Wisdom is not the product of schooling but the lifelong attempt to acquire it.
- Albert Einstein

Software Design

Gerhard Fischer and Leysia Palen
Spring Semester 1999

February 17, 1999

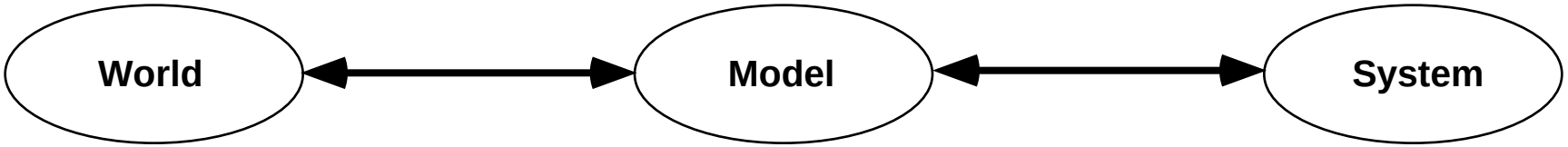
Problems of Software/System Design

- problems in semantically rich domains ----> thin spread of application knowledge
- modeling a changing world ----> changing and conflicting requirements
- turning a vague idea about an ill-defined problem into a specification ----> “design disasters”, “up-stream activities”
- symmetry of ignorance ----> communication and coordination problems
- reality is not user-friendly ----> useful *and* usable

Answers to Problems of System Design

- problems in semantically rich domains ----> thin spread of application knowledge — **domain-orientation**
- modeling a (changing) world ----> changing and conflicting requirements — **evolution**
- turning a vague idea about an ill-defined problem into a specification ----> "design disasters", "up-stream activities" — **integration of problem framing and problem solving**
- symmetry of ignorance ----> communication and coordination problems — **representation for mutual understanding and mutual learning**
- reality is not user-friendly ----> useful *and* usable — **collaborative work practices, power users**

Upstream <--------> Downstream



Upstream

Design

design disaster

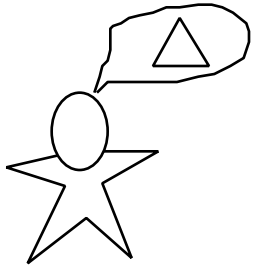
Downstream

Engineering

implementation diaster

Why Upstream — Understanding the Context

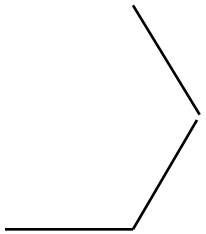
Intention of the Designer:



Procedure Written by the Designer:

```
define triangle
       repeat 3 [forward 100 right 60]
```

Feedback from the Environment:

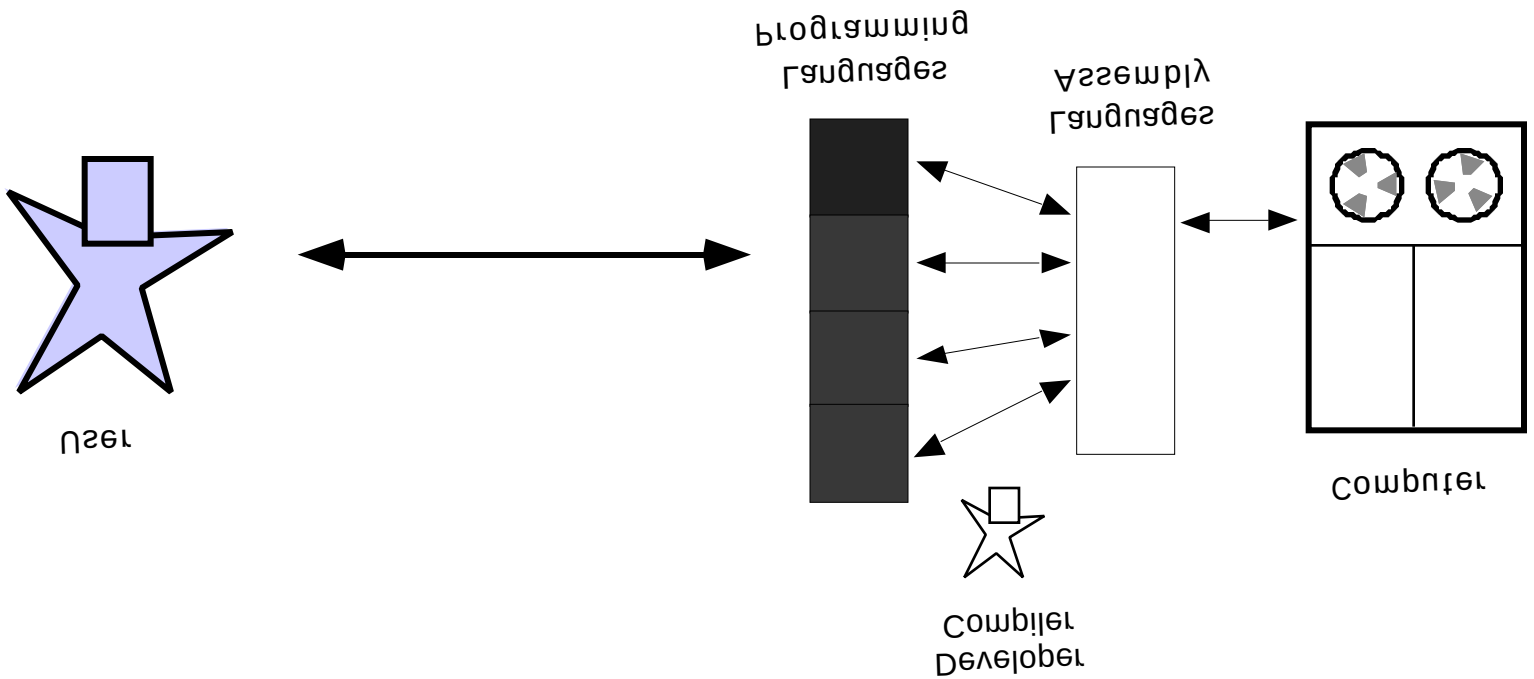


"Intent" Articulation and Communication (communicated to the system): `closed figure`

