Requirements Wrap Up

Feature list

feature list: a list of user visible aspects of your product that reflect value-add

Which of these aren't features/well defined features?

- Rich text/music editor
- Global search
- Ability to create/update/delete accounts
- Beta release
- Chat facility
- Performance improvements

Pulling it all together

How much is enough?

You have to find a balance

- comprehensible vs. detailed correctness
- graphics vs. explicit wording and tables
- short and timely vs. complete and late

Your balance may differ with each customer depending on your relationship and flexibility

Words of Wisdom 5

After you create a specification, go over it to:

- Eliminate all requirements not absolutely necessary
- Simplify those that are more complicated than necessary
- Substitute cheaper options when available
- Move non essentials to future releases

Words of Wisdom 5’

Agile Principle – Simplicity is Essential

Software Architecture

"Good software architecture makes the rest of the project easy."

McConnell, Survival Guide

Outline

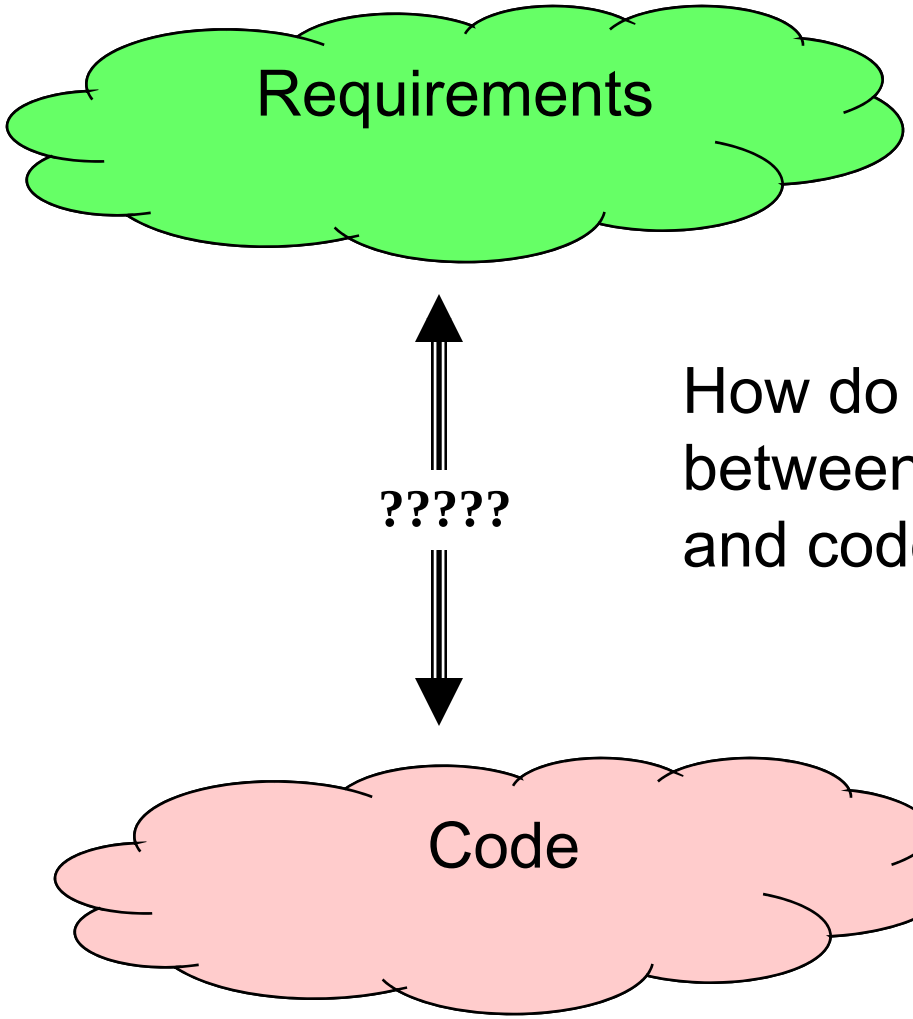
- Part 1 – Basics of software architecture
- Part 2 – Common architecture models
- Part 3 – Real examples

Readings

- *The architecture of Web applications*, Grady Booch
- *Software Project Survival Guide*, McConnell, p144-150
- [optional] *Software Architecture*, Garlan, http://www.cs.cmu.edu/afs/cs/project/able/ftp/encycSE2001/encyclopedia-dist.pdf

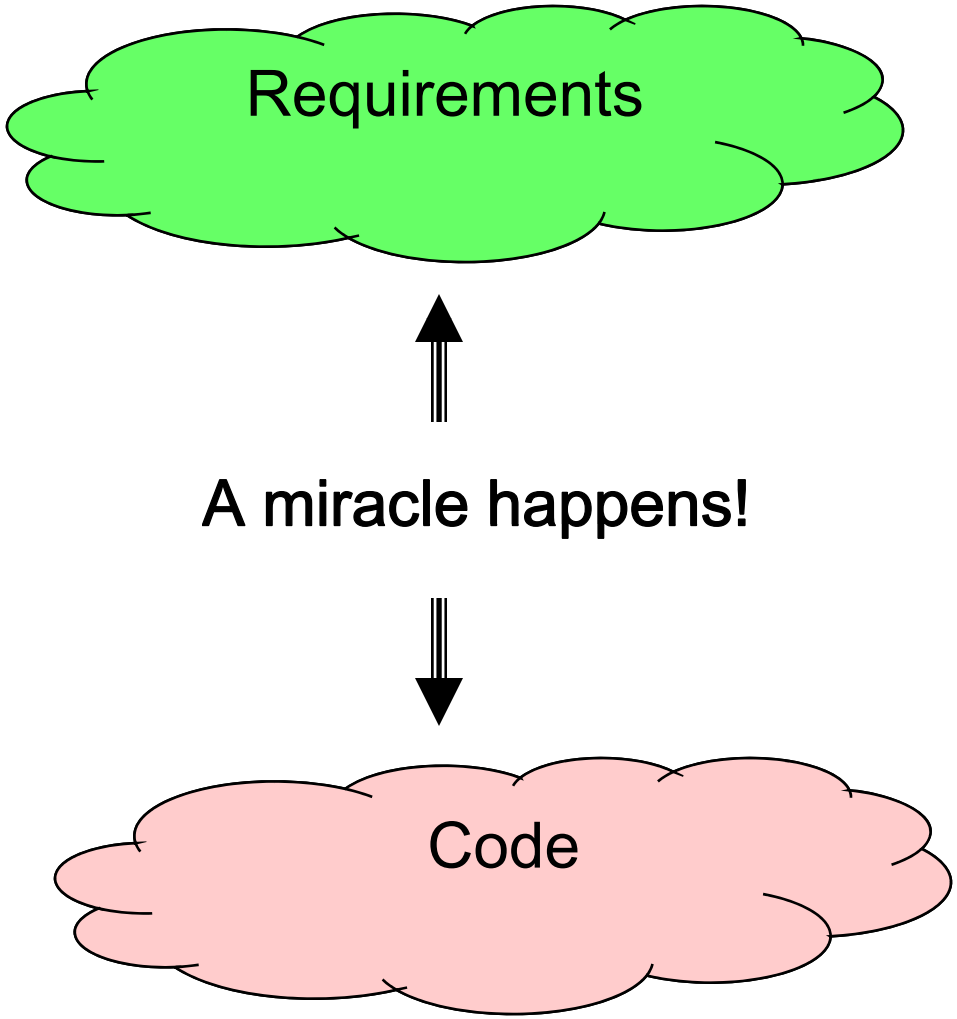
Summary 3 is due today!

