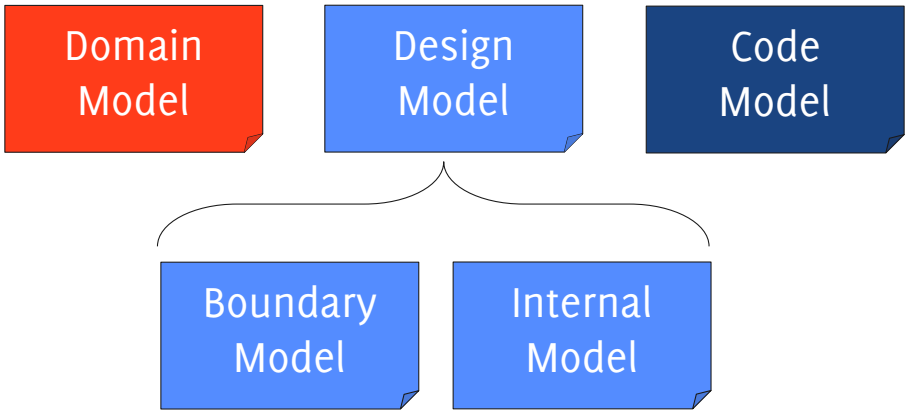
Università della Svizzera italiana

Faculty of Informatics

Software Architecture Modeling

Prof. Cesare Pautasso
http://www.pautasso.info
cesare.pautasso@usi.ch
@pautasso

Canonical Models



Domain Model

- Problem domain description:
  - Information (invariants, navigation, snapshots)
  - Functionality (use-case scenarios)
- Main concerns: usability and interoperability
- Define shared vocabulary and understanding
- **Avoid analysis paralysis**: stop modeling the domain when all your questions about the problem have been answered by the domain experts

Example Domain Model

- Music songs are organized in albums
- The same song can be authored by many artists
- Listening to each song costs 0.99 CHF, but short samples can be heard for free
- Songs can be downloaded and also live streamed
- Songs are stored in files of standard MP3 format
- Files contain embedded metadata and watermarks
- A music player can carry 10'000+ albums

Design Model

- Interfaces:
  - Externally visible behavior
  - Interchange data
- Quality Attributes
- System Context
- Refinement of the boundary model
- Component assembly
- Internal behavior

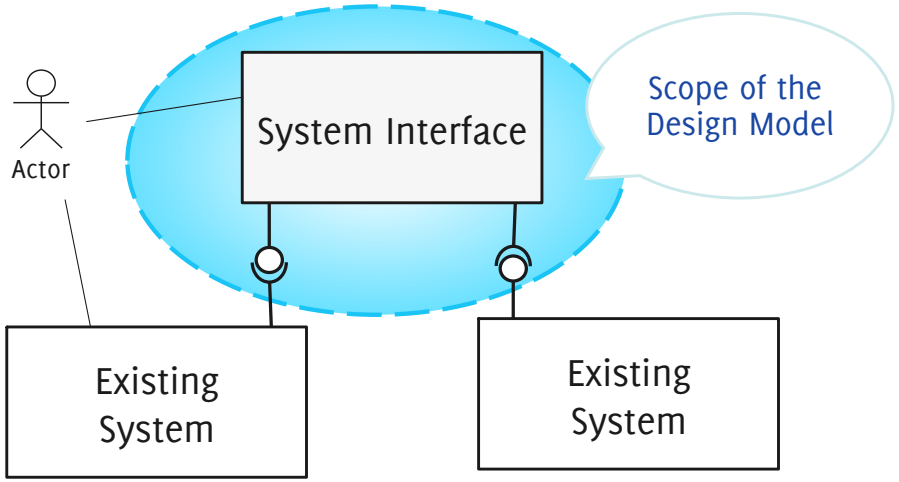
Example Boundary Model

- Streamed songs should begin to play after a max delay of 5 seconds
- Payment messages should be transferred with an encrypted standard protocol
- Songs are stored in files of standard MP3 format
- Songs can be downloaded and also live streamed
- Playlists cannot be modified while they are being played

