Software Architecture

Short Introduction

Alar Raabe

Content

Software architecture is what software architects do

Kent Beck

- What is Software Architecture
  - Design vs. Architecture
  - Early Views and Software Architecture as Discipline
  - Software Architecture related Concepts and Terminology
- Group Work
- Software Architectural Styles
  - Short overview
  - Example software architectural style – Dataflow Systems
- Discussion
  - Our common language
  - Value of architecture for us

Architecture

Architecture is about
→ Durability
→ Utility
→ Beauty

Vitruvius

- Merriam-Webster
  - architecture is the art or science of building
  - **unifying or coherent structure**
  - **the manner in which the components of the system are organized or integrated**
- Wikipedia
  - The term architecture (from Greek word αρχιτεκτονική, pronounced *architektonike*) can refer to
    - a process – the activity of designing and constructing any kind of system,
    - a profession – the role of those persons or machines providing architectural services, or
    - documentation – usually based on drawings, **architecture defines the structure and/or behavior of a system that is to be or has been constructed**

Design vs. Architecture

All architecture is design but not all design is architecture

Grady Booch

- Design = Plan
  - adaptation of means to ends
- Software Design can be viewed on many levels
  - design of higher levels is architecture for lower levels
- Booch
  - architecture represents the significant design decisions that shape a system, where *significant is measured by cost of change*
- Eden
  - architectural decisions are non-local intensional design decisions

Non-Local	*Intensional*	*Architecture*
Local	*Intensional*	*Design*
Local	*Extensional*	*Implementation*

Early Views on Software Architecture

Structure Matters!

- Turing & Wheeler (1946-50)
  - subroutine library (reusability, reliability, unit testing (testability), multiple versions with different non-functional qualities, ...)
- Brooks & Iverson (1964-69)
  - architecture is a conceptual structure
  - architecture is the complete and detailed specification of the user interface
- Dijkstra, Parnas & Jackson (1972-76)
  - separation of concerns – isolation, encapsulation, modularization
  - program families can be described by a decision trees
  - *structure influences non-functional 'qualities' of systems*
  - *structure of program is defined by domain structures*

Software Architecture as Discipline

elements + form/structure + rationale/principles

- Perry and Wolf (1992)
  - a set of architectural (or, if you will, design) elements that have a particular form (processing, data, and connecting elements)
  - the structure of the components of a program/system, their interrelationships, and principles and guidelines governing their design and evolution over time
- Garlan and Shaw (1994)
  - the organization (structure) of the overall system (incl. gross organization and global control structure; protocols for communication, synchronization, and data access; assignment of functionality to design elements; physical distribution; composition of design elements; scaling and performance; and selection among design alternatives)

Software Architecture as Discipline

global design constraints

- Bass, Clements, Kazman (1997)
  - the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them

- Eden, Kazman (2003)
  - strategic design statements (global design constraints like programming paradigms, architectural styles, component-based software engineering standards, design principles, and law-governed regularities)

Agile Software Architecture

the important stuff – whatever that is

Ralph Johnson

- Beck (1992)
  - what the software architects do
- Johnson (...)
  - a shared understanding of the system design of the expert developers working on the project (incl. how the system is divided into components and how the components interact through interfaces)
  - the decisions that you wish you could get right early in a project
- Fowler (2003)
  - a word we use when we want to talk about design but want to puff it up to make it sound important

Software Architecture Standards

≠ architectural description

- Open Group TOGAF 9 Enterprise Architecture Framework
  - a formal description of a system, or a detailed plan of the system at component level to guide its implementation
  - the structure of components, their interrelationships, and the principles and guidelines governing their design and evolution over time
- IEEE 1741-2000 | ISO/IEC 42010:2007 Systems and Software Engineering – Architecture description
  - the fundamental conception of a system in its environment embodied in elements, their relationships to each other and to the environment, and principles guiding system design and evolution

Architecture Descriptions are for

ISO/IEC 42010:2007

- Communicating
  - the system's architecture throughout its life cycle among the system's stakeholders, to guide desired and acceptable change