The basic problem

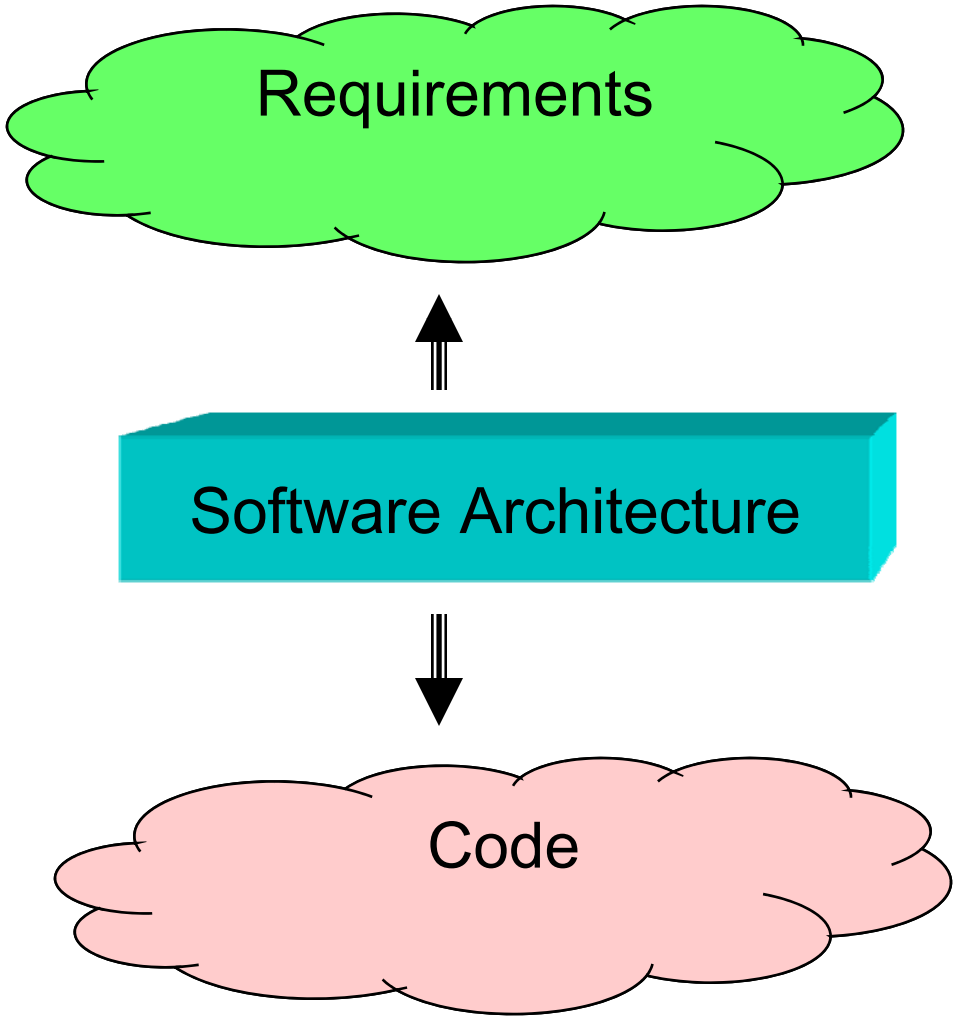


How do you bridge the gap between requirements and code?

One answer



A better answer



Provide a high level framework to build and evolve the system

What exactly is an architecture?

At the core of most all definitions is the notion that the **architecture** of a system describes its gross structure --

1. Main components and their behaviors
2. Connections between the components

-- using one or more ***views***

An architecture bridges the gap between requirements and code

An architecture helps with …

- System understanding - describes the interactions between modules
- Reuse – by defining the high level components, we can see if there is opportunity for reuse
- Construction – breaks development down into work items; provides a path from req to code
- Evolution - see reuse
- Management – helps understand the work to do, and the work done; track progress
- Communication – gives you a vocabulary; pictures say 1000 words

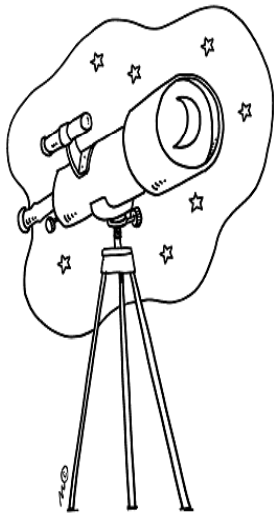
An architecture helps with …

- System understanding — level of abstraction, vocabulary, constraints
- Reuse — of components, identifies commonalities
- Construction — partial blueprint
- Evolution — organization for change, separates “what” from “how”
- Management — good milestone
- Communication — between stakeholders

Back to “views”

