4. Processes

Overview

- 4.1 Fundamentals
- 4.2 Elicitation
- 4.3 Specification
- 4.4 Verification
- 4.5 Validation

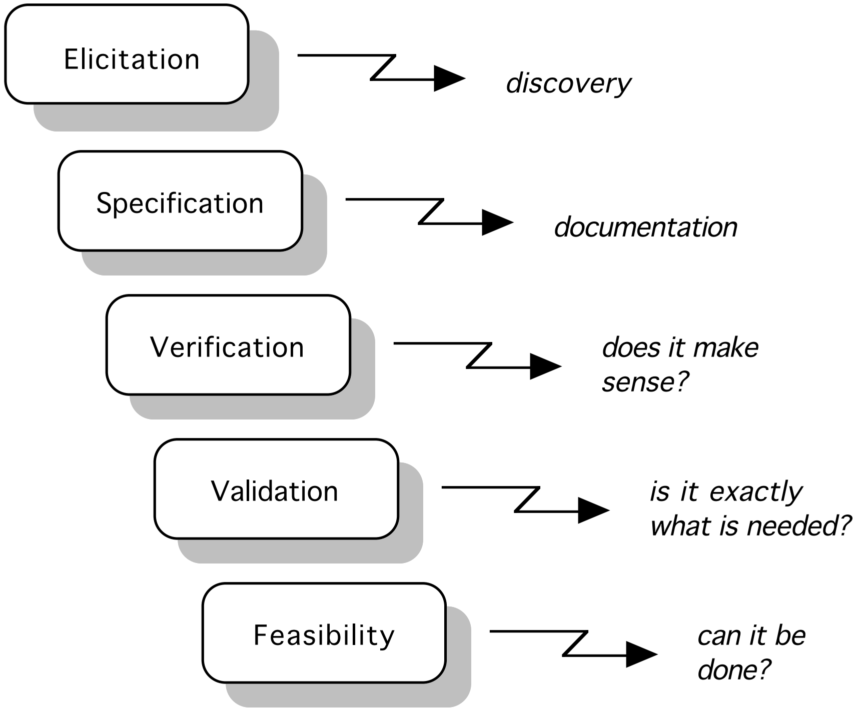
4.1 Fundamentals

- The fundamental goals of the requirements definition phase are
  - to understand the nature of the problem
  - to establish a baseline for the software development process
  - to facilitate communication among participants in the development effort

Observations

- Problem understanding is a prerequisite to starting any software development
- The establishment of a baseline involves
  - formal recording of the requirements (documentation)
  - analyzing them (feasibility)
  - accepting them as the basis for planning and development
- Requirements definition is a communication-intensive phase whose goal is not only to extract information but to lay a firm foundation for communication
  - between customers and developers
  - among various groups of developers

Activities



Elicitation → *discovery*

Specification → *documentation*

Verification → *does it make sense?*

Validation → *is it exactly what is needed?*

Feasibility → *can it be done?*

Controls

- Since the requirements are a baseline for the project, any changes can have major implications
- Requirements documents must be placed under configuration control at the end of the requirements definition phase
- Any requirements change must be evaluated with respect to its implications on
  - software design
  - development plans
  - customer/user
- Requirement changes must be subject to formal approval and notification—a center of responsibility is required
- Project measurement must monitor the type, source, gravity, and detection point for requirements errors and changes

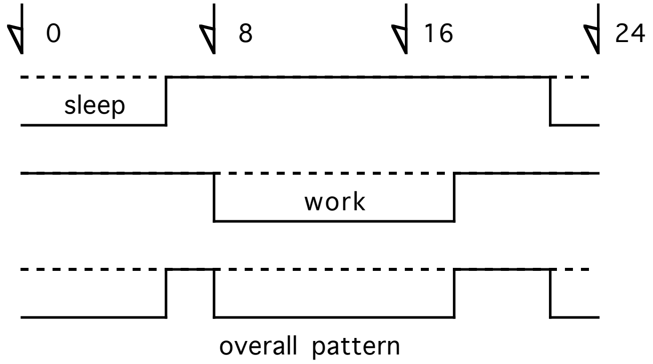
Case Study: Thermostat

Elicitation

- Develop a thermostat controller for a heating system.
- Provide an energy saver feature designed to reduce the temperature setting by a fixed amount while the residents are at work and during the night.