- Planning and Managing
  - the activities of system development
  - the effective utilization of a system's elements and resources throughout its life cycle (*operations*)

- Evaluating
  - and comparing of systems architectures in a consistent manner
  - completeness, consistency and correctness of requirements
  - and verification of a system's implementation for compliance with its intended architecture

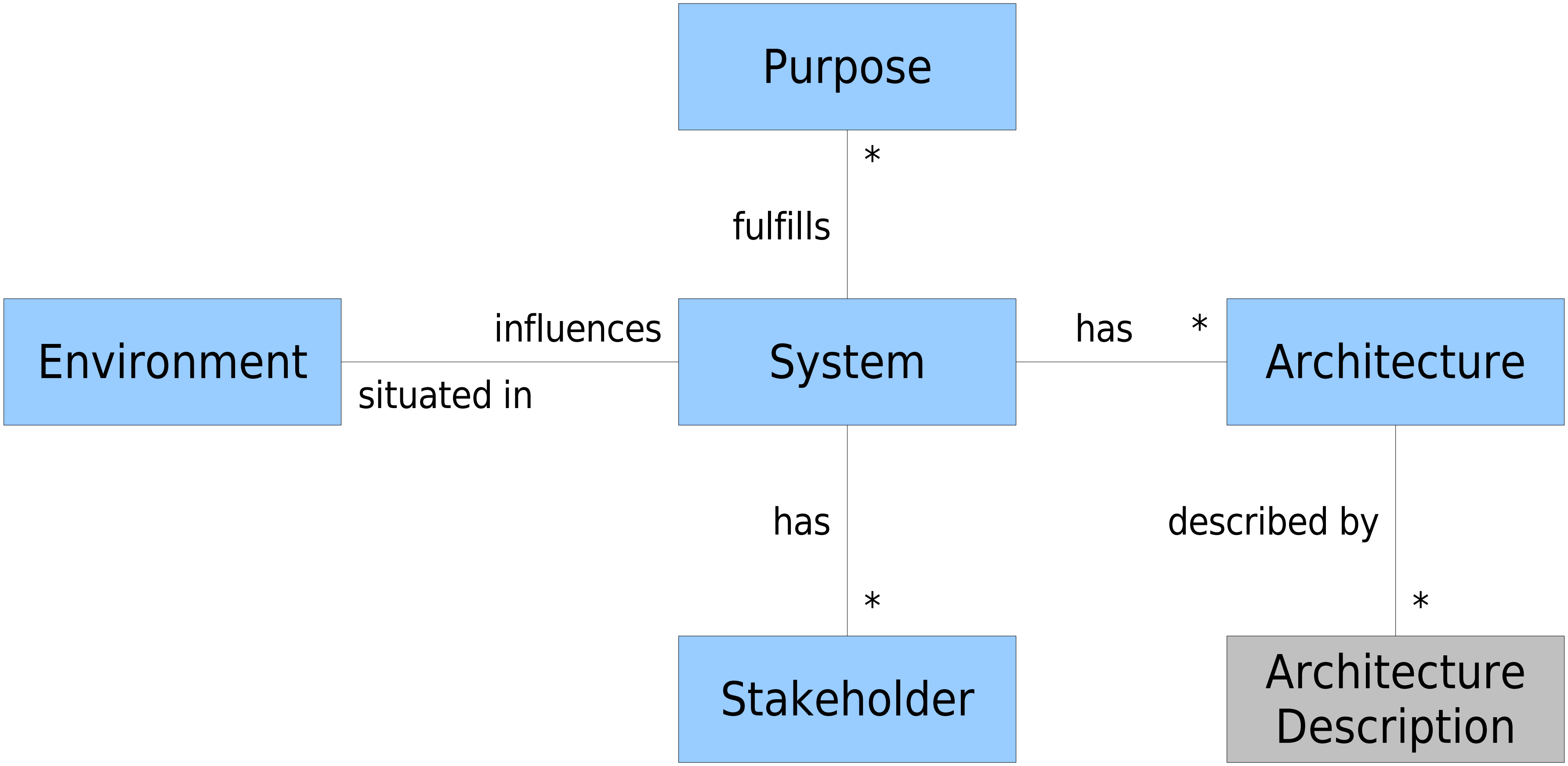
Terms (Glossary)

