CpSc 875

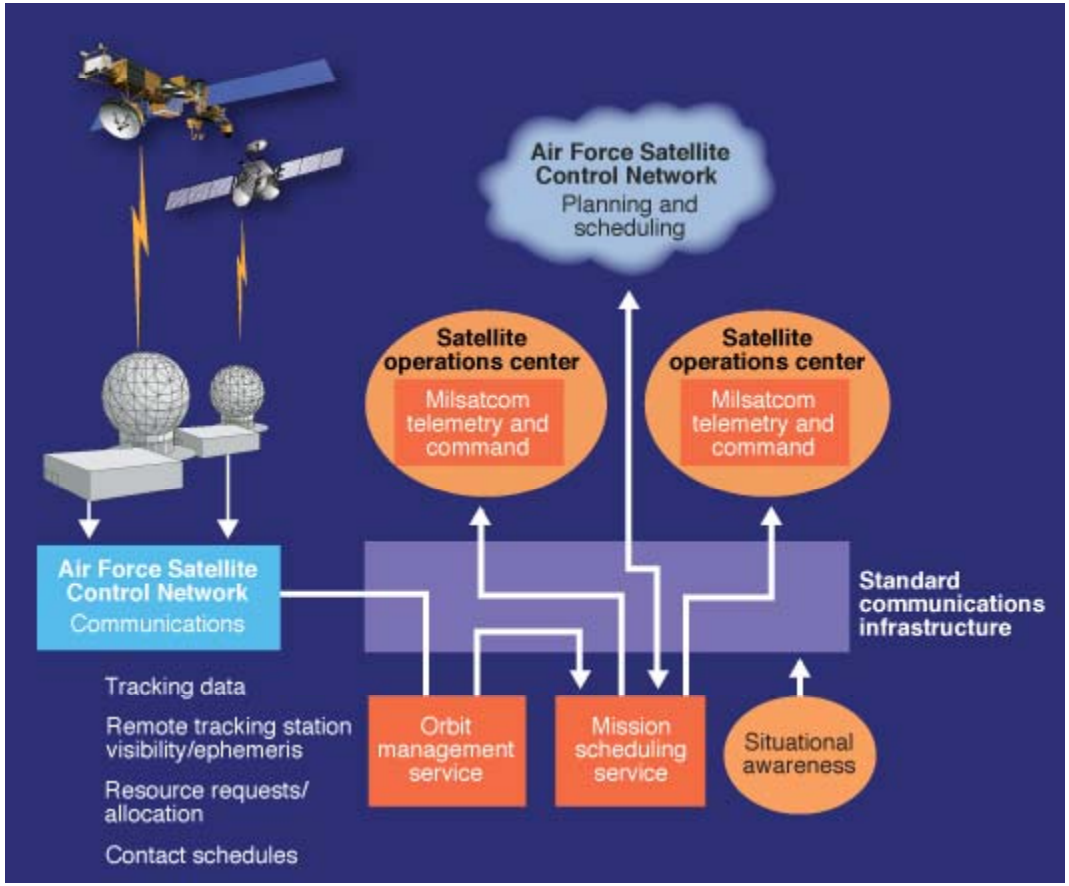
John D. McGregor

Initial decomposition

C3

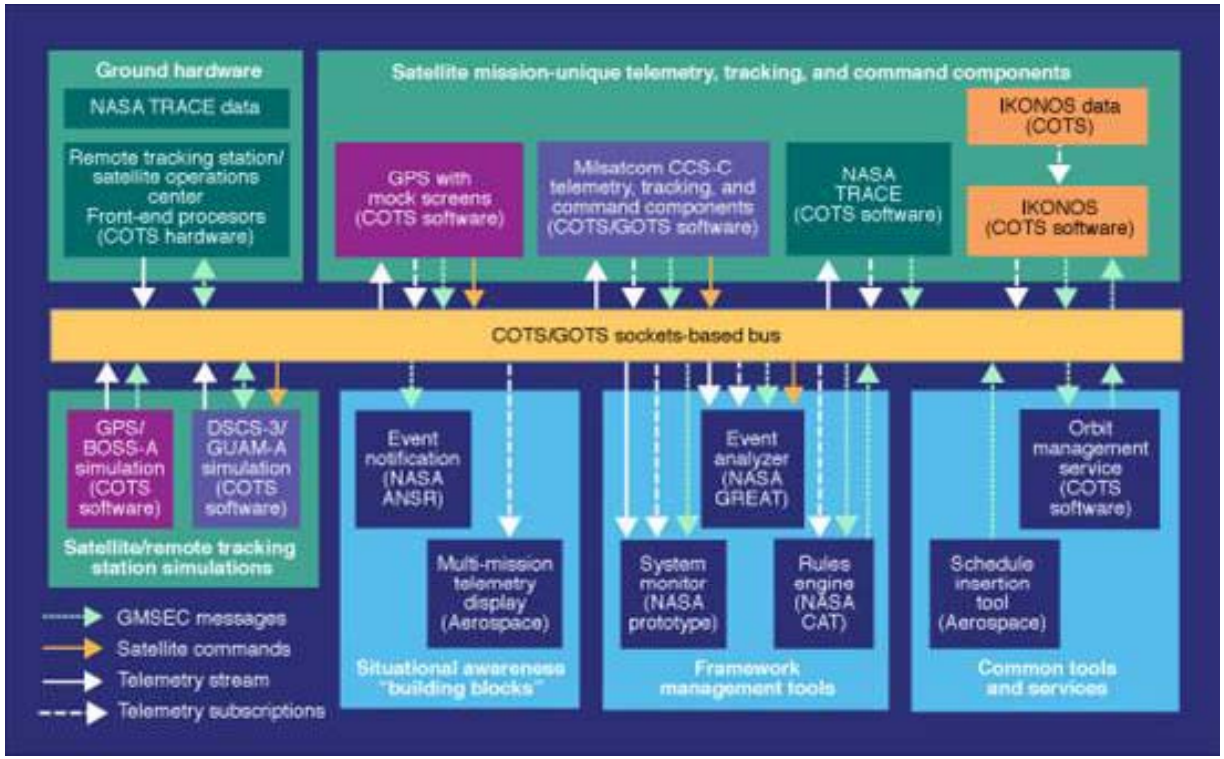
Ground station

- Relays signals between satellites and ground controllers
- It has to be: very fast and very reliable.



Example architecture

- Satellite ground station
- Bus architecture
- Any module on the bus can communicate with any other
- Modifiability +



Ground station design principles

- One look at qualities
- But there are more formal ones.

Architecture Principle	Definition	Actions
Utility	• The measure of usefulness of a capability provided to a customer (measure of benefit)	• Fully understand the capability needed (not solution preferred) • Focus on user requirements that flow from needed capability (e.g., on call deployment)
Interoperability	• The ability of two or more systems to exchange and mutually use information	• Adopt and implement common standards • Establish common requirements with mission partners
Flexibility	• The ease with which one can alter the architecture to include a capability to perform a new or unanticipated requirement without adding a component	• Adopt and implement common standards • Leverage research and technology • Investigate novel concept of operations for current systems
Adaptability	• The ability to add a new capability component to the architecture to perform a new or unanticipated requirement	• Adopt and implement common standards • Leverage research and technology • Look at commercial applications