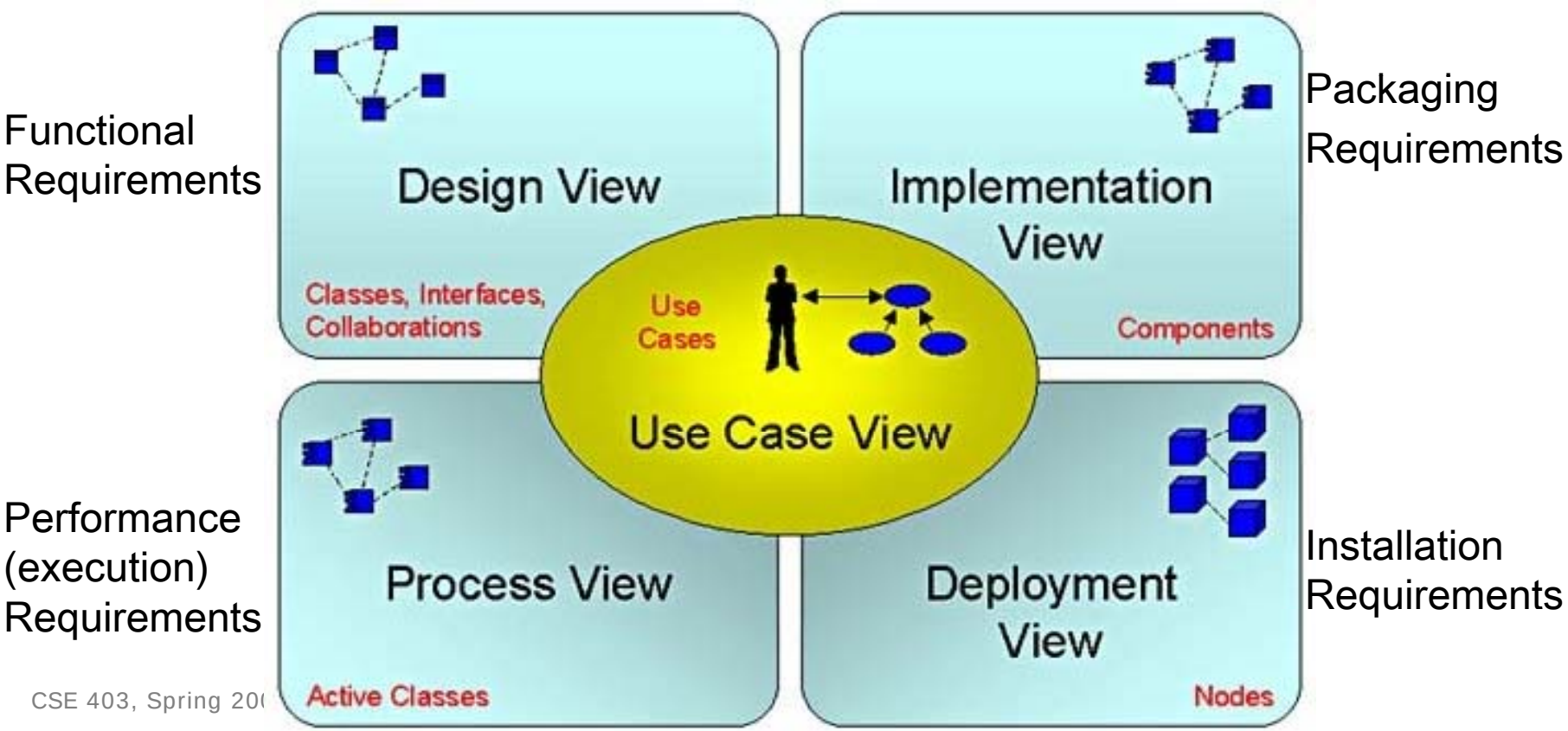
A **view** illuminates a set of top-level design decisions

- how the system is composed of interacting parts
- where are the main pathways of interaction
- key properties of the parts
- information to allow high-level analysis and appraisal