Specification

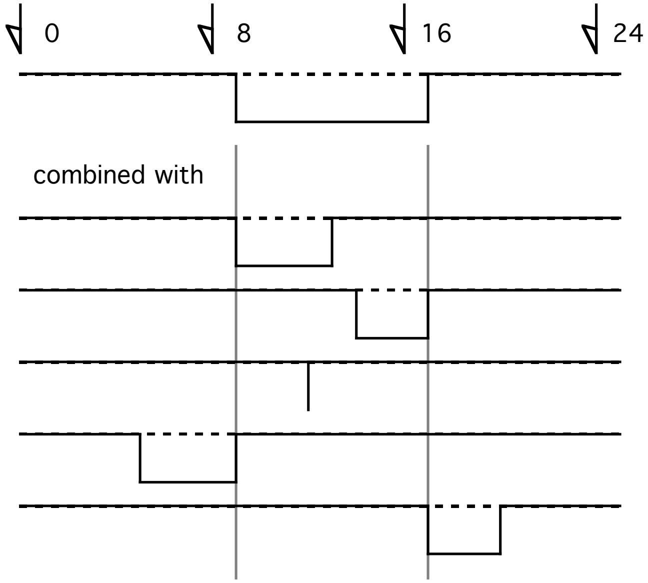
- Use a 24 hour profile diagram to capture the desired meaning for the control logic.
- View the falling and rising edges as events (offset on and off)



Thermostat

Verification

- Evaluate against special cases where points on the diagram overlap.



Validation

- Evaluate against standard behavior patterns. Consider vacations (24 hour offset), weekends (override), etc.

Feasibility

- Check that all sensor and actuator controls are actually available.

4.2 Elicitation

- Discover and catalogue application needs
- Identify constraints
- Identify and prioritize objectives
- Reconcile conflicting views
- Define standard terminology
- Separate concerns
- Organize the information
- Pave the way to conceptualization
- Make technical specifications feasible

Issues

- Multiplicity of sources
- Conflicting interests
- Hidden objectives
- Unclear priorities
- Limited understanding of technology
- Communication difficulties
- Limited understanding of the application

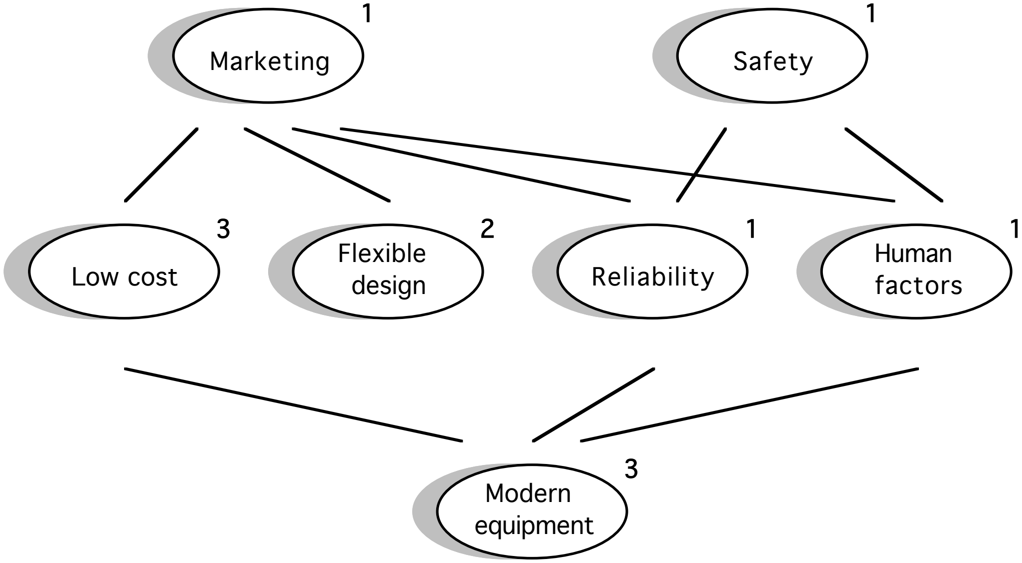
Mechanics

- Systematic techniques can overcome the apparently ad-hoc nature of the process
- A simple five-step method
  - collect information
  - formulate working hypotheses
  - define terms
  - validate hypotheses and terms
  - separate concerns

Clarifying Objectives

- Systematic acquisition of information must be accompanied by deeper understanding
- The relation among competing objectives is critical in carrying out technical trade-offs