Agility	• Measure of ability to make required changes to an architecture	• Break down barriers to agility (e.g., processes, authorities, etc.)

Quality attributes

- IEEE Std. 1061 subfactors:

Efficiency
- **Time economy**
- **Resource economy**

Functionality
- **Completeness**
- **Correctness**
- **Security**
- **Compatibility**
- **Interoperability**

Maintainability
- **Correctability**
- **Expandability**
- **Testability**

Portability
- **Hardware independence**
- **Software independence**
- **Installability**
- **Reusability**

Reliability
- **Non-deficiency**
- **Error tolerance**
- **Availability**

Usability
- **Understandability**
- **Ease of learning**
- **Operability**
- **Comunicativeness**

http://en.wikipedia.org/wiki/ISO/IEC_9126

Qualities

- Trade-offs - a trade off is when one quality degrades another quality as the first quality increases
  - Testability & modifiability
  - Performance and modularity
- Develop a catalog of trade-offs during this course

Perspectives on quality

- The executive
- The customer
- The developer
- The tester

Quality without a name

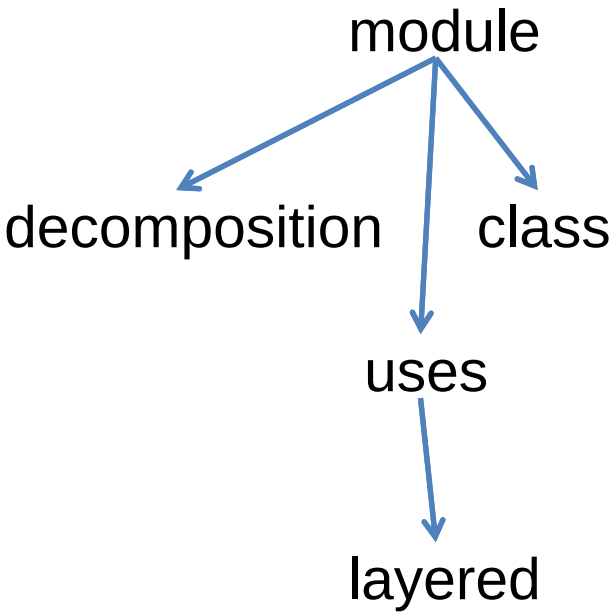
Naming something denotes certain properties more than others. By not putting into words what we see or feel about this scene we allow each viewer to emphasize what is important to them.

Standard architecture structures

- Module structures
  - Which piece is responsible for what?
  - How are pieces defined?
- Component and connector structures
  - How do the major pieces interact at runtime?
- Allocation structures
  - Associates pieces of the architecture with pieces of the external environment?