Importance of views

Multiple views are needed to understand the different dimensions of systems



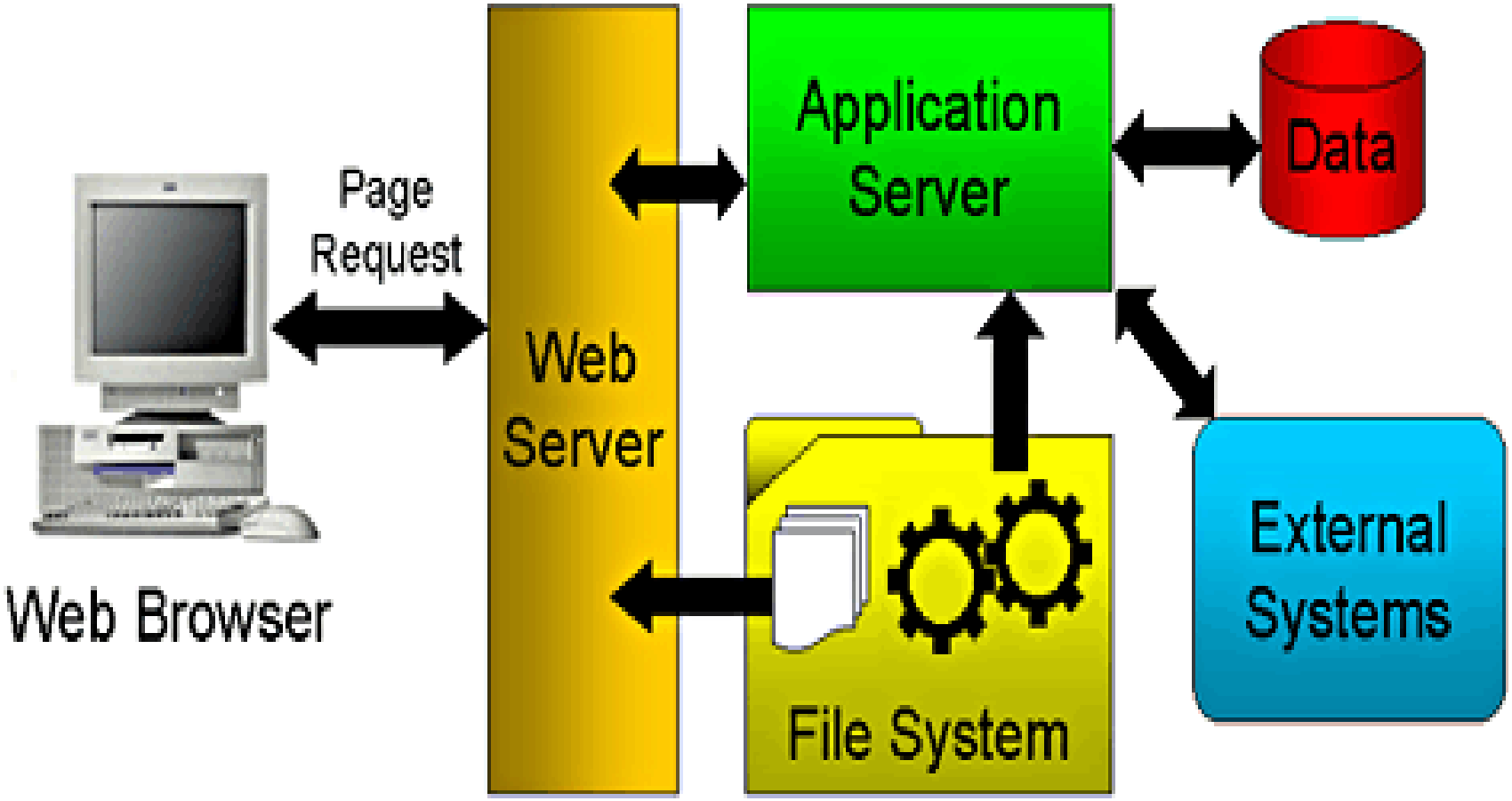
Booch

Let's make this more concrete

We'll start with diagrams of some familiar architectures

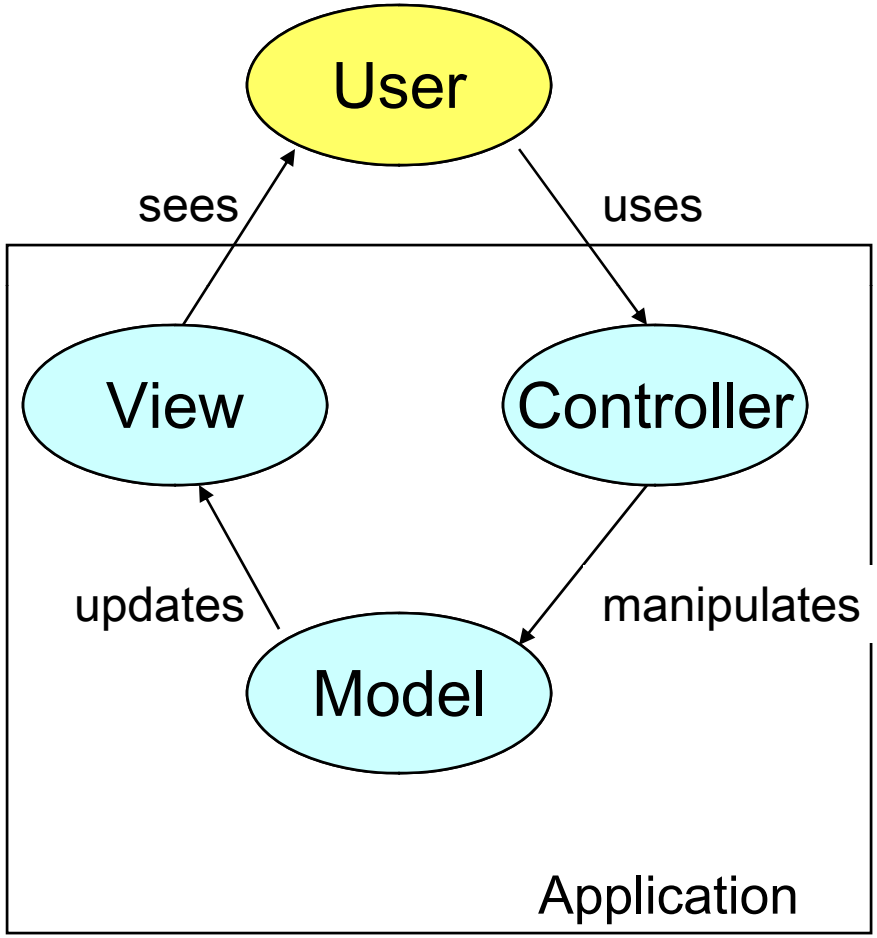
And then look at several views of the Cray RS system architecture

Web application (client-server)