ISO/IEC 42010:2007

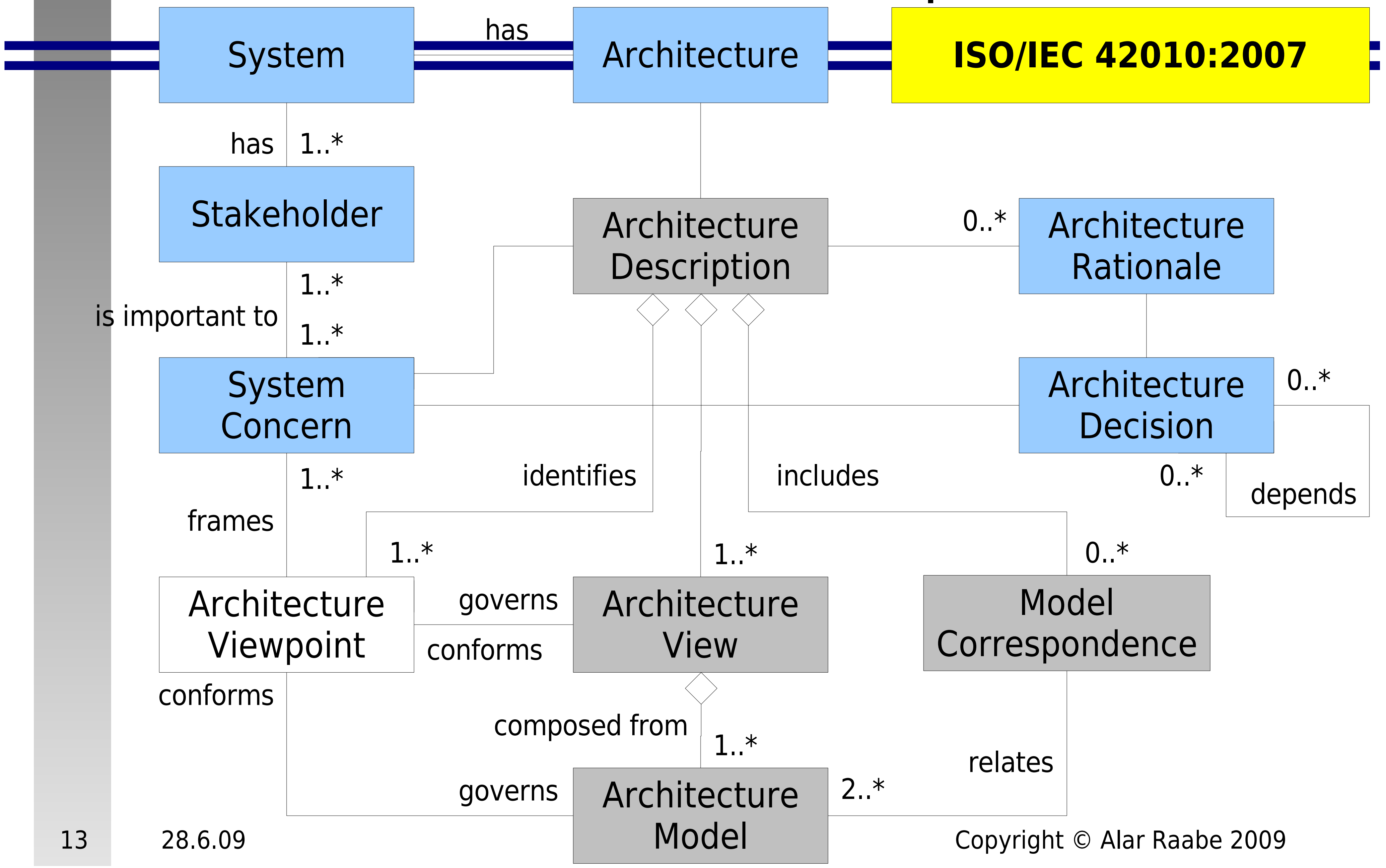
architecture	fundamental conception of a system in its environment embodied in elements, their relationships to each other and to the environment, and principles guiding system design and evolution
architecture decision	choice made from among possible options that addresses one or more architecture-related concerns
architecture description	collection of work products used to describe an architecture
architecture model	work product from which architecture views are composed
architecture rationale	explanation or justification for an architecture decision
architecture view	work product representing a system from the perspective of architecture-related concerns
architecture viewpoint	work product establishing the conventions for the construction, interpretation and use of architecture views
environment	context determining the setting and circumstances of developmental, technological, business, operational, organizational, political, regulatory, social and any other influences upon a system
model correspondence	relation on two or more architecture models
stakeholder	individual, team, organization, or class thereof, having concerns with respect to a system
purpose	*{one of system concerns}*
system	*{a conceptual entity defined by its boundaries}*
system concern	area of interest in a system pertaining to developmental, technological, business, operational, organizational, political, regulatory, social, or other influences important to one or moreof its stakeholders