Module structures

- Decompose – module into sub modules. Pieces related to the whole
- Uses – one module expects another to be present
- Layered – decomposition in which there is an ordering
- Class – specialization relationships

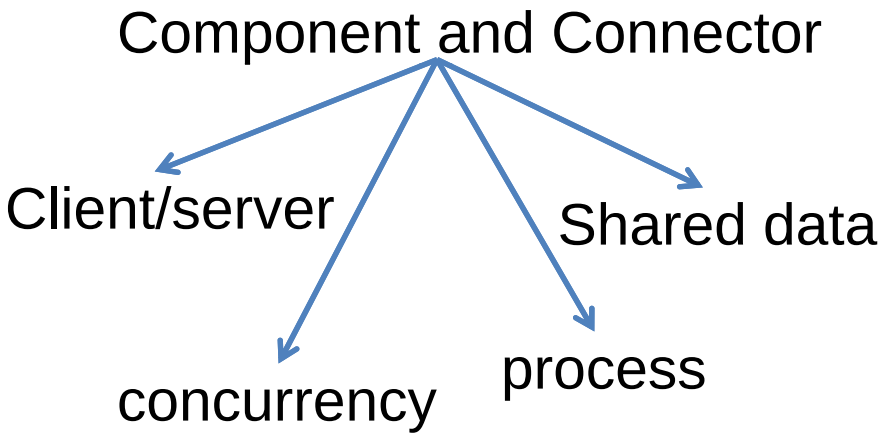


Decomposition

- Taking one big thing and making it several smaller things
- The relationships among these pieces determines what qualities the design enhances and which it degrades.
- Other operations such as combination also affect the product qualities.

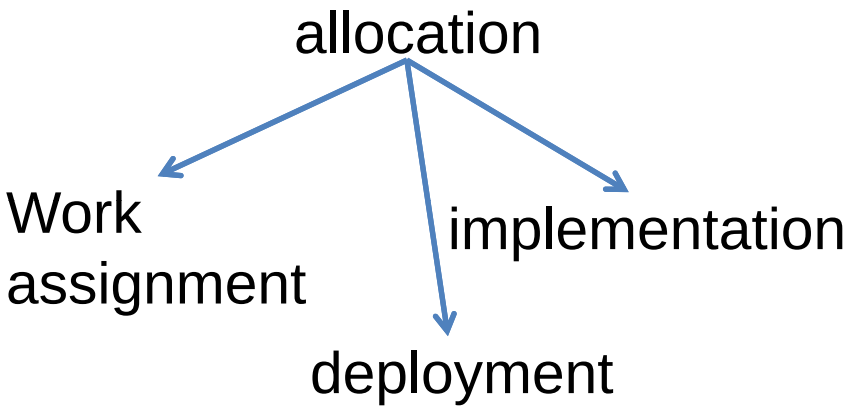
Component and Connector

- Client/server – multiple modules go to a common module for the same action
- Concurrency – logical threads
- Process – actual threads/ processes of the system
- Shared Data – how is data stored and accessed



Allocation structures

- work assignment– module assigned to a team
- deployment – which processor has which threads
- implementation – where in CM are the files for this module



