PROGRAMMING LANGUAGES LAB

Lab Members

Faculty:
