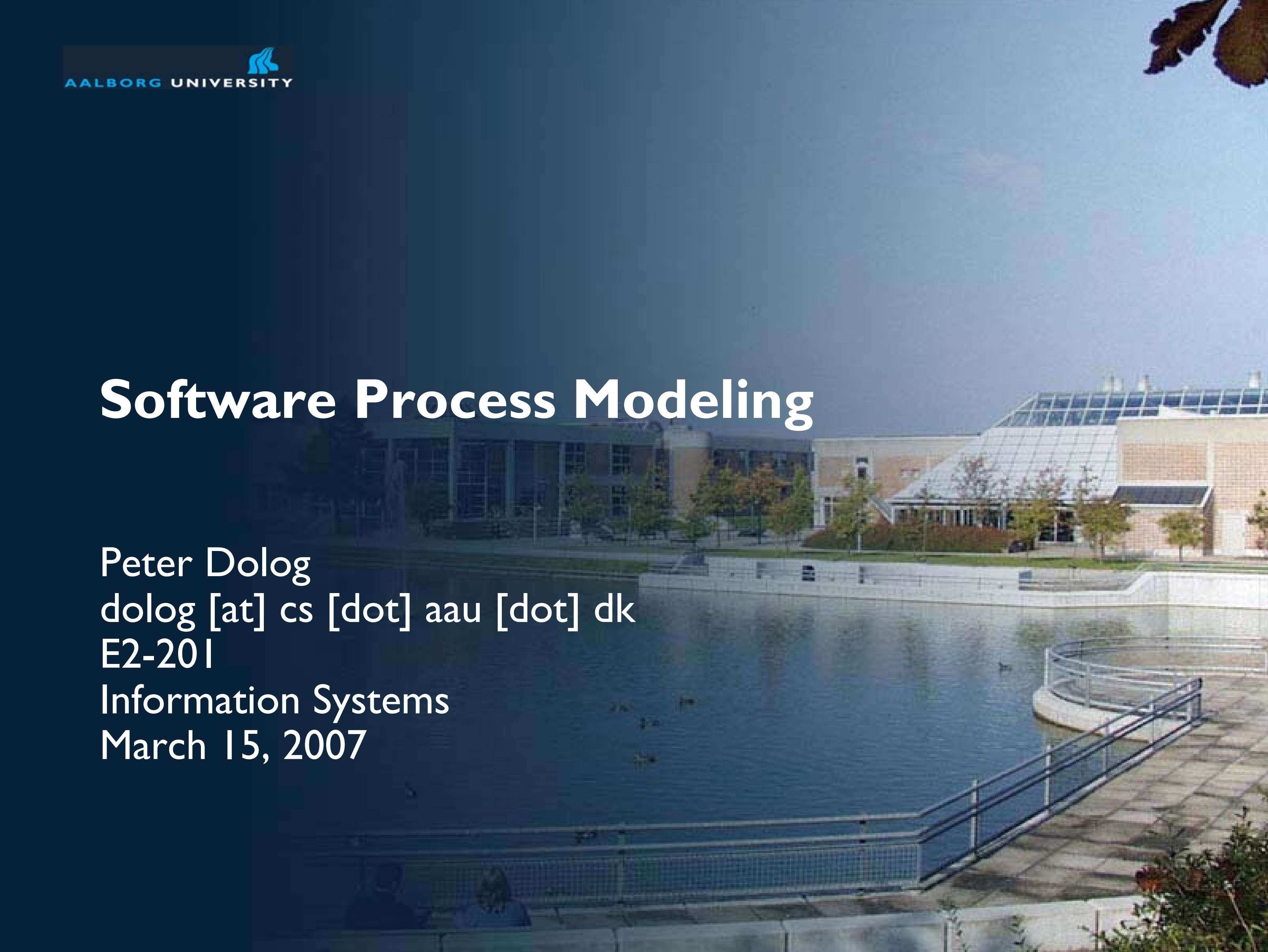
AALBORG UNIVERSITY



Software Process Modeling

Peter Dolog
dolog [at] cs [dot] aau [dot] dk
E2-201
Information Systems
March 15, 2007

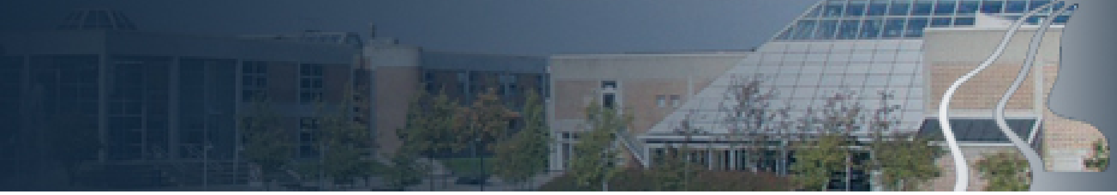
Process programming

Our suggestion is that we describe software processes by "programming" them much as we "program" computer applications. We refer to the activity of expressing software process descriptions with the aid of programming techniques as process programming, and suggest that this activity ought to be at the center of what software engineering is all about.

Osterweil, L. "Software Processes are Software Too," Proceedings of the Proceedings of the Ninth International Conference on Software Engineering, Washington, DC, 1987, pp. 2-13.

Osterweil

http://sunset.usc.edu/events/2001/dod/Event/dod_presentations/4.3-Osterweil.ppt



Perspectives on Process Formalisms for Enhanced Software Effectiveness

Leon J. Osterweil (ljo@cs.umass.edu)
University of Massachusetts
Amherst, MA 01003
URL: laser.cs.umass.edu

DOD Software Summit
USC -- 7 August 2001

What do we mean by “process”

Activities related to the development, verification, evolution, evaluation, management, etc. of software products

Examples of high level processes:

- Develop requirements
- Do Object Oriented Design
- Formally verify

Examples of low level processes

- Archive result of running a test case
- Compile a piece of code

Tend to be concurrent (coordination of people, systems)

Usually (regrettably) informal or undefined

© Osterweil

Why the interest in process?

Superior quality from superior processes

- Build quality in, don't "test in" quality (manufacturing)
- Many observed "process errors"
  - Real processes are intricate, have intricate interconnections
  - Machines can help people with process complexities
- Process effectiveness from human/machine synergy

Processes

Are intangible and invisible

- Although their effects are substantial

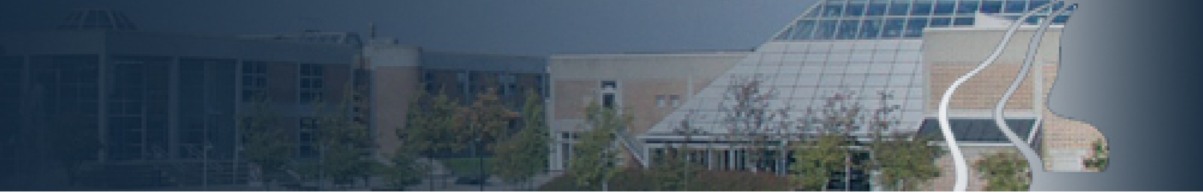
How to improve their effectiveness?

