Requirements, Specifications, and Programs

Notes on "The World and the Machine," Michael Jackson, ICSE 95

Engineering?

Jackson: "We do not speak of engineering mathematical theorems, or works of literature, or business policies*"

So in what sense is (or is not) software development a species of engineering?

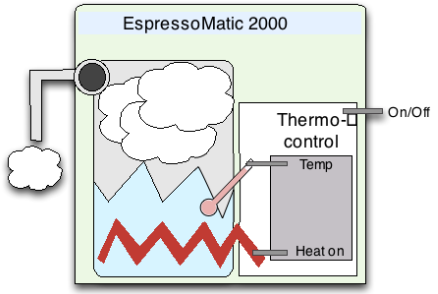
*Actually the term "reengineering" has been applied lately to business processes, though not in 1995 when Jackson wrote this paper.

World and Machine

What does Jackson mean by "the world"?

What does he mean by "the machine"?

Why bother with the distinction? What does he think is broken?



Espresso machine pressure control:

- Maintain 9 +/- 2 bar pressure when on
- Pressure is proportional to temperature

What's "the world" relative to pressure control program? What's "the machine"?

Four Facets of Relationship Between World and Machine

Modeling (system reflects the world)

One of the roots of OO programming & design, starting with Simula 67

Interface

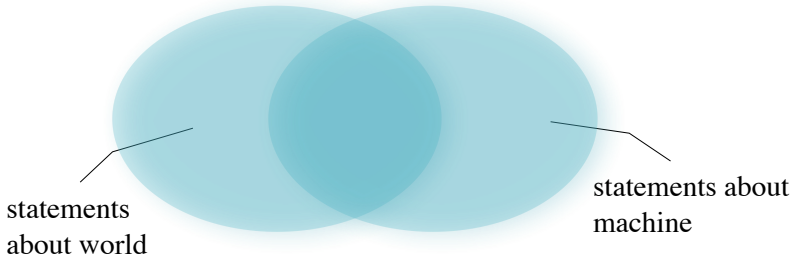
World ∩ Machine

Engineering

Specification ties Requirements to Programs

Problem Structure

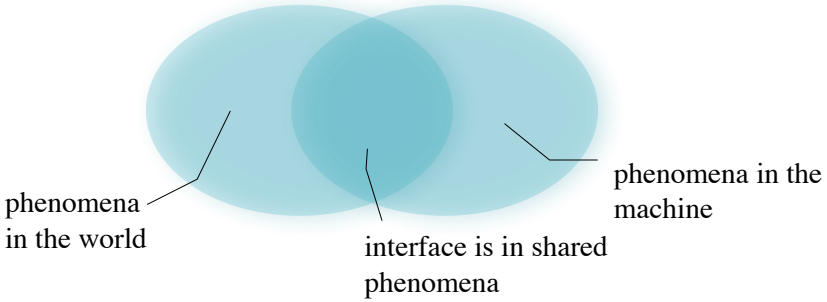
Mapping Model to World



Mapping concerns only part of each, where they overlap. Why?

What does it mean for something be outside the mapped part of the world?

Interface



Is this the same diagram again?

And what is the consequence of only partial overlap?

Specification Koan

"A specification is both a requirement and a program. It is a requirement because it is concerned only with phenomena in the world. It is a program because it is concerned only with phenomena in the machine."

Really? How can that be?

And what is the practical consequence to how we gather and describe requirements?

Problem Structure

"Problems usually exhibit a parallel structure, in which the key connective is 'and'. Nor do they allow homogenous decomposition."

What's he talking about? What does it have to do with gathering requirements?

Requirements Engineering

"Rightly or wrongly, wisely or foolishly, we answered this question [whether problem analysis is part of software development] long ago. We said that we would do it."

So: Rightly, or wrongly? Wisely, or foolishly? And why us, more than engineers of roads and coffee machines?

Not Unique to SE

Auto engineers don't (often) ask "what is the purpose of a car"

But auto designers do often ask "what is the purpose of *this* car? Who will drive it?"

Designers at Steelcase and Herman Miller do design work patterns, of which desks and chairs are just parts.

Precedented and Unprecedented

Precedented: Another system of type X, customized and modestly extended

A minivan; a compiler; a database-backed web information system

Unprecedented: In the problem domain (world) or in the solution domain (system)

Domain Descriptions

"Traditionally, I am claiming, we pay too little attention to the world in which our problems are found. In software development, paying attention to a subject must mean describing it carefully and precisely."

Known as domain analysis

Neat Models of Messy Problems

Others have said: All models are wrong, but some models are useful

- We need models that are useful and also neat, well-structured, comprehensible

But Jackson says: The world is not neat

- Is he right? And if so, how can we have a manageable problem description and specification?

Abstraction in modeling

Consider a *model* of an airplane

- For analyzing lift and strength of the wings, during aircraft design
- For analyzing time required to load passengers, during aircraft design
- For designing a new wing of an airport

The airplane is the same; the models will differ

Is Formalism the Answer?

"There is no sense in being precise when you don't know what you are talking about." (von Neumann, via Jackson)

- Which doesn't mean there is no use in being precise. It does mean that we must know what we are talking about.
- "We must begin by establishing the vocabulary of ground terms"
  - Often missed, or messed, in OOD

Some Practical Approaches

Domain analysis and modeling; domain-specific notations

Very useful when many systems are built in a single problem domain, over a period of time. Often leads to additional automation.

Prototype, prototype, prototype

Learning about the world by building machines

Distinguishing problem description (requirements) from solution description (specification)

Organizing the work

Sometimes "analyst" is a distinct job

- But less often in modern "agile" development; problem analysis is intertwined with solution development

Sometimes "systems engineering" is distinct from "software engineering"

- But mostly in embedded systems and mega-projects; and even then the roles are fluid

Often problem analysis is incremental

- We start from a conventional problem analysis (e.g., "interactive map", "first person shooter")

Summary: World & Machine

We want to build "useful machines"

By programming a "universal machine"

We (mostly) know how to build *correct* machines

"Correct" = "consistent with specification"

What we learn in computer science courses, mostly

We also need to *design the specification*

And that involves building a clear, useful model of the world in which the machine is situated