A Comparison of Two Software Architectural Styles for Space-Based Control Systems

Why State Encapsulation is Not a Universal Good Thing

Daniel Dvorak, Ph.D.
Information Technologies and Software Systems Division
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

Abstract

In the hardware/software design of control systems it is almost an article of faith to decompose a system into loosely coupled subsystems, with state variables encapsulated inside device and subsystem objects. The engineering advantages of such an approach are so attractive that it is sometimes applied inappropriately, yielding a design that hides a tangle of special-case subsystem-to-subsystem couplings behind a façade of modular decomposition.

The limitations of a subsystem/device architecture become apparent in the design of resource-limited control systems, such as planetary rovers, where the world is full of physical side-effects that have little "respect" for conventional subsystem boundaries. Here, the very notion of decomposition by subsystem, and its attendant state encapsulation, actually complicates the design. Fundamentally, there is a clash between a subsystem-device-object metaphor and the laws of physics. A more appropriate architectural approach is to acknowledge the underlying physics and to elevate the concepts of *state* and *models* to first-class design elements that are *not* encapsulated within subsystem objects.

The Road to Encapsulation

Object-Orient Analysis

↓

Anthropomorphic Story-Telling

↓

Ownership ---------- Appealing Metaphor

↓

Subsystem/Device Architecture ---------- Appealing Pattern Reuse *and* Work Breakdown

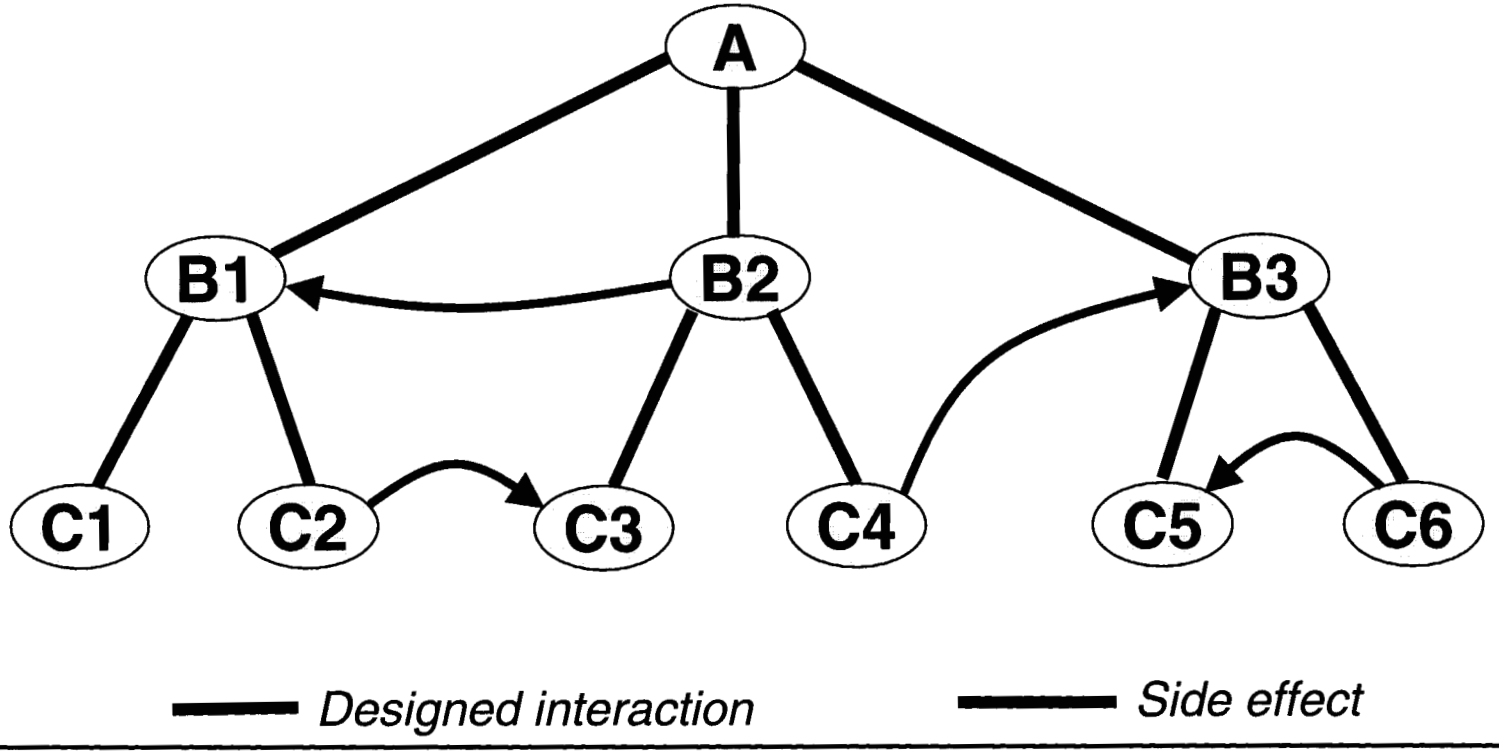
↓

State Encapsulation ---------- Article of Faith

Clash with Laws of Physics

The Problem:

- Physics has no respect for our mental simplifications
- "Side effects" (couplings) are everywhere
- And we can't ignore them in some control systems