Example Internal Model

- The playback thread should have a high priority
- Sound decoder component built with a MP3 library
- Payment messages should be processed with a single-pass parser
- Playlists are programmed with Arrays to simplify random access

System Context



- Distinguish what needs to be built from what already exists and define the interconnection points

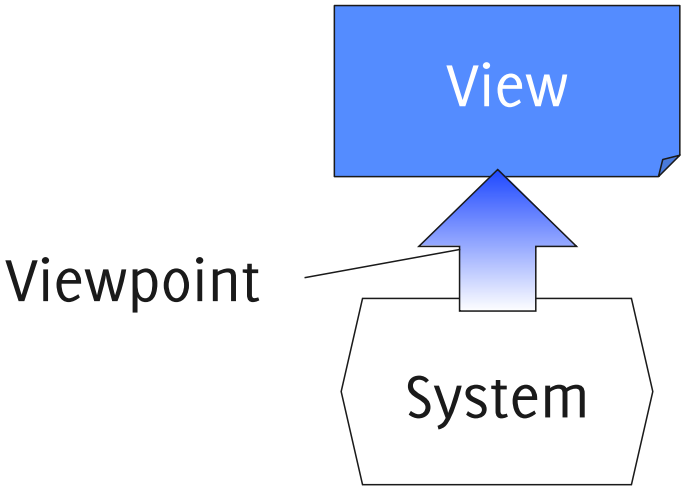
Scope of the Model

1. Parts of the system for different use cases
2. The architecture of the entire system
3. The style of the system (constrain the actual architecture without describing it)

- Note: *Add details to the model only where they are needed to minimize risk*

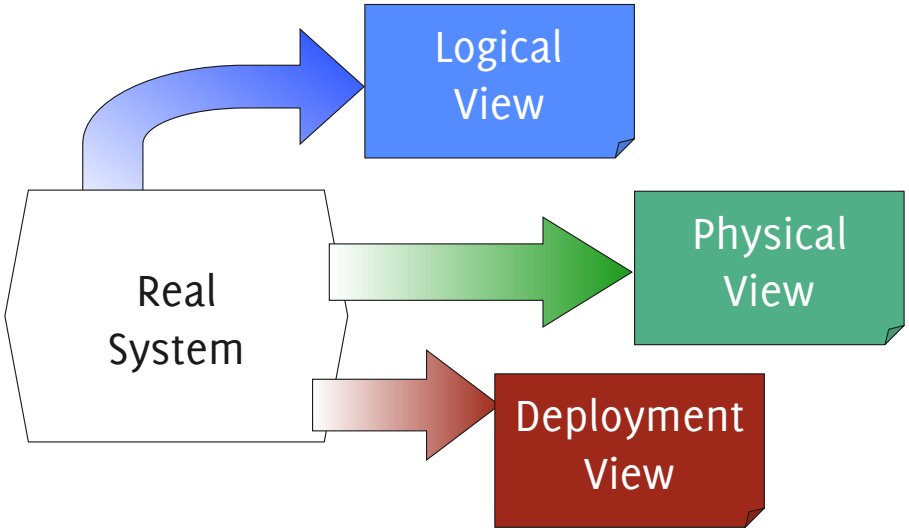
What is a view?

- No single modeling approach can capture the entire complexity of a software architecture
- Various parts of the architecture (or views) may have to be modeled with a different:
  - Notation
  - Level of detail
  - Target Audience
- A **view** is a set of design decisions related by common concerns (the viewpoint)