- Make them tangible, visible
- Engineer them as first-class objects

Use computer science formalisms to define processes

© Osterweil

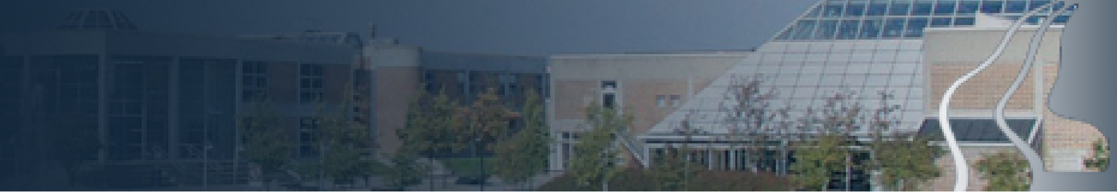
The Capability Maturity Model (CMM) is a Test Plan for Software Processes



The Capability Maturity Model (CMM) is a Test Plan for Software Processes

It supports a Black Box Testing Approach to Software Process Improvement

© Osterweil



The Capability Maturity Model (CMM) is a Test Plan for Software Processes

It supports a Black Box Testing Approach to Software Process Improvement

We advocate White Box Analysis based on explicit process definition

© Osterweil

Approaches to Process Definition

Process Definition may have different goals

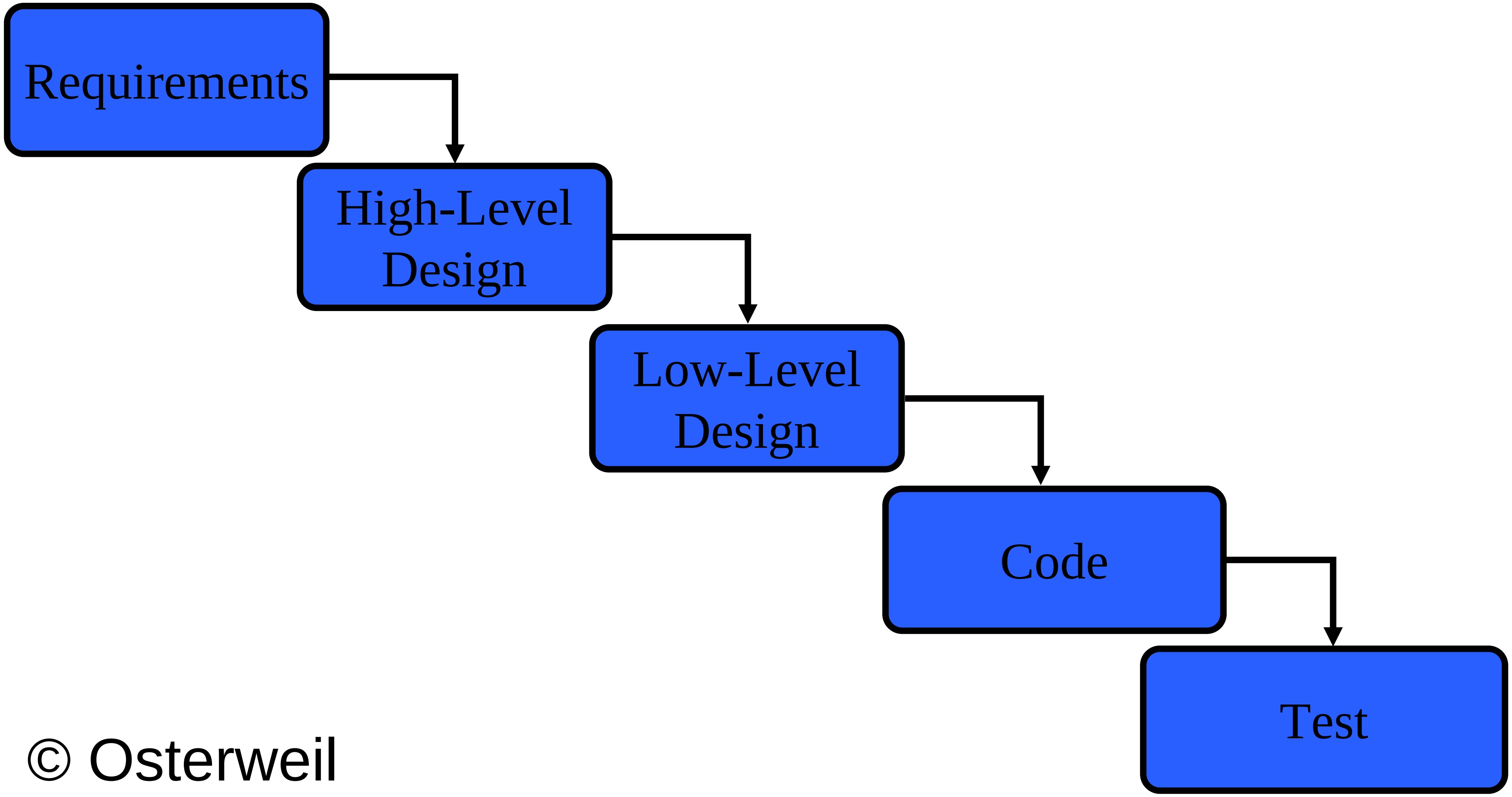
- Communication
- Coordination
- Intuitive understanding
- Verification
- Training
- Automation
- Deep understanding
- Etc.

Different definition formalisms support different of these goals to different degrees