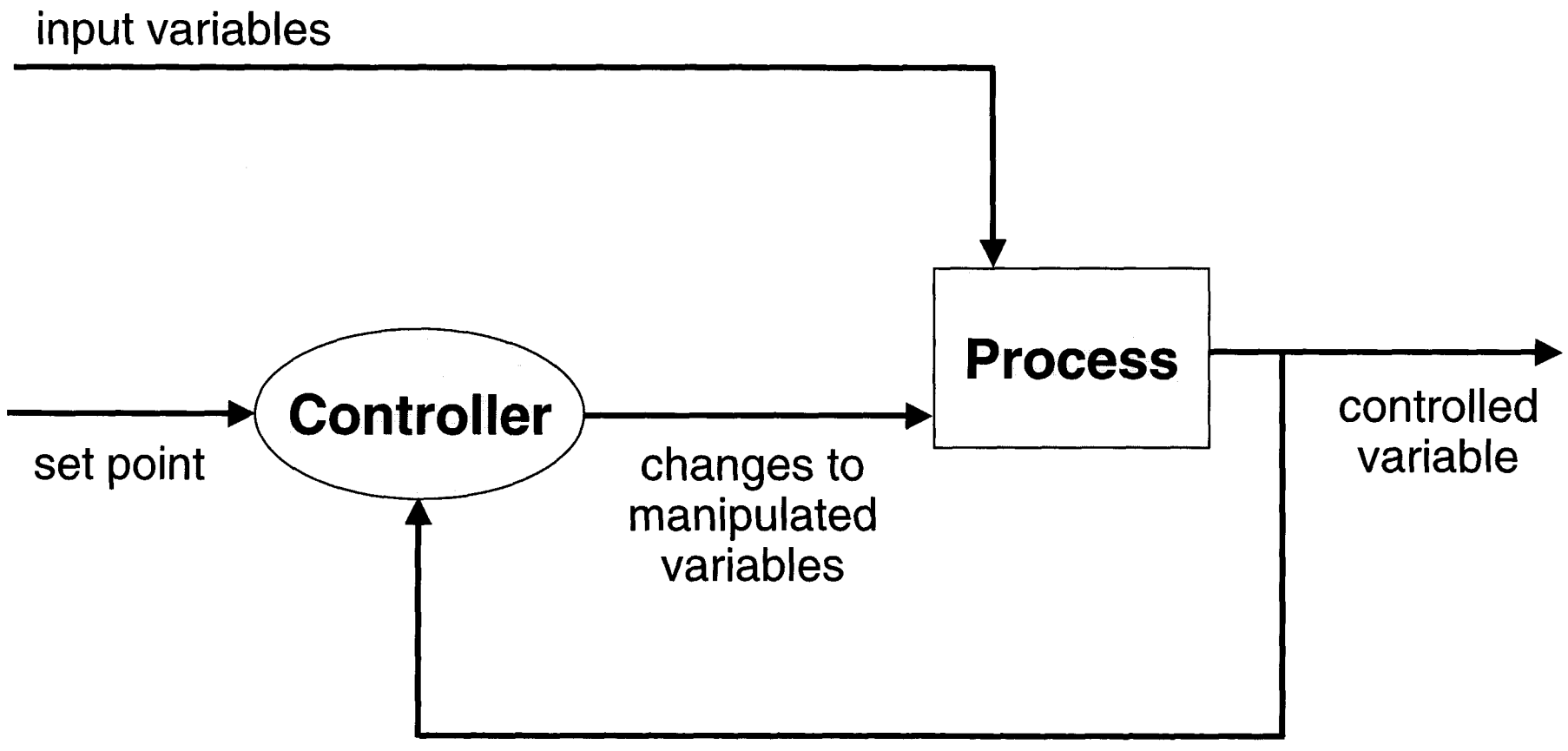
Outline

- Problem domain: control systems
- Home heating system
  - Design by subsystem decomposition
- Deep space systems and the world of “side effects”
- Ordinary versus high-risk control systems
- A better approach: State/Model architecture
- Conclusions

Control System Domain

- **Characteristics**:
  - Interacts with world via imperfect sensors & actuators
  - Designed for continuous operation
  - Real-time closed-loop control
  - Embedded systems, often
- **Examples**:
  - Petroleum refining
  - Pharmaceutical manufacturing
  - Nuclear power plant
  - Spacecraft control

Feedback Control System



* Diagram from "Software Architecture: Perspectives on an Emerging Discipline", Shaw & Garlan, 1996

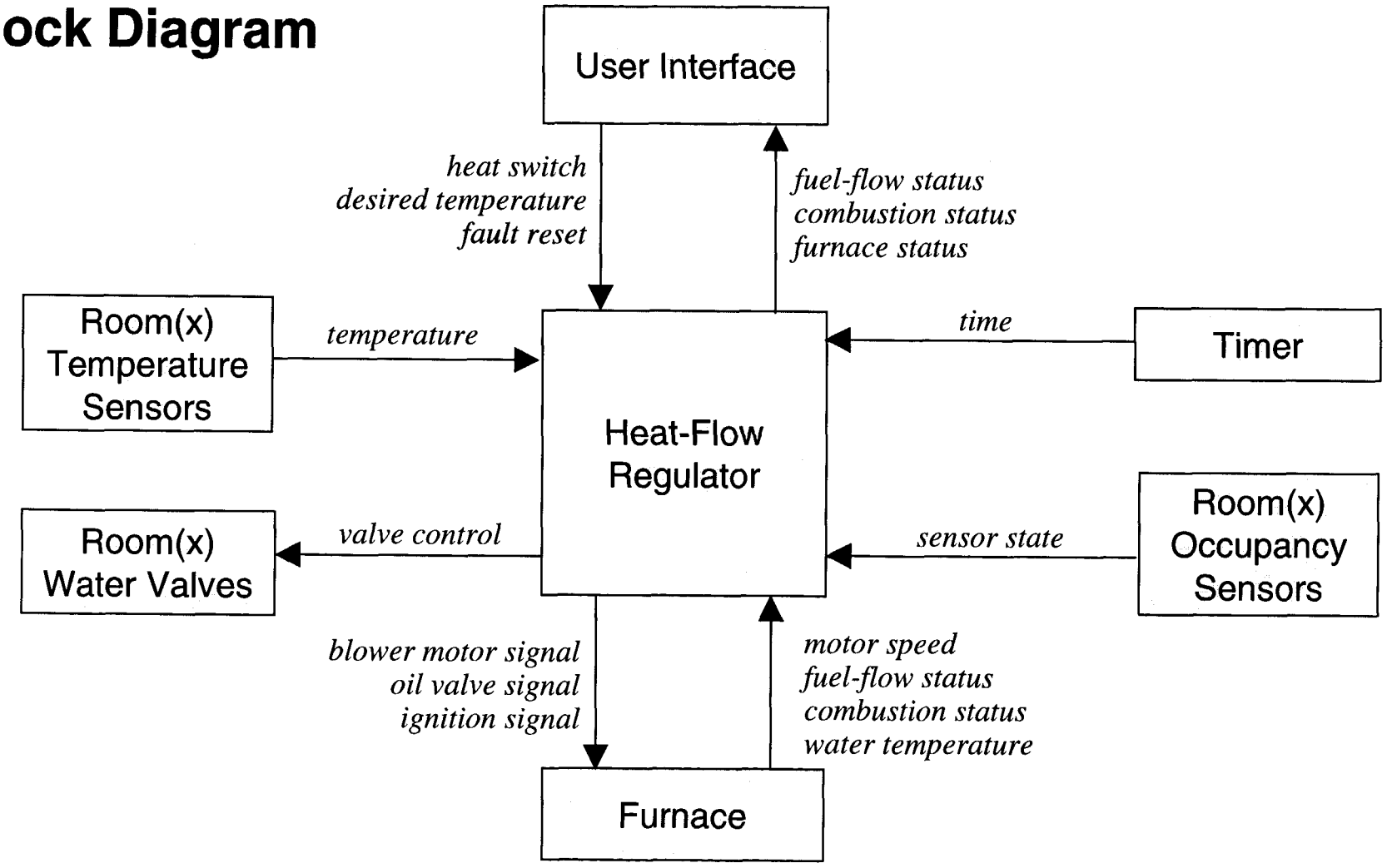
Home Heating System

An example of good design by subsystem decomposition

Example taken from "Smalltalk Home Heating System", Grady Booch, *Object Oriented Design with Applications*, 1991.