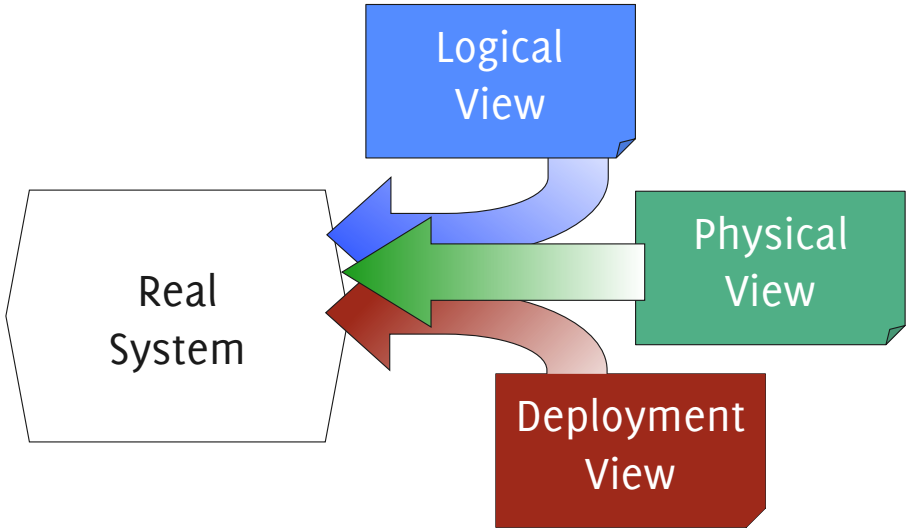
Multiple Views

- There is too much information to model: we need multiple views



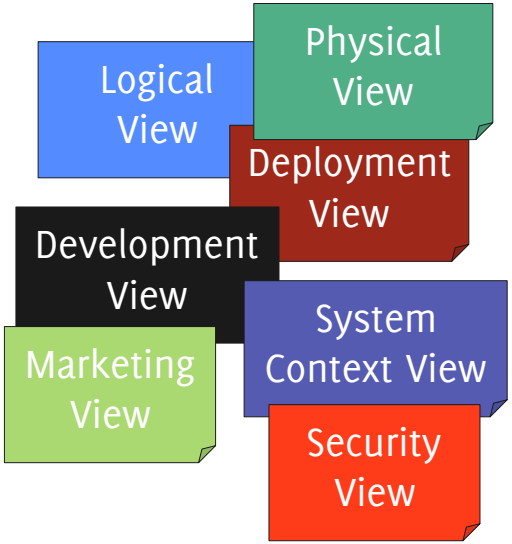
View Consistency

- Views are not always orthogonal and should not become inconsistent

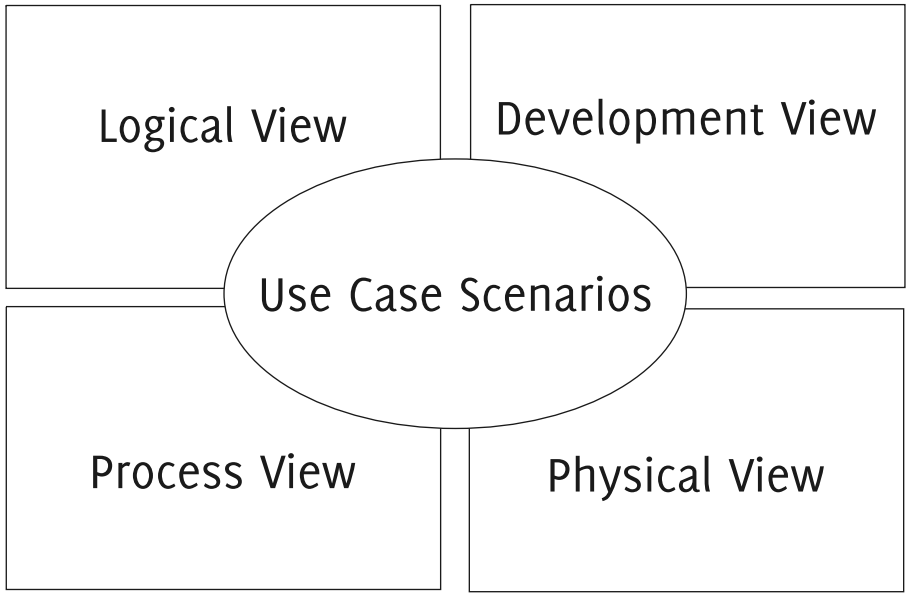


How many views?