Formalisms vary in rigor, precision (semantic detail), semantic scope, clarity

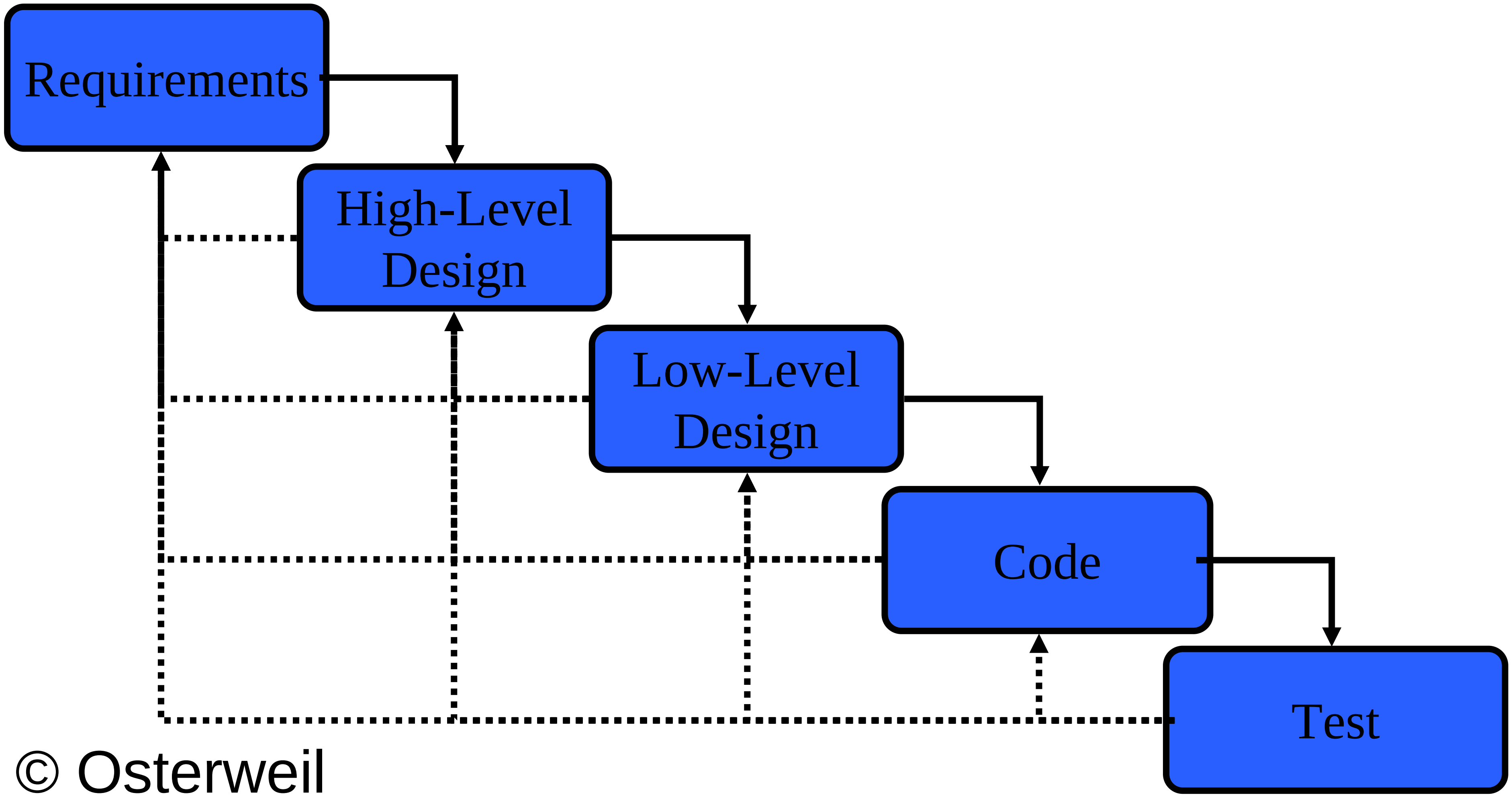
© Osterweil

Data Flow Diagram of Traditional Waterfall Model



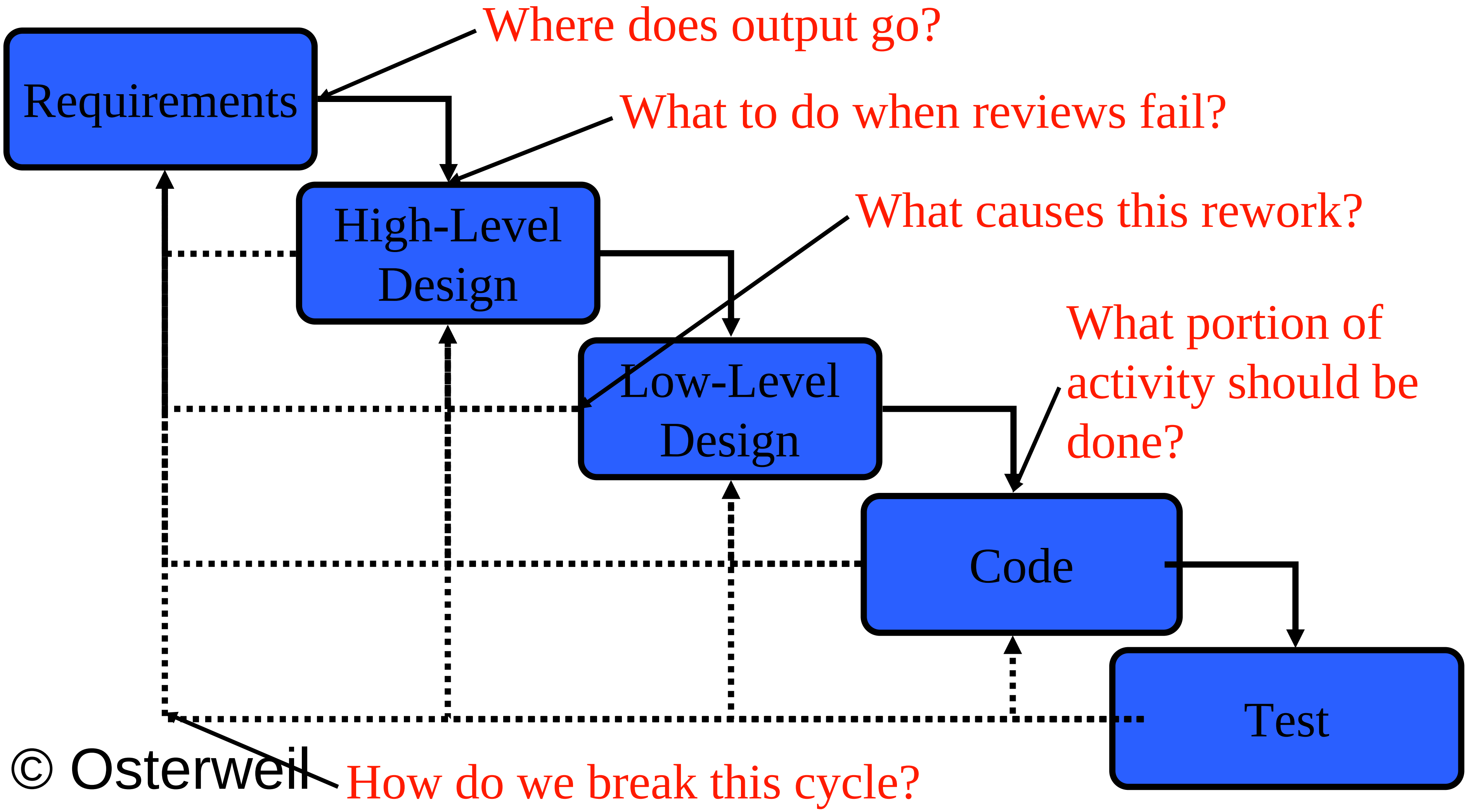
© Osterweil

With More Rework

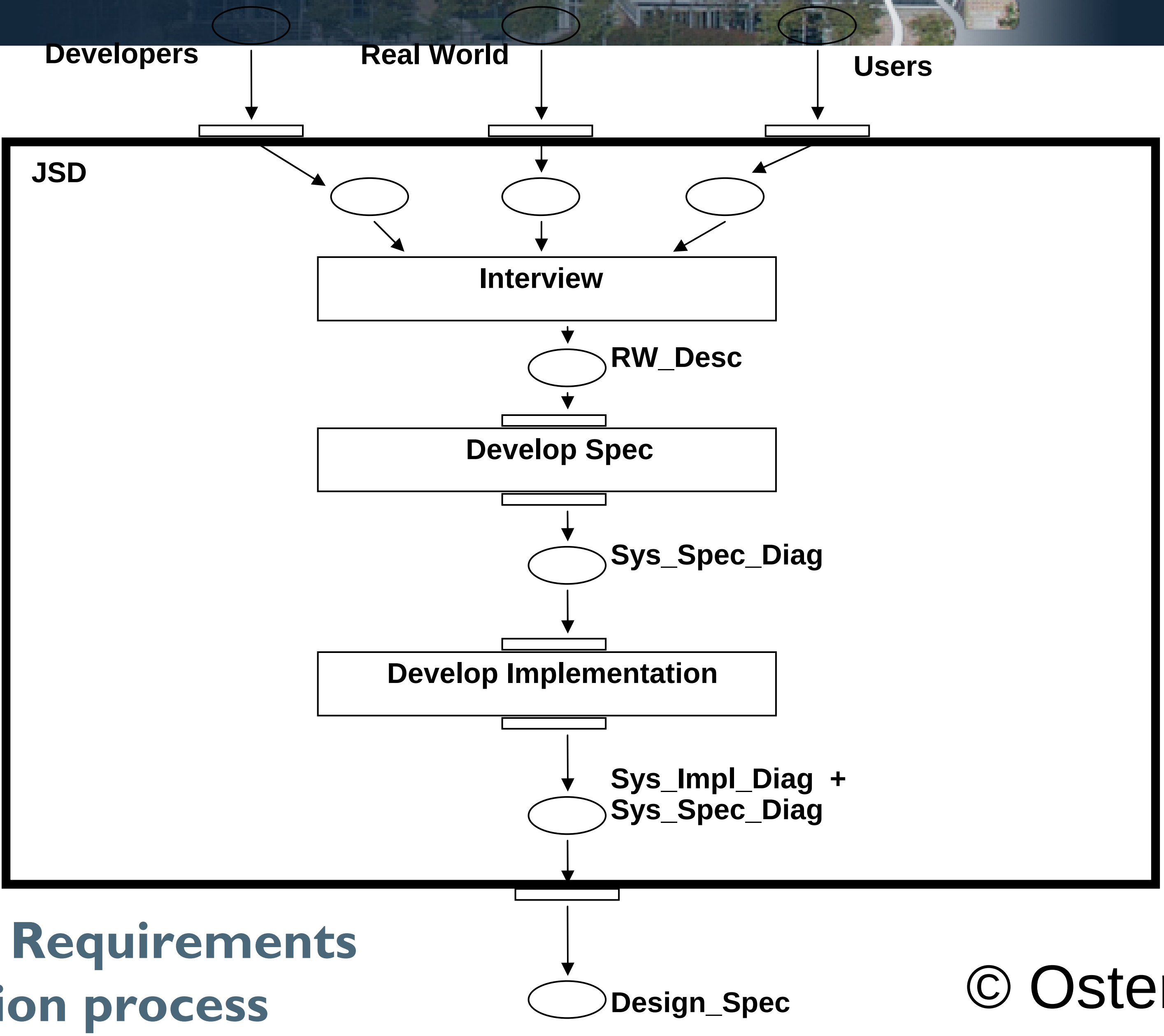


© Osterweil

The Trouble with Dataflow Diagrams