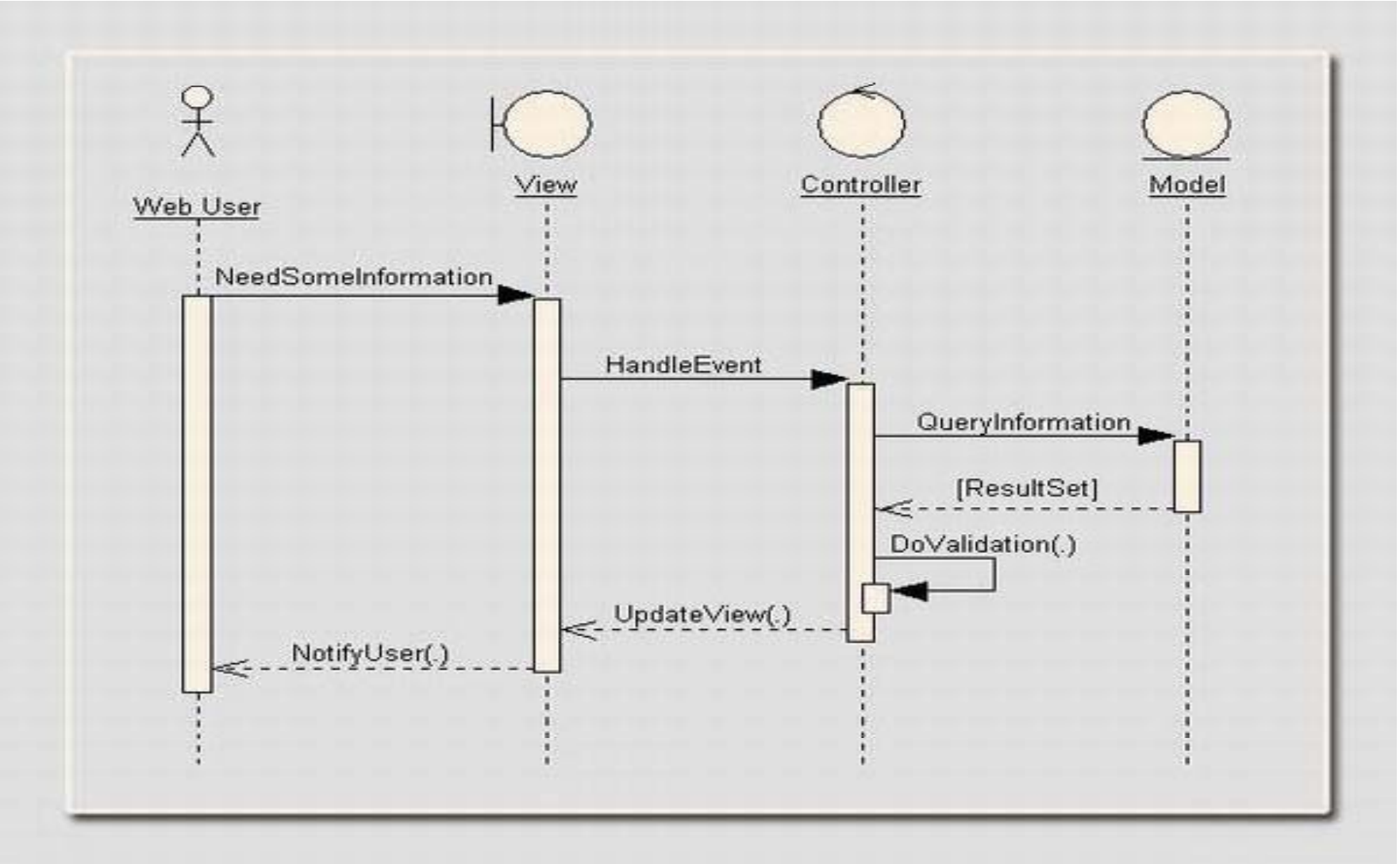
Booch

Model View Controller

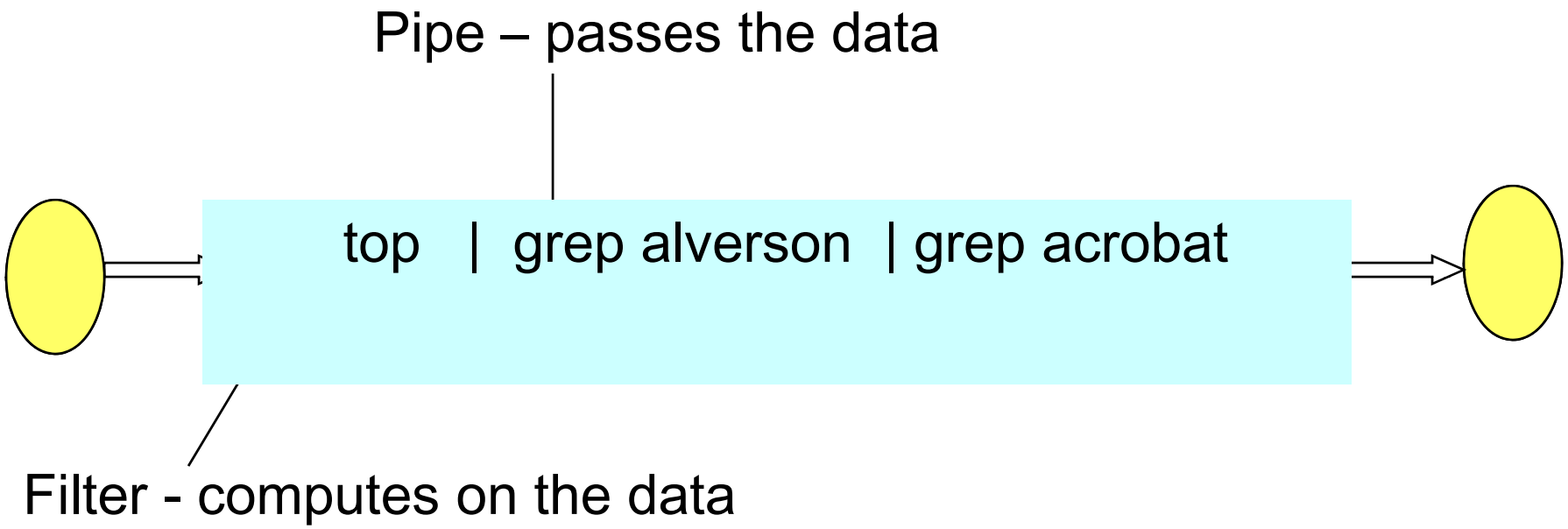


Separates the application object (model) from the way it is represented to the user (view) from the way in which the user controls it (controller).

MVP – Sequence diagram (UML)