Home Heating System

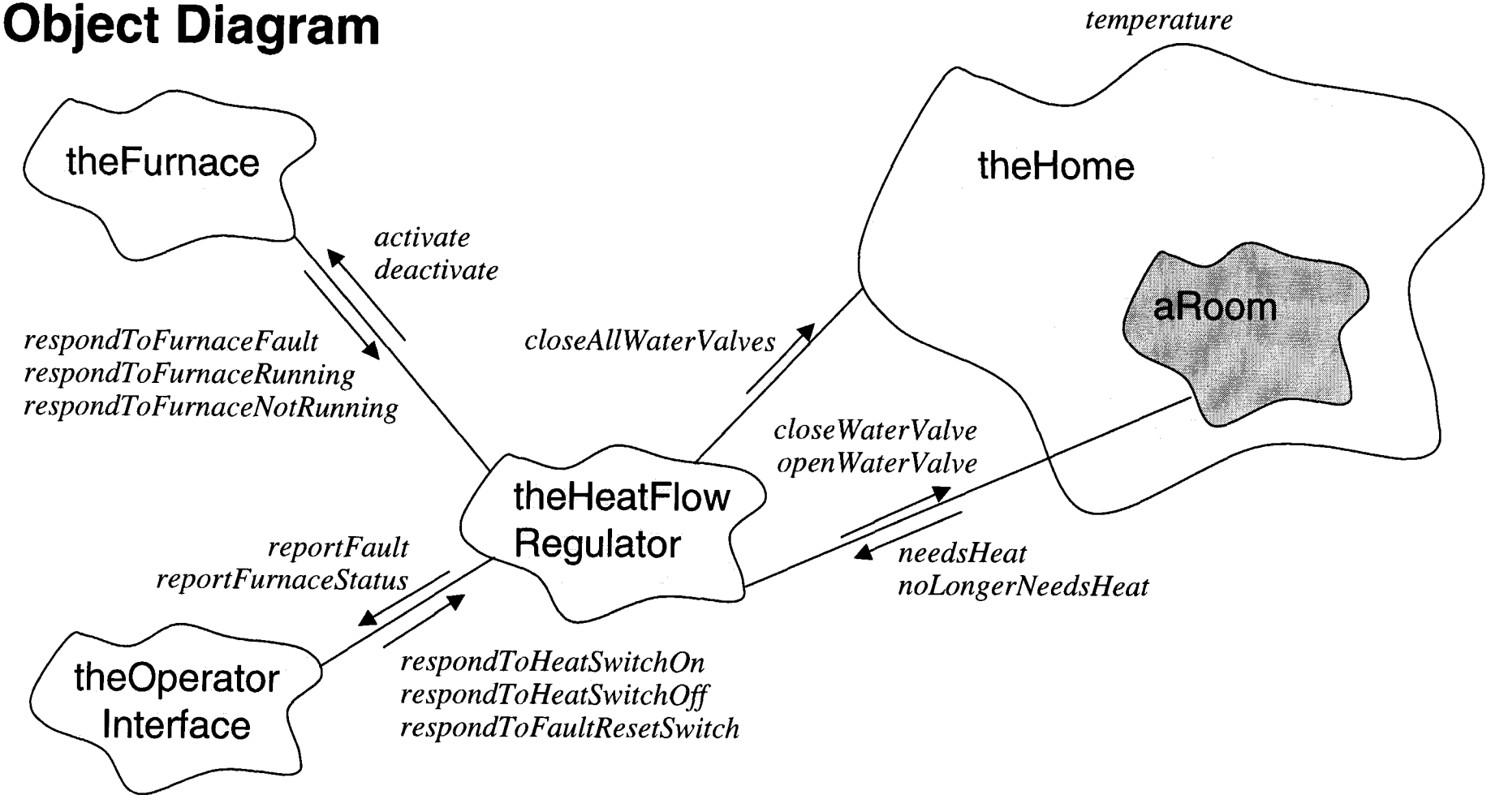
Block Diagram



* Grady Booch, 1991, *Object Oriented Design with Applications.*

Home Heating System

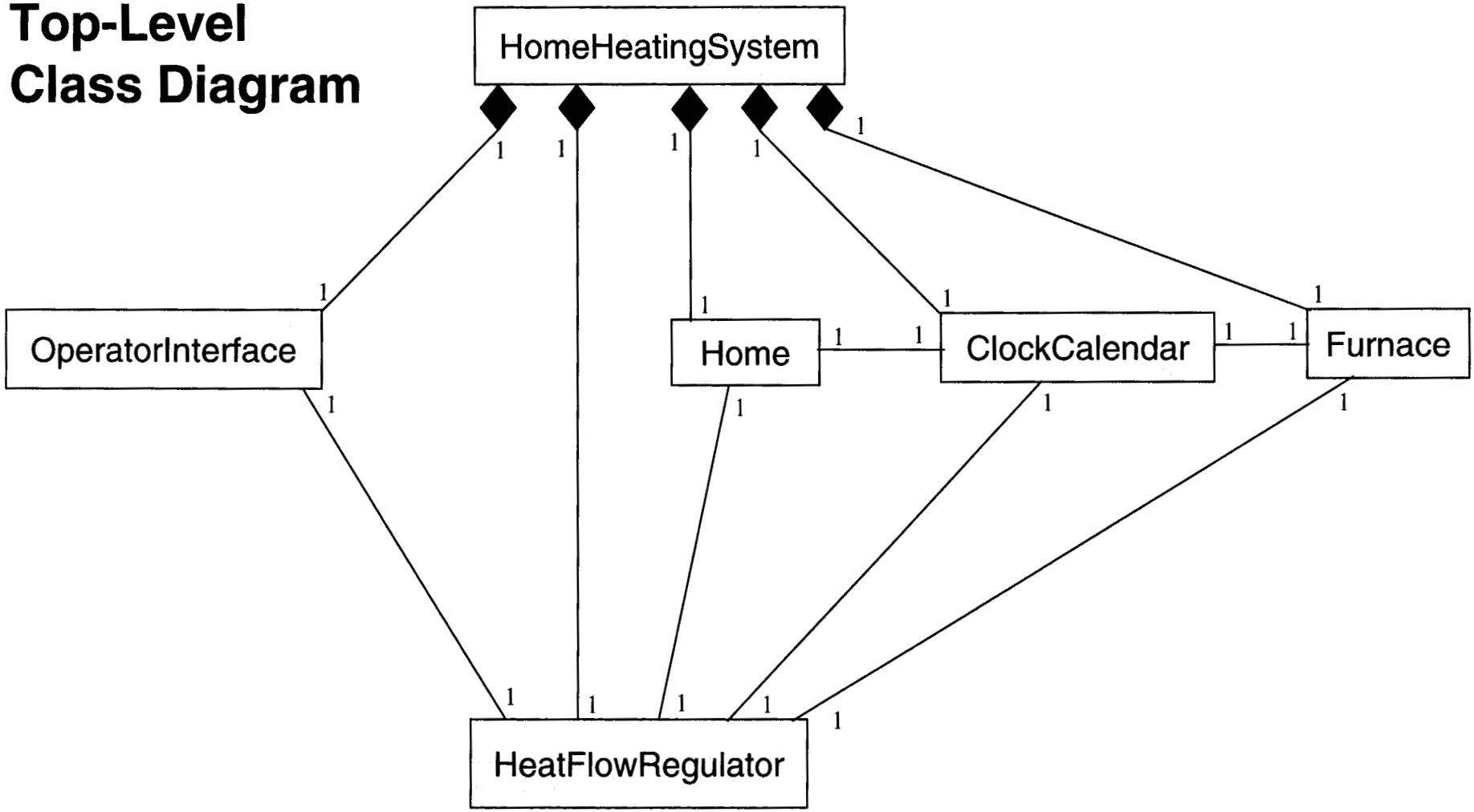
Object Diagram



* Grady Booch, 1991, *Object Oriented Design with Applications.*

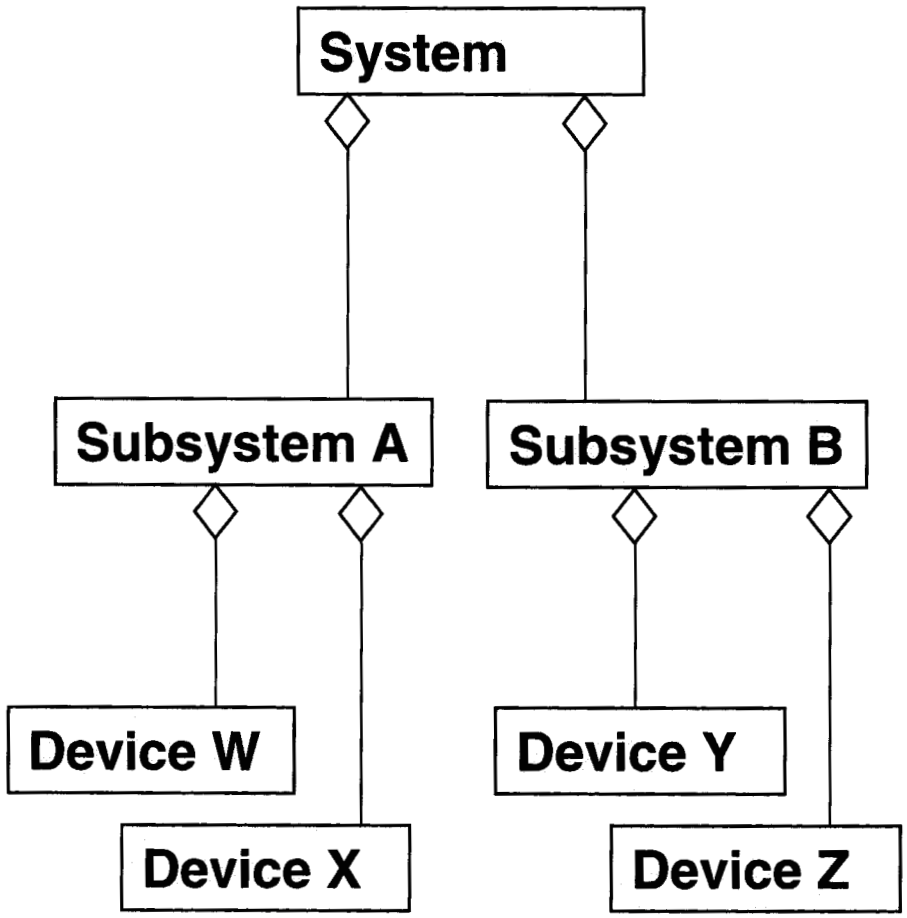
Home Heating System

Top-Level Class Diagram



* Grady Booch, 1991, *Object Oriented Design with Applications.*

Decomposition by Subsystem & Device



- System decomposed into subsystems and devices
- System manages subsystems
- Subsystem manages devices
- Assumes loose coupling
  - Most data flows along aggregation relationships

Benefits / Enticements of Subsystem Approach

Convenient Work Breakdown Structure:

- Subsystems mirror engineering disciplines
- Design mirrors organization chart

Desirable State Encapsulation:

- Each state variable is encapsulated in lowest level object that is responsible for its estimation and control

Reminder

Subsystem-based design is appropriate for loosely coupled subsystems

The World of Side Effects

Turning on a disk drive has the following side effects:

- It reduces available power
- It causes heating
- It causes vibration
- It causes electromagnetic radiation
- It imparts rotational torque
- It stabilizes orientation around axis of rotation

In a server room on Earth, these side effects are negligible.