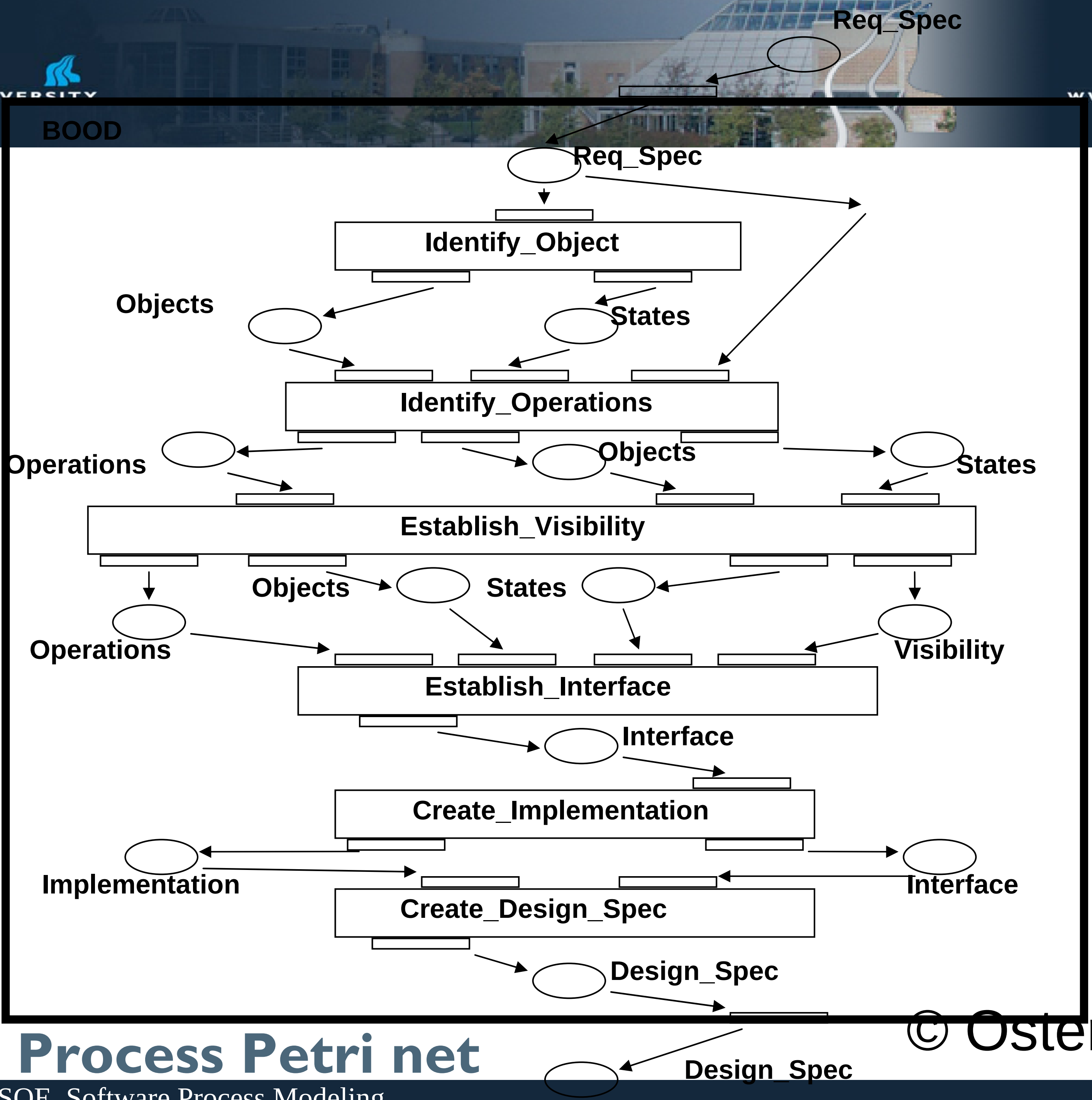


© Osterweil



Petri Net Requirements specification process

© Osterweil



© Osterweil

Design Process Petri net

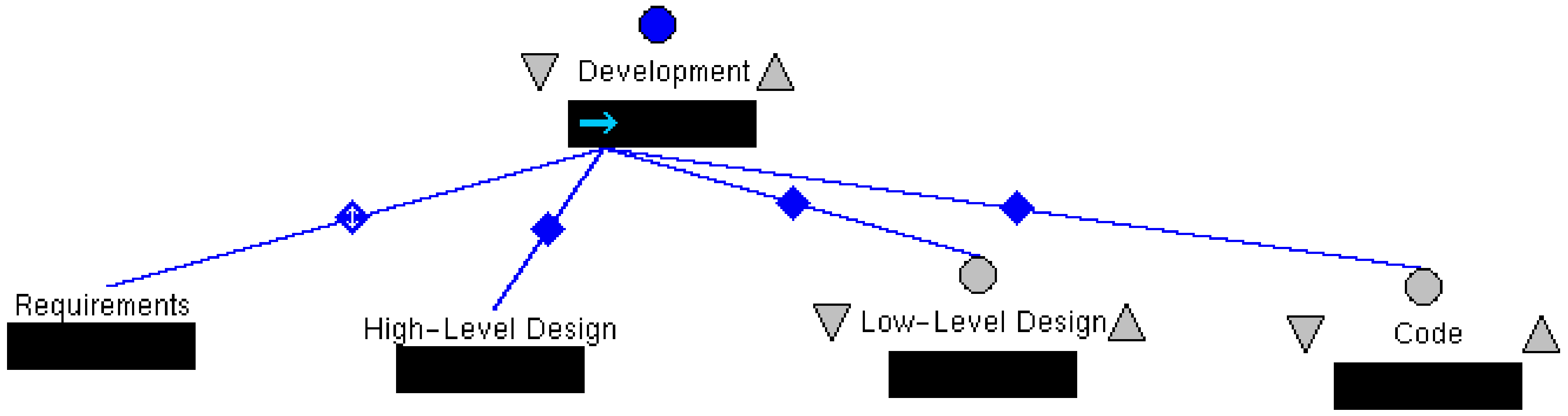
Little JIL

Graphical language with execution semantics

- Expresses process coordination
- Designed to be used interchangeably with other product, resource, scheduling factors