- Choose the right set of views to model the architecture of the specific system.
- Each design may use a different set of views
- You do not have to always use all views for every project



4+1



Use Case Scenarios

- The modeling elements of the 4 views are linked by scenarios that show how they work together to fulfill the use cases of the system
- The actual model of the architecture can be broken down in scenarios, each illustrated using the other views
- Scenarios help to ensure that the architectural model is complete with respect to requirements
- Scenarios can be prioritized to help driving the development of the system (most critical for success, most expensive to build, most useful, most risky) according to different stakeholders expectations

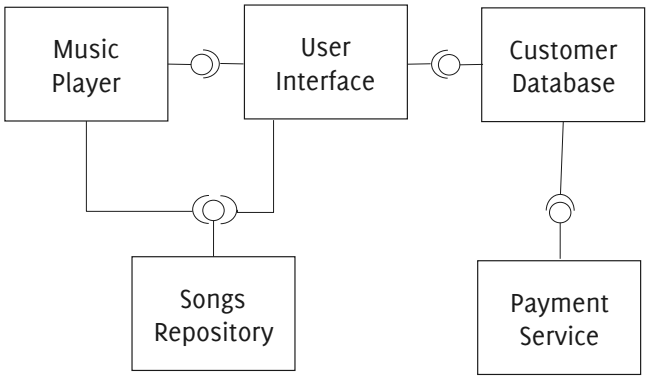
Music Player Scenarios

1. Browse for new songs
2. Search for interesting songs
3. Play the song sample
4. Pay to hear the entire song
5. Download the purchased song on the device
6. Play the song
7. Play multiple songs on a predefined playlist
8. Play multiple songs in random order
9. Share songs with friends
10. Make a backup of the device's content
11. Suggest related songs
12. Generate a tasteful playlist
13. Display album cover image
14. Show the device's battery status
15. Record sounds with a microphone

Logical View

- Decompose the system structure into software components and connectors
- Map functionality/requirements/use cases onto the components
- Concern: Functionality
- Target Audience: Developers and Users

Example Logical View

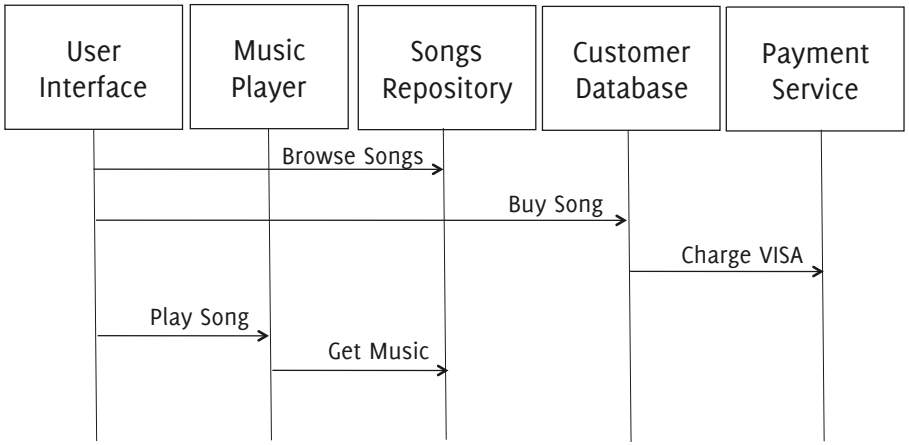


Process View

- Model the dynamic aspects of the architecture:
  - Which are the active components?
  - Are there concurrent threads of control?
  - Are there multiple distributed processes in the system?
  - What is the behavior of (parts of) the system?
- Describe how processes/threads communicate (e.g., RPC, Messaging connectors)
- Concern: Functionality, Performance
- Target Audience: Developers