System and Architecture

ISO/IEC 42010:2007

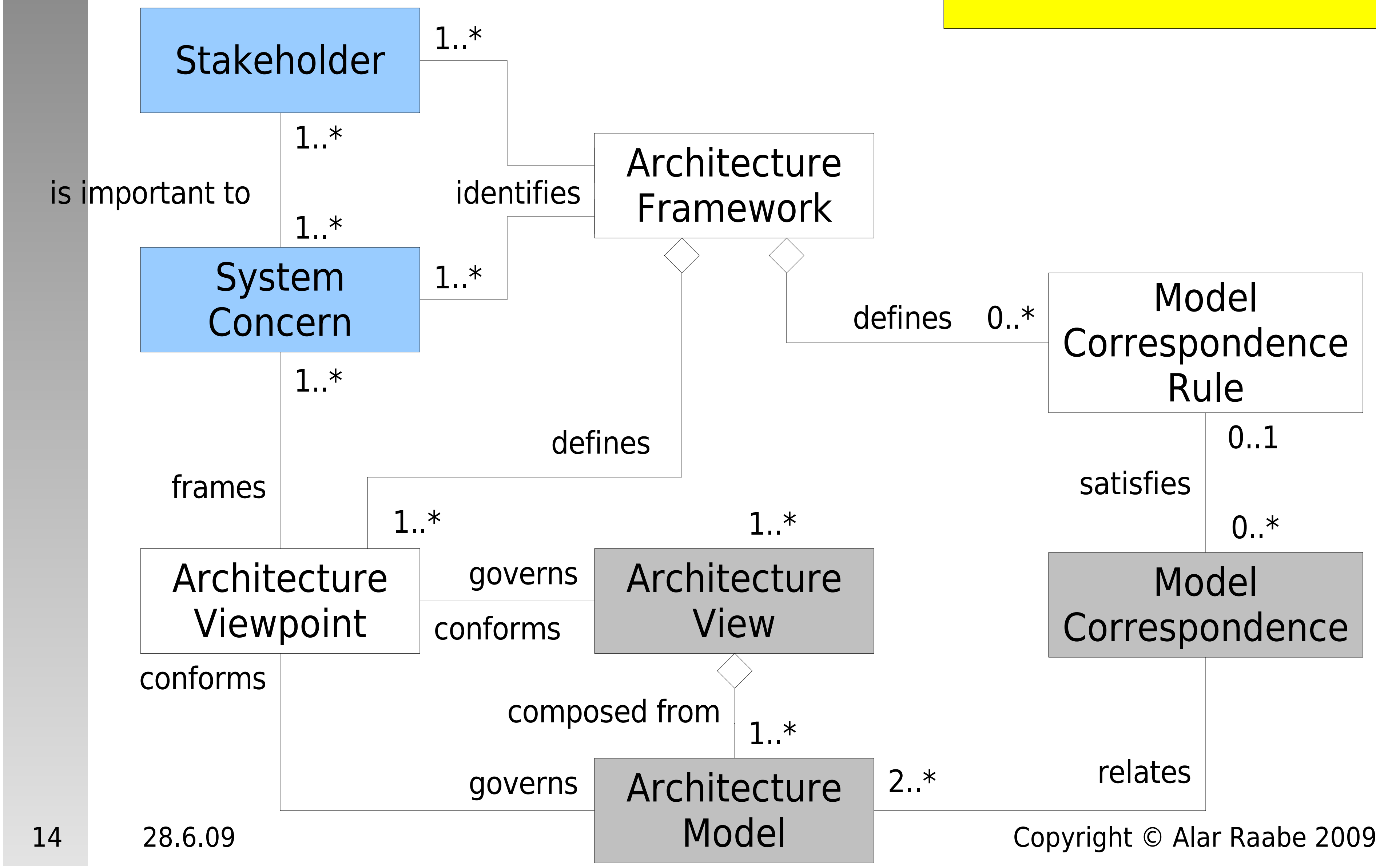


Architecture Description



Architecture Framework

ISO/IEC 42010:2007



Stakeholders

ISO/IEC 42010:2007

- users and operators of the system
- acquirers and owners of the system
- suppliers and developers of the system
- builders and maintainers of the system

System Concerns

ISO/IEC 42010:2007

- the purpose of the system
- the suitability of the architecture for achieving the system's purposes
- the feasibility of constructing the system
- the potential risks of the system to its stakeholders throughout its life cycle
- maintainability, deployability, and evolvability of the system

Architecture Decisions

ISO/IEC 42010:2007

- decisions regarding architecturally significant requirements
- decisions needing a major investment of effort or time to make
- decisions affecting key stakeholders or a number of stakeholders
- decisions needing intricate or non-obvious resoning
- decisions that are highly sensitive to changes
- decisions that could be costly to change

What is (Software) Architecture

- (Software) Architecture is a
  - **fundamental conception** of a (software) system in its
  - **environment** embodied in its
  - **elements**,
  - their **relationships** to each other and to the environment, and
  - **principles** guiding (software) system design and evolution

elements that correspond to system concerns (e.g. utility, cost, risk, ...)

- (Software) Architecture description is a
  - collection of **models** in **correspondence** (relations),