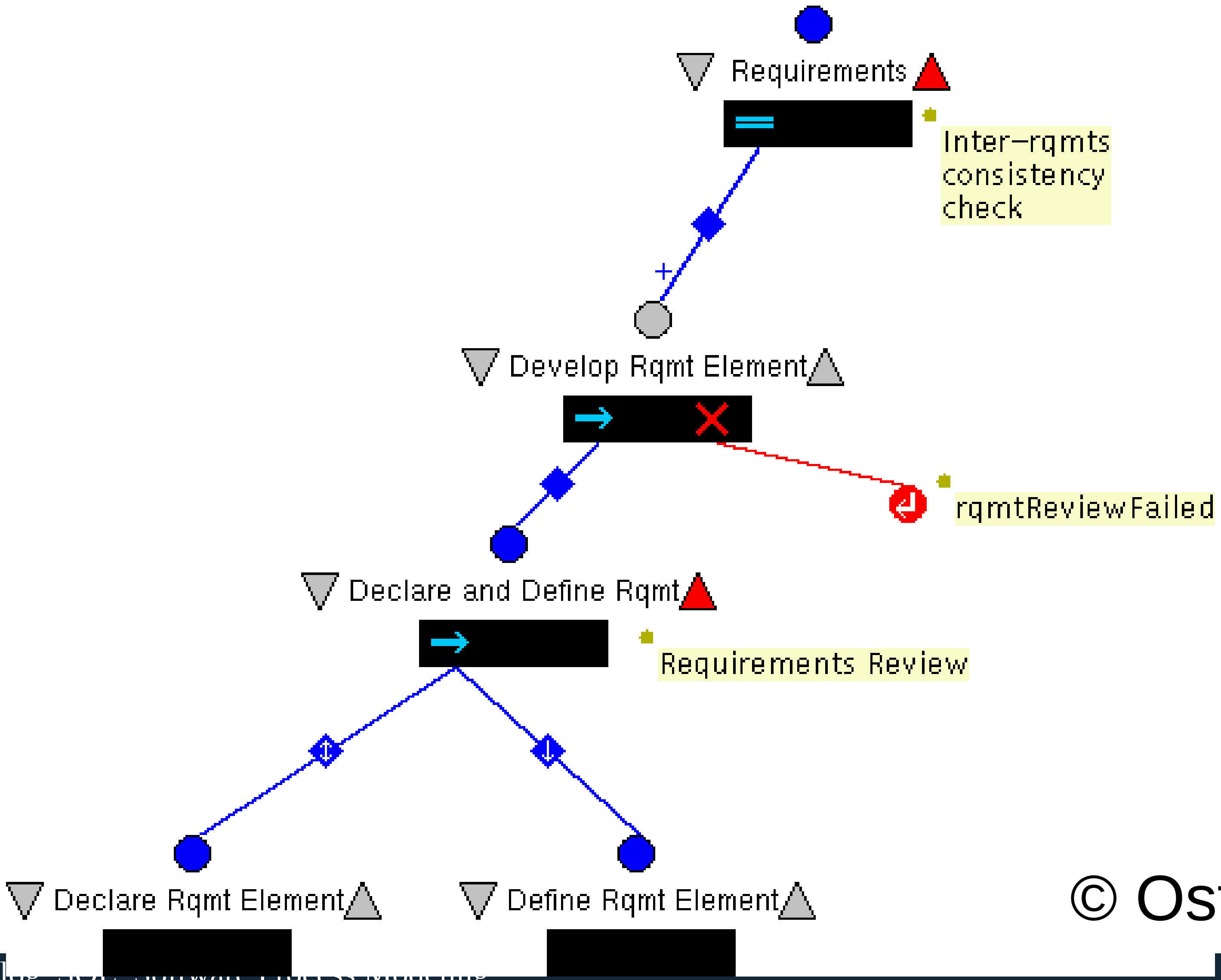
Visual JIL is the graphical interface to Little JIL

