DEPENDABLE
COMPUTING

The Problem-Derived World-Situated Machine Model

M. Anthony Aiello
Dependable Computing
9 June 2015

DEPENDABLE COMPUTING

Systems of Systems
The Challenge of Reuse

- Reuse is a key feature of systems of systems
  - Systems — with prior missions — are reused to complete a new mission

DEPENDABLE COMPUTING

Systems of Systems
The Challenge of Reuse

- Reuse is a key feature of systems of systems
  - Systems — with prior missions — are reused to complete a new mission
- Reuse brings significant challenges

DEPENDABLE COMPUTING

Systems of Systems
The Challenge of Reuse

- Reuse is a key feature of systems of systems
  - Systems — with prior missions — are reused to complete a new mission
- Reuse brings significant challenges

DEPENDABLE COMPUTING

Systems of Systems
The Challenge of Reuse

- Reuse is a key feature of systems of systems
  - Systems — with prior missions — are reused to complete a new mission
- Reuse brings significant challenges

Arianne V

Therac 25

DEPENDABLE COMPUTING

Systems of Systems
The Challenge of Reuse

- Reuse is a key feature of systems of systems
  - Systems — with prior missions — are reused to complete a new mission
- Reuse brings significant challenges
  - Syntax and semantics of the *interfaces* must be completely defined

DEPENDABLE COMPUTING

Systems of Systems
The Challenge of Reuse

- Reuse is a key feature of systems of systems
  - Systems — with prior missions — are reused to complete a new mission
- Reuse brings significant challenges
  - Syntax and semantics of the *interfaces* must be completely defined
  - Novel use may bring a system outside of the *context* for which it was designed & certified

DEPENDABLE COMPUTING

Systems of Systems
The Challenge of Reuse

- Reuse is a key feature of systems of systems
  - Systems — with prior missions — are reused to complete a new mission
- Reuse brings significant challenges
  - Syntax and semantics of the *interfaces* must be completely defined
  - Novel use may bring a system outside of the *context* for which it was designed & certified
- Existing approaches struggle with both of these challenges

DEPENDABLE COMPUTING

Systems of Systems
The Challenge of Reuse

- Reuse is a key feature of systems of systems
  - Systems — with prior missions — are reused to complete a new mission
- Reuse brings significant challenges
  - Syntax and semantics of the *interfaces* must be completely defined
  - Novel use may bring a system outside of the *context* for which it was designed & certified

We introduce a solution based on
Problem Frames, Real-World Types, and Argumentation

DEPENDABLE COMPUTING

The Problem-Derived World-Situated Machine Model

- A *reference model* for safety- and security-critical systems engineering
  - from the problem to be solved
  - into the world the solution will transform
- *What* must the system accomplish?
- *Where* must the solution take place?

DEPENDABLE COMPUTING

The Problem-Derived World-Situated Machine Model

- A *reference model* for safety- and security-critical systems engineering
  - from the problem to be solved
  - into the world the solution will transform
- *What* must the system accomplish?
- *Where* must the solution take place?

DEPENDABLE COMPUTING

The Problem-Derived World-Situated Machine Model

- A *reference model* for safety- and security-critical systems engineering
  - from the problem to be solved
  - into the world the solution will transform

- *What* must the system accomplish? — Interface

- *Where* must the solution take place? — Context

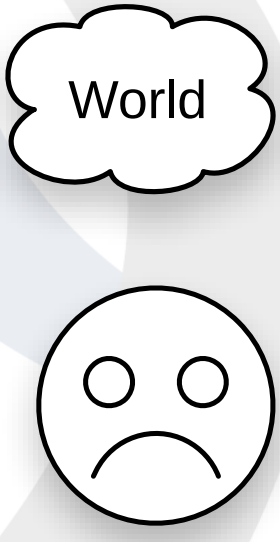
DEPENDABLE COMPUTING

Why Do We Build Systems?

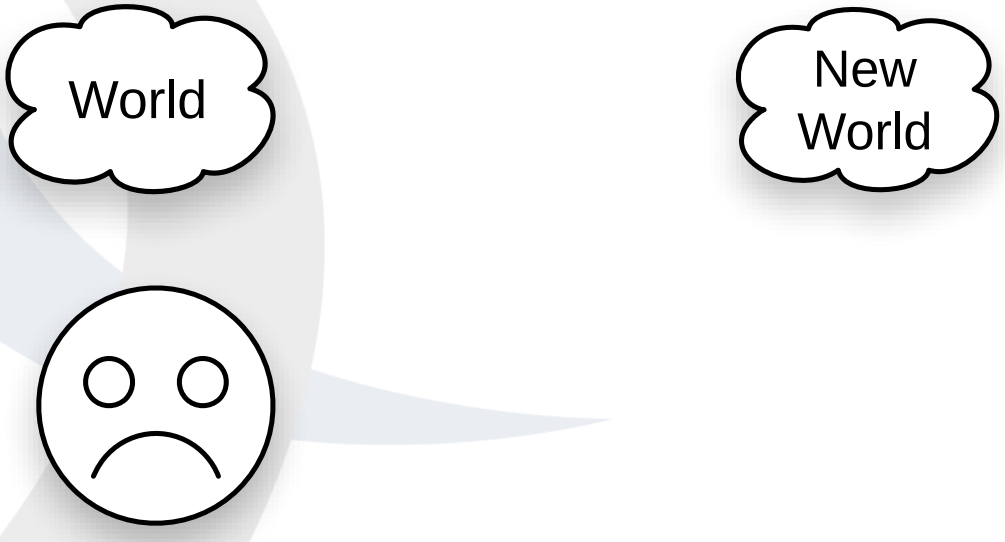
Why Do We Build Systems?

DEPENDABLE COMPUTING

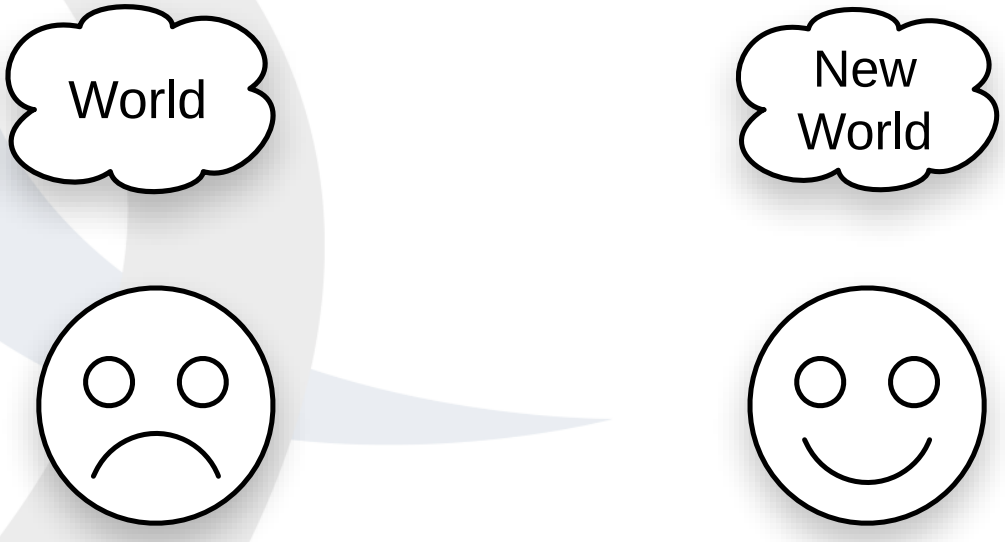
Why Do We Build Systems?



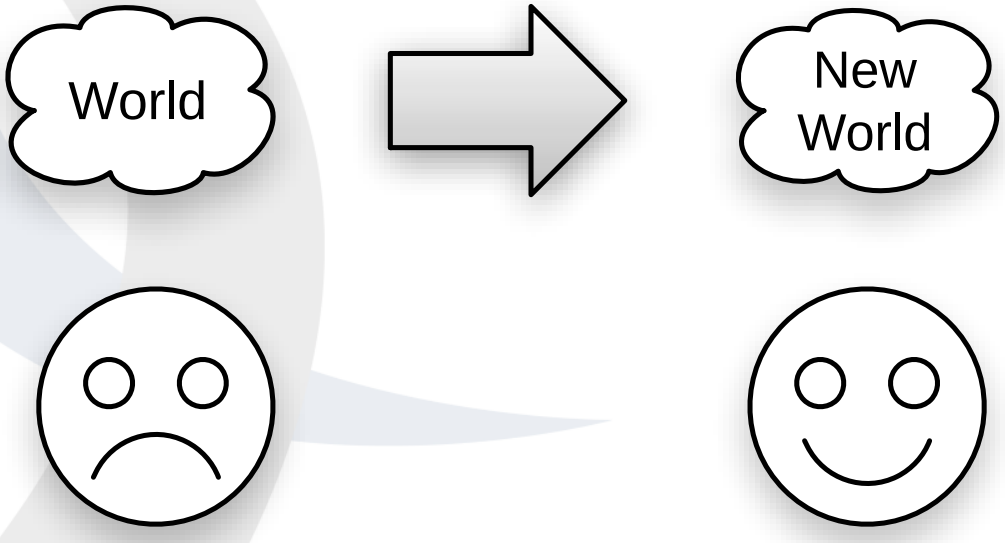
Why Do We Build Systems?



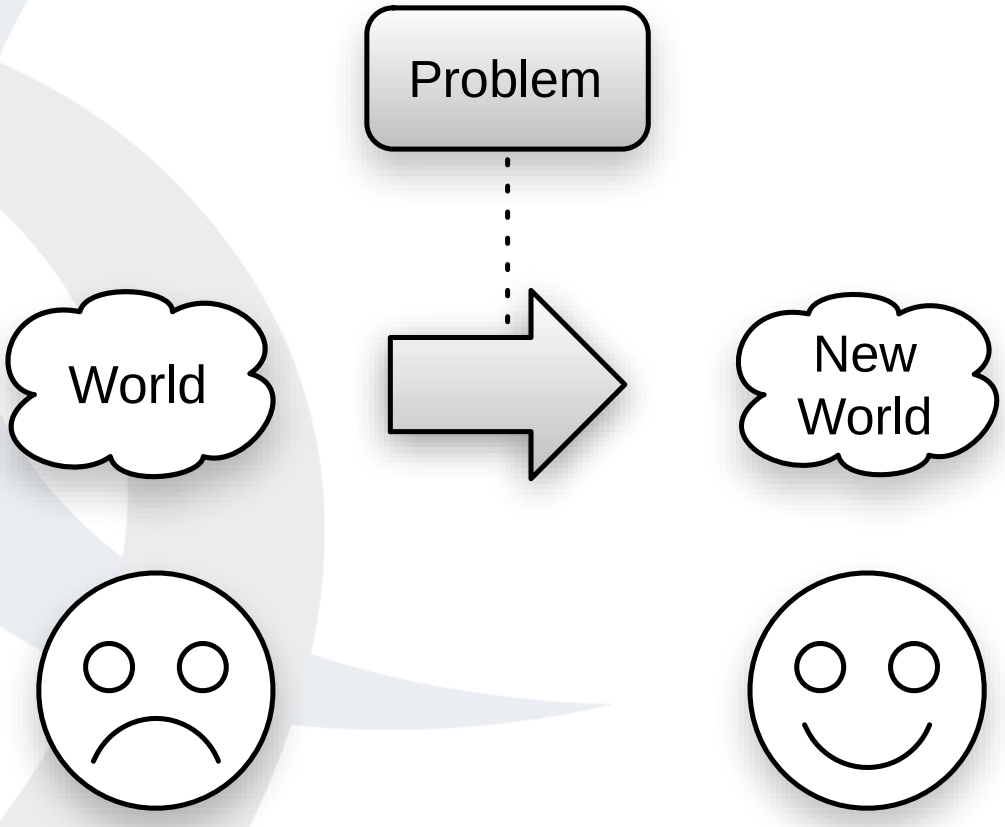
Why Do We Build Systems?



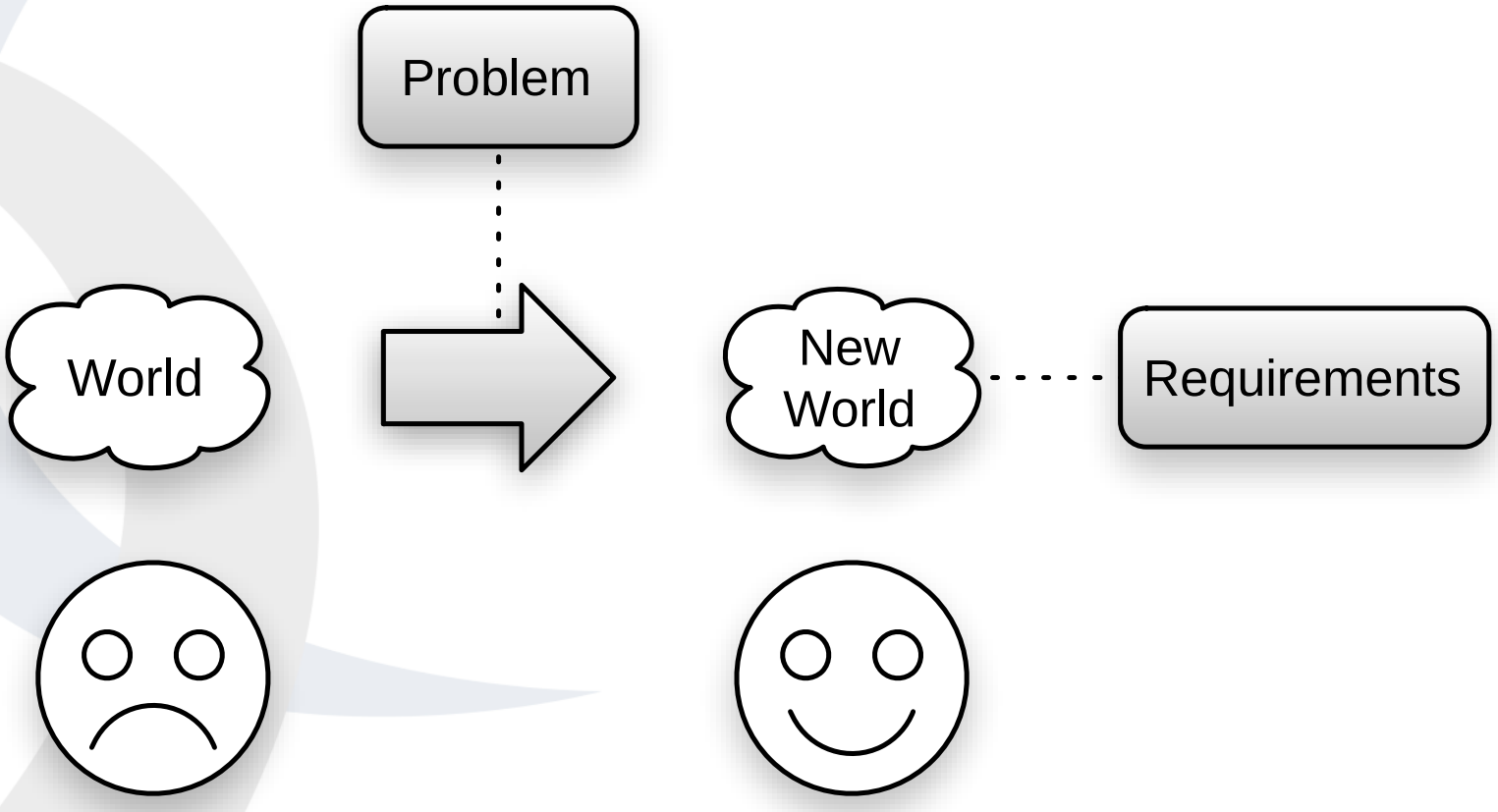
Why Do We Build Systems?