  - organized into *synthetic* or *projective* **views** (cohesive groups, defined by **viewpoints**) according to te **concerns** addressed

- (Software) System Model
  - **anything** that can be used to answer questions about system

Example: Subscription-Based Sensor Collection Service

- stakeholders
  - users, developers, operators
- concerns (by stakeholders)
  - ROI (operators)
  - timely delivery of sensor data (users)
  - understanding of interactions between system elements (developers)
- viewpoints (by concerns)
  - financial : cash-flow spreadsheet (ROI)
  - operational : timeline diagram (timely delivery of sensor data)
  - system : system interface diagram (understanding of interactions between system elements)
- views (by viewpoints)
  - profit spreadsheet & profitability curve (cash-flow spreadsheet)
  - timeline diagram (timeline diagram)
  - dataflow diagram (system interface diagram)
- view consistency and correspondence rules
  - each node in dataflow diagram should appear at least once in timeline diagram

Group Work: CDI-Hub

- stakeholders
- concerns
- viewpoints
- *views*
- *model correspondence rules*
- *rationale*
- *decisions*

Software Architectural Styles

- Different Levels of Commonality: Idioms, Patterns, Styles
- What is a Software Architectural Style
- Examples of Different Software Architectural Styles

Different Levels: Idioms, Patterns, Styles

reuse of (design) knowledge

- Specific to a (programming) language
  - Software Idioms – coding/programming
    - describe usage of (programming) language for certain (simple) problems
  - Programming Style – programming
    - a consistent set of idioms (e.g. fluent style, functional style)