High-Level Process



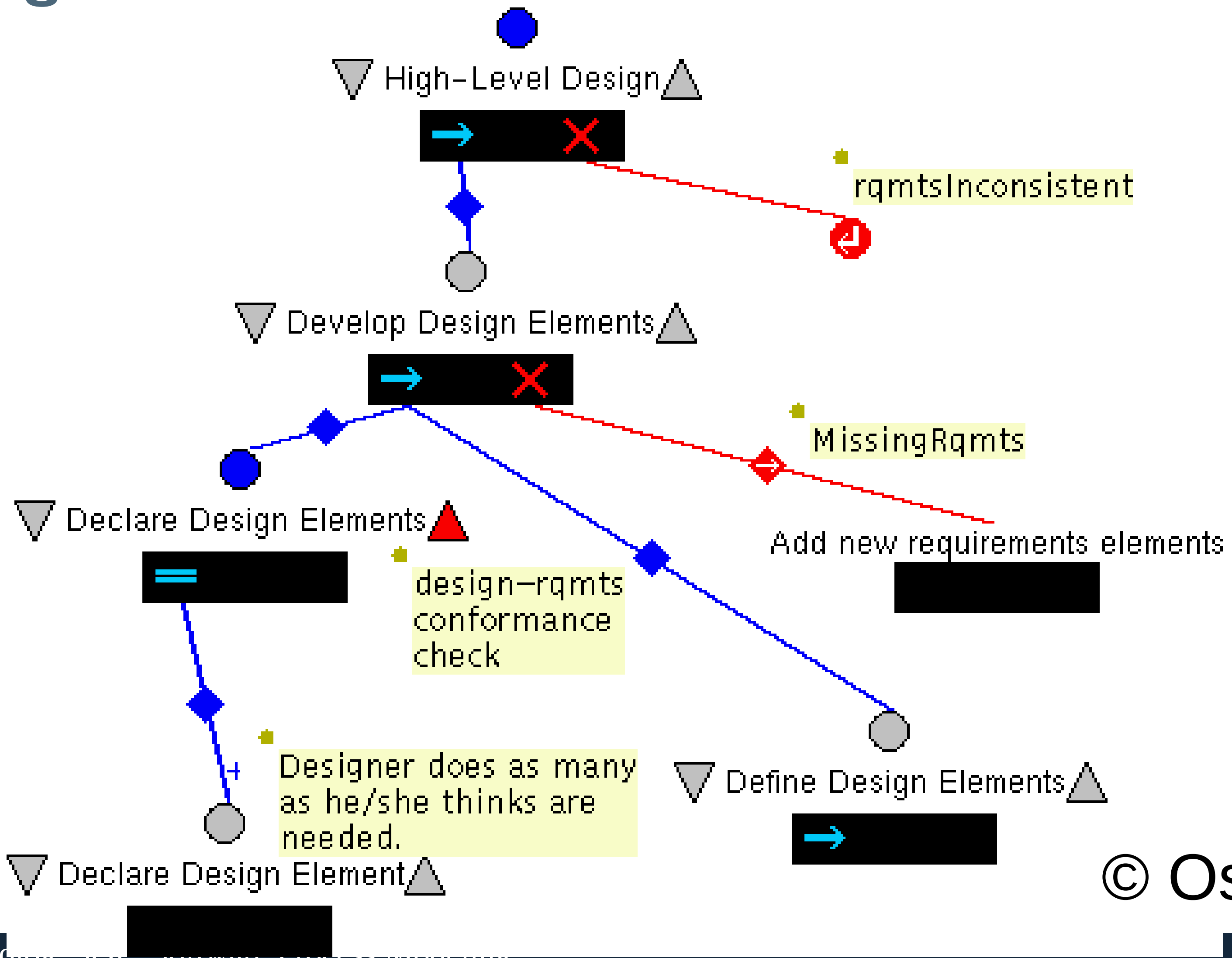
© Osterweil

Requirements Details



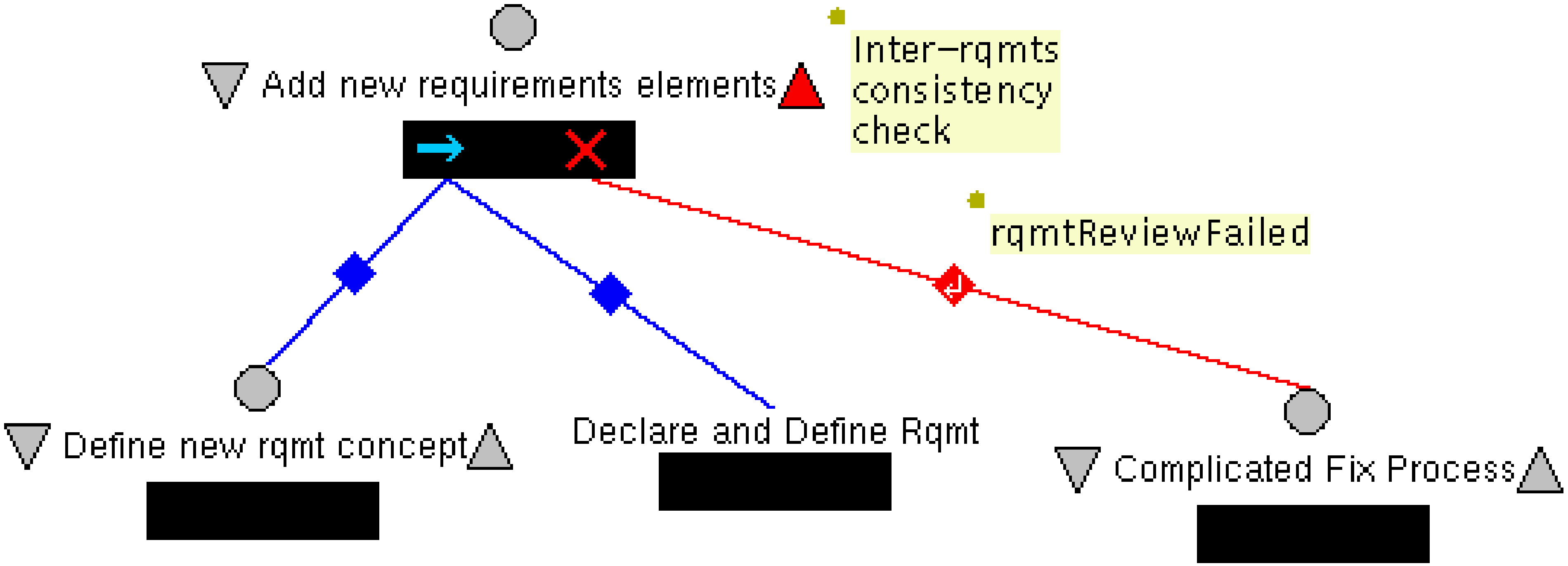
© Osterweil

Design Details

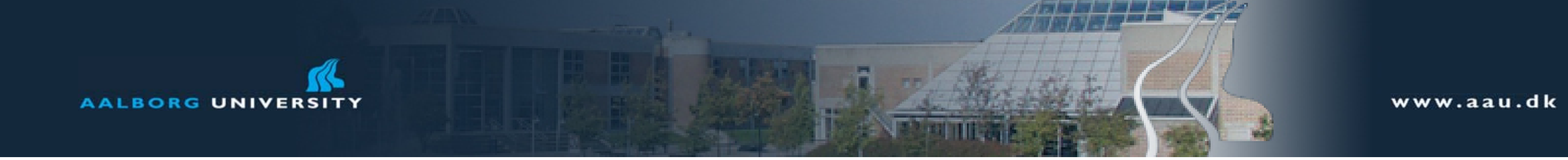


© Osterweil

Requirements Rework



© Osterweil



Opportunities and Directions

Evolve and Improve Software Processes and Products

Through improving process languages by

- Defining key processes
- Using them to integrate tools
- Automating processes
- Analyzing and improving processes