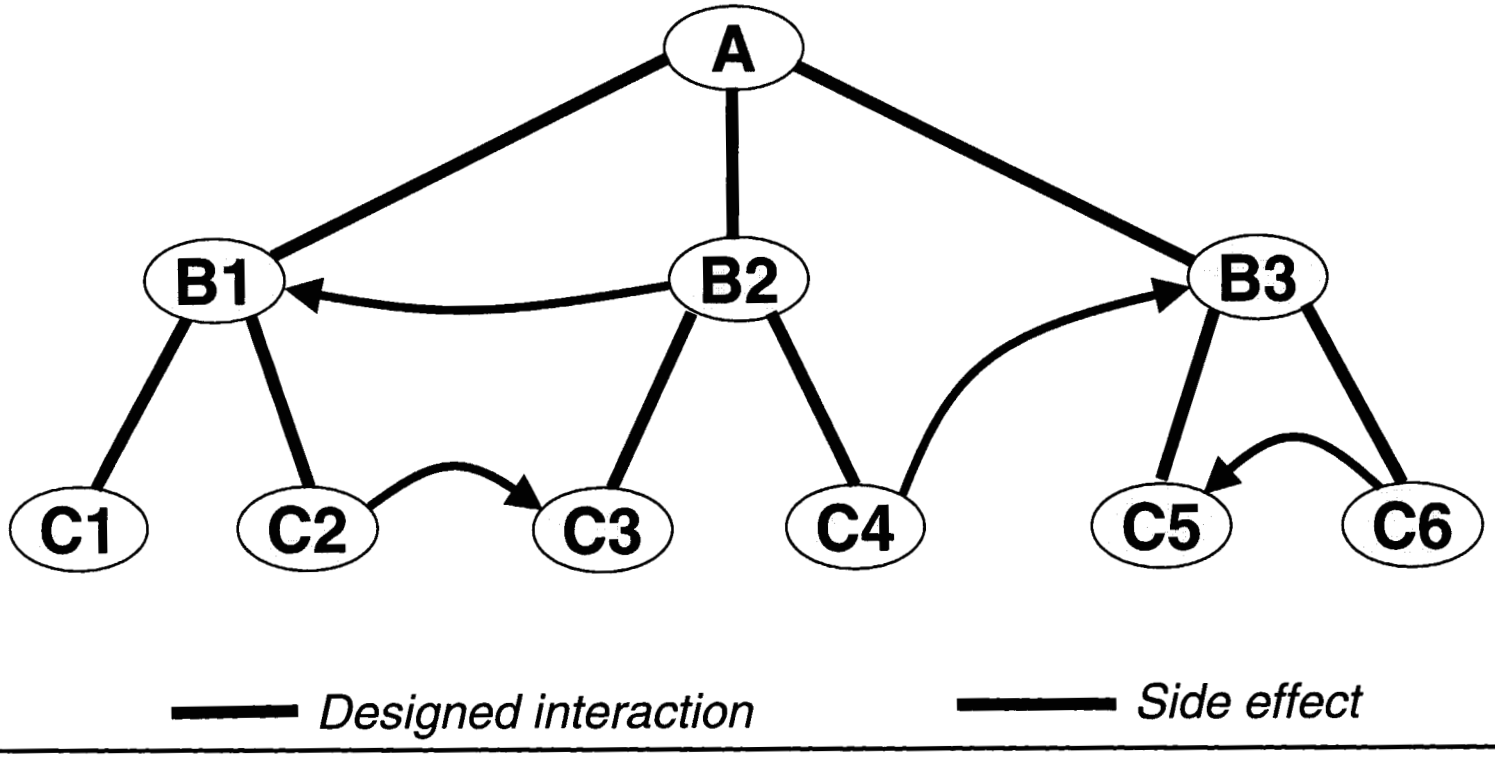
In a spacecraft, every one of these side effects is significant and must be managed!

"Everything affects everything" (a slight exaggeration)

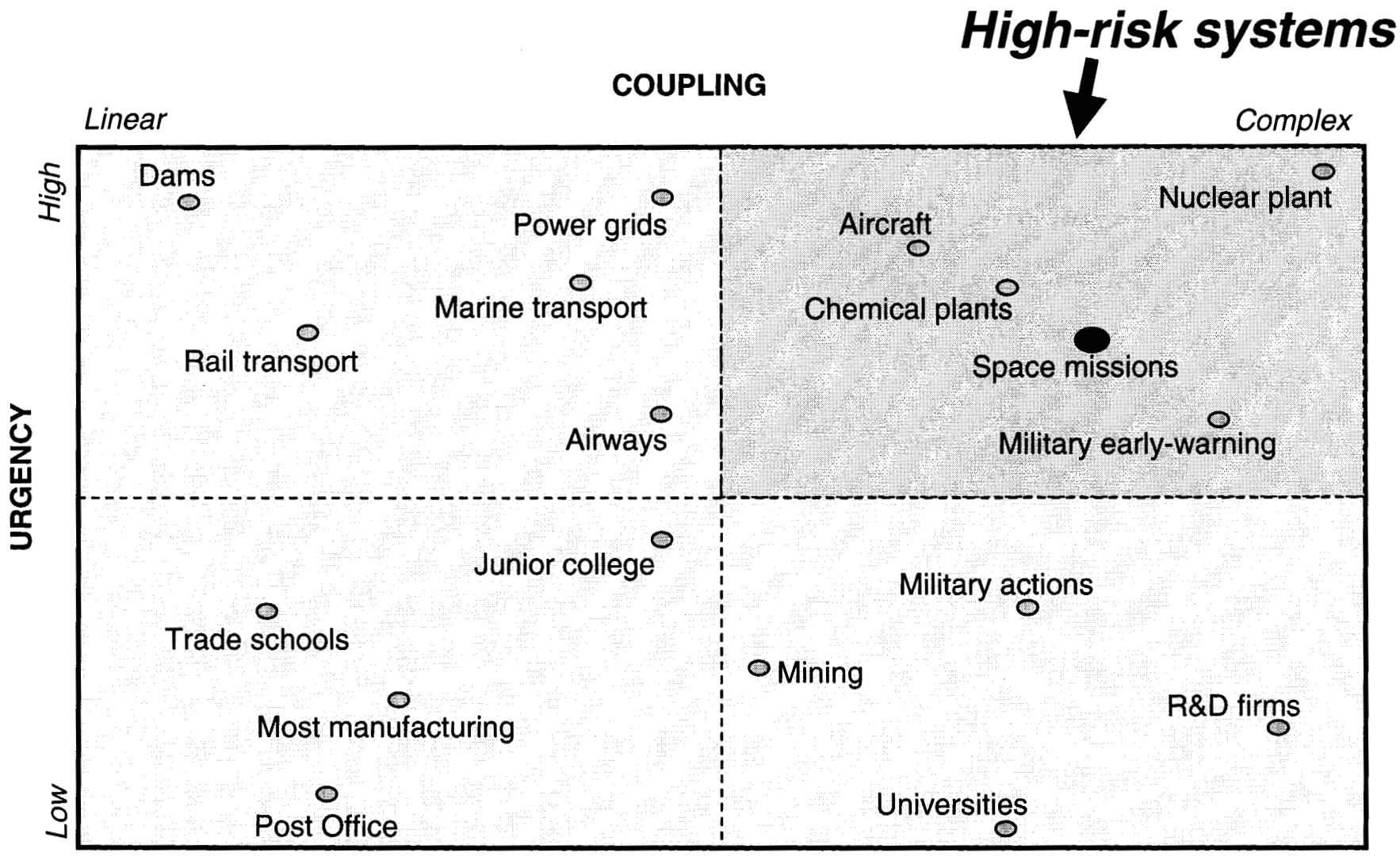
Clash with Laws of Physics

The Problem:

- Physics has no respect for our mental simplifications
- "Side effects" (couplings) are everywhere
- And we can't ignore them in some control systems



Risk due to Complex Coupling and High Urgency



Adapted from Charles Perrow, "Normal Accidents: Living with High-Risk Technologies", 1984.

Coupling in Space Systems

Complex couplings arise from physics and design

- Amount of mass launched determines a ***big*** mission cost
- Therefore, minimize size of batteries, size of solar panels, amount of memory, articulation mechanisms, shielding, smaller antenna, low-power transmitter, etc
- That means:
  - Slower CPU and busses and less memory
  - Can't drive and transmit concurrently
  - Can't run heaters while firing thrusters
  - Can't independently point camera and antenna
  - Lower signal-to-noise ratio, so lower data communication rates, so science downlink is limited
  - Must hold reserve power for surviving the night

Revisiting Thermal Control

- Ideal in Home Heating System:
  - Thermal subsystem *owns* temperature state
  - Has *complete* responsibility for estimating and controlling it
- Reality on a Mars Rover:
  - Competes for power with driving, science, telecom, etc
  - Is affected by heating from nearby powered-on devices
  - Is affected by position of Sun relative to rover
  - May produce electromagnetic interference that precludes use of certain science instruments
  - If temperature sensor fails, must rely on thermal model
  - If heater fails, must turn on nearby devices for heating effect
- **Observation**: The very concept of self-contained thermal control falls apart because it rests on an assumption of loose coupling

Addressing the Couplings

A State/Model Architecture

Managing Interactions

- Interactions often cross subsystem boundaries
- Managing interactions is key to good design
- Need to elevate interactions to architectural level
- Need to describe how one thing affects another
  - The variables are *states*
  - The equations are *models*

State/Model Architecture

Estimators interpret measurement and command evidence to estimate state

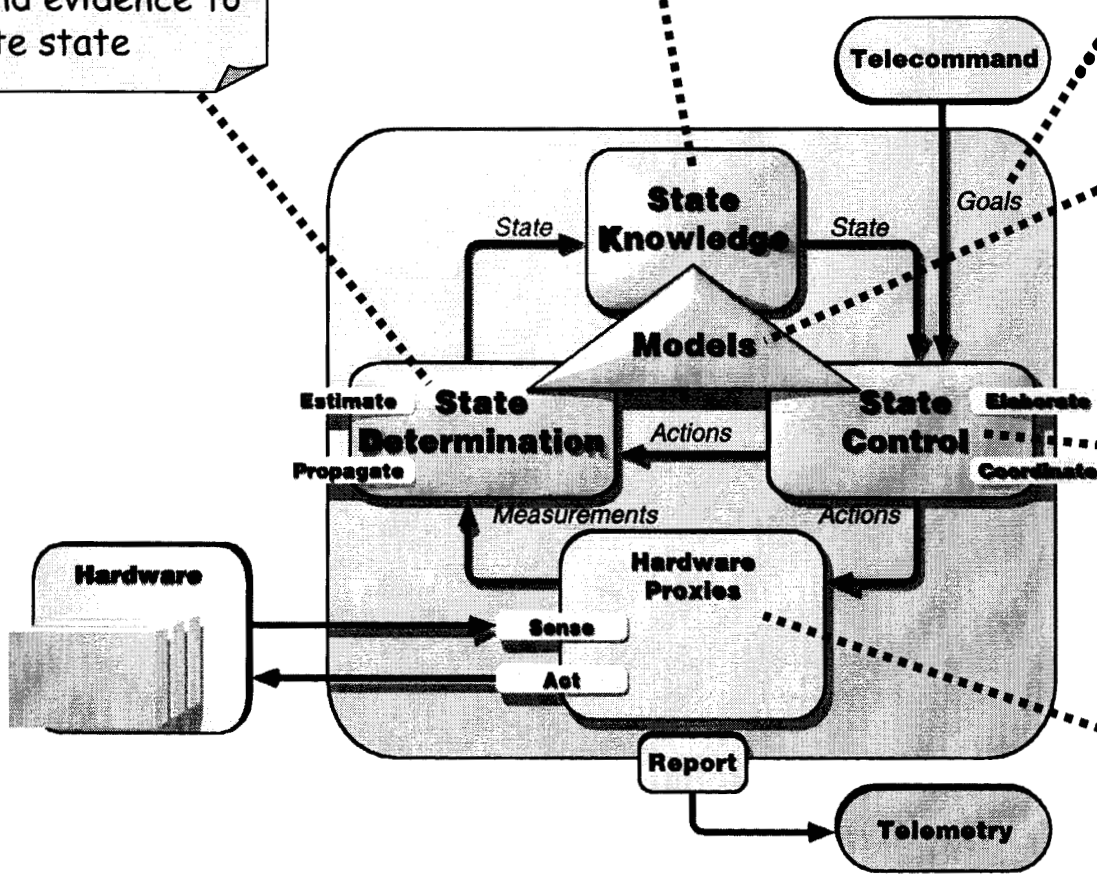
State variables hold state values, including degree of uncertainty

A goal is a constraint on the value of a state variable over a time interval

Models express mission-specific relations among states, commands, and measurements

Controllers issue commands, striving to achieve goals

Hardware proxies provide access to hardware busses, devices, instruments



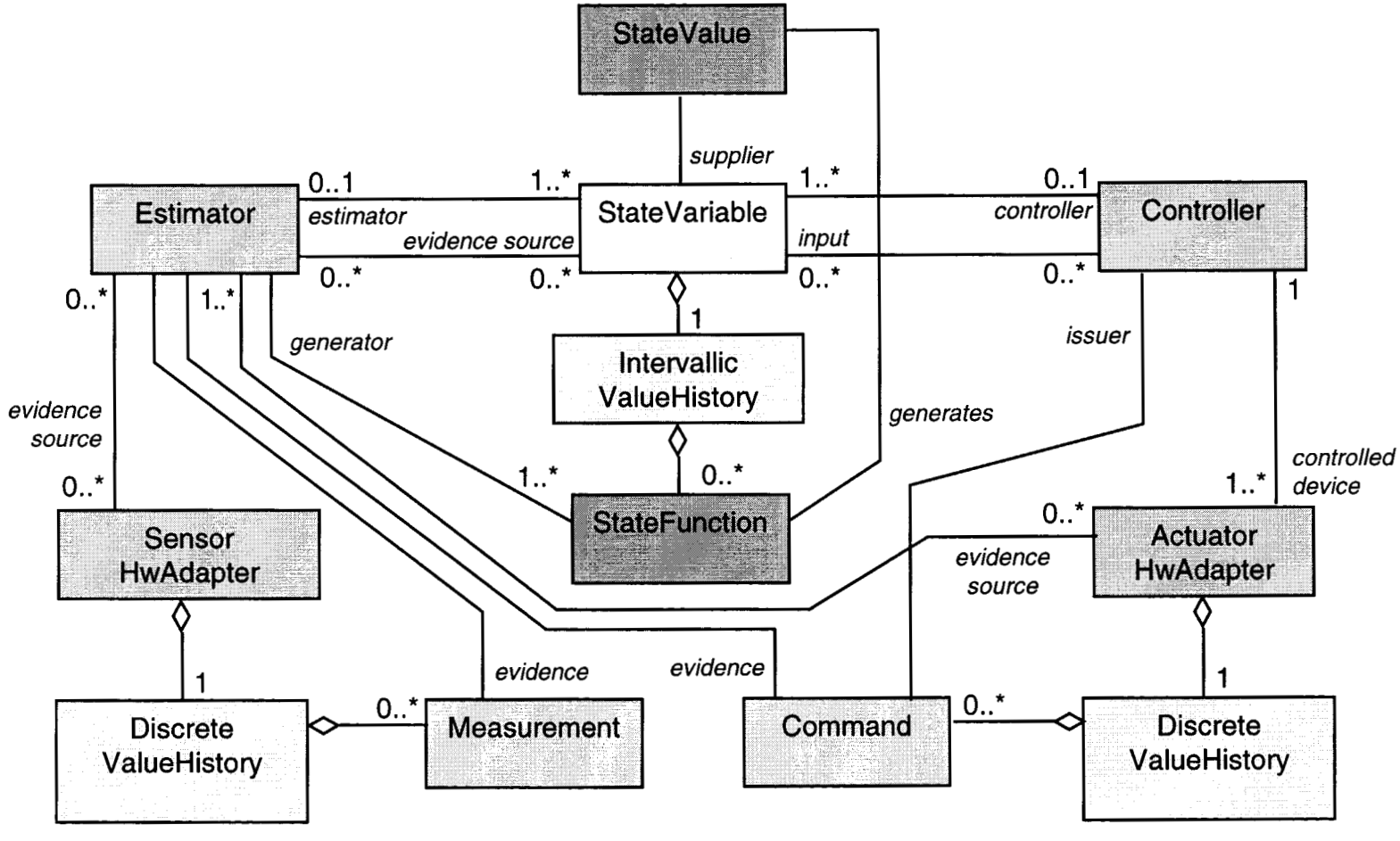
Example States

- Dynamics
  - Vehicle position & attitude, gimbal angles, wheel rotation, …
- Environment
  - Ephemeris, light level, atmospheric profiles, terrain, …
- Device status
  - Configuration, temperature, operating modes, failure modes, …
- Parameters
  - Mass properties, scale factors, biases, alignments, noise levels, …
- Resources
  - Power & energy, propellant, data storage, bandwidth, …

Example Models

- Relationships among states
  - *Power varies with solar incidence angle, temperature, & occultation*
- Relationships between measurement values and states
  - *Temperature data depends on temperature, but also on calibration parameters and transducer health*
- Relationships between command values and states
  - *It can take up to half a second from commanding a switch to full on*
- Sequential state machines
  - *Some sequences of valve operations are okay; others are not*
- Dynamical state models
  - *Accelerating to a turn rate takes time*

Architectural Relationships



The color coding conveys similarities, e.g., estimators and controllers are goal achievers, sensors and actuators are devices, measurements and commands are time-tagged items.

Concluding Observations

Paradigm Shift

- "Side effects" *always* exist, but can be ignored in systems that have plentiful resources
- Resource-limited systems have no choice but to acknowledge and manage such interactions
- Trying to do this in a conventional subsystem decomposition leads to an appealing fiction
  - a tangle of special-case subsystem-to-subsystem couplings behind a façade of modular decomposition
- Solution requires a paradigm shift in architecture
  - State/model structure vs. subsystem/device structure