Architecture Styles

Greek Revival

French Colonial

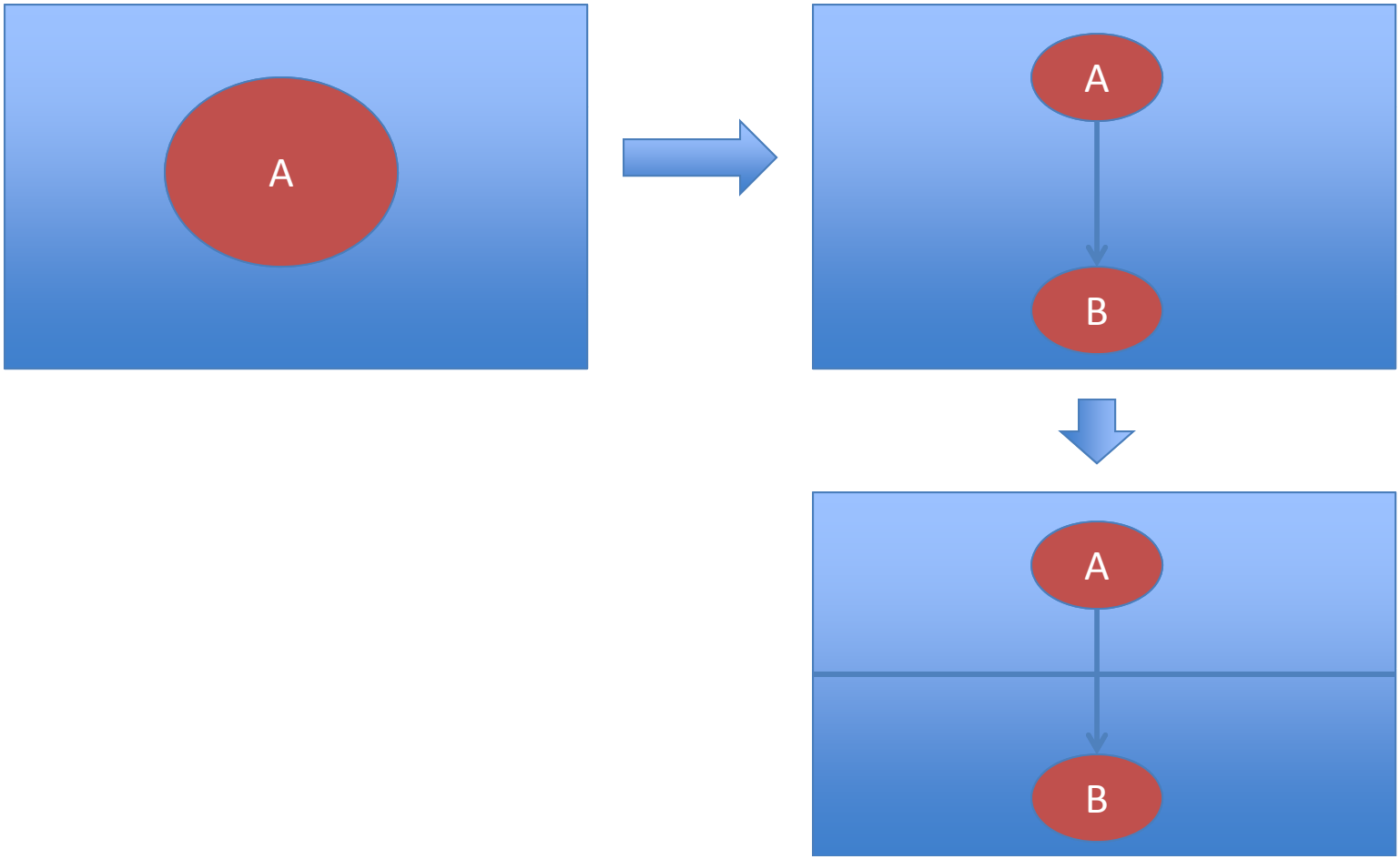
Queen Anne

Architectural styles

- Set of element types
  - Pipes and filters
- A topological layout
  - A pipe connects two filters
- Set of semantic constraints
  - A filter transforms its inputs to create its outputs
- Set of interaction mechanisms
  - The output of a filter is carried in a pipe to another filter