Effecting the definition and automation of superior processes for

- Software development
- Management
- Procurement

To assure superior software products

© Osterweil

Osterweil 1987

Software Processes are Software Too

End-Users and Applications

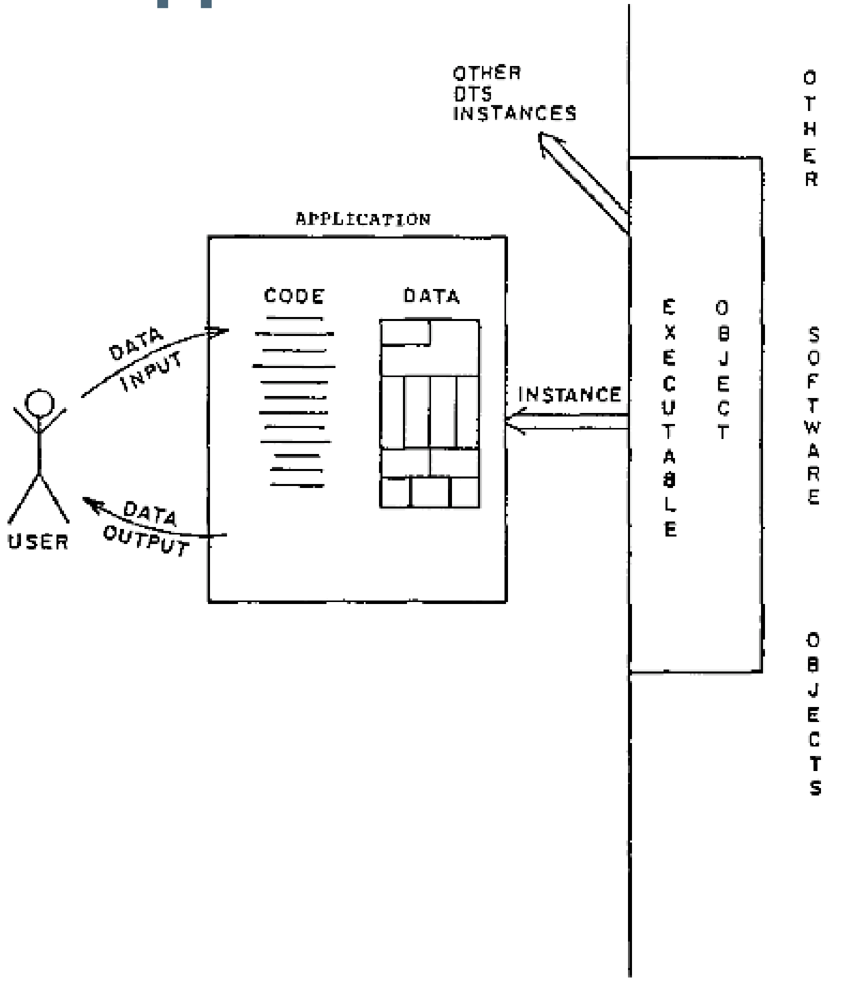


Figure 1

The Software Practitioner

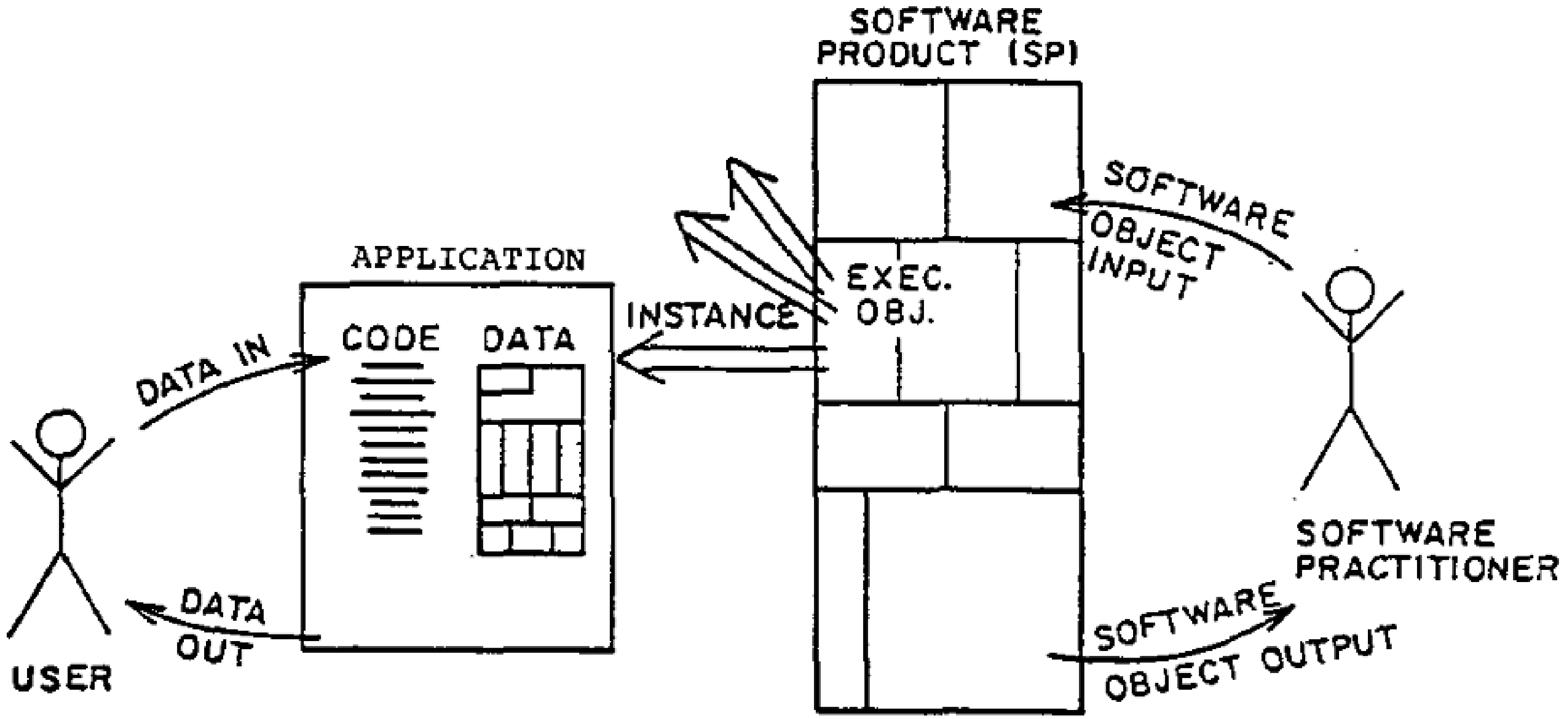


Figure 2

Deriving the Software Practitioner's Product

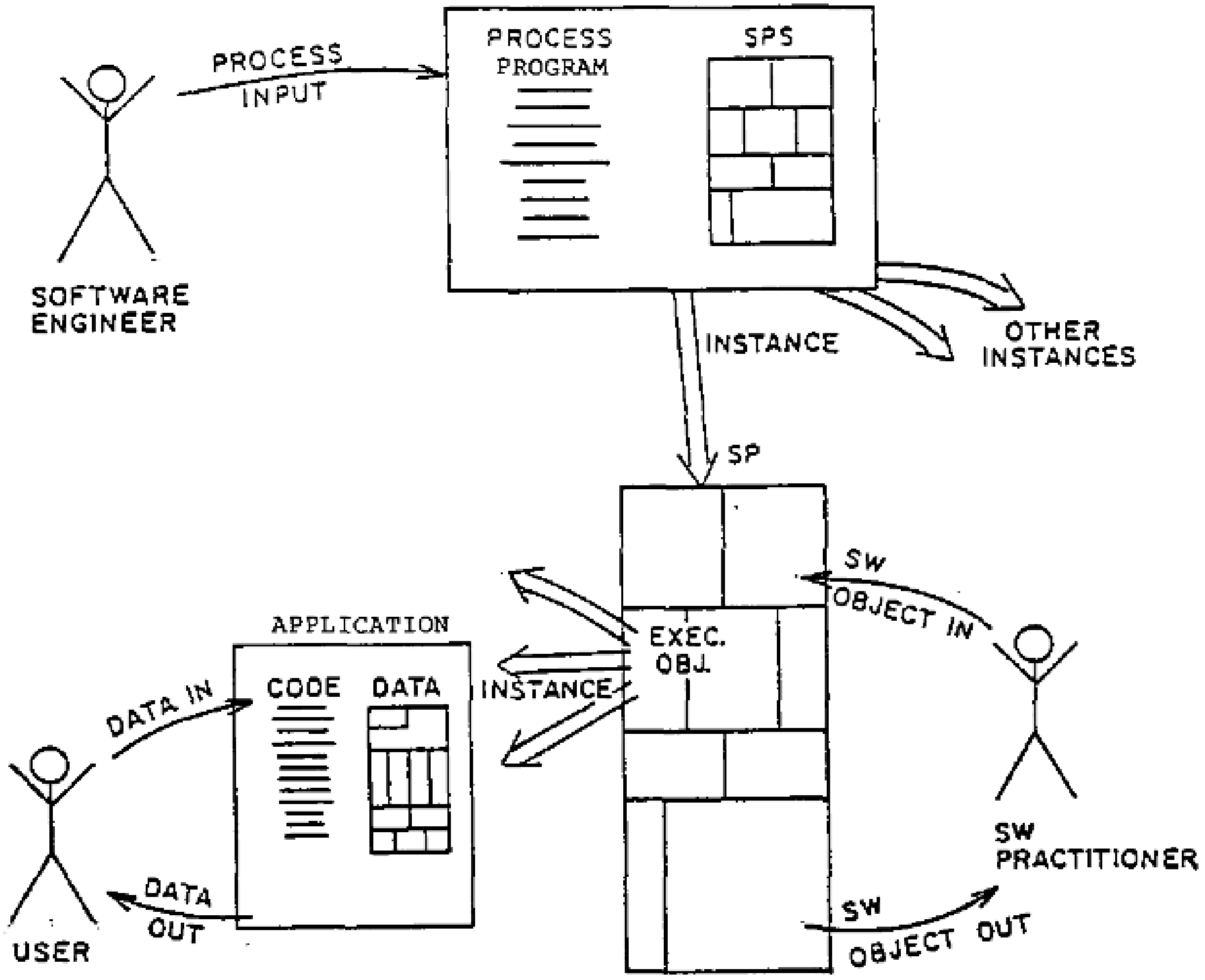


Figure 3

Process Programs and Process Descriptions

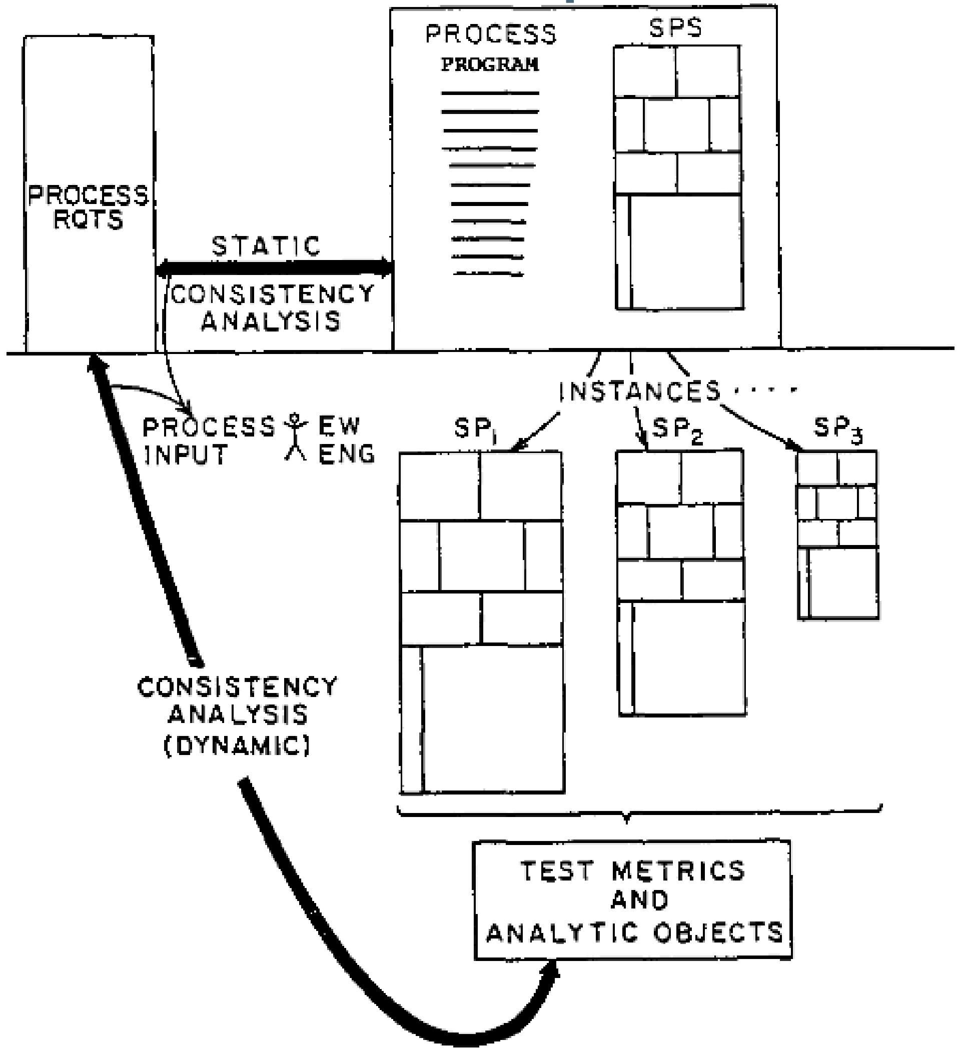


Figure 4

Process vs. Process Description

a process is a vehicle for doing a job, a process description is a specification of how the job is to be done

Process Programs and Humans

Process programs need not treat humans like ***robots***

Human tasks should be represented as ***functions or procedures*** *for which the definition is omitted, thereby leaving the human free to execute the task as he or she sees fit.*

The level to which any human task is elaborated by the process code is the ***choice of the process coder****, who, in doing so, specifies the extent to which the process is authoritarian and prescriptive, or permissive and relaxed.*

Computer Software Processes

software development as a ***manufacturing*** *activity*
parts fabrication *have considerable intuitive appeal, although they have not been entirely satisfactory.*

set of activities carded out in order to effect the development or evolution of a software product ***should be*** *considered to be a process.*

There seems to be ***a clear analogy*** *between this sort of process and the process by which airplane wings are built, gourmet meals are prepared and autos are assembled.*

Process Potential

process description/instantiation/execution capabilities are powerful indeed.

we are able to ***improvise*** *effective process descriptions on the fly*

maintain them in our heads

modify them as necessary and

guide others in their effective execution.

Hypothesis: Just think *how much we might be able to achieve were we able to externalize these process descriptions, express them rigorously, promulgate them publicly, store them archivally, and exploit computing power to help us in guiding and measuring their execution.*

Advantages of Process Programming

… the greatest advantage offered by process programming is that it provides ***a vehicle for the materialization*** *of the processes by which we develop and evolve software.*

… it is startling to realize that we develop and evolve so large, complex, and intangible an object as a large software system without the aid of suitably ***visible, detailed and formal*** *descriptions of how to proceed.*

Through a process program the ***manager*** *of a project can* ***communicate*** *to workers, customers and other managers just what steps are to be taken in order to achieve product development or evolution goals.*