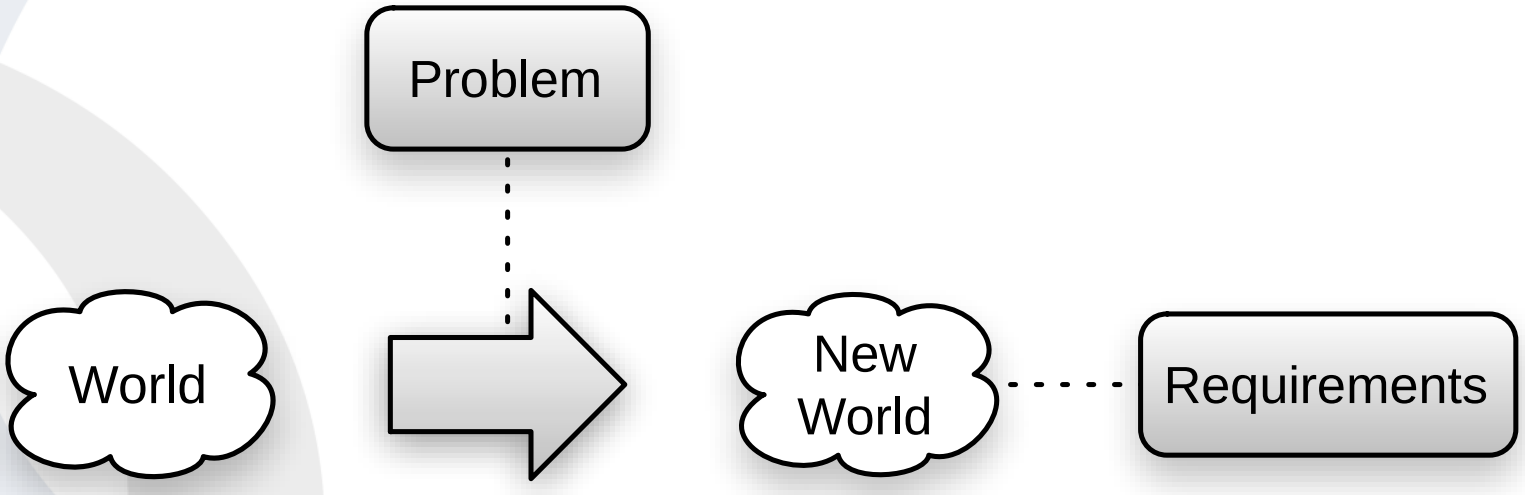
Why Do We Build Systems?



Why Do We Build Systems?



Why Do We Build Systems?



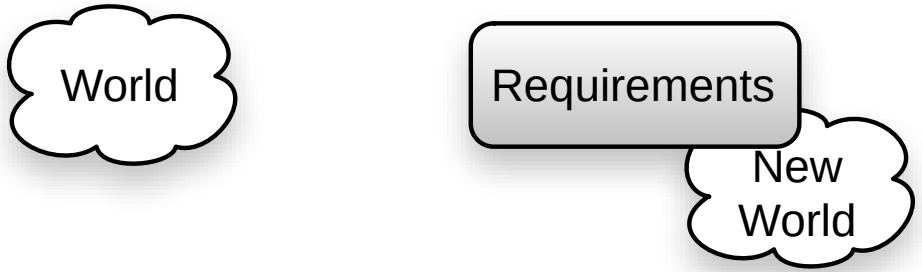
We build systems to solve a problem, thereby changing the world.

DEPENDABLE COMPUTING

How Do We Solve the Problem?

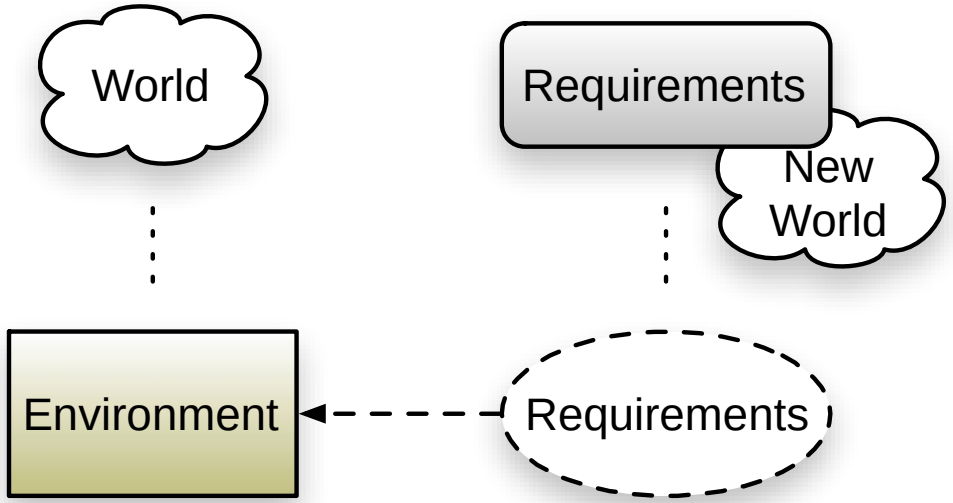
DEPENDABLE COMPUTING

How Do We Solve the Problem?

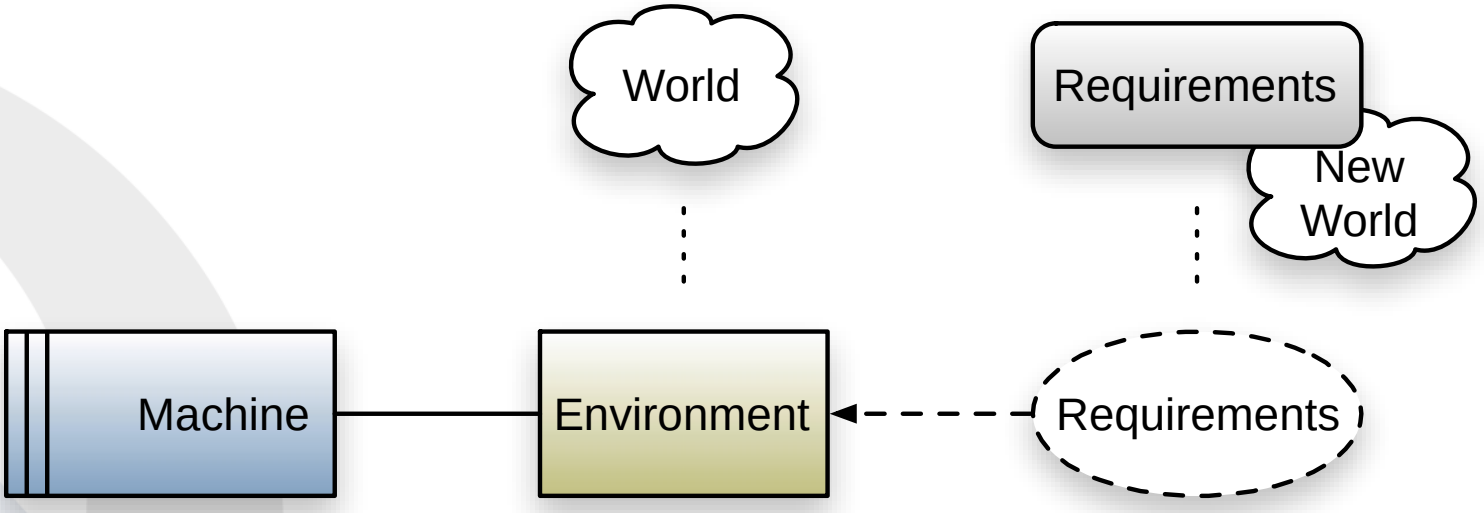


DEPENDABLE COMPUTING

How Do We Solve the Problem?

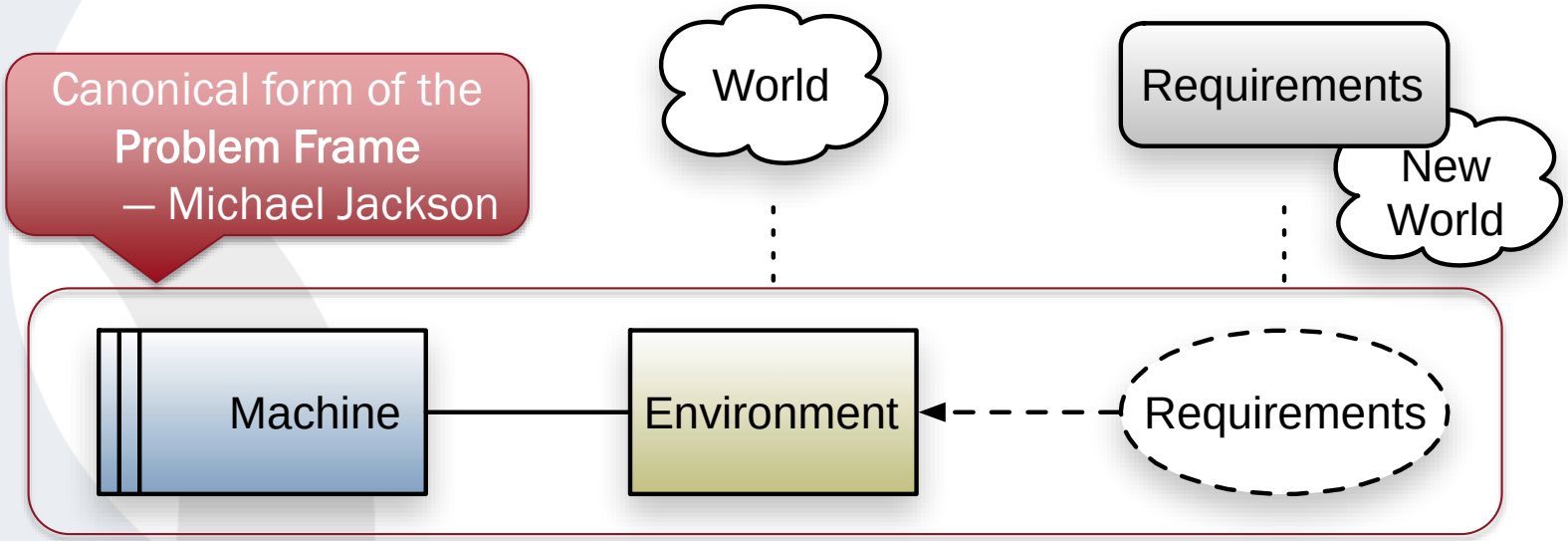


How Do We Solve the Problem?



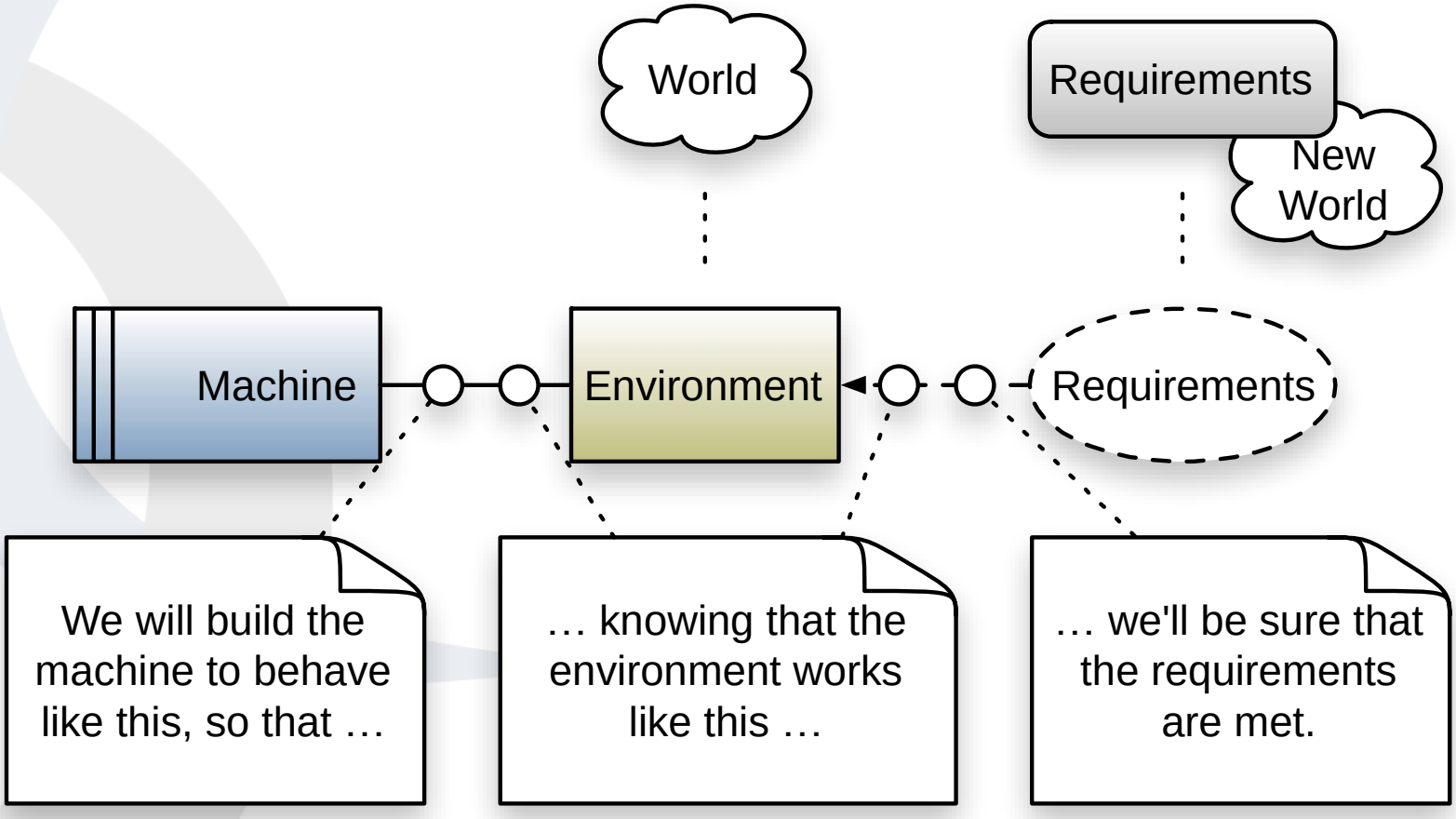
DEPENDABLE COMPUTING

How Do We Solve the Problem?



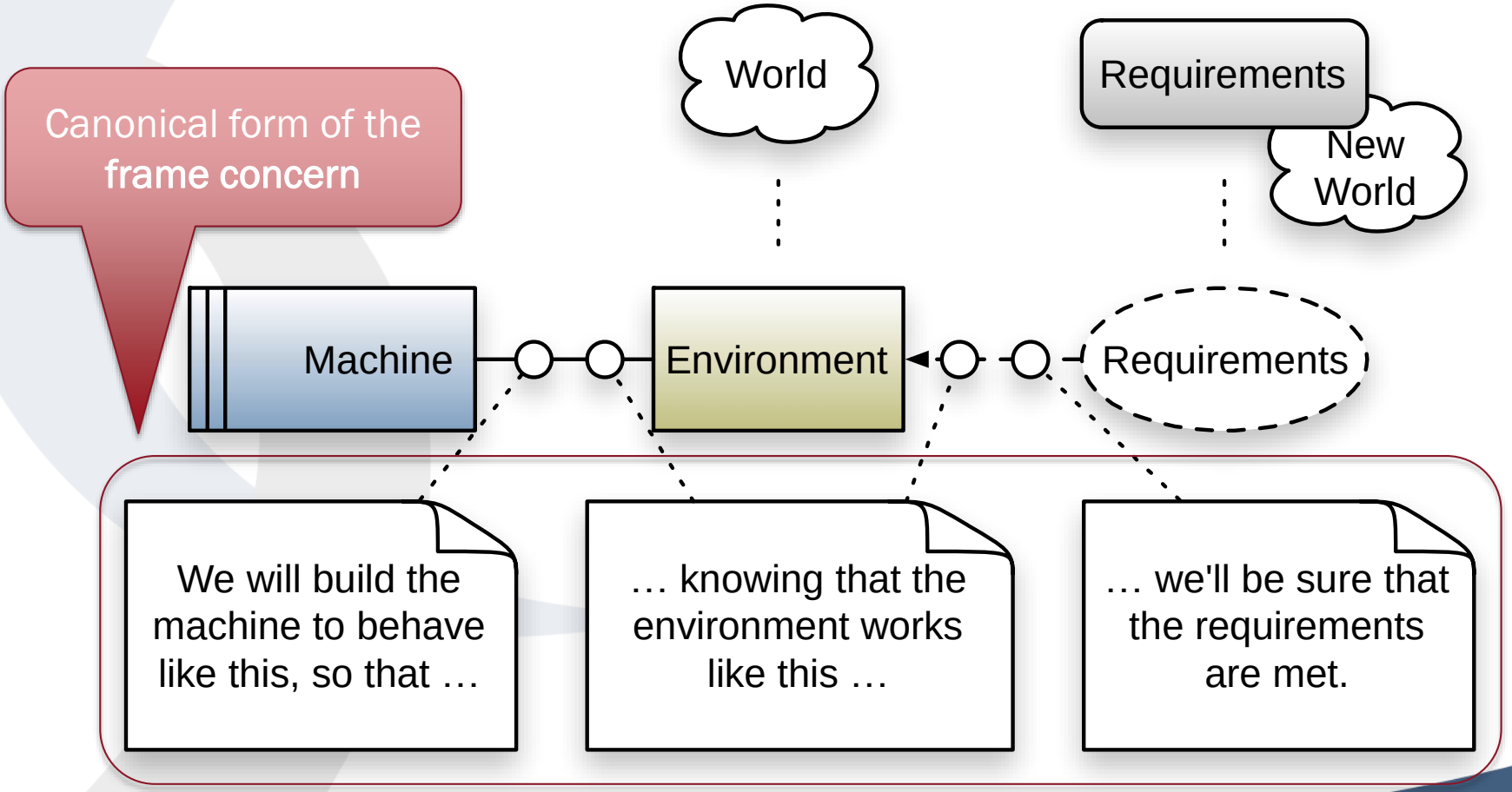
DEPENDABLE COMPUTING

How Do We Solve the Problem?