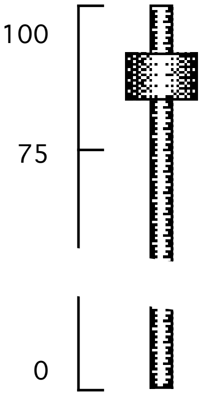
- Illustration: rail traffic control system

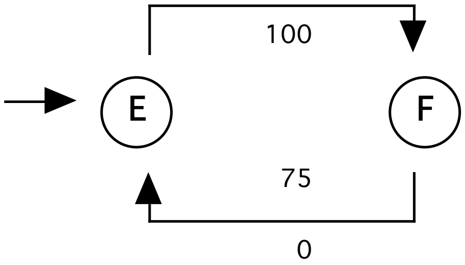


Seeking Simplicity

- The development of simple conceptual models helps clarify basic functional relationships
- Models also prepare the transition for the specification

- Illustration: a tank refilling procedure





Technicalities

- While application-specific strategies to requirements elicitation may be attractive, the software is ultimately built by software engineers
- Specification strategies undoubtedly lead away from the application to the computing domain
- It is helpful to prepare early for this transition by attempting to
  - identify the boundary between the system and its environment
  - separate constraints from functional requirements

Principal Product

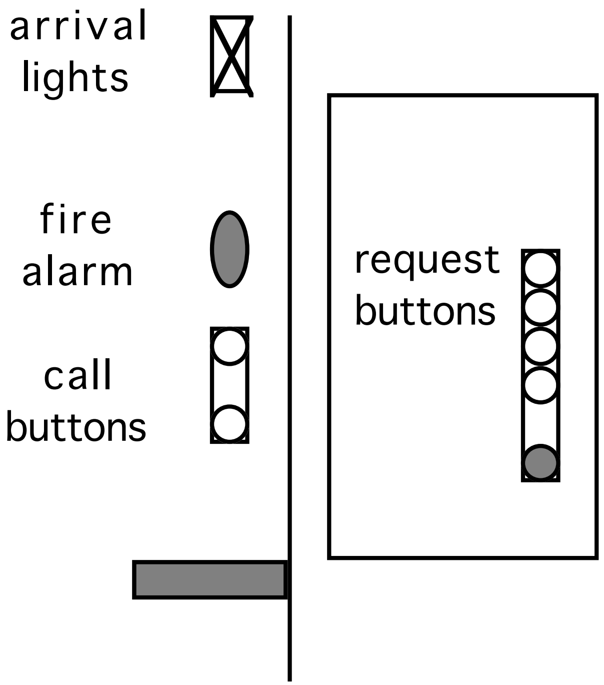
- Requirements Definition Document (RDD)
  - is relatively high level
  - does not provide yet a baseline for the development (due to incompleteness)
  - does provide the basis for specification
  - is the starting point for a number of specialized preliminary studies
- The document must be accessible to a broad range of readers
  - customers, users, managers, designers

By-products

- Feasibility study
- Cost analysis
- Planning
- Market analysis
- Component selection and evaluation
- Technology evaluation
- Human factors studies

Case Study: Elevator

- Consider the development of an elevator control system for a 10-story residential building.