Workers*, in particular, can benefit from process programs in that reading them should indicate the way in which work is to be coordinated and the way in which each individual's contribution is to fit with others' contributions.*

a critical necessity

At present key software process information is ***locked in the heads*** *of software managers.*

It can be reused only when these individuals instantiate it and apply it to the execution of a specific software process.

When these individuals are promoted, resign or die their software process ***knowledge disappears****.*

Others who have studied their work may have anecdotal views of the underlying process descriptions, but these descriptions themselves ***vanish with the individual*** *who conceived them.*

Obviously such process knowledge is a valuable commodity and ***ought to be preserved and passed on****. Materializing it is a critical necessity.*

Non-traditional Domain

*… we believe that software **process descriptions should be considered to be software**. They are created in order to describe the way in which specific information products are to be systematically developed or modified. In this respect they are no different than **a payroll program** or a **matrix inversion program**.*

… our early work indicates that process programs can be written in compilable, interpretable programming languages

*We believe that **the primary difference** between the process programs which we are suggesting and conventional programs is that process programs represent programming in a non-traditional application domain.*

Standard Processes

Work in this area is just now beginning to proliferate, and it seems that this kind of work could be most critical to fostering the ***maturation of software engineering****.*

If software engineering is to mature into an orderly discipline it seems that it must develop ***a core set*** *of well-understood, well-supported* ***standard processes****, and a cadre of practitioners who understand what the processes are and how to use them.*

Certainly the older, better established engineering disciplines, such as ***Chemical Engineering*** *and* ***Industrial Engineering****, exemplify this sort of use of process.*

Basis of the Paper

this notion is widespread and

exploitation of it is done very effectively by humans.

Processes are used to effect ***generalized****, indirect problem solving.*

The essence of the process exploitation paradigm seems to be that humans solve problems by ***creating*** *process descriptions and then* ***instantiating*** *processes to solve individual problems.*

Rather than repetitively and directly solving individual instances of problems, humans prefer to create generalized solution specifications and ***make them available*** *for instantiation (often by others) to solve individual problems directly.*

Relation to Watts Humphrey

The earliest impetus for the ideas of process programming arose out of meetings and conversations with Watts Humphrey and his team at IBM in the early 1980's.