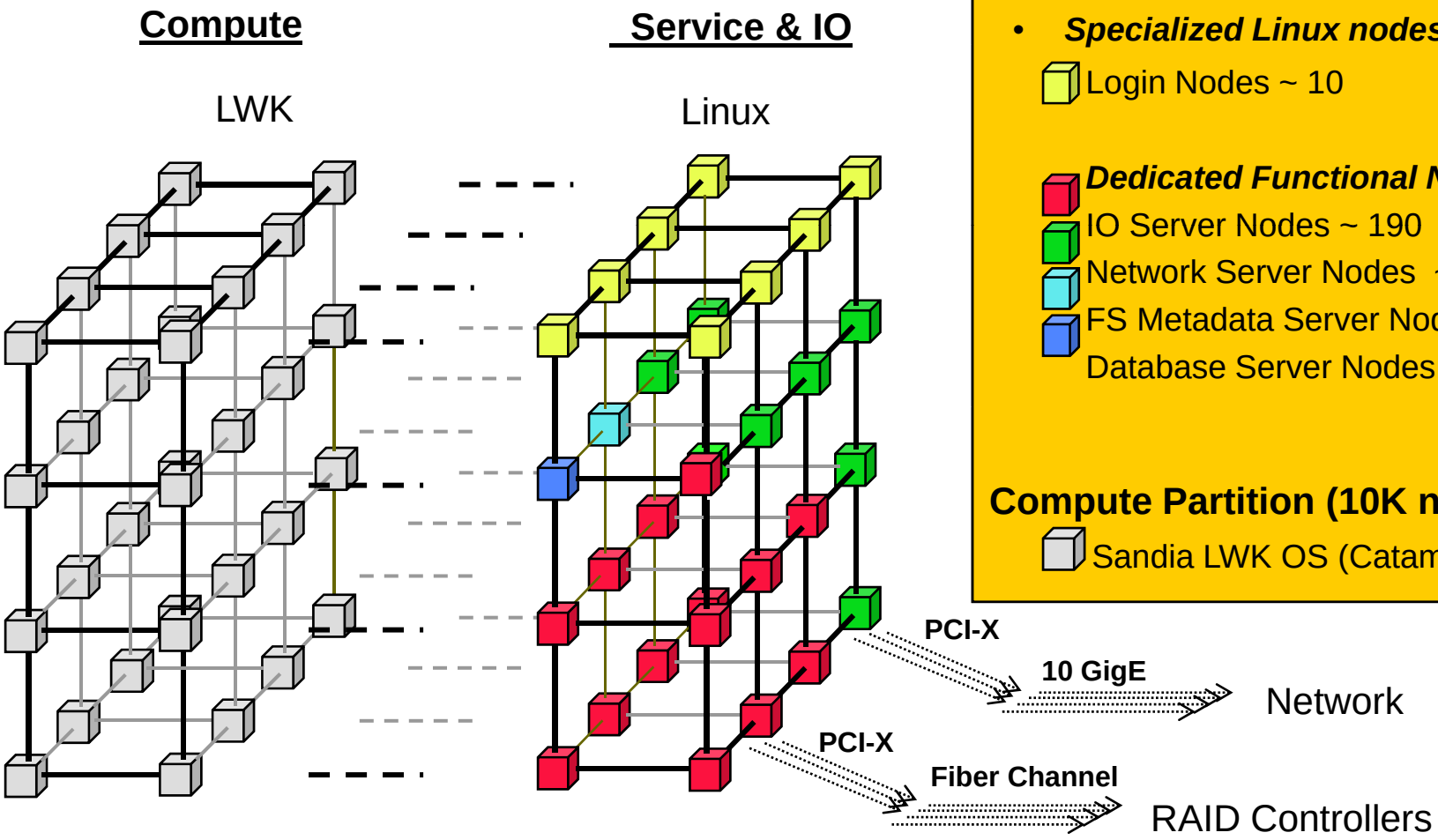


Pipe and filter



Each stage of the pipeline acts independently of the others. Can you think of a system based on this architecture?

RS Software Architecture



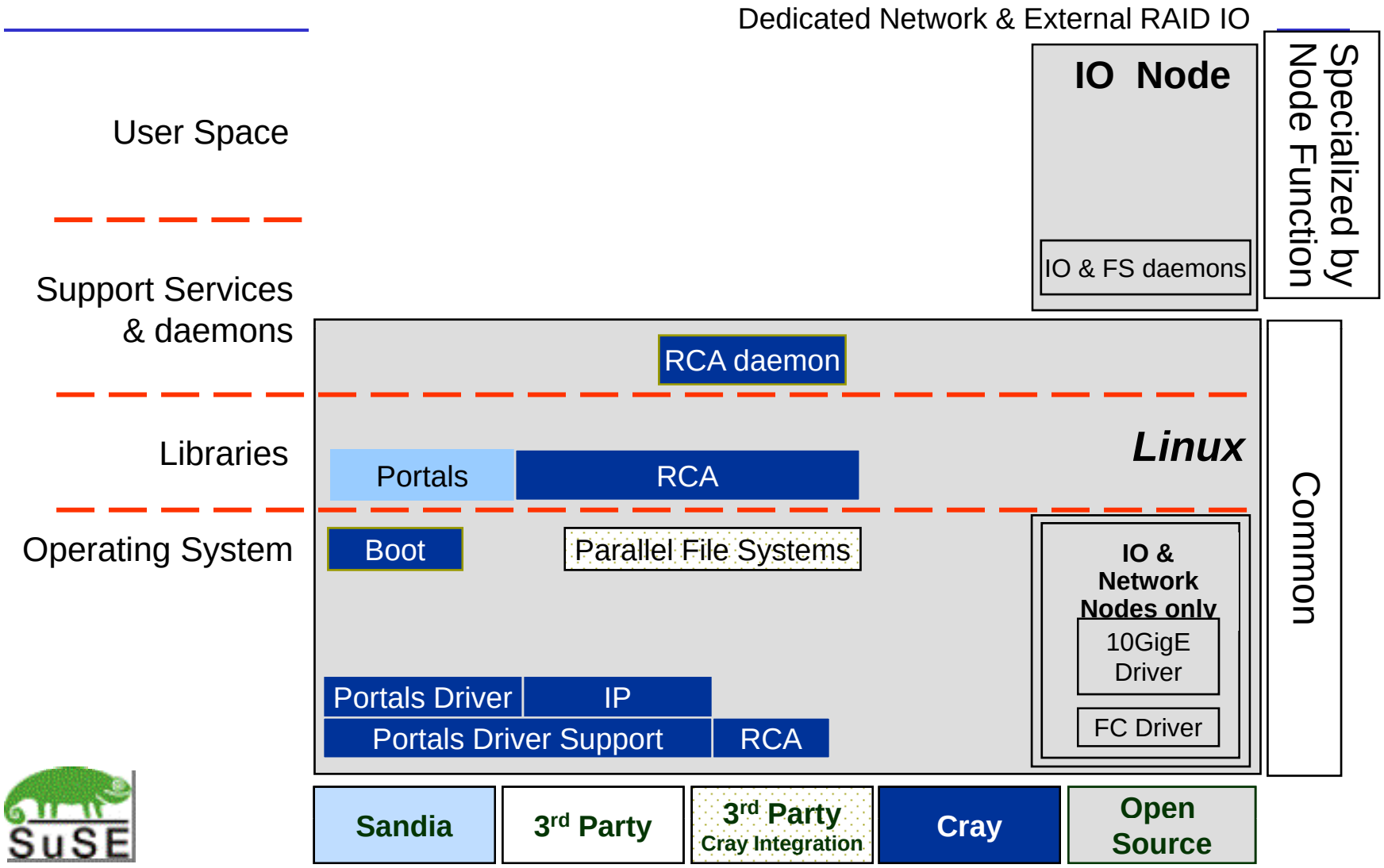
Service Partition (256 nodes)

- Linux OS
- ***Specialized Linux nodes***
  - Login Nodes ~ 10
  - ***Dedicated Functional Nodes***
  - IO Server Nodes ~ 190
  - Network Server Nodes ~ 50
  - FS Metadata Server Nodes ~ 4
  - Database Server Nodes ~ 2

Compute Partition (10K nodes)