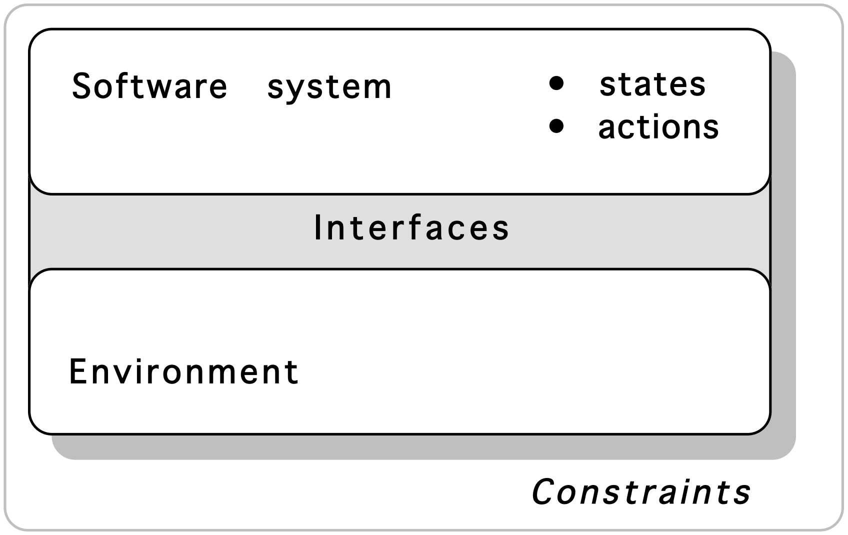


4.3 Specification

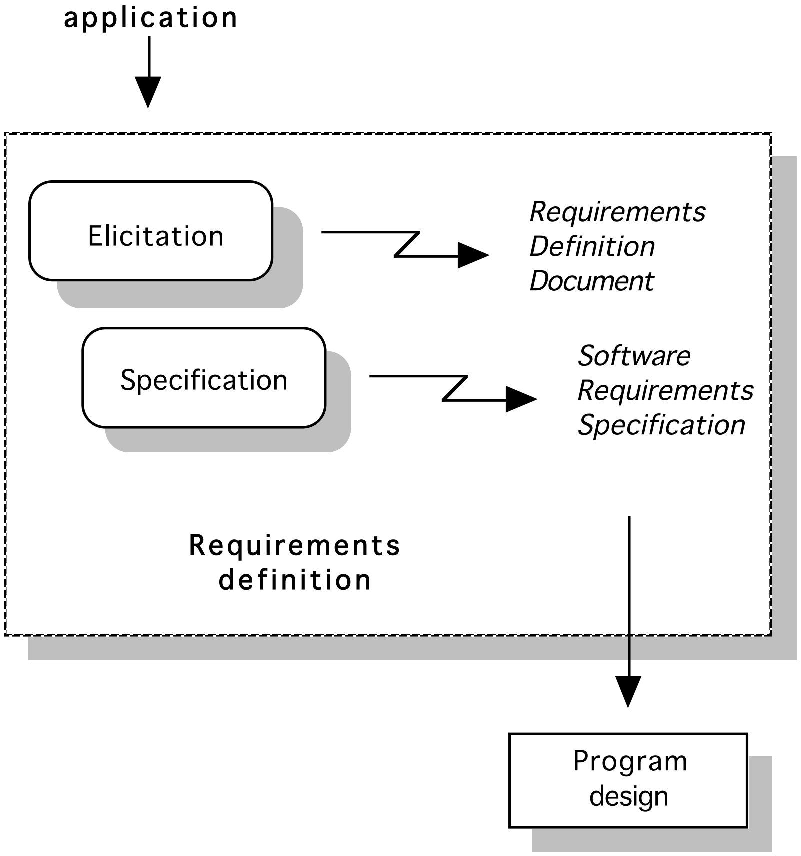
- The Software Requirements Specification (SRS) is a description of the functionality and constraints that must be delivered by the software
  - precise
  - detailed
  - technical
- The SRS becomes the baseline for the entire software development process
- The boundary between the system and its environment must be known at this time
- The SRS assumes that the system functions have been allocated over the architecture

Technical Contents

- The proper contents of the SRS is determined by fundamental technical considerations having to do with how we view computing
- The specific form of an SRS reflects the specific computational model underlying the specification methodology being employed