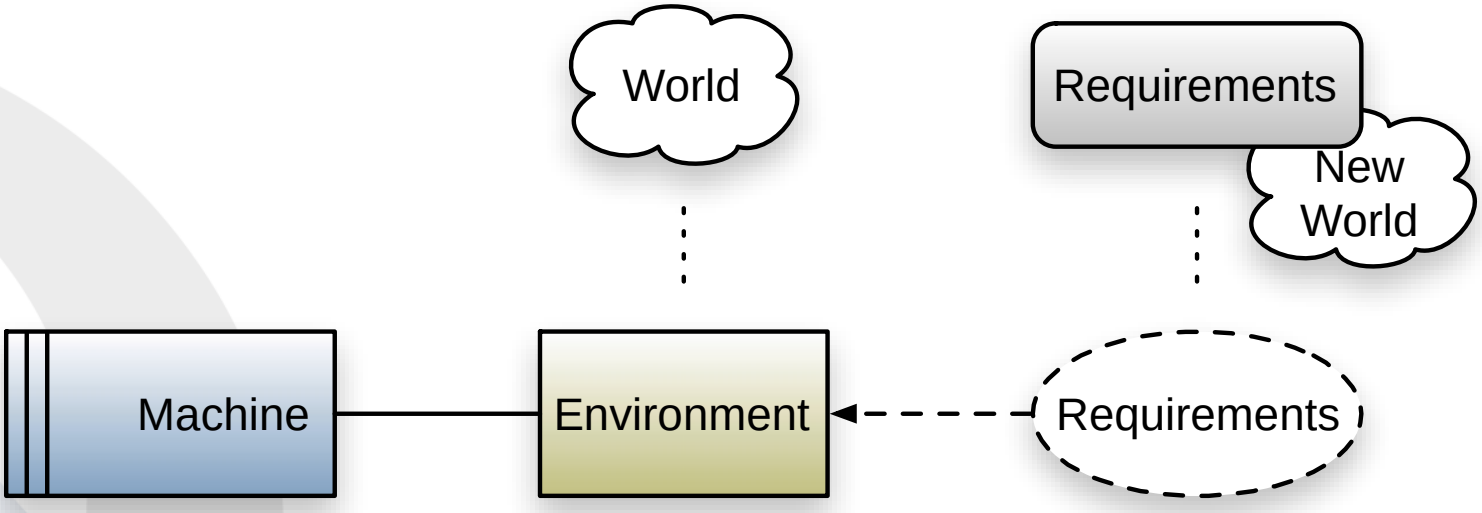
DEPENDABLE COMPUTING

How Do We Solve the Problem?



DEPENDABLE COMPUTING

How Do We Solve the Problem?



We solve the problem by satisfying the *frame concern*:

The Machine and the World satisfy the Requirements

DEPENDABLE COMPUTING

The Problem-Derived World-Situated Machine Model

DEPENDABLE COMPUTING

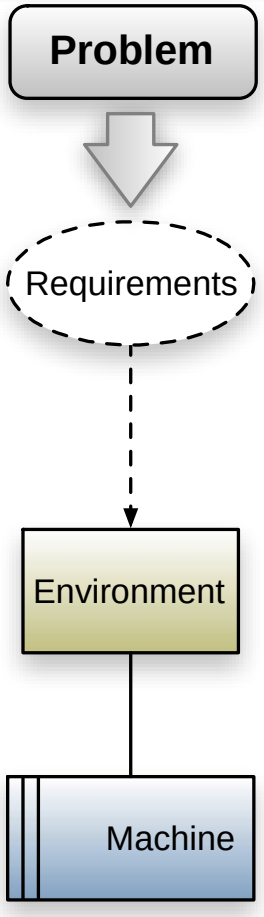
The Problem-Derived World-Situated Machine Model

Problem

Explicitly identify the problem to be solved

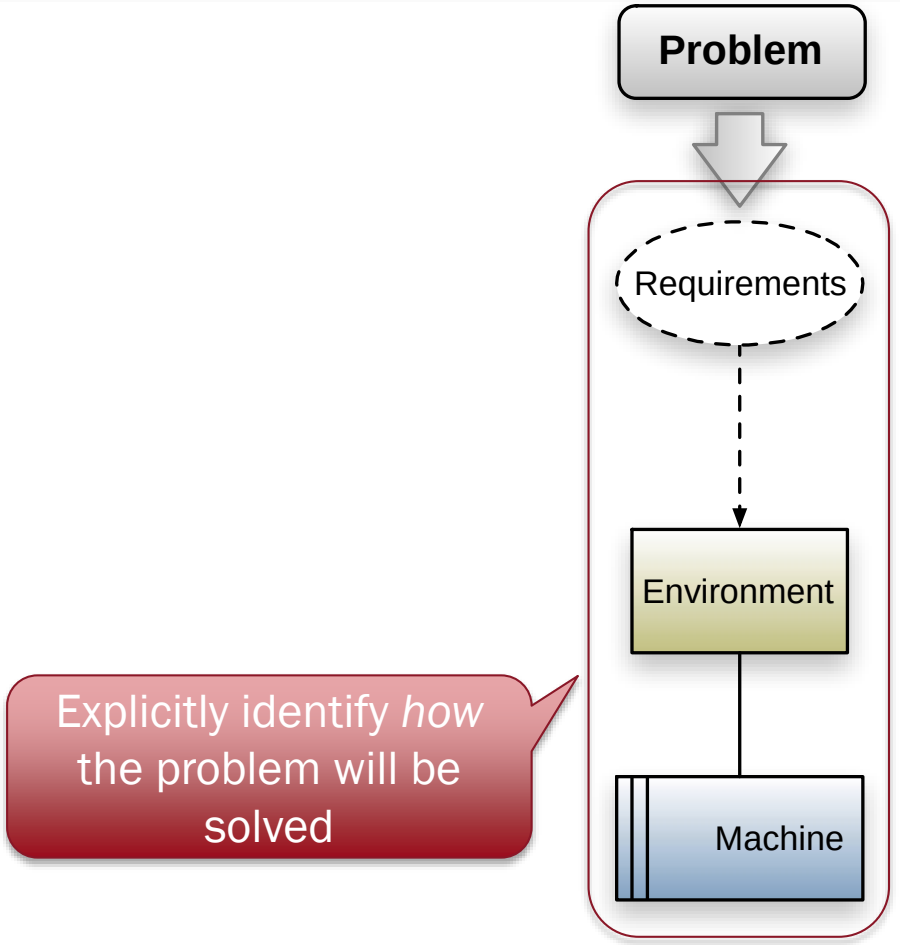
DEPENDABLE COMPUTING

The Problem-Derived World-Situated Machine Model



DEPENDABLE COMPUTING

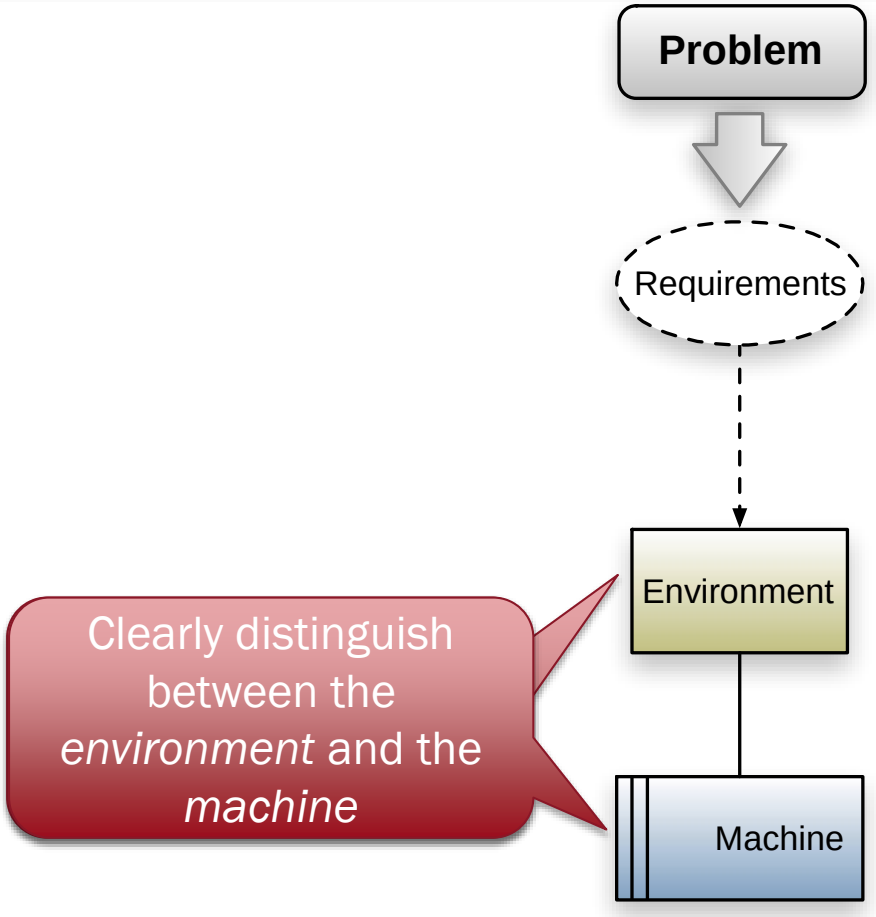
The Problem-Derived World-Situated Machine Model