- (Programming) language independent
  - Design Patterns – design
    - describe standard solutions to certain common problems
  - Architecture Styles – architectural design
    - specific vocabulary and rules for architectural design
    - defines a class of systems with specific properties
    - describes standard solution to a class of problems

What is a Software Architectural Style

- Characterizes a family of systems that are related by shared structural and semantic properties

- Defines
  - a vocabulary of design elements
  - design rules, or constraints (incl. topology)
  - semantic interpretation
  - analyses that can be performed on systems built in that style

Examples of Architectural Styles

SOA conform to Independent Components

- Dataflow Systems
  - Batch sequential, Pipes and filters
- Call-and-return Systems (*explicit calls*)
  - Main program and subroutines, OO systems, Hierarchical layers
- Independent Components (*implicit calls*)
  - Communicating processes, Event Systems
- Virtual Machines
  - Interpreters, Rule-based systems
- Data-Centered Systems (Repositories)
  - Databases, Hypertext system, Blackboards

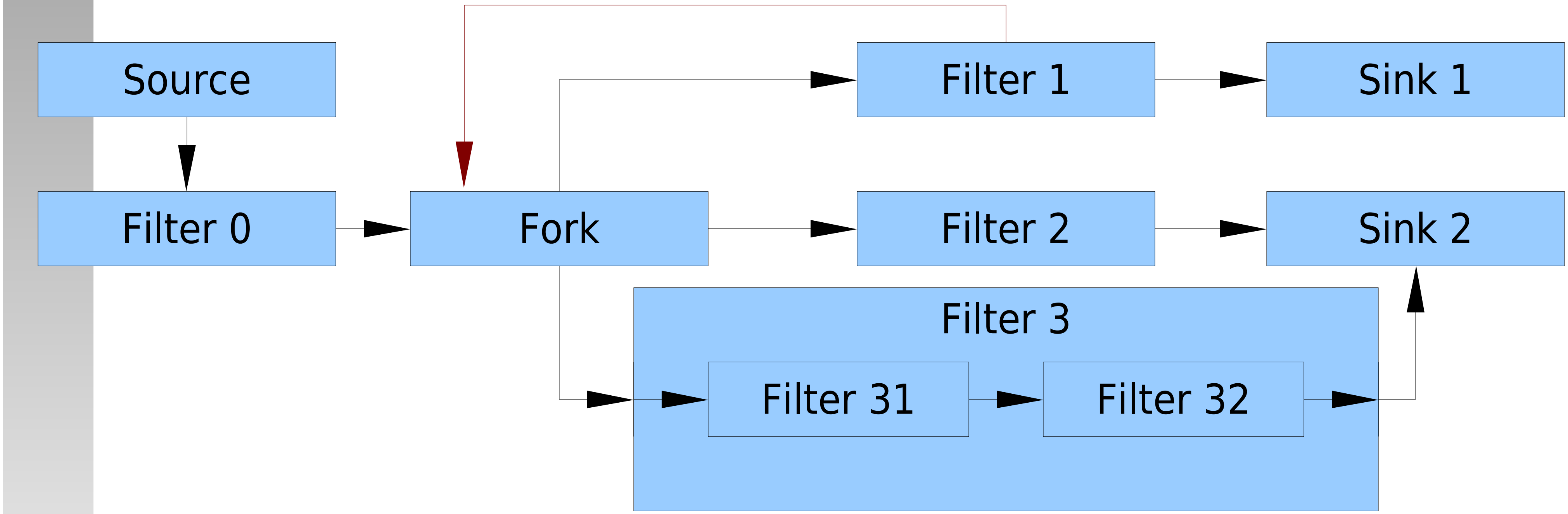
Benefits of Architectural Style

- Design Reuse
  - Well-understood solutions applied to new problems
- Code reuse
  - Shared implementations of invariant aspects of a style
- Understandability of system organization
  - A phrase such as ‘client-server” conveys a lot of information
- Interoperability
  - Supported by style standardization
- Style-specific analysis
  - Enabled by the constrained design space
- Visualizations
  - Style-specific descriptions matching engineer’s mental models

Dataflow Systems

shared nothing!

