub-condition Means = Step	DETAILED REPRESEN-TATIONS (OUT-OF CONTEXT) *Sub-Contractor*
FUNCTIONING ENTERPRISE	e.g. DATA	e.g. FUNCTION	e.g. NETWORK	e.g. ORGANIZATION	e.g. SCHEDULE	e.g. STRATEGY	FUNCTIONING ENTERPRISE