Explicitly identify *how* the problem will be solved

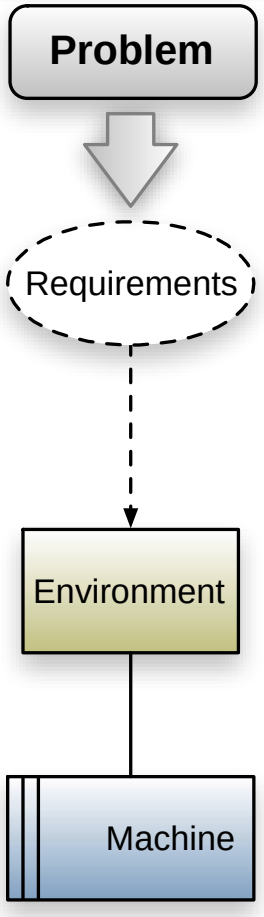
DEPENDABLE COMPUTING

The Problem-Derived World-Situated Machine Model



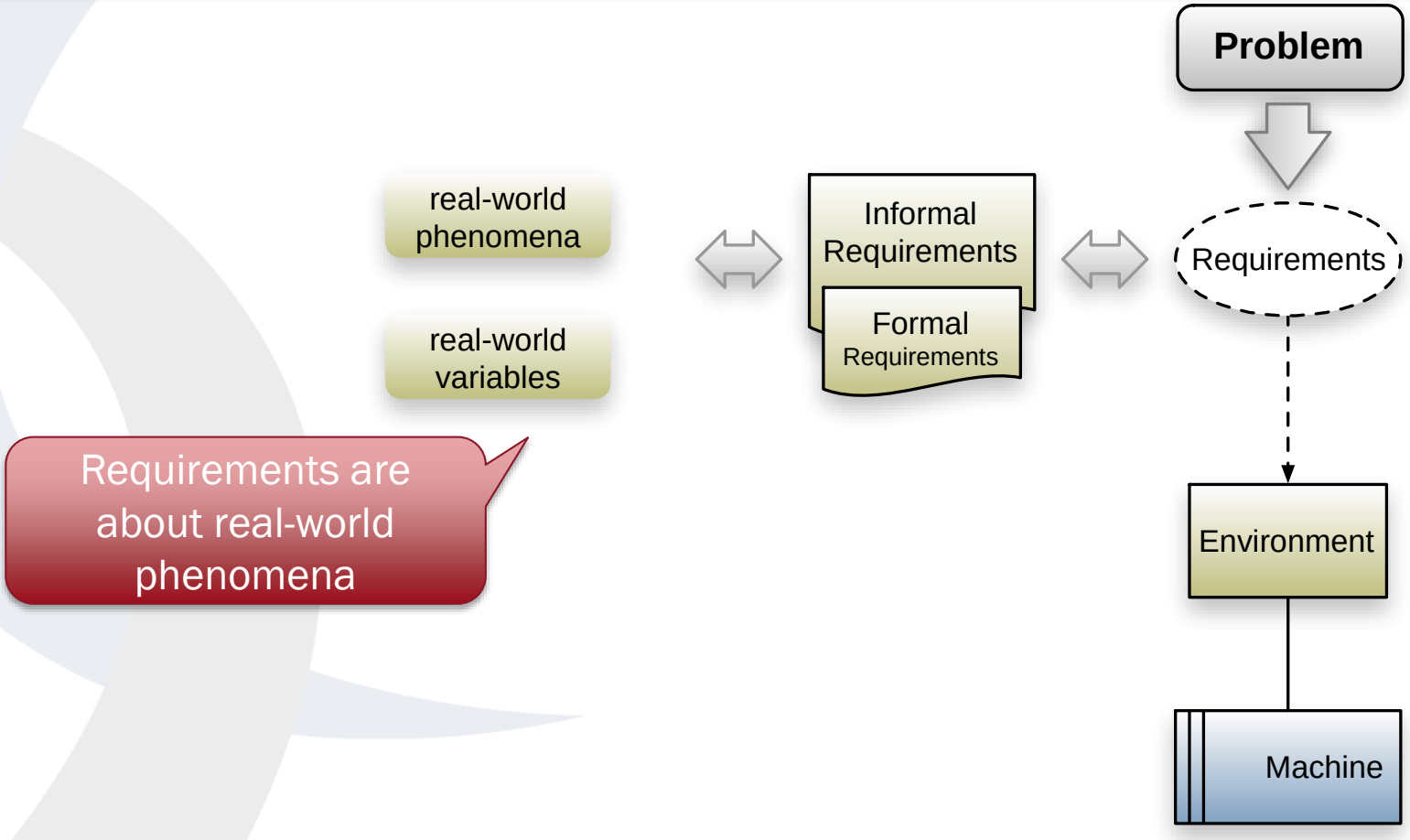
DEPENDABLE COMPUTING

The Problem-Derived World-Situated Machine Model



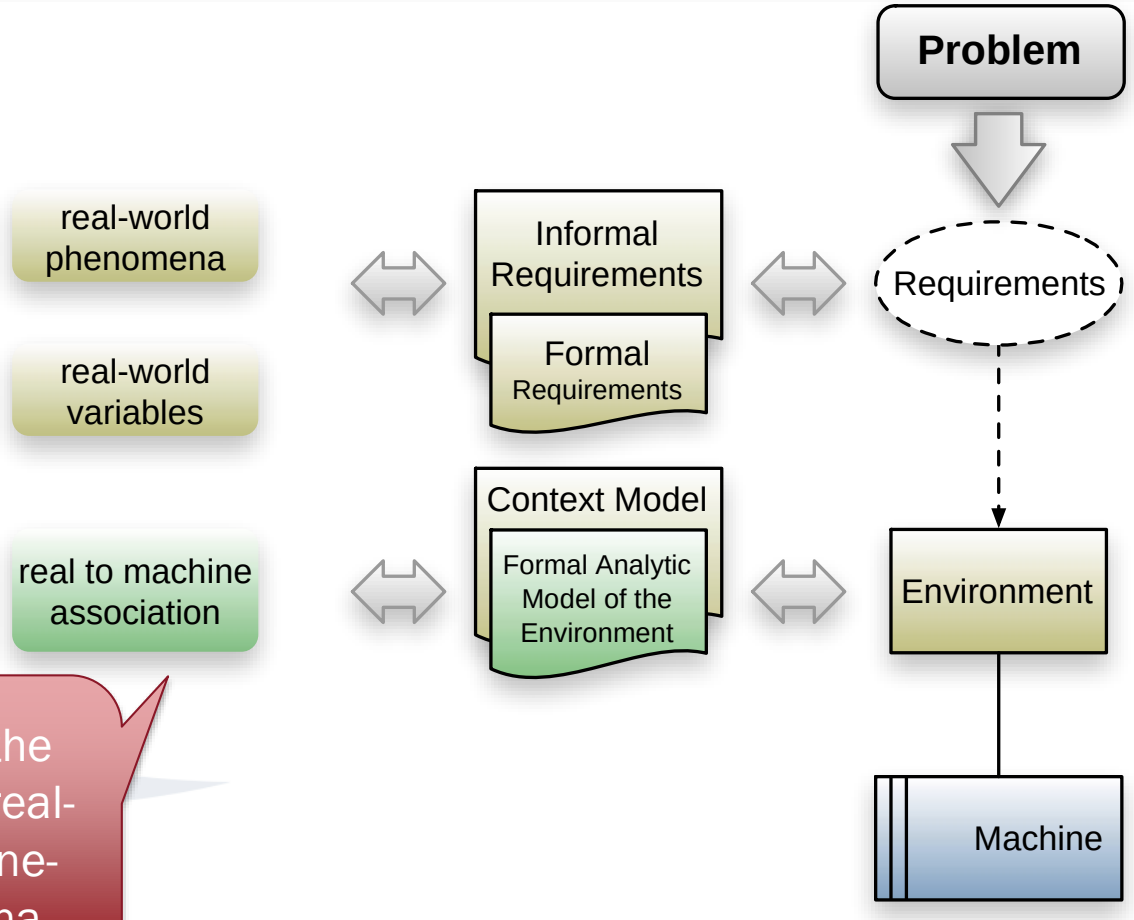
DEPENDABLE COMPUTING

The Problem-Derived World-Situated Machine Model



DEPENDABLE COMPUTING

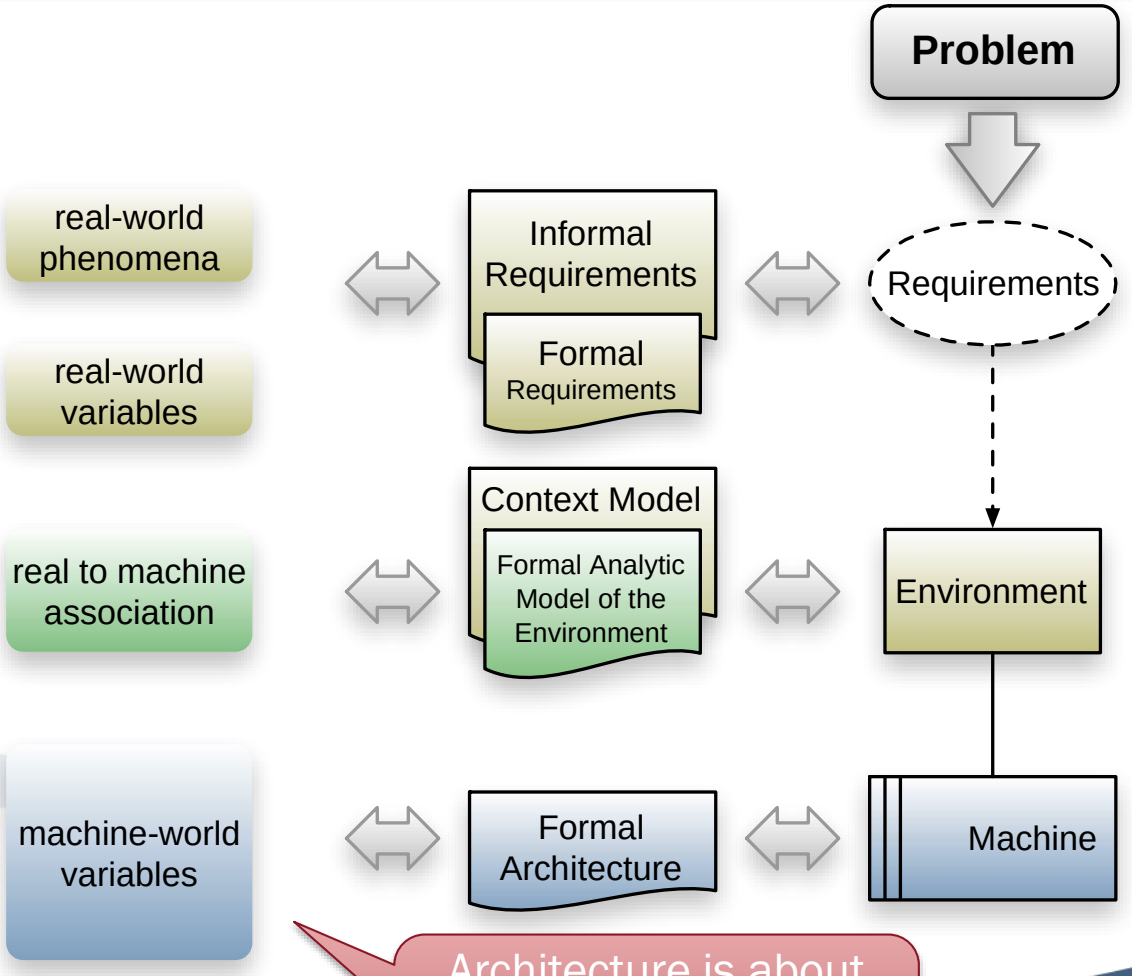
The Problem-Derived World-Situated Machine Model



Model provides the linkage between real-world and machine-world phenomena

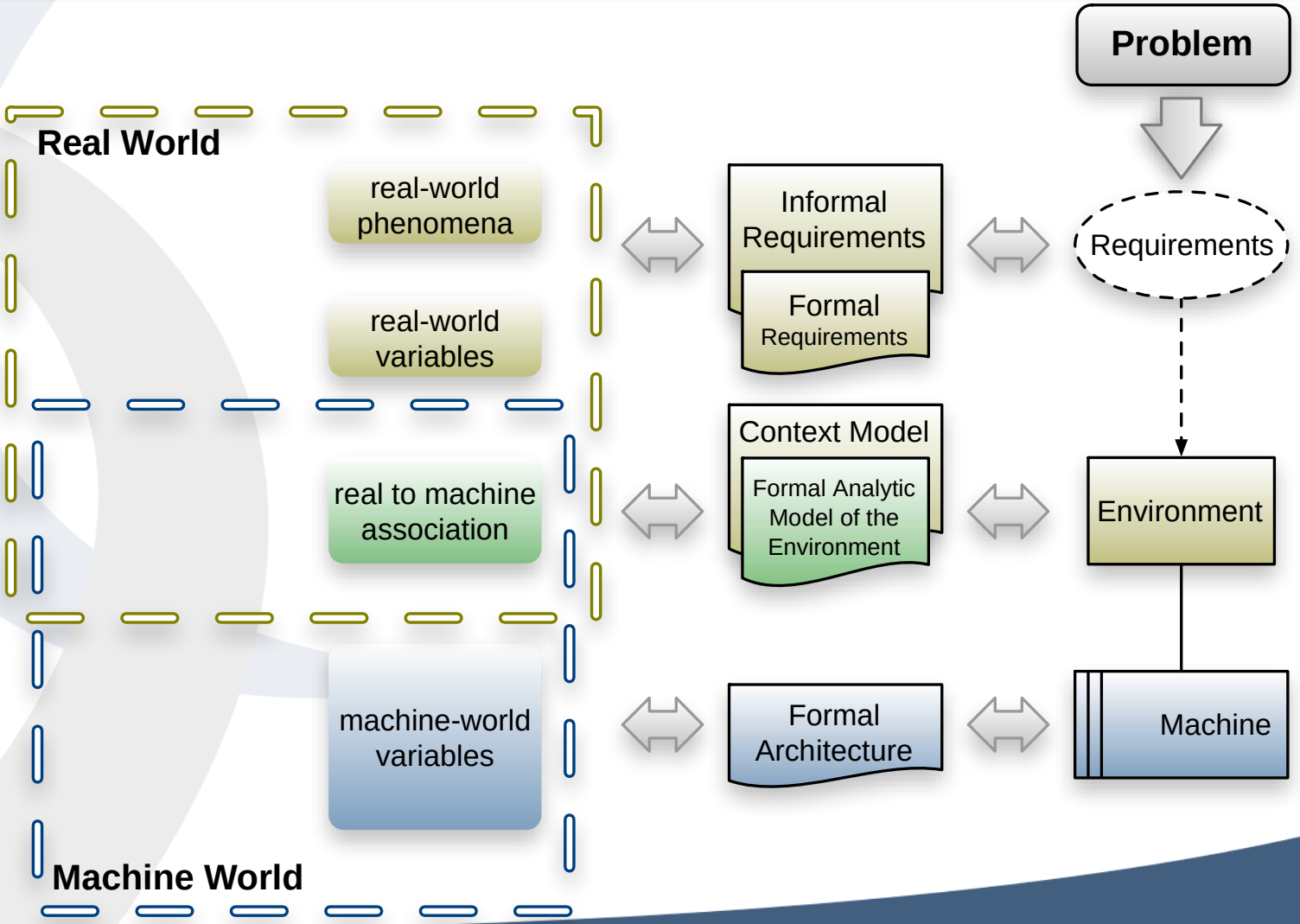
DEPENDABLE COMPUTING

The Problem-Derived World-Situated Machine Model



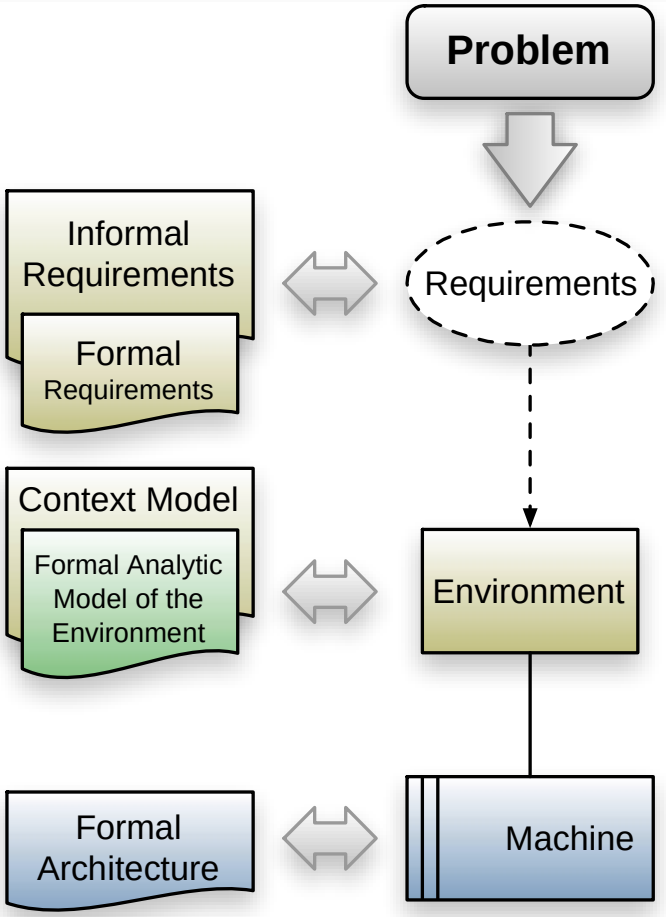
DEPENDABLE COMPUTING

The Problem-Derived World-Situated Machine Model



DEPENDABLE COMPUTING

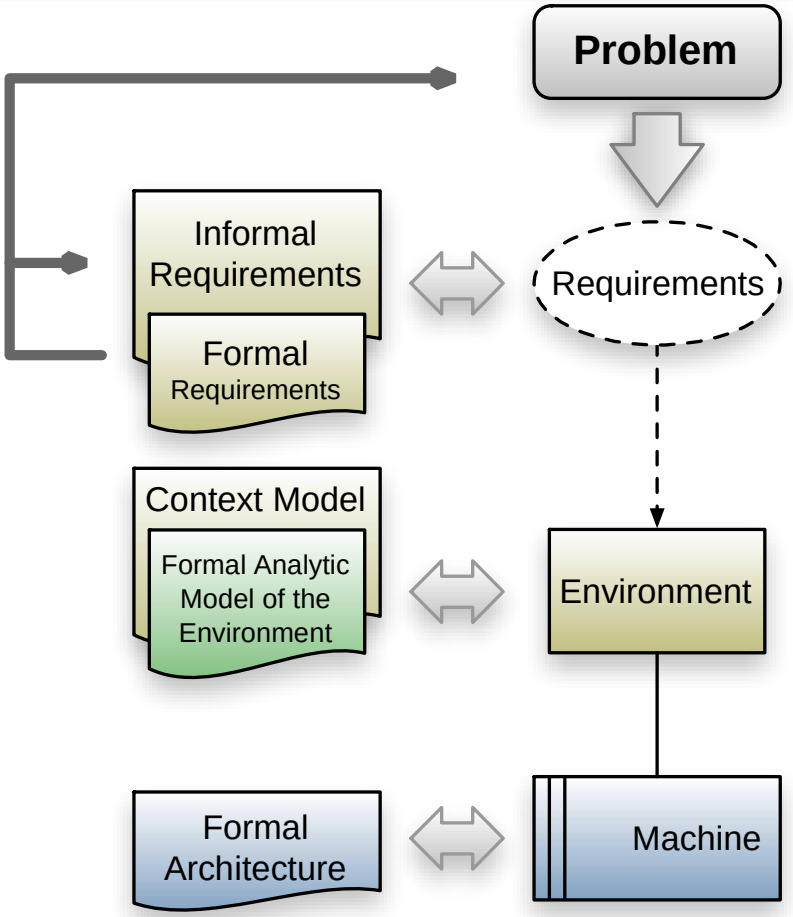
Satisfaction of the Frame Concern



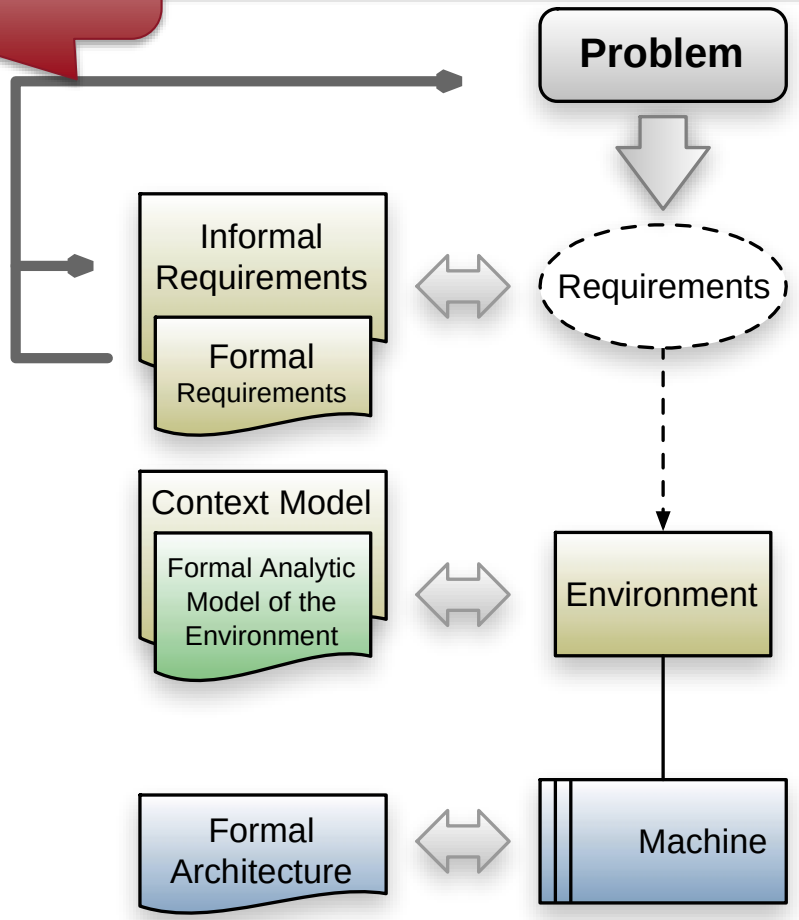
DEPENDABLE COMPUTING

Satisfaction of the Frame Concern

- Informal Validation:
  - Formal properties "imply" informal requirements
  - Informal requirements solve the problem