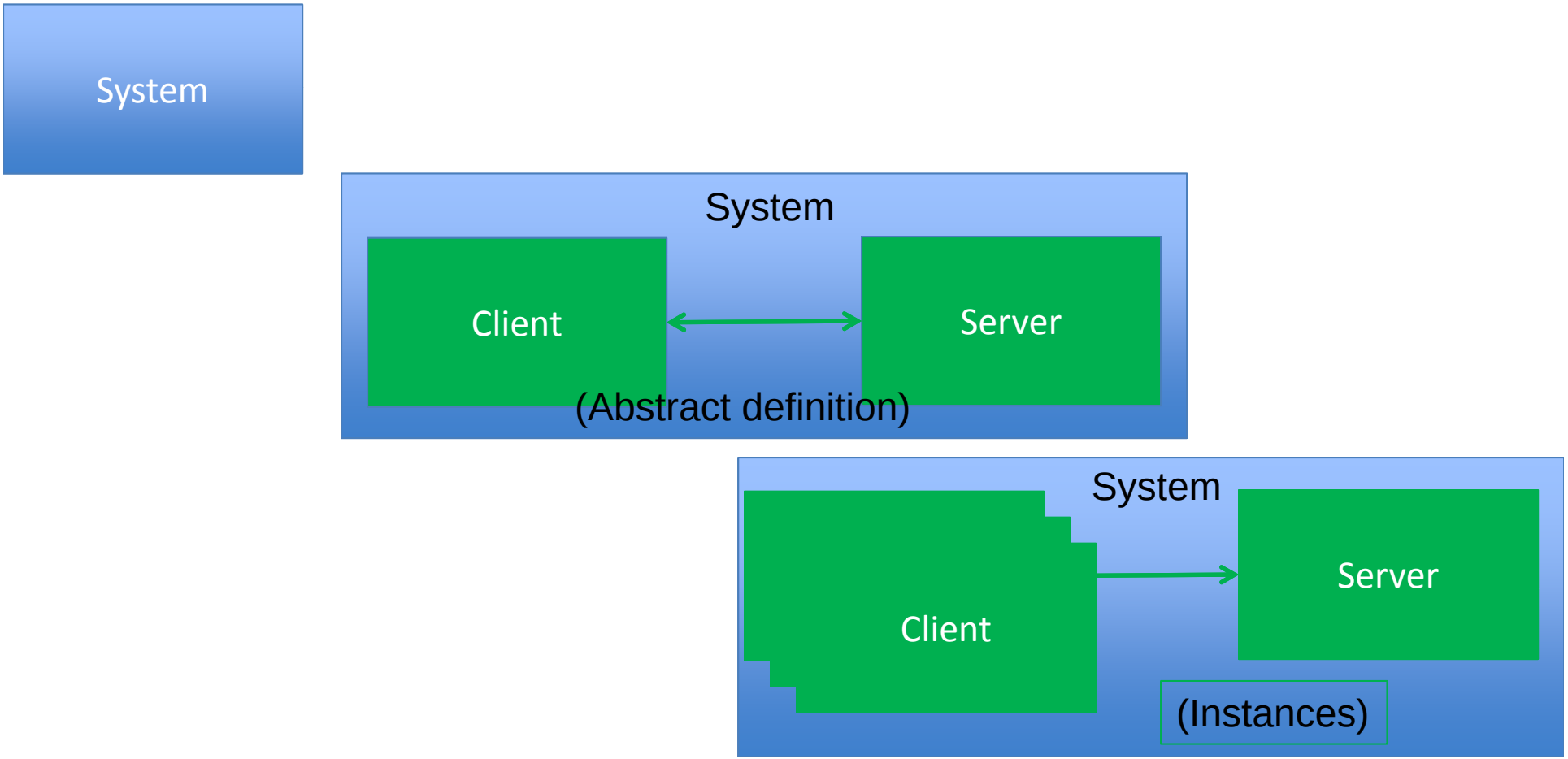
Decomposition

Module A is getting too large to be understandable. Split it into pieces. Separate the context for



Client/server

- Server provides some service that we wish to keep centralized.
- Many clients may all go to the same source for a function

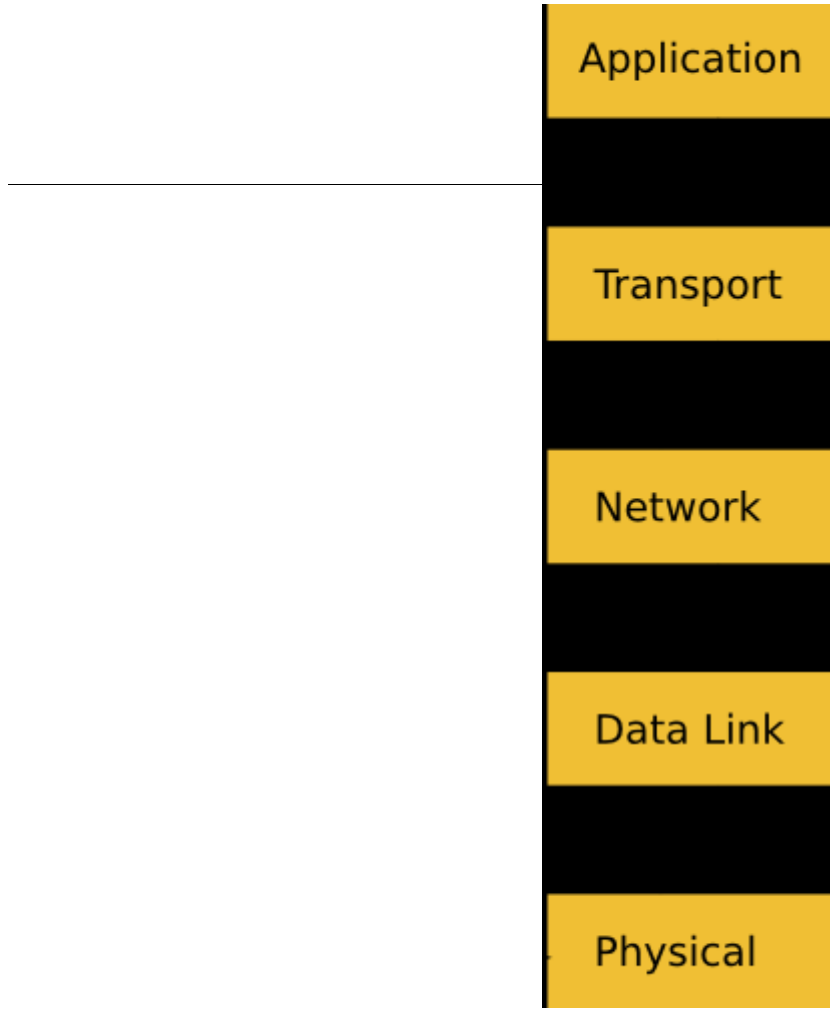


Layered style

- Functionality is divided into buckets
- The buckets are arranged in a hierarchy
- Data and control can flow up and down the hierarchy
- Functionality in a bucket may only invoke functionality in an adjacent bucket

Layered style

- Example: OSI network protocol stack
- each layer provides a specific type of network service. It illustrates why groups of related protocols are frequently called *protocol stacks*



Pipe and filter style

- "The ***Pipes and Filters*** architectural pattern provides a structure for systems that process a stream of data. Each processing step is encapsulated in a filter component. Data [are] passed through pipes between adjacent filters. Recombining filters allows you to build families of related filters." [Buschmann]
- Example: ray tracer