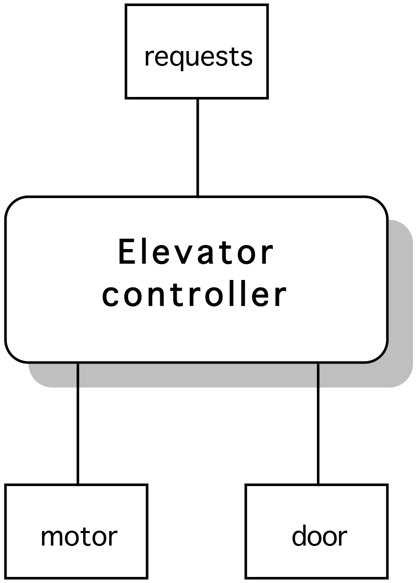


Specification after Elicitation

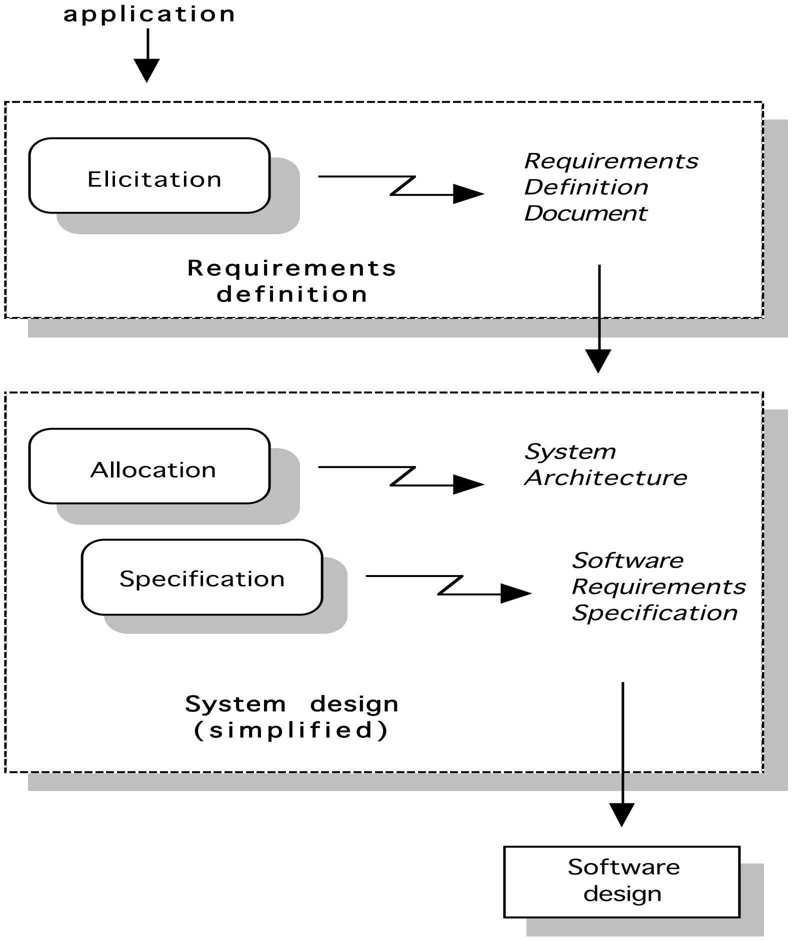


Centralized Controller

- All external interfaces have been identified
- The specification does not rule out a distributed implementation