Sa[illegible]the Frame Concern



- Informal Validation:
  - Formal properties "imply" informal requirements
  - Informal requirements solve the problem

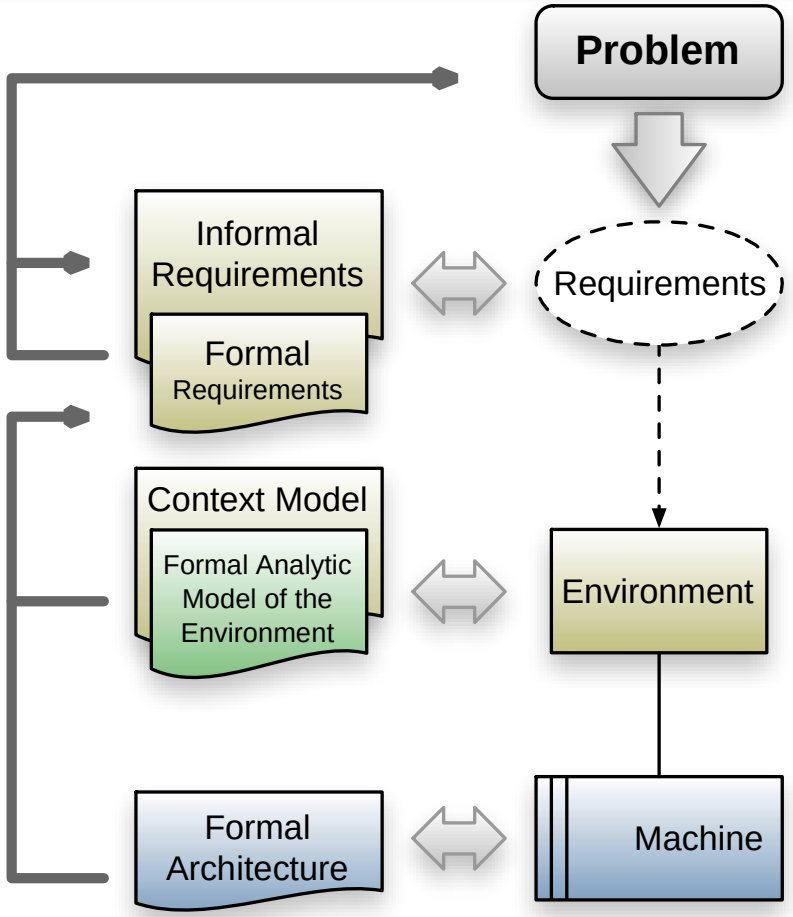
DEPENDABLE COMPUTING

Satisfaction of the Frame Concern

- Formal Validation:

$$\mathbf{M} \wedge \mathbf{W} \vdash \mathbf{R}$$

- Informal Validation:
  - Formal properties "imply" informal requirements
  - Informal requirements solve the problem



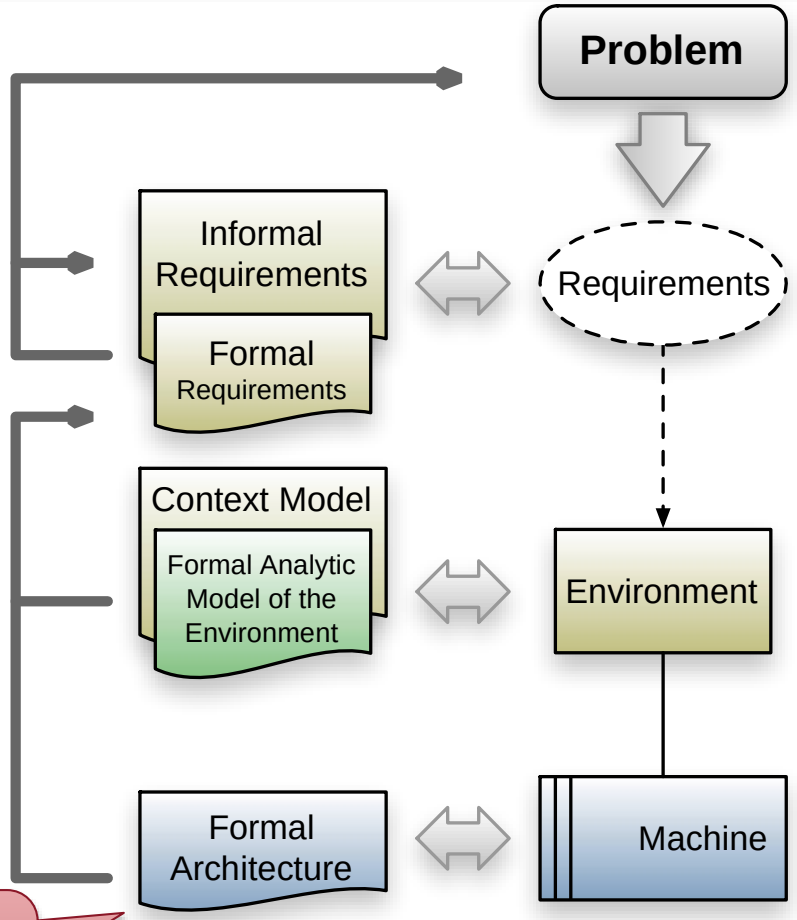
DEPENDABLE COMPUTING

Satisfaction of the Frame Concern

- Formal Validation:

$$\mathbf{M} \wedge \mathbf{W} \vdash \mathbf{R}$$

- Informal Validation:
  - Formal properties "imply" informal requirements
  - Informal requirements solve the problem



From **interfaces** through the architecture

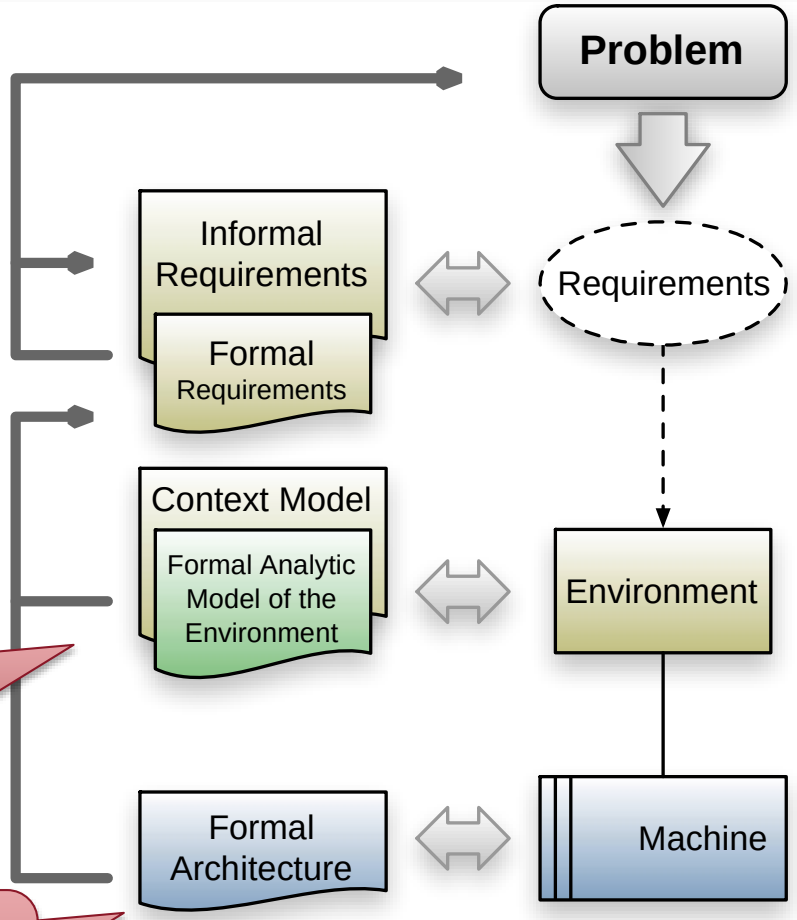
DEPENDABLE COMPUTING

Satisfaction of the Frame Concern

- Formal Validation:

$$\mathbf{M} \wedge \mathbf{W} \vdash \mathbf{R}$$

- Informal Validation:
  - Formal properties "imply" informal requirements
  - Info
    solv



Given the **context** of the environment

From **interfaces** through the architecture

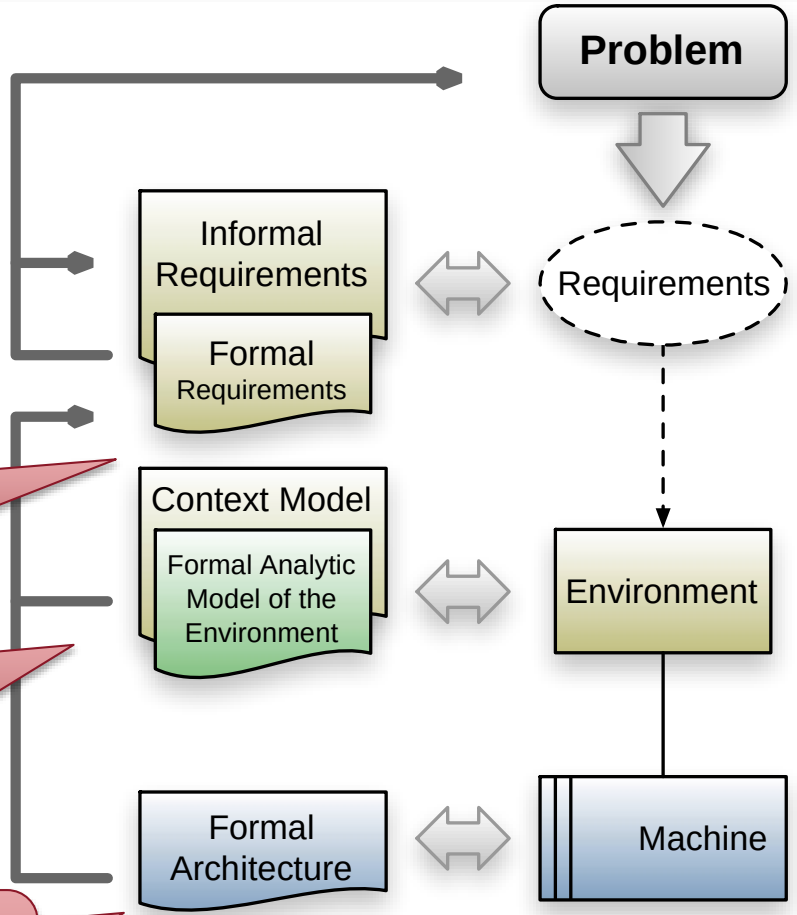
DEPENDABLE COMPUTING

Satisfaction of the Frame Concern

- Formal Validation:

$$\mathbf{M} \wedge \mathbf{W} \vdash \mathbf{R}$$

- Informal Validation:
  - For[illegible] "imp[illegible] requirements
  - Info[illegible] solv[illegible]



Requirements are met

Given the **context** of the environment

From **interfaces** through the architecture

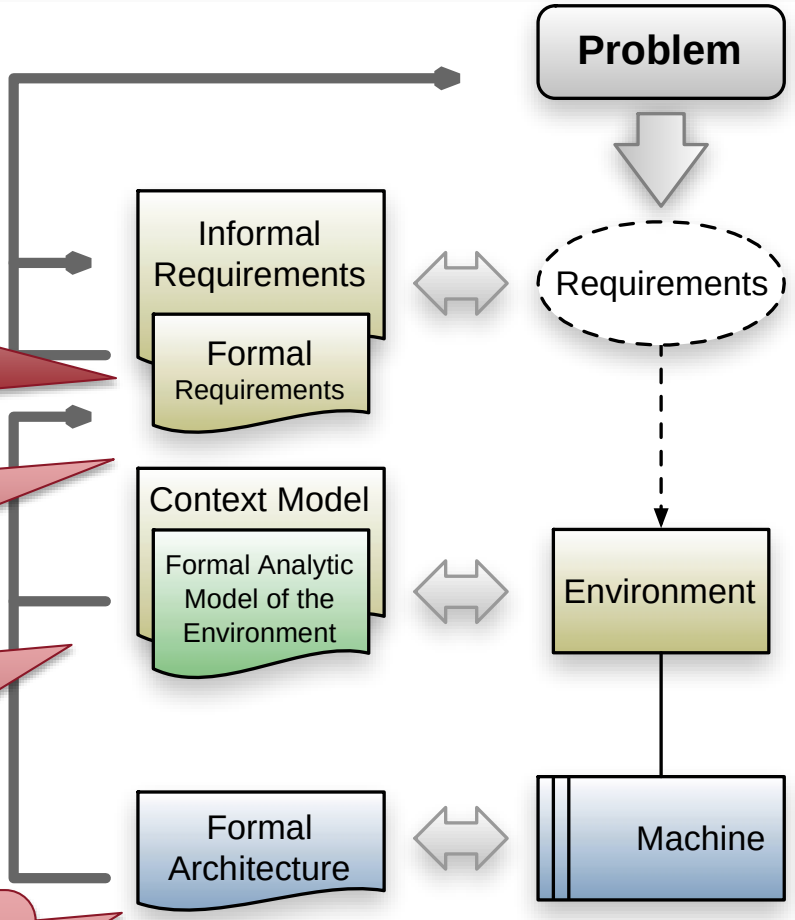
DEPENDABLE COMPUTING

Satisfaction of the Frame Concern

- Formal Validation:

$$M \wedge W \vdash R$$

- Informa
  - For
    "imp
    requ
  - Info
    solv



Interfaces at the next level up

Requirements are met

Given the **context** of the environment

From **interfaces** through the architecture

DEPENDABLE COMPUTING

Satisfaction of the Frame Concern

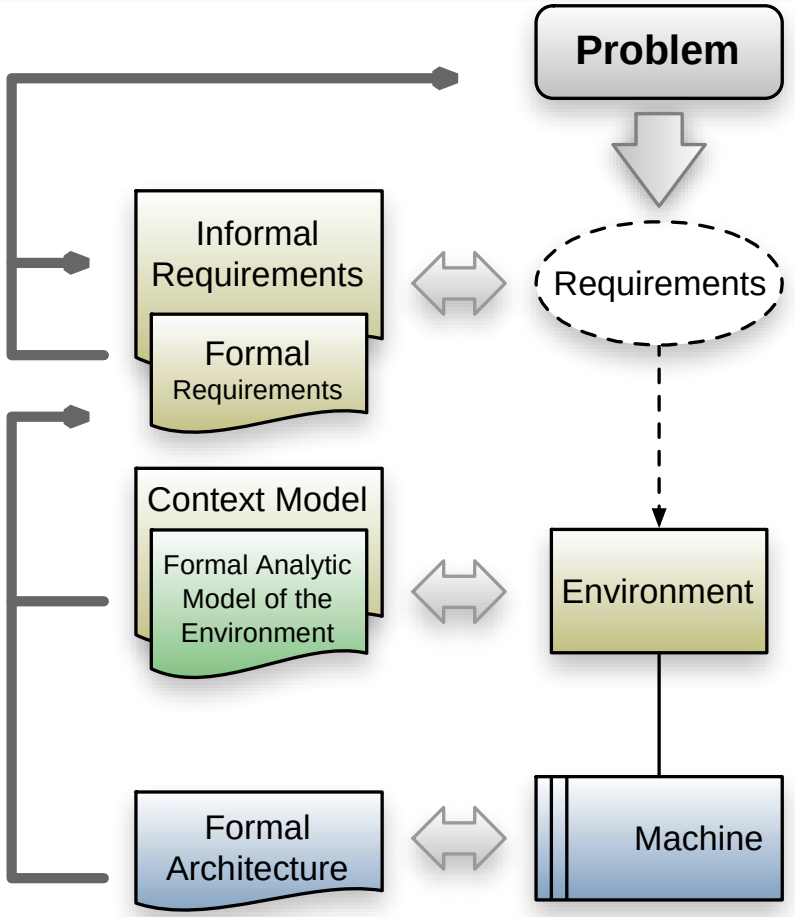
- Formal Validation:

$$M \wedge W \vdash R$$

Proof

– "imply" informal requirements

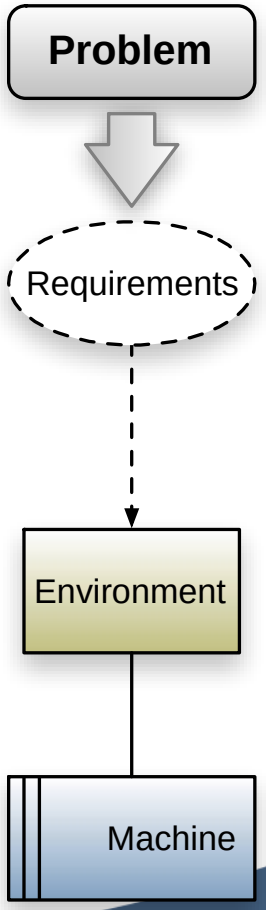
– Informal requirements solve the problem