Ray tracing architecture

- Each step has a specific purpose
- Each step can be developed independently
- Each step can be replaced or modified independently (assuming the results are the same type of information)

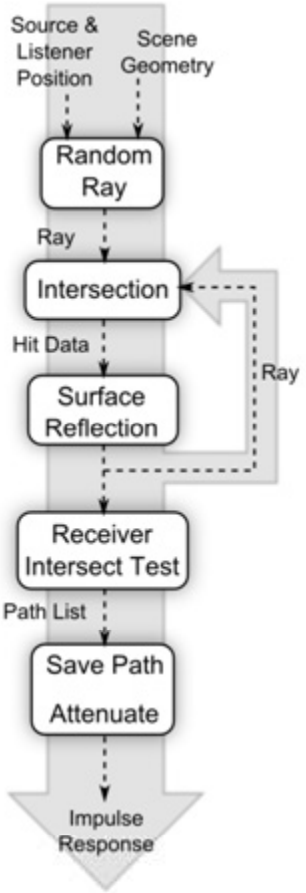


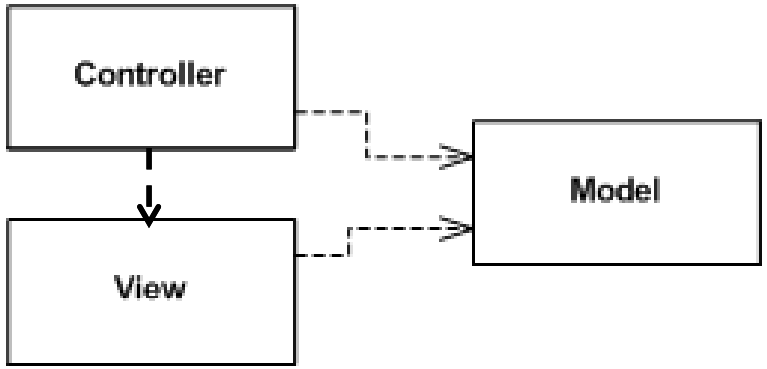
Figure 3: Unified ray engine: Both (left) frustum tracing and (right) ray tracing share similar rendering pipeline.

Similar

- The layered and pipe and filter styles are very similar
- What could be different between them?
- It isn’t geometry obviously

GUI design

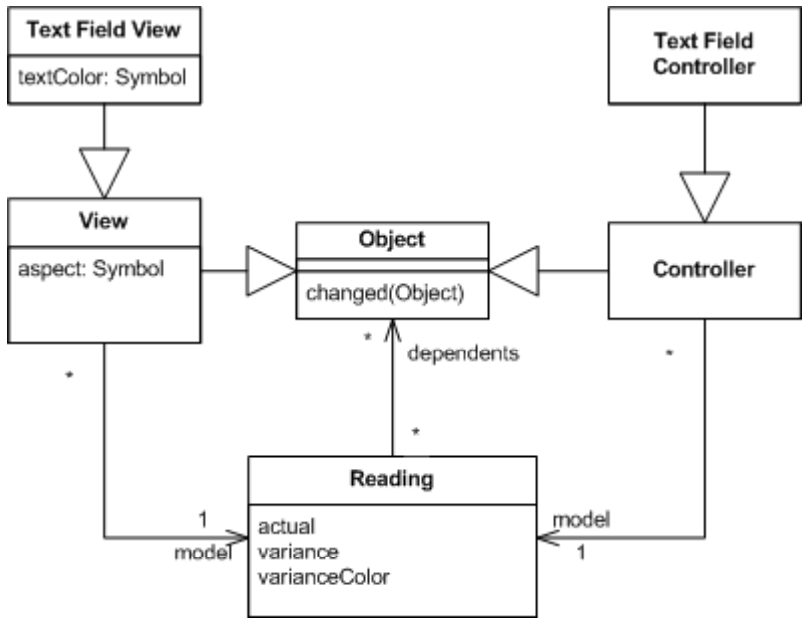
- Separates the data model from the means of viewing it
- Interaction is handled by the controller(s)
- Data is presented in the view(s)
- Multiple views can register with the model. The model does not know how many views are registered.
- There is one or more controllers associated with each view.