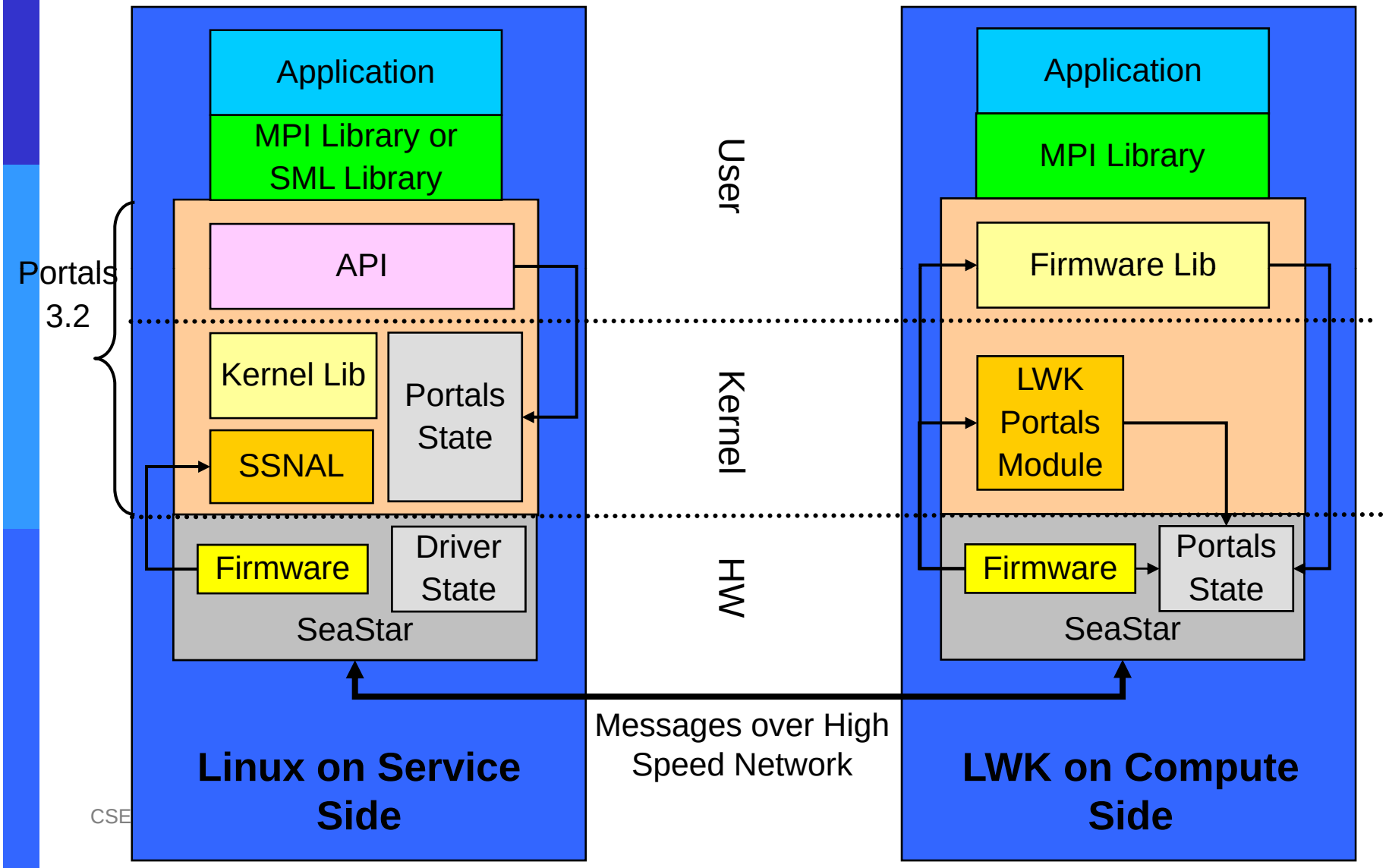
- Sandia LWK OS (Catamount)