- Dataflow Systems – Pipes and Filters
  - Components (sources, filters, sinks)
  - Connectors (pipes)
  - Constraints (is feedback allowed or not, are pipes buffering, ...)
  - Theory (Queueing Theory (K. Erlang 1909))



Dataflow Systems Advantages (1)

- Modifiability & Reuse (low coupling, encapsulation)
  - filters stand alone and can be treated as black boxes
  - filters interact with other components in limited ways
- Ease of construction
  - systems can be hierarchically composed – higher order filters can be created from any combination of lower order pipes and filters
- Flexibility
  - the construction of the pipe and filter sequence (system configuration) can often be delayed until runtime (late binding)

Dataflow Systems Advantages (2)

- Run-time scalability
  - because the process performed by the filter is isolated from the other components in the system, it is easy to run a pipe-and-filter system on parallel processors
- Understandability/Analyzability
  - system behavior is a succession of component behaviors
  - support certain analyses (throughput, latency, deadlock)

Dataflow Systems Disadvantages

- Difficult to create interactive applications
  - because the problem is decomposed into sequential steps
- Common data representation
  - data has to be represented as the lowest common denominator (typically byte or character streams)
- Parsing overhead
  - if processing must be based on information, every filter may introduce parsing and unparsing of the data stream
- Unknown memory requirements and deadlock possibility
  - if a filter can not produce any output until it has received all of its input, the filter will require a buffer of unlimited size
  - if fixed size buffers are used, the system could deadlock (e.g. sort filter has this problem)
- Data sharing is difficult

Dataflow Systems Examples

- Batch systems
- Many compilers
- Unix pipelines
- Spreadsheets
- JDPF (Java Data Processing Framework)
- Apache Camel (?)

Discussion

What is/are for us ...

- Concepts of
  - (Software) System
  - (Software) Architecture
  - (Software) Architecture Description
- Value of
  - (Software) Architecture
  - (Software) Architecture Description
- Most Relevant (Software) Architecture Styles
- Main Stakeholders
- Main System Concerns
- (Software) Architecture Framework

Conclusion

- Value of (Software) Architecture
  - as fundamental conception of (software) system, architecture allows us to reason (answer questions) about the (software) system
  - as specific architectural styles address certain concerns (cause certain properties/qualities) of (software) systems, architecture allows us to address concerns (achieve required properties or qualities) of (sofware) systems

- Value of Architecture Description
  - as document, it provides guidance for constructing and evolving the (software) system, and allows us to record and communicate our knowledge and decisions about the (software) system architecture
  - as model, it allows us to reason (answer questions) about the (software) system architecture

Thank You!

Leftovers

- Conway's law (1968)
  - organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations

Definitions $_1$

- System
  - a collection of interacting components organized to accomplish a specific function or set of functions within a specific environment
- Interface (Connection)
  - a shared boundary between two functional units, defined by various characteristics of the functions
  - component that connects two or more other components for the purpose of passing information from one to the other
- Module (Component)
  - a logically separable part of a system
- Encapsulation
  - isolating some parts of the system from the rest of the system
  - a module has an outside that is distinct from its inside (an external interface and an internal implementation)

Definitions $_2$

- Modularity
  - the degree to which a system is composed of discrete components such that a change to one component has minimal impact on other components
  - the extent to which a module is like a black box
- Coupling
  - the manner and degree of interdependence between modules
  - the strength of the relationships between modules
  - a measure of how closely connected two modules are
- Cohesion
  - the manner and degree to which the tasks performed by a single module are related to one another
  - a measure of the strength of association of the elements within a module

Definitions $_3$

- Model
  - an interpretation of a theory for which all the axioms of the theory are true
  - a semantically closed abstraction of a system or a complete description of a system from a particular perspective
  - anything that can be used to answer questions about system
    - to an observer B, an object M_A is a model of an object A to the extent that B can use M_A to answer questions that interest him about A

      Marvin Minsky
    - M is a model of A with respect to question set Q if and only if M may be used to answer questions about A in Q within tolerance T

      Doug Ross