http://martinfowler.com/eaaDev/uiArchs.html

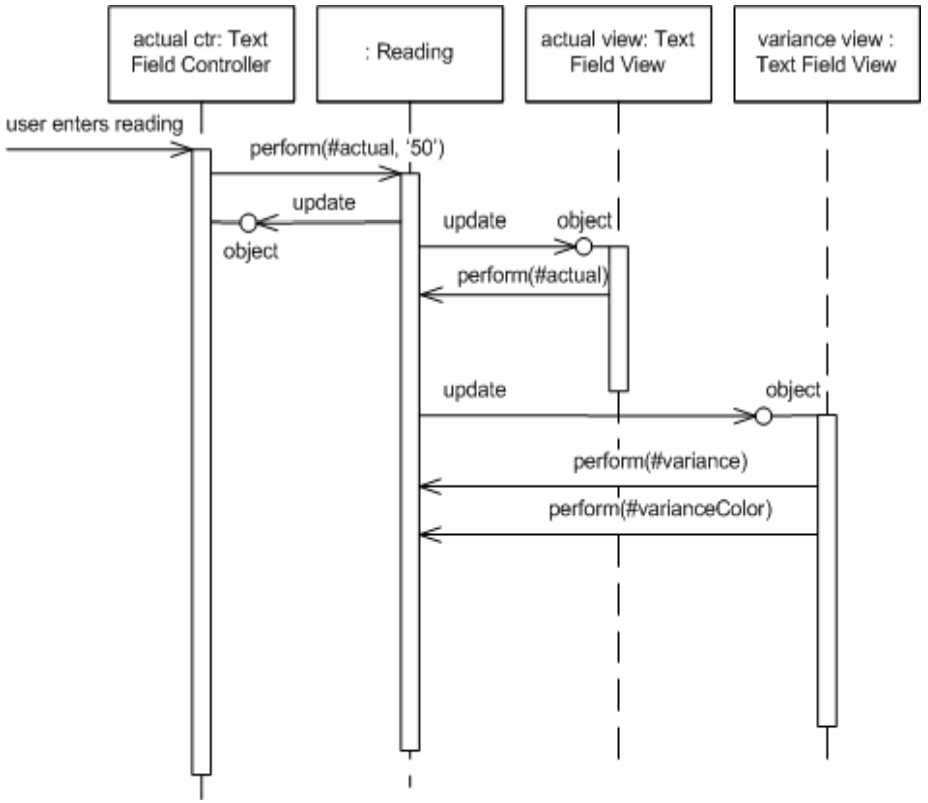
Model-View-Controller

- View and Controller are used to create specialized Views and Controllers which can be polymorphically substituted
- Reading is the model of a reading from some sensor