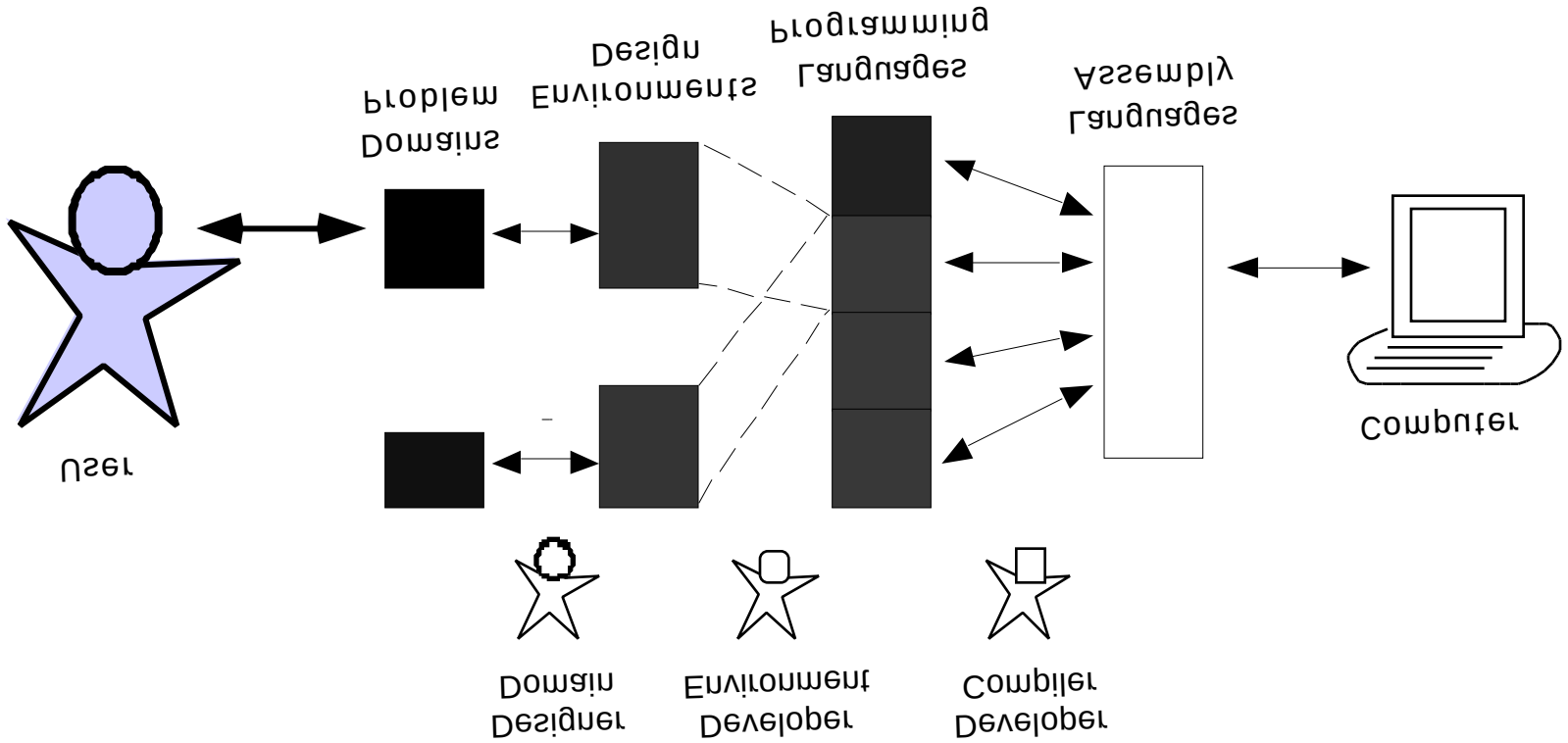
Upstream <--------> Downstream

	Upstream	Downstream
type of problem	ill-defined problems	well-defined problems
criteria to judge solutions	adequate, understandable, enjoyable	correct, robust, reliable, meets functional specifications
breakdowns	design diasters (we solve the wrong problem)	implementation diasters (wrong solution to the “right” problem)
primary source of knowledge	domain workers	software designers
support environments	domain-oriented design environments	knowledge-based software assistants, programming environments
interaction paradigm	languages of doing:prototypes, scenarios, mock-ups, conceptual models of users	(formal) specifications
externalization	(semi-formal) objects-to-think-with; understood by all stakeholders	computationally interpretable objects
focus	embedding in larger context, user experience	computational mechanisms
evolution	participatory design, use situations	debugging, verification, validation

The 1960s: High-Level, General Purpose Programming Languages



Domain-Oriented Design Environments Supporting Human Problem-Domain Communication



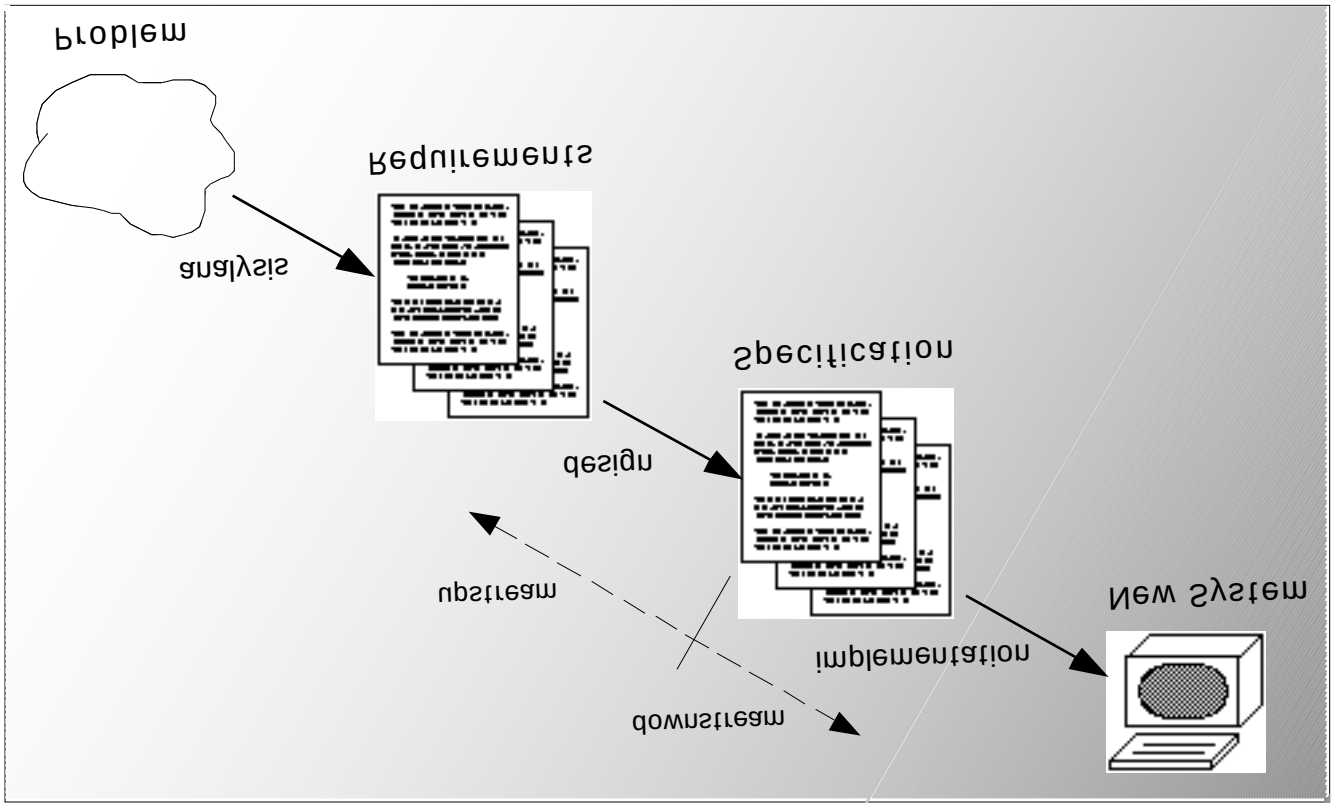
Three Generations of Design Methods from the History of Architectural Design