RS: Service Nodes

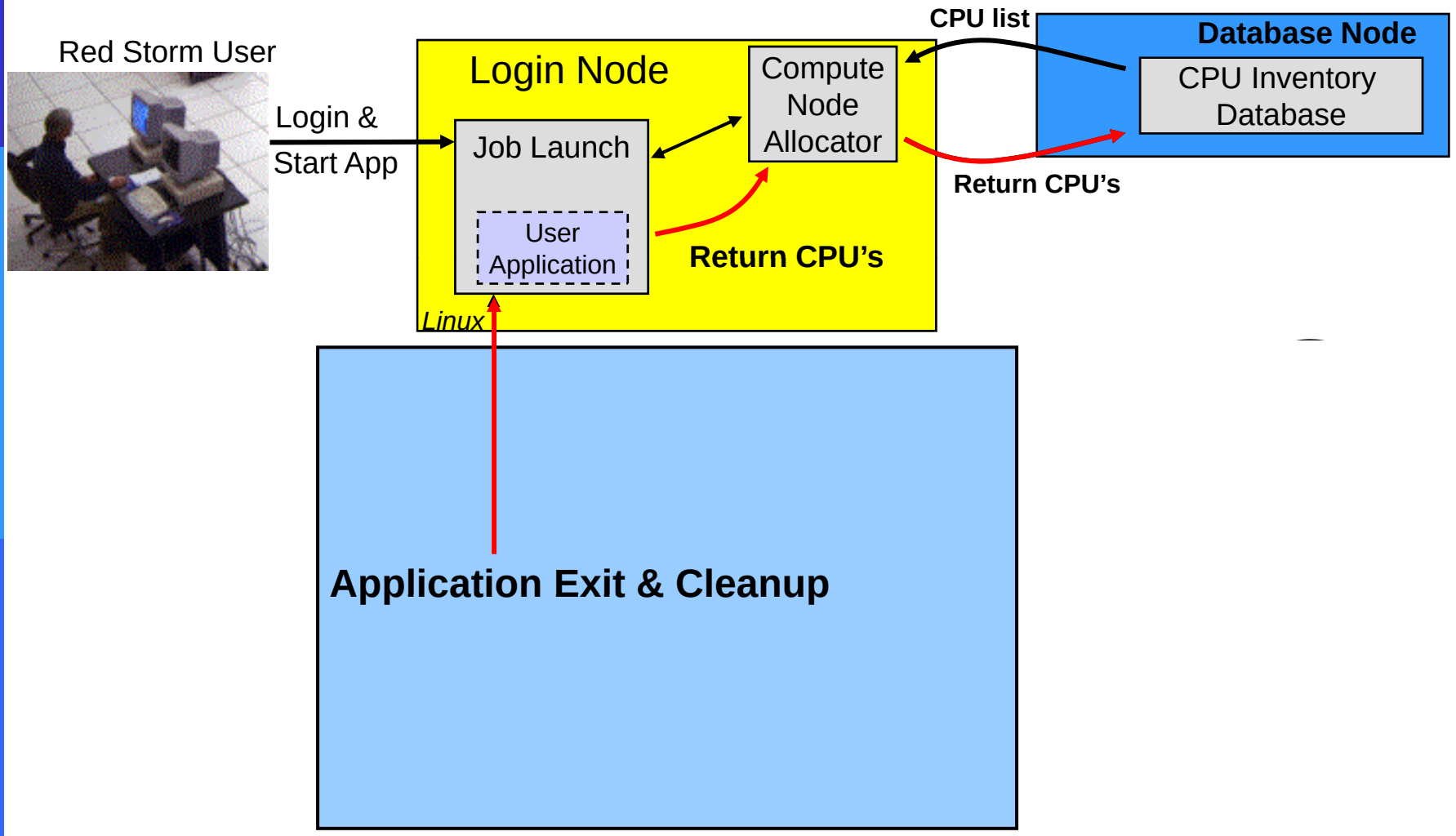


Runs on all Service and IO Nodes

RS: Message Passing

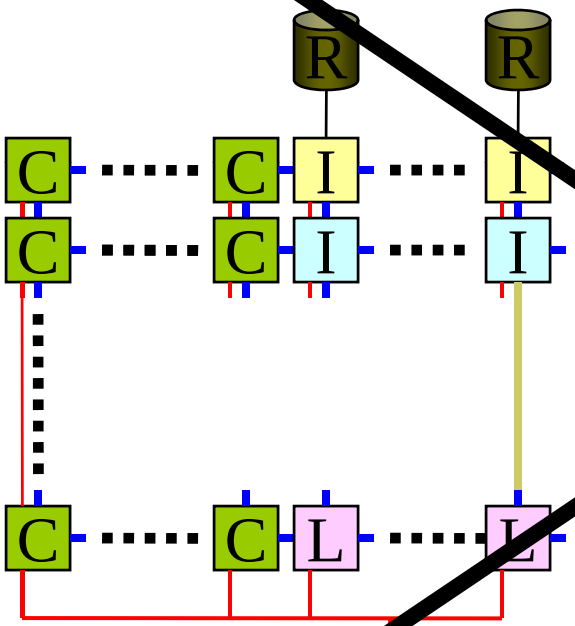


RS: Job Launch



RS: System boot

Requirement 5.11 The full system must be able to be booted in less than 15 minutes after a clean shutdown.



Service Nodes

1. RAS broadcasts kernel to Service nodes
2. Service nodes verify Unix image is valid
   If not, first loads image from RAS
   and others, load from peer
3. Service nodes are up and running

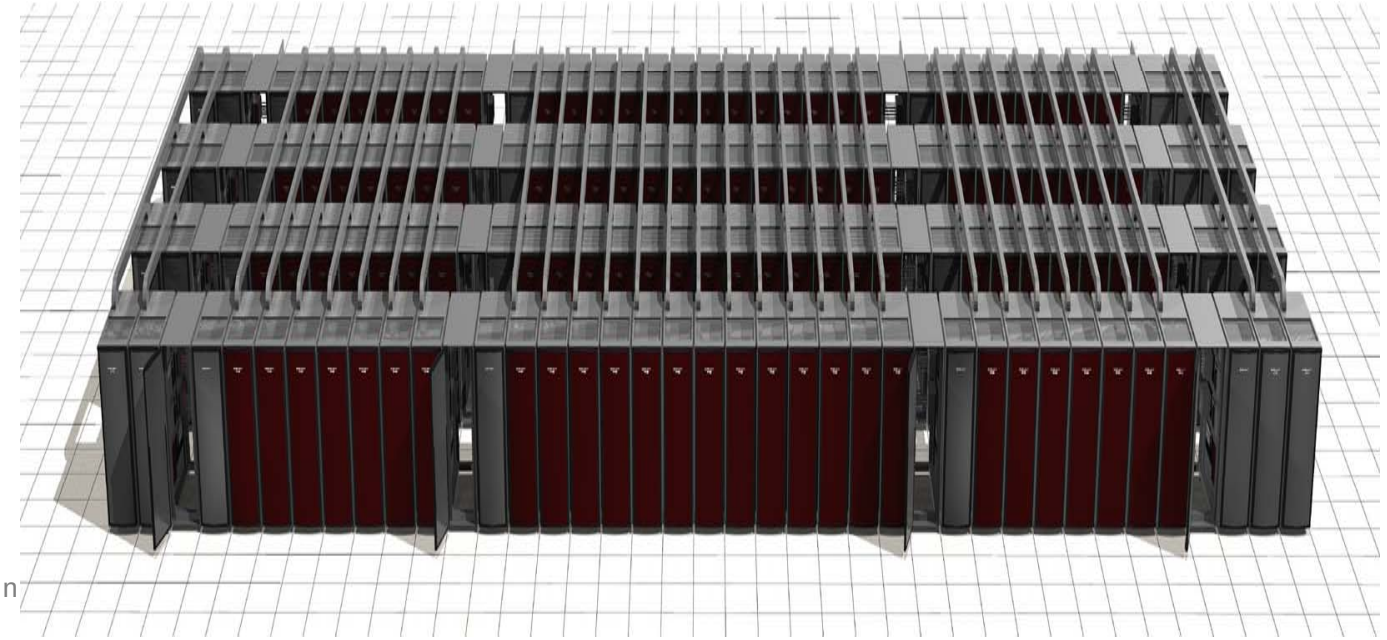
Compute Nodes

1. RAS broadcasts kernel to cage controllers
2. Cage controllers broadcast to other nodes
3. Compute nodes are up and running

Let's reflect …

- Did you find the multiple views of the RS system helpful in understanding its operation?

- Would one view have sufficed?



Let's discuss…

- What does it mean for an architecture to have conceptual integrity?

- What makes a software architecture different from software design? Is there a difference?

Architecture – spans the system

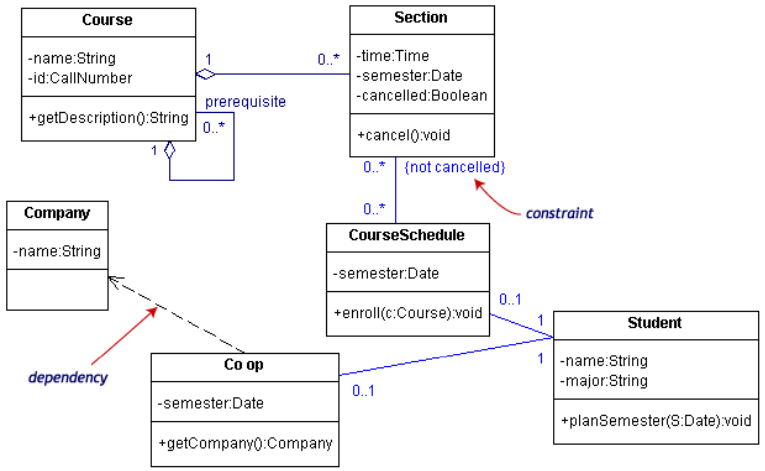
Design – spans a component of the system

… shades of gray …

Another question …

- Could you start coding from any of these diagrams? If you had the Model component and your partner had the View, could you both go off and develop, returning in a month to integrate?

- What information would a design view need to allow you to get started, somewhat independently?

Coming soon to theatres near you

- UML
  - Unified Modeling Language
    or Union of all Modeling Languages ☺
- Language to describe the architecture/design of a system
- We'll look at a subset:
  - Class diagrams
  - Sequence diagrams