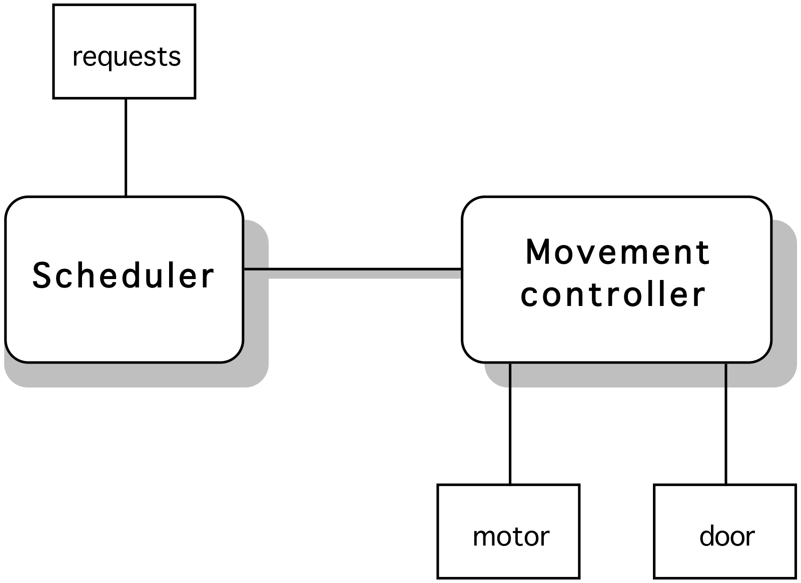


Specification after Allocation



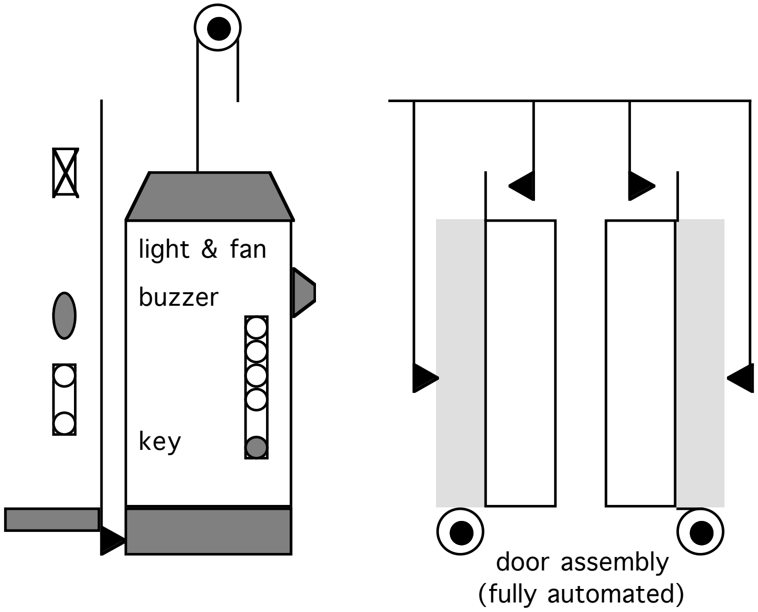
Distributed Controller

- Additional internal interfaces have been identified
- The specification rules out a centralized implementation



Case Study: Elevator

- The full technical specification cannot start until all interfaces are well defined.

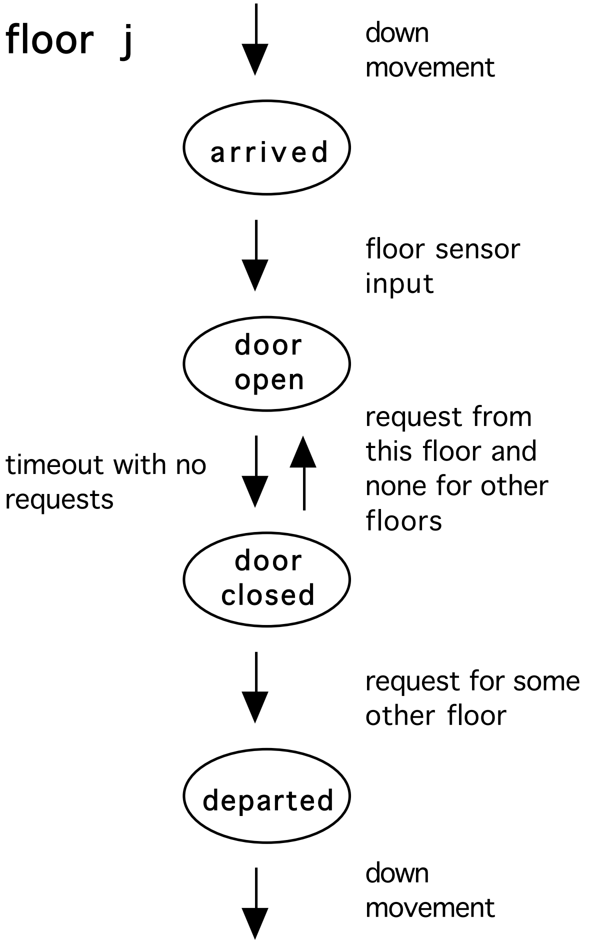


4.4 Verification

- Requirements verification is an activity directed towards the discovery of specification errors
- The ultimate goal is to ensure that the specification (when considered on its own) is
  - correct
  - consistent
  - complete
- The verification must be carried out against a model (formal or informal)
- Formal and semi-formal specifications can be checked out by tools

Door Control Logic Analysis

- Consider a deterministic finite state representation of the elevator movement logic
- Some errors can be detected simply by the very nature of the model, e.g., initial state, missing transitions, non-deterministic transitions, possible livelock, etc.



4.5 Validation

- Requirements validation is concerned with establishing that the specified requirements do represent the needs of the customer or user
- Since the needs are not reflected by any model or document, the validation cannot be performed in a mechanical way
- Good communication is the key to a successful validation
  - well-defined terminology
  - well-written and simple specifications
  - formal reviews
  - rapid prototypes
  - simulations

Case Study: Elevator

- Consider an elevator movement policy which
  - takes the elevator up and down, from top to bottom, and services requests as it goes
- The policy satisfies the customer stated requirements
  - every request is eventually serviced
  - there is a defined upper bound on the time it takes for a request to be serviced
- Nevertheless
  - the time it takes to service a request during low demand periods is unacceptable
  - unnecessary energy utilization emerges as a new issue