End-User Debugging in Distributed Environments

Institute of Parallel and Distributed Systems (IPVS)
University of Stuttgart
Germany

Research Group Context-based Systems

Dipl.-Inform. Andreas Heil
Supervised by Dr. Torben Weis (University of Stuttgart) and Dr. Alexander Brändle (Microsoft Research Cambridge)

In a Nutshell

"Millions for compilers but hardly a penny for understanding human programming language use. Now, programming languages are obviously symmetrical, the computer on one side, the programmer on the other. I an appropriate science of computer languages, one would expect that half of the effort would be on the computer side, understanding how to design languages that are easy or productive to use… The human and computer parts of programming languages have developed in radical asymmetry."

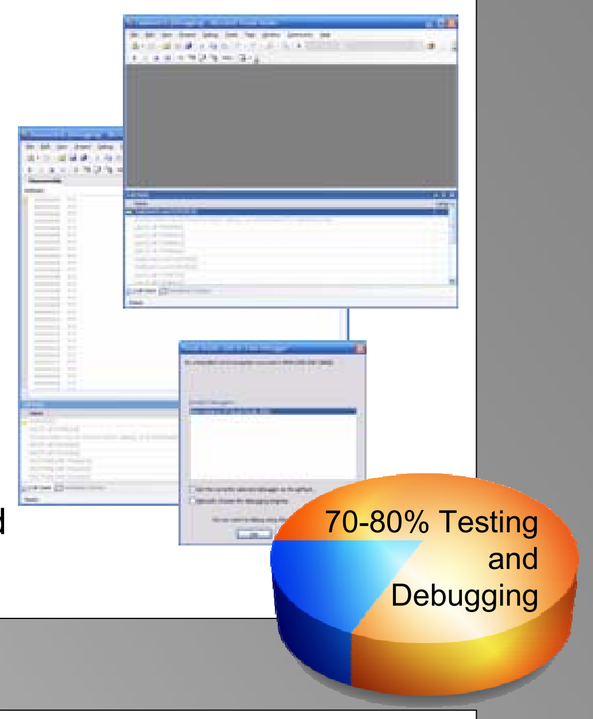
Allen Newell and Stuart Card, 1985

Challenge

Designing testing and debugging strategies for distributed systems

Common debugging strategies fail in distributed systems (breakpoints, step-through, test-first…)

Reproduction of faults in real world pervasive systems

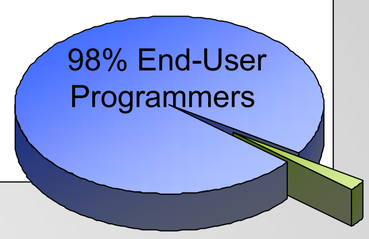


Definition*

End-User Programmer
People who write programs, but not as their primary job function, but to achieve their main goals which are usually something complete different

Professional Programmer
Someone whose primary job function is to write or maintain software. Typically having significant training in programming (e.g., BS in CS)

Novice Programmers
Someone who is learning to be a professional programmer

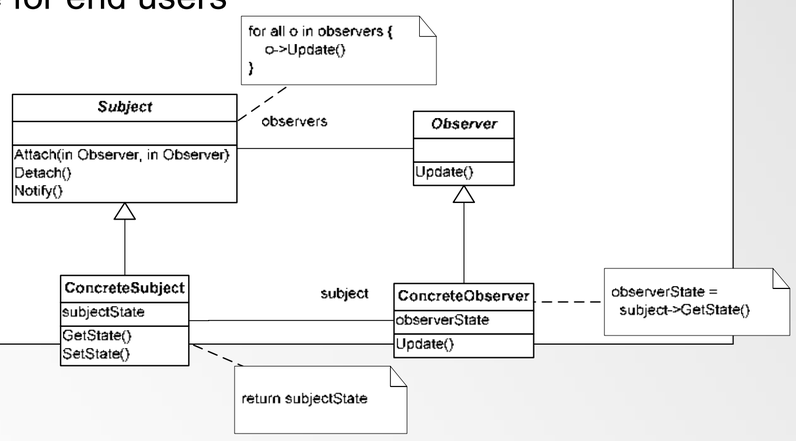


** by Brad Myers, 2006*

Software Crafting

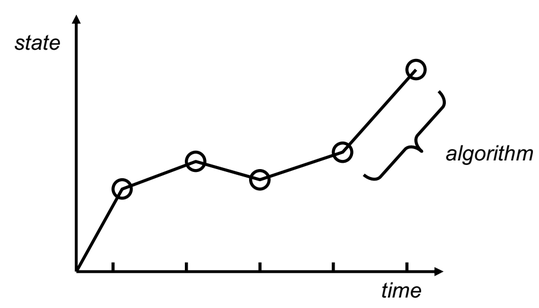
Not aware of software engineering techniques and debugging strategies, end-users need a new set of techniques and tools, being the counterpart of software engineering and debugging for end-users

Typical software engineering techniques (e.g., design pattern) are not accessible for end users



Changing Paradigm

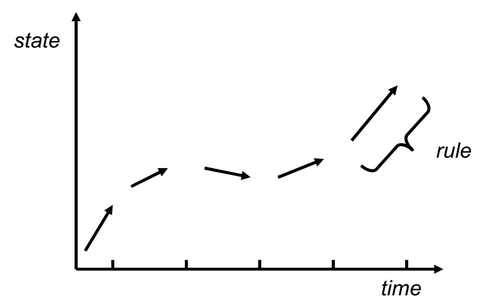
1. State changes within programs are described by algorithms (*how* to achieve a state)




```
foreach (Observer o in observers)
{
    o.Update(this);
}
```

Knowledge about software engineering techniques is required

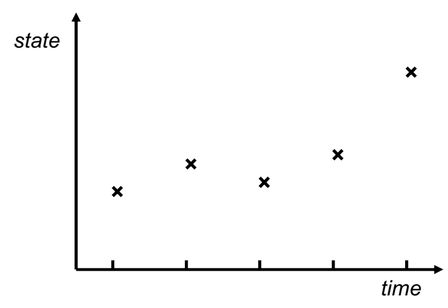
2. Rules allow to describe the conditions for state changes (*why* to reach a state)




```
sensor(ESB3).
sensor(ESB4).
…
light(on) :- sensor(X),
             range(2),
             event(motion).
```

Detailed knowledge about the system is required to define facts and rules

3. By expressing a *desire*, the end-user can define a state (*which* state to reach)



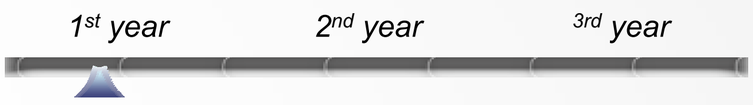
Defining the context

```
while a meeting takes place
and no beamer is used
the room shall be bright
```

Expressing the desire

Defining states is possible in pervasive computing and allows to query why a system did or did not reach a certain state

Achievements:
Prototype for evaluating and simulating different approaches in debugging distributed systems. Allows to trace and evaluate the control flow and the system's state



Next Steps:
Adapting current research results in end-user development to debugging strategies of distributed systems

Applying commercial products (e.g., the Microsoft Robotics Studio) to evaluate against real world distributed systems