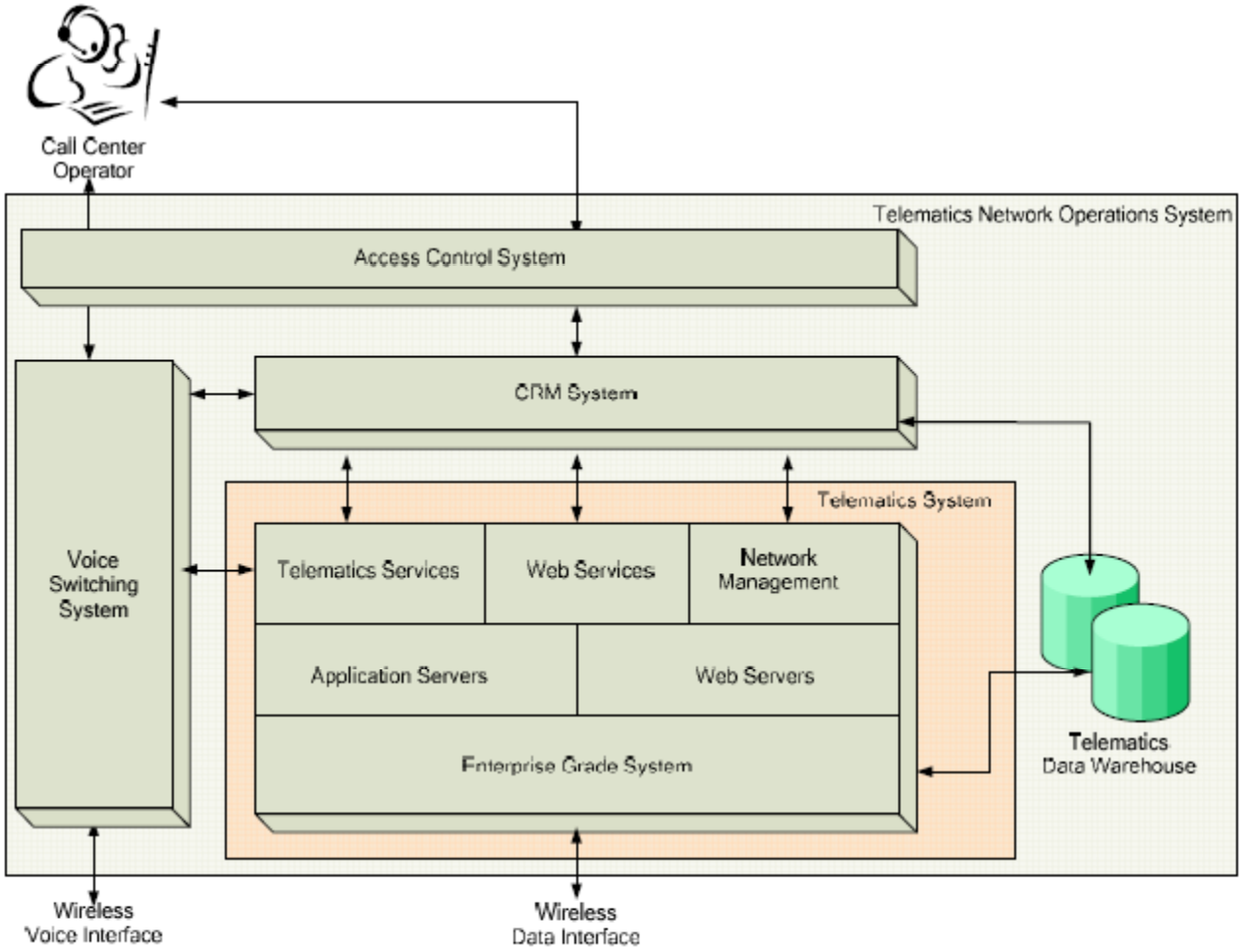
Model-View-Controller

- This shows normal operation
- Update – is the way Reading notifies Views when new data is available
- Perform – the Views are asking the Model to do some standard action



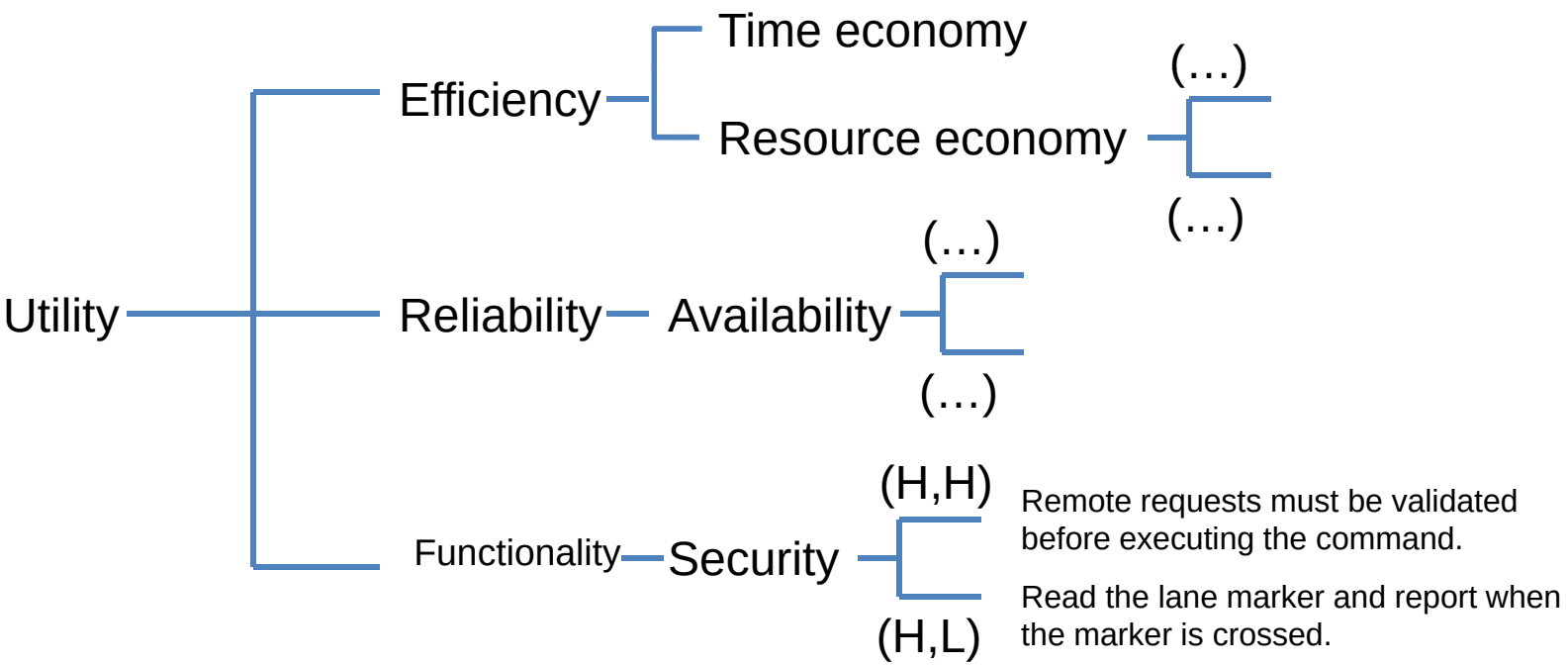
Telematics

- A start
- Many competing architectures



http://www.telematicsinfo.jp/whitepaper/779.pdf

Utility tree



(criticality,risk)

Next steps

- Each team member takes on the identity of one of the stakeholders for the part of the product you are designing(auto owner, auto designer, auto executive, telematics architect). Each of you creates a list of quality attributes and scenarios in priority order based on your role. Draw the utility tree complete with criticality and risk rating.
- Consolidate the lists within your group. Submit by 11:59pm Monday Jan 28th.
- Read about the QAW using at least the first of the two tech reports
- Begin reading: https://wiki.sei.cmu.edu/aadl/images/7/73/AADLV2Overview-AADLUserDay-Feb_2010.pdf