Example Process View

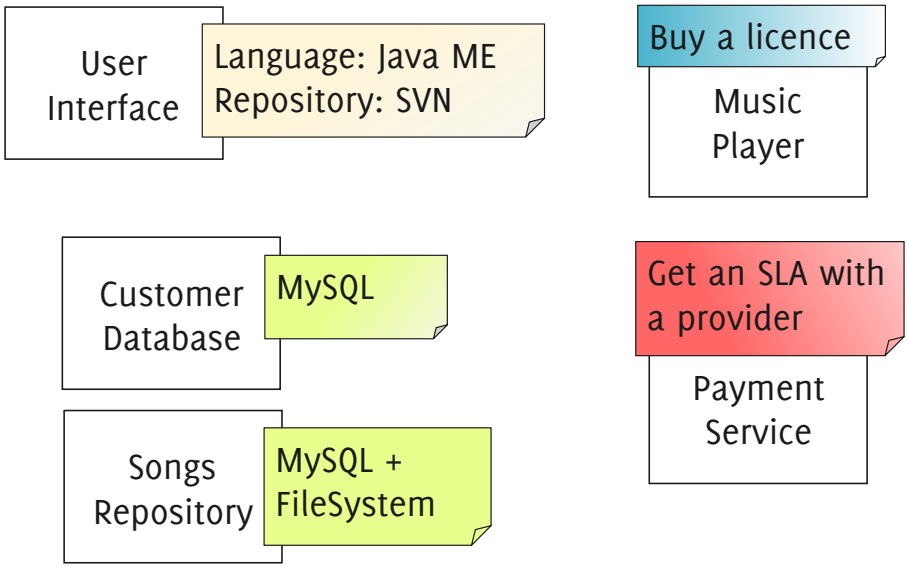
Use Cases: Browse, Pay and Play For Songs



Development View

- Static organization of the software code artifacts (packages, modules, binaries…)
- A mapping between the logical view and the code is also required
- Concern: Reuse, Portability, Build
- Target Audience: Developers

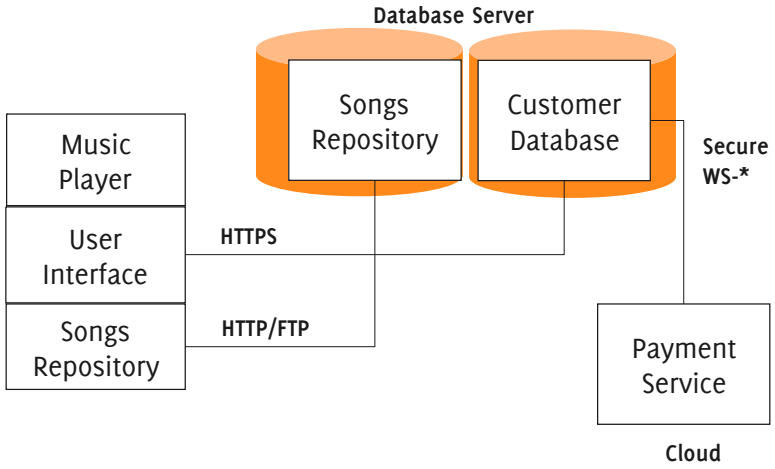
Example Development View



Physical View

- Define the hardware environment (hosts, networks, storage, etc.) where the software will be deployed
- Different hardware configurations may be used for providing different qualities
- A Mapping between logical and physical entities is also necessary (sometimes found in a separate Deployment View)
- Concern: Performance, Scalability, Availability, Reliability
- Target Audience: Operations

Example Physical View



References

- Michael Jackson, Problem Frames: Analyzing and structuring software development problems, Addison-Wesley, 2001
- Richard N. Taylor, Nenad Medvidovic, Eric M. Dashofy, Software Architecture: Foundations, Theory and Practice, John-Wiley, January 2009
- Philippe Kruchten, Architectural Blueprints—The "4+1" View Model of Software Architecture, IEEE Software 12 (6). November 1995, pp. 42-50
- Scott W. Ambler, Agile Modeling, http://www.agilemodeling.com/
- IEEE Software, Twin Peaks of Architecture and Requirements (http://www.computer.org/portal/web/computingnow/software/multimedia/-/blogs/march-april-2013:-twin-peaks-of-architecture-and-requirements) , March/April 2013