The Road to Encapsulation is Paved with Good Intentions

Object-Orient Analysis

Anthropomorphic Story-Telling

This is where it went astray

Ownership — — — — — — — — — — — — Appealing Metaphor

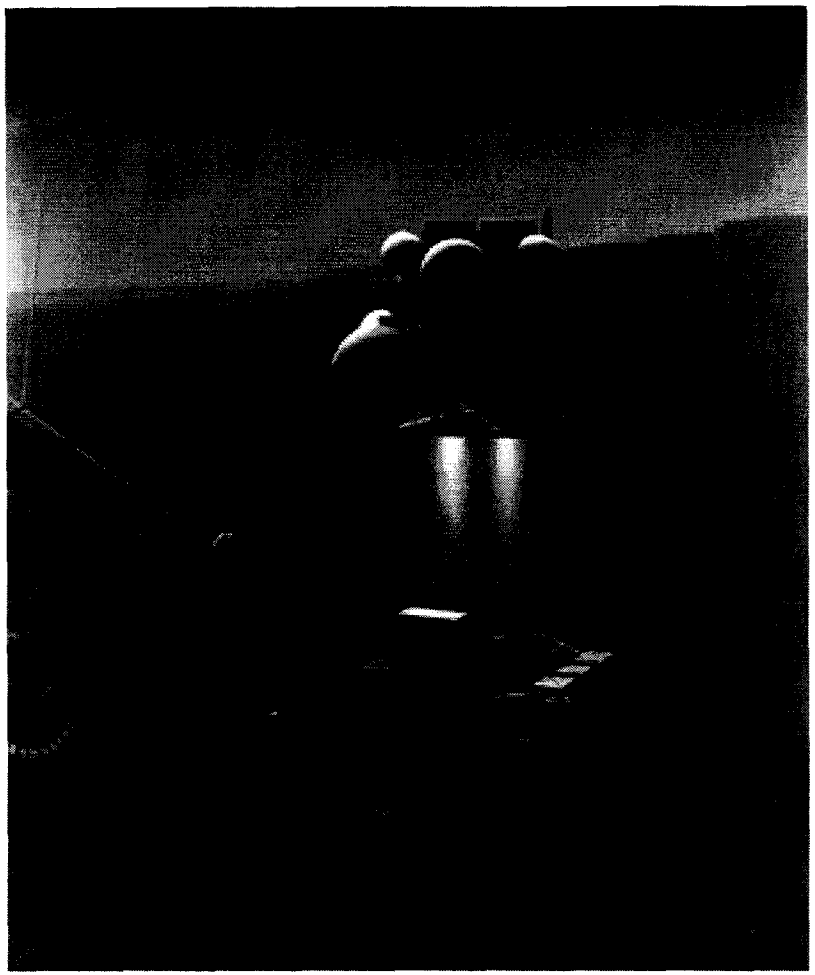
Subsystem/Device Architecture ---------- Appealing Pattern Reuse *and* Work Breakdown

State Encapsulation ---------- Article of Faith

Summary

- State encapsulation is not a universal GOOD THING
- Metaphors like “ownership” are seductive
  - … and sometimes clash with the laws of physics
- Managing interactions is the key to good design
  - Think about the interactions that complicate your domain
  - Elevate those interactions into your architecture

Questions?



Artist's conception:
A Mars sample-return mission blasting off from Mars

Reserve Slides

Coupling and Urgency

"***Linear couplings*** are those in expected and familiar production or maintenance sequence, and those that are quite visible even if unplanned."

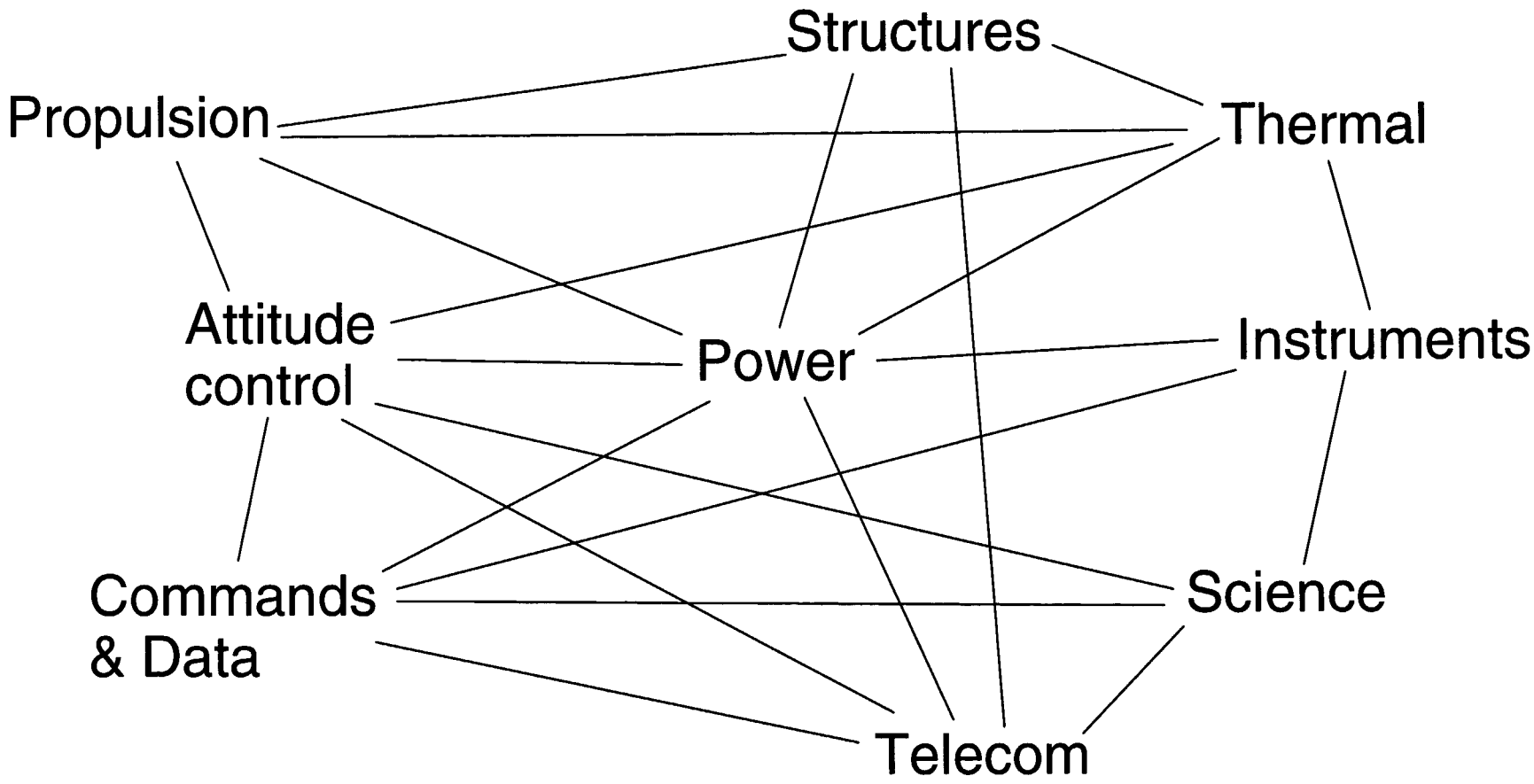
"***Complex couplings*** are those of unfamiliar sequences, or unplanned and unexpected sequences, and either not visible or not immediately comprehensible."

"***Low urgency*** systems can incorporate shocks and failures and pressures for change without destabilization. ... Loosely coupled systems tend to have ambiguous or perhaps flexible performance standards..."

"***High urgency*** systems have more time-dependent processes: they cannot wait or stand by until attended to. ... Reactions, as in chemical plants, are almost instantaneous and cannot be delayed or extended."

Adapted from *Normal Accidents: Living with High-Risk Technologies*, Charles Perrow, 1984.

Coupling in Space Systems



** Some domains of concurrent design in JPL's Project Design Center*