DEPENDABLE COMPUTING

The Problem-Derived World-Situated Machine Model

- Composite model of the machine and its environment
- Separates the descriptions of
  - Machine with *machine-world* phenomena
  - Environment with *real-world* phenomena
- Clearly identifies
  - *Interfaces* at the requirements and machine
  - *Context* in the environment
- Enables **formal validation** and **assured reuse**



DEPENDABLE COMPUTING

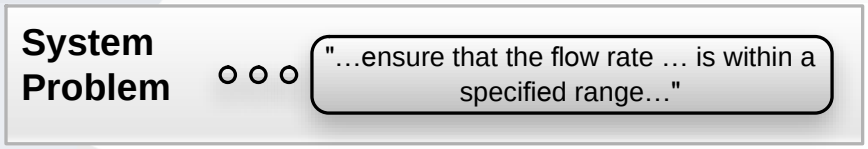
Liquid Tank: Problem Statement

System Problem

"…ensure that the flow rate … is within a specified range…"

DEPENDABLE COMPUTING

Liquid Tank: Problem Statement



A buffering tank is needed for a chemical processing application. Currently, a pump provides liquid at a constant flow rate for a subsequent stage of processing. However, the pump occasionally introduces gas or vacuum cavities into the liquid, reducing the efficiency of processing in the subsequent stage. By buffering the liquid in a tank, the cavities can dissipate, improving overall production efficiency.

[…]

1. The tank system must ensure that the flow rate for liquid leaving the tank is within a specified range, or zero.

[…]

DEPENDABLE COMPUTING

Liquid Tank: Problem Statement

System Problem

"…ensure that the flow rate … is within a specified range…"

A buffering tank is needed for a chemical processing application. Currently, a pump provides liquid at a constant flow rate for a subsequent stage of processing. However, the pump occasionally introduces gas or vacuum cavities into the liquid, reducing the efficiency of processing in the subsequent stage. By buffering the liquid in a tank, the cavities can dissipate, improving overall production efficiency.

[…]

1. The tank system must ensure that the flow rate for liquid leaving the tank is within a specified range, or zero.

[…]

DEPENDABLE COMPUTING

Liquid Tank: Requirements

System Problem

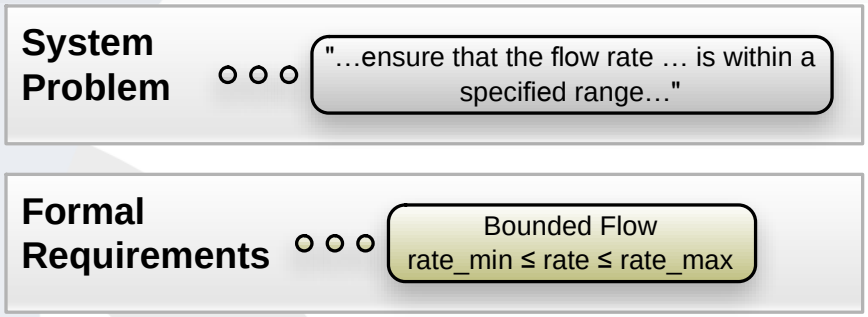
"…ensure that the flow rate … is within a specified range…"

Formal Requirements

Bounded Flow
rate_min ≤ rate ≤ rate_max

DEPENDABLE COMPUTING

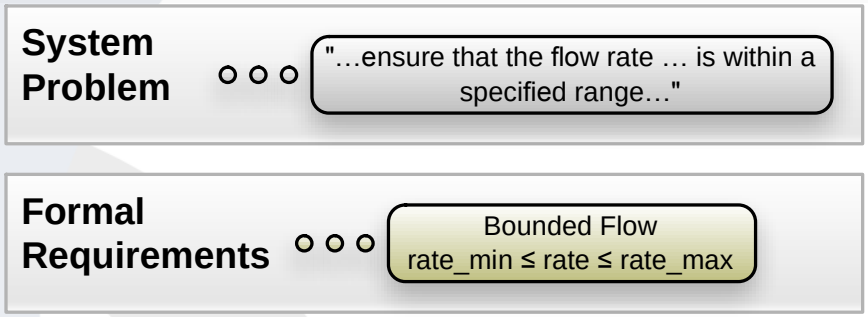
Liquid Tank: Requirements



- **Invariant Requirements**
  - SI1: The tank system shall have two modes: normal and emergency.
  - […]
- **Normal-Mode Requirements**
  - SN1: The liquid outflow rate shall be between *rate_min* and *rate_max*.
  - […]
- […]

DEPENDABLE COMPUTING

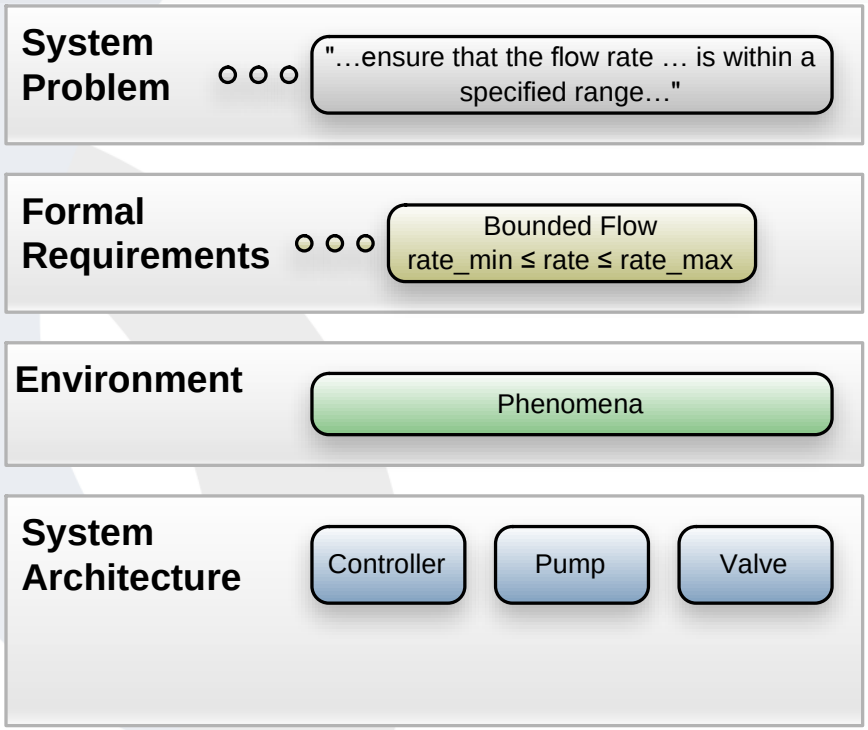
Liquid Tank: Requirements



- **Invariant Requirements**
  - SI1: The tank system shall have two modes: normal and emergency.
  - […]
- **Normal-Mode Requirements**
  - SN1: The liquid outflow rate shall be between *rate_min* and *rate_max*.
  - […]
- […]

DEPENDABLE COMPUTING

Liquid Tank: Architecture



DEPENDABLE COMPUTING

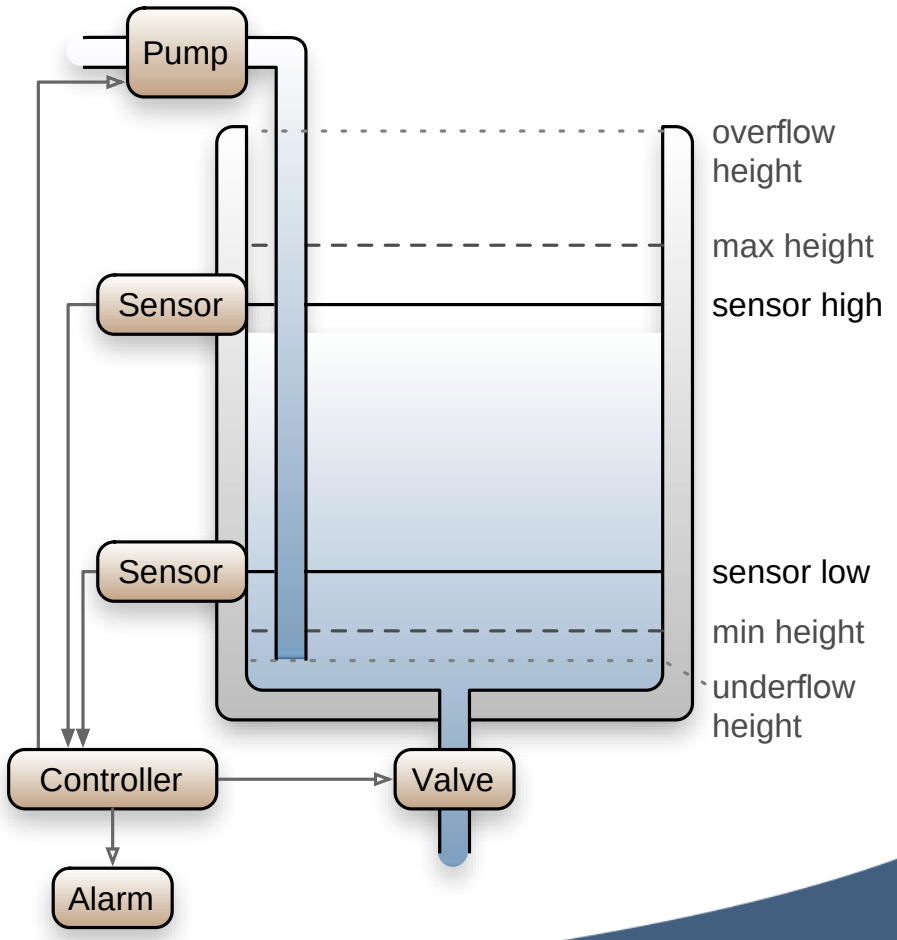
Liquid Tank: Architecture

System Problem "…ensure that the flow rate … is within a specified range…"

Formal Requirements Bounded Flow rate_min ≤ rate ≤ rate_max

Environment Phenomena

System Architecture Controller, Pump, Valve



DEPENDABLE COMPUTING

Liquid Tank: Architecture

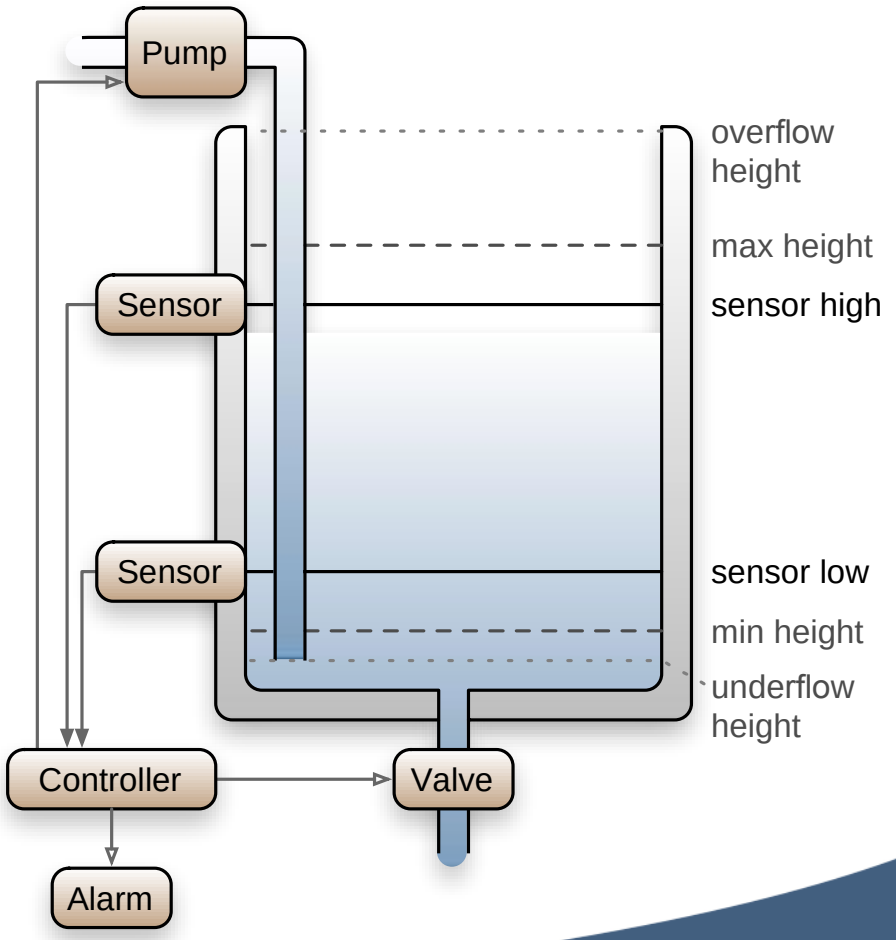
System Problem "…ensure that the flow rate … is within a specified range…"

Formal Requirements Bounded Flow: rate_min ≤ rate ≤ rate_max

Environment Phenomena

System Architecture Controller, Pump, Valve

Controller Contract, Pump Contract, Valve Contract



DEPENDABLE COMPUTING

Liquid Tank: Architecture

System Problem

"…ensure that the flow rate … is within a specified range…"

Formal Requirements

Bounded Flow
rate_min ≤ rate ≤ rate_max

Environment

Phenomena

System Architecture

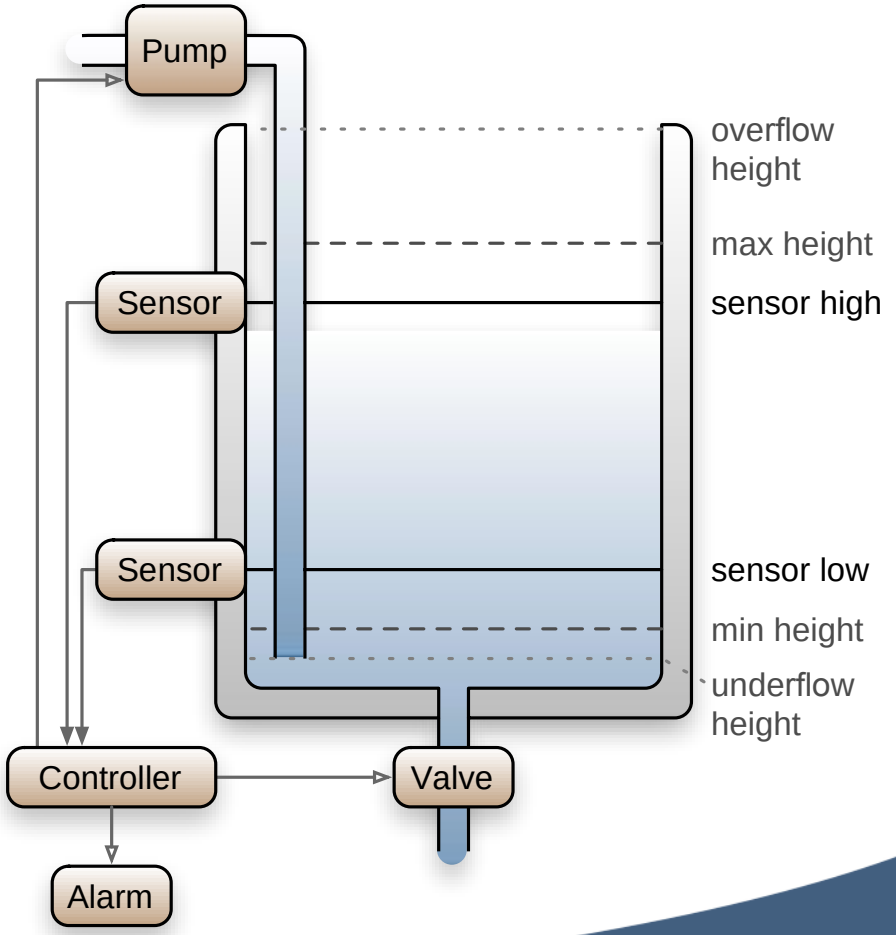
Controller | Pump | Valve

Controller Contract | Pump Contract | Valve Contract

Phenomena | Phenomena | Phenomena

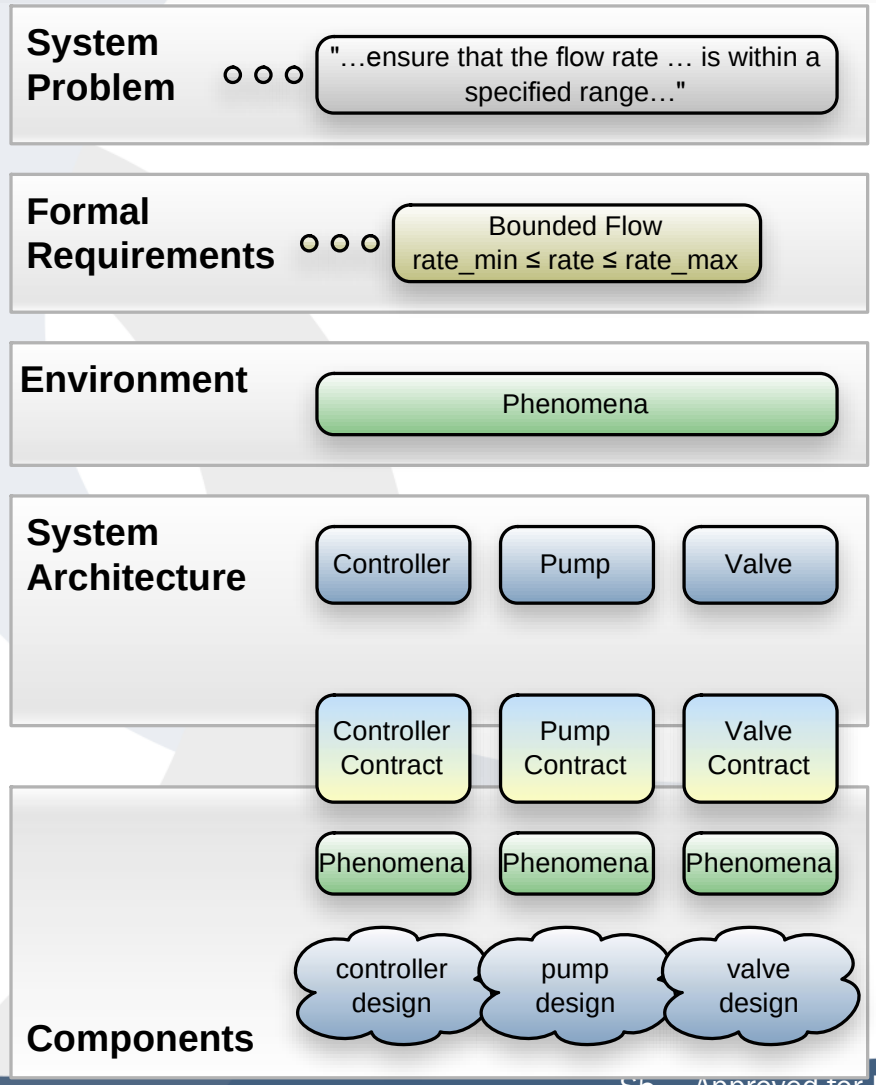
Components

controller design | pump design | valve design



DEPENDABLE COMPUTING

Liquid Tank: Validation



DEPENDABLE COMPUTING

Liquid Tank: Validation

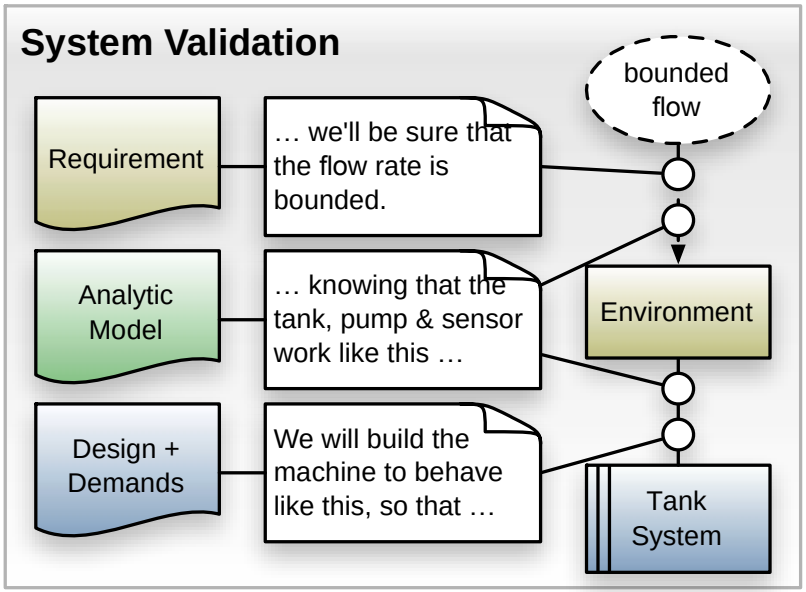
System Problem "…ensure that the flow rate … is within a specified range…"

Formal Requirements Bounded Flow: rate_min ≤ rate ≤ rate_max

Environment Phenomena

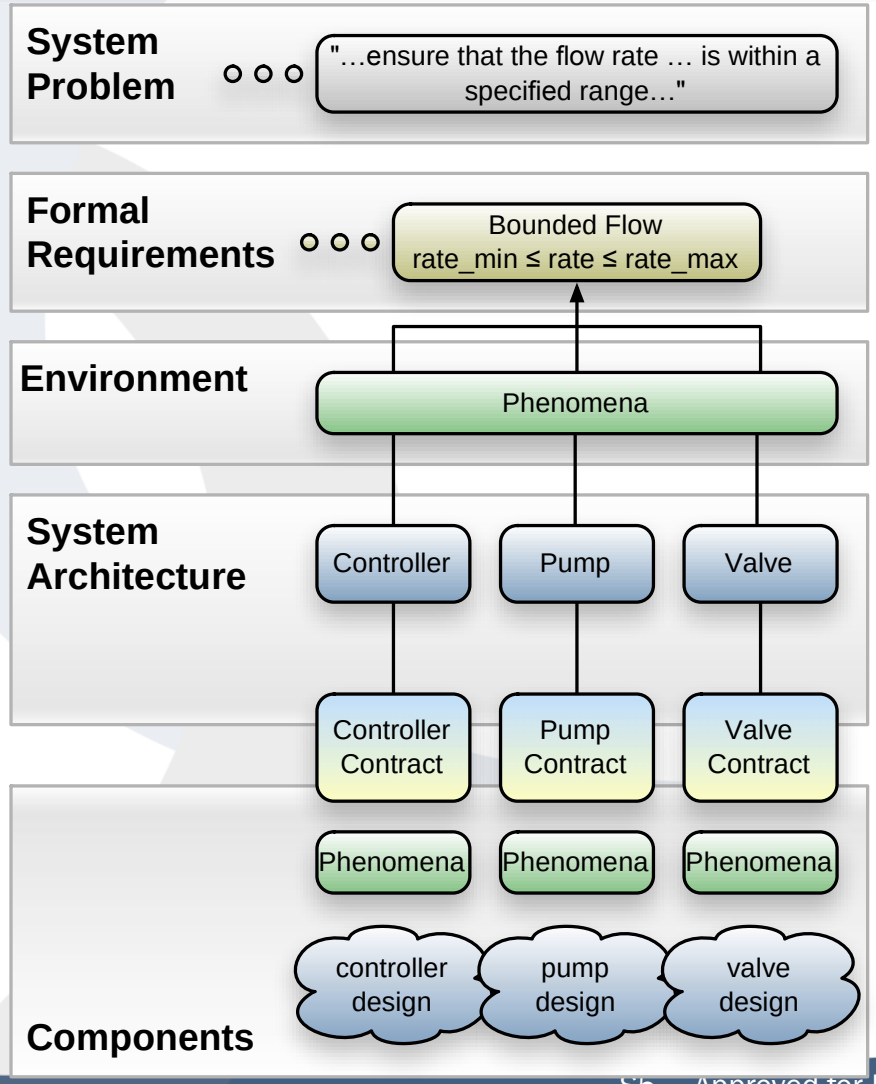
System Architecture Controller, Pump, Valve

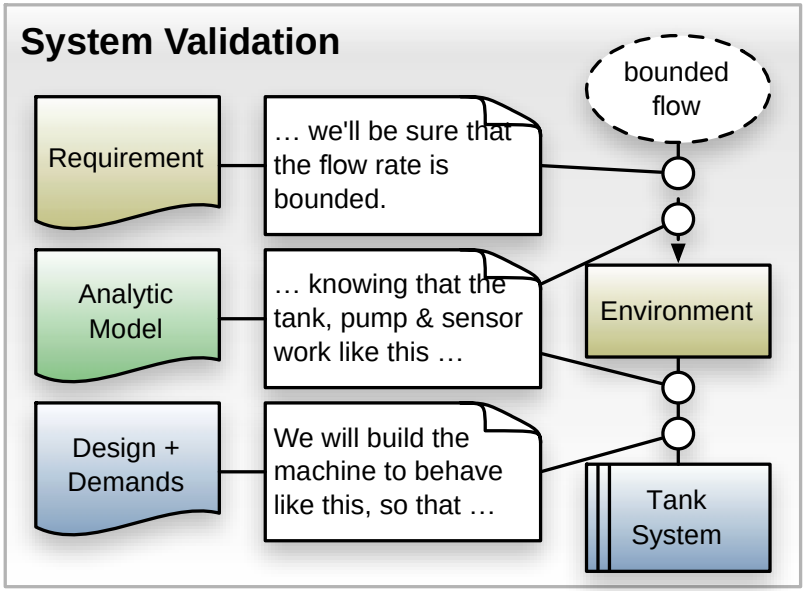
Components Controller Contract, Pump Contract, Valve Contract; Phenomena, Phenomena, Phenomena; controller design, pump design, valve design



DEPENDABLE COMPUTING

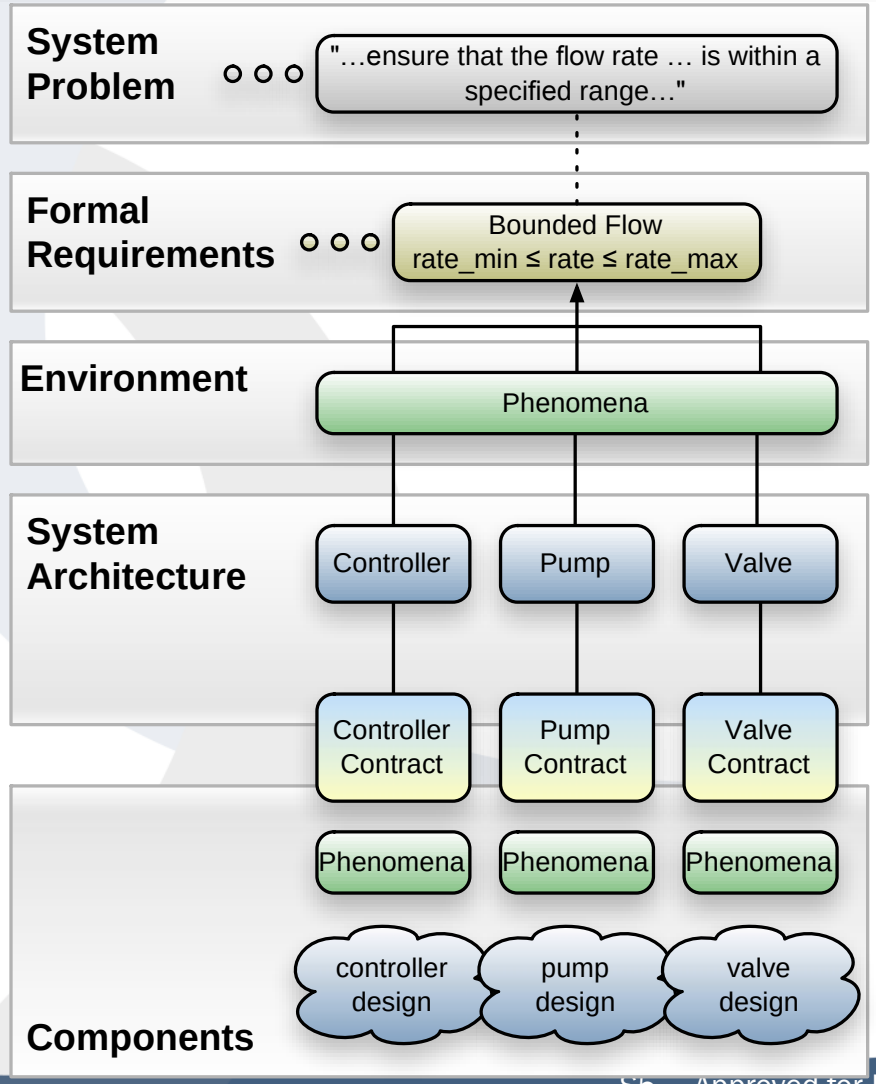
Liquid Tank: Validation

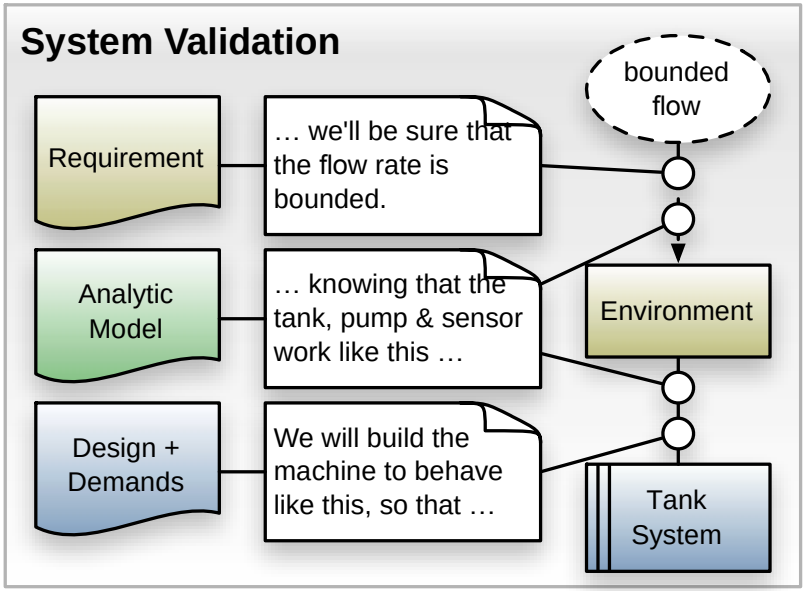




DEPENDABLE COMPUTING

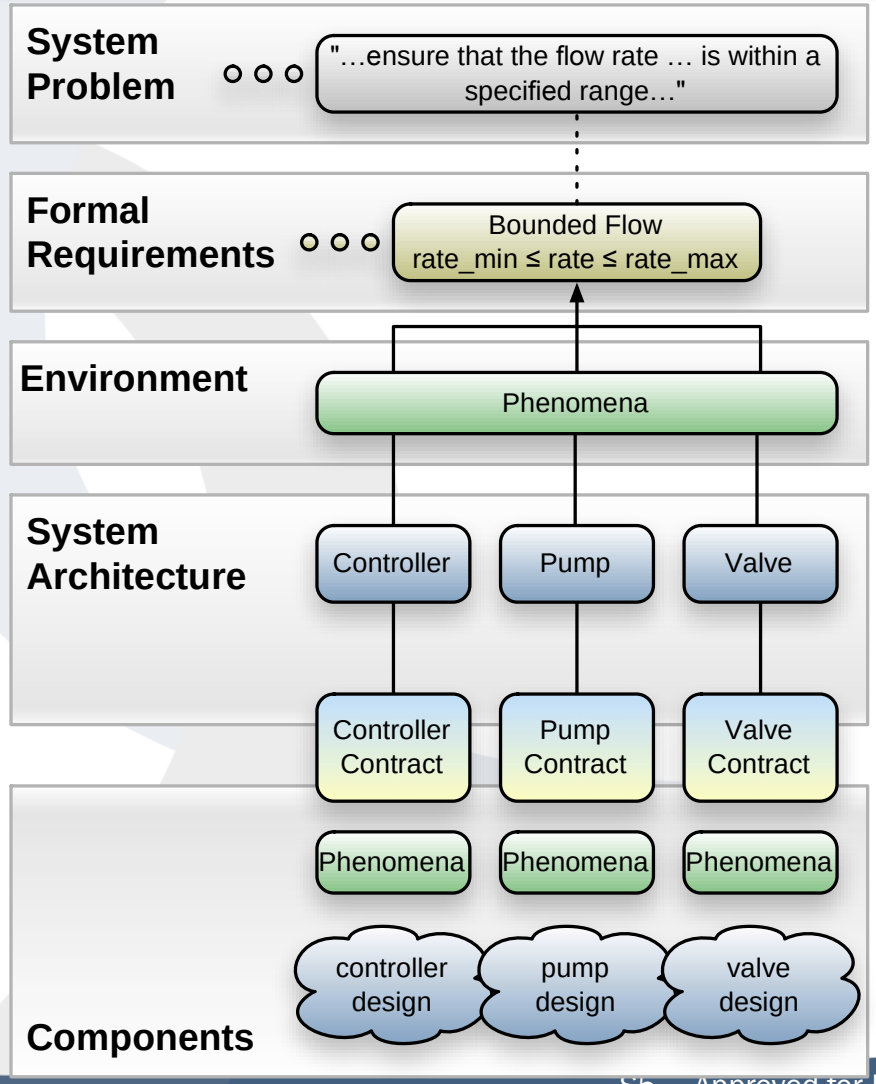
Liquid Tank: Validation

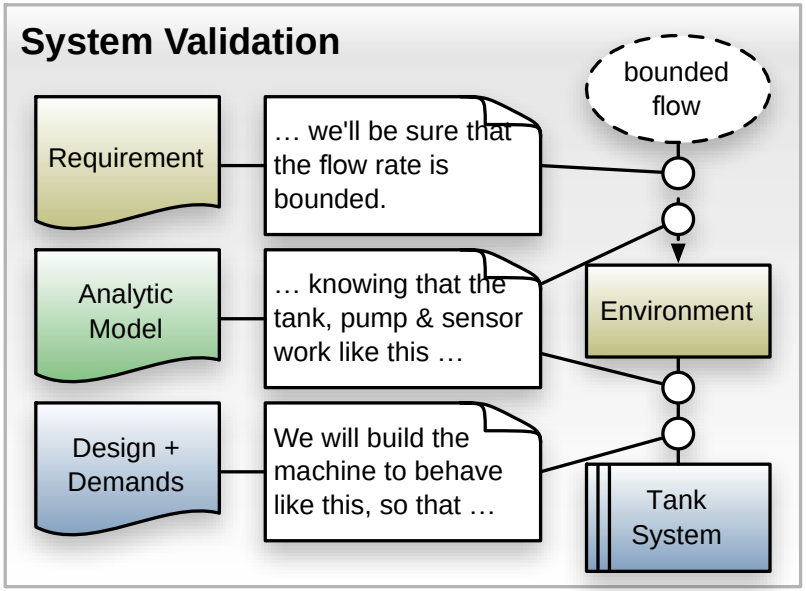




DEPENDABLE COMPUTING

Liquid Tank: Validation

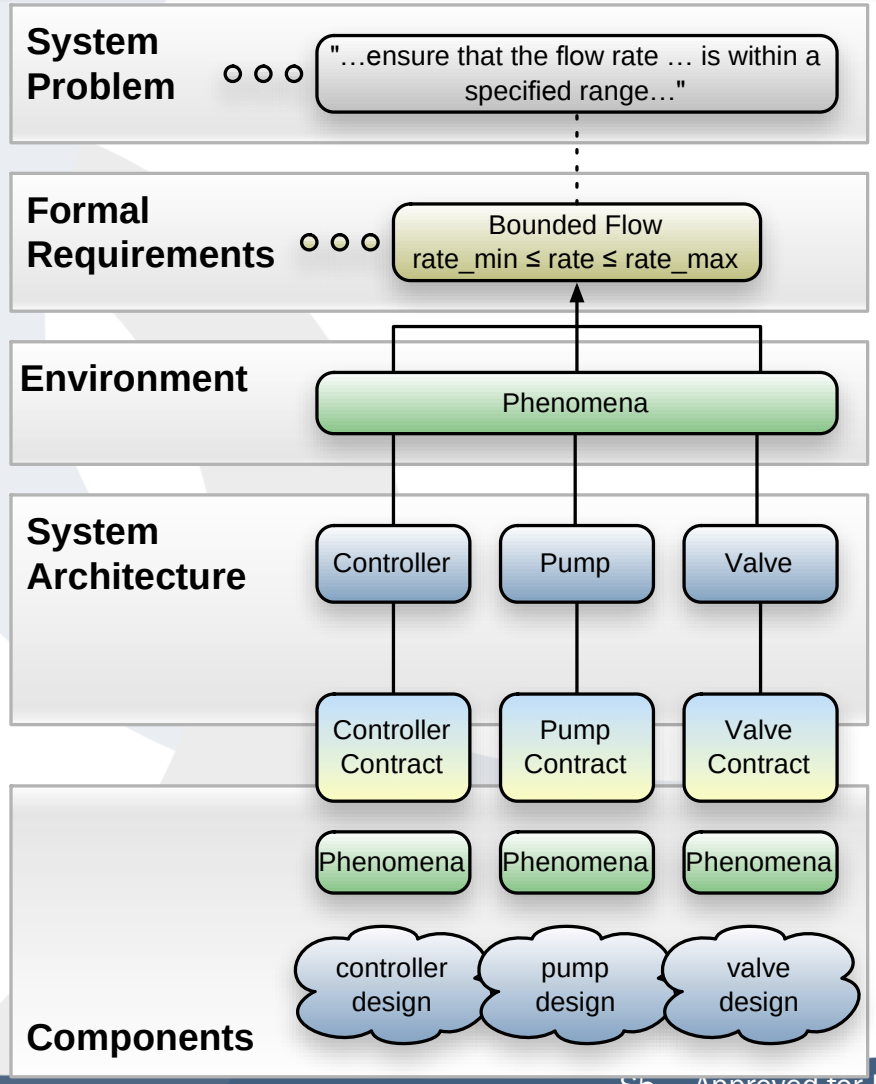


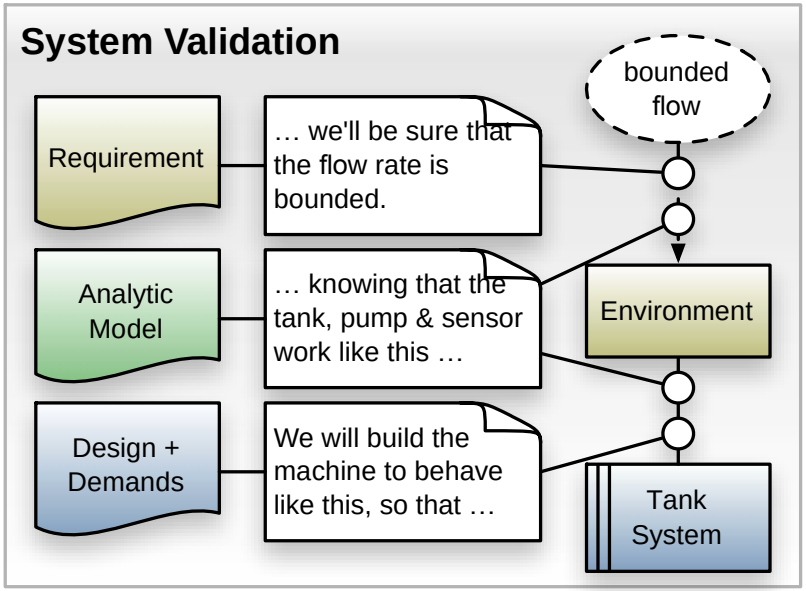


Analysis of Contract Fulfillment

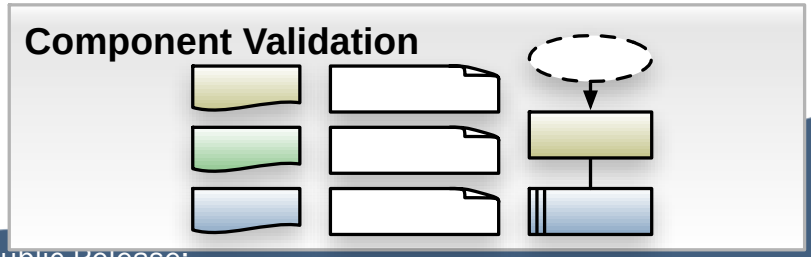
DEPENDABLE COMPUTING

Liquid Tank: Validation



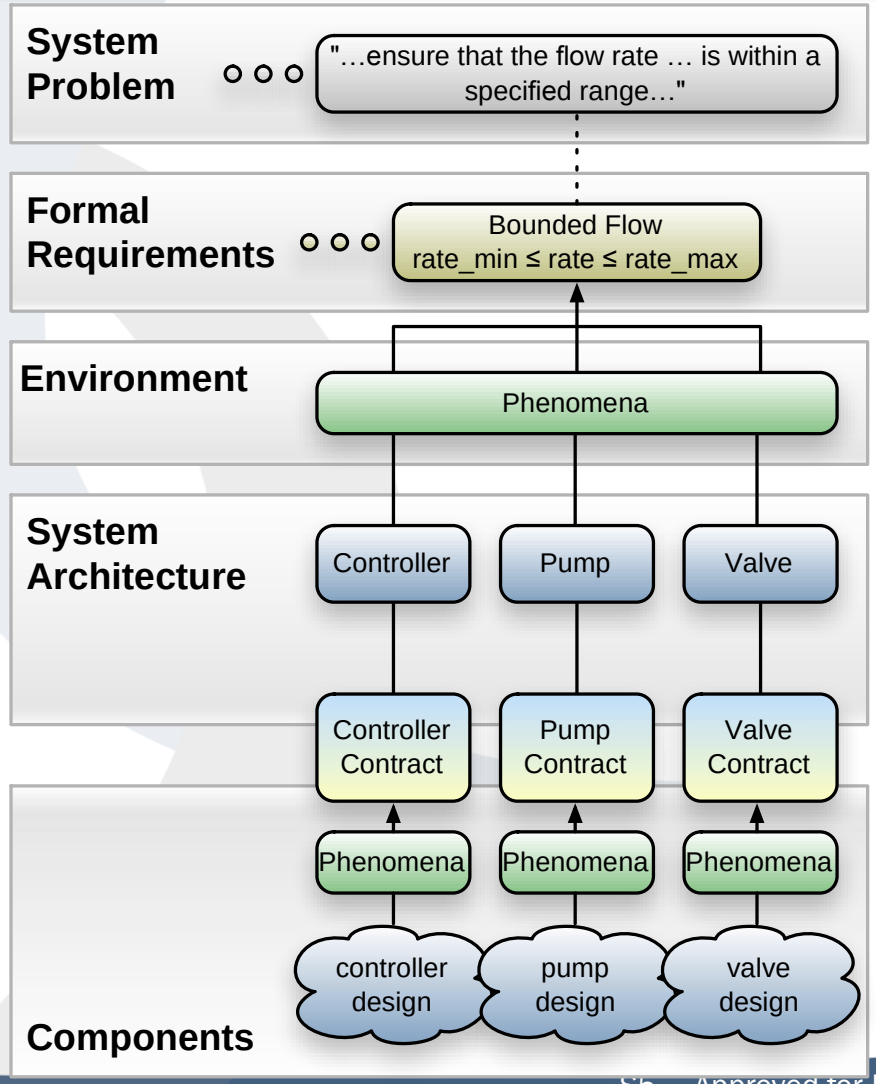


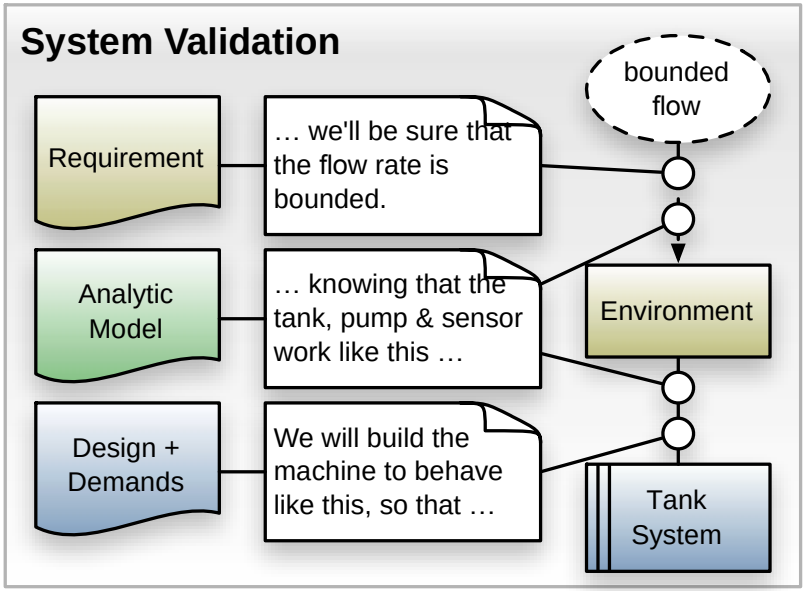
Analysis of Contract Fulfillment



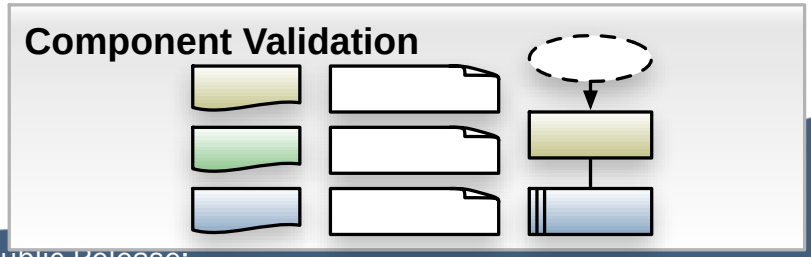
DEPENDABLE COMPUTING

Liquid Tank: Validation





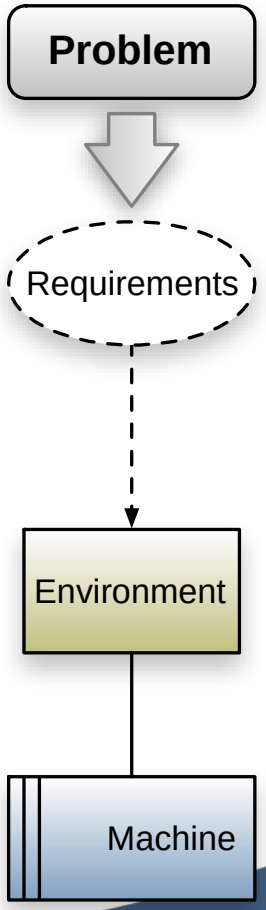
Analysis of Contract Fulfillment



DEPENDABLE COMPUTING

The Problem-Derived World-Situated Machine Model

- Composite model of the machine and its environment
- Separates the descriptions of
  - Machine with *machine-world* phenomena
  - Environment with *real-world* phenomena
- Clearly identifies
  - *Interfaces* at the requirements and machine
  - *Context* in the environment
- Enables **formal validation** and **assured reuse**



DEPENDABLE
COMPUTING

Questions?