- **1st Generation (before 1970):**
  - directionality and causality
  - separation of analysis from synthesis ---> waterfall model
  - major drawback: (a) perceived by the designers as being unnatural, and (b) does not correspond to actual design practice

- **2nd Generation in the early 70'es:**
  - participation — expertise in design is distributed among all participants
  - argumentation — various positions on each issue
  - major drawback: insisting on total participation neglects expertise possessed by a well-informed and skilled designer

- **3rd Generation (in the late 70'es):**
  - inspired by Popper: the role of the designer is to make expert design conjectures
  - these conjectures must be open to refutation and rejection by the people for whom they are made (---> end-user modifiability)

The Waterfall Model



Software and Design — Some Claims

- although there is a huge diversity among design disciplines, we can find **common concerns and principles** that are applicable to the design of any object, whether it is a poster, a household appliance, a housing development, a software environment

- **software design** is a user-oriented field, and as such will always have the human openness of disciplines such as architecture and graphic design, rather than the hard-edged formulaic certainty of engineering design (Winograd)

- system development is **difficult** not because of the complexity of technical problems, but because of the **social interaction** when users and system developers learn to create, develop and express their ideas and visions (Greenbaum & Kyng)

- **questions to be asked about software design?**
  - how does it differ from programming, software engineering, software architecture, human factors and interface design?
  - how is it related to other fields that call themselves design, such as industrial design, graphic design, information design, urban design, and even fashion design?

Readings about Software Design

- Curtis, B., Krasner, H., & Iscoe, N. (1988) "A Field Study of the Software Design Process for Large Systems," Communications of the ACM, 31(11), pp. 1268-1287.
- Ehn, P. (1988) Work-Oriented Design of Computer Artifacts, Almquist & Wiksell International, Stockholm, Sweden (discussion of the three generations of design methodologies)
- Greenbaum, J. & Kyng, M. (Eds.) (1991) Design at Work: Cooperative Design of Computer Systems, Lawrence Erlbaum Associates, Inc., Hillsdale, NJ.
- Winograd, T. (1995) "From Programming Environments to Environments for Designing," Communications of the ACM, 38(6), pp. 65-74.
- Winograd, T. (Ed.) (1996) Bringing Design to Software, ACM Press and Addison-Wesley, New York.
- Tognazzini, B. (1996) Tog on Software Design, Addison-Wesley Publishing Company, Reading, Massachusetts (the author of the Starfire Video)
- Fischer, G., K. Nakakoji, & J. Ostwald (1999): “Domain-Oriented Design Environments — A New Understanding of Design and Its Computational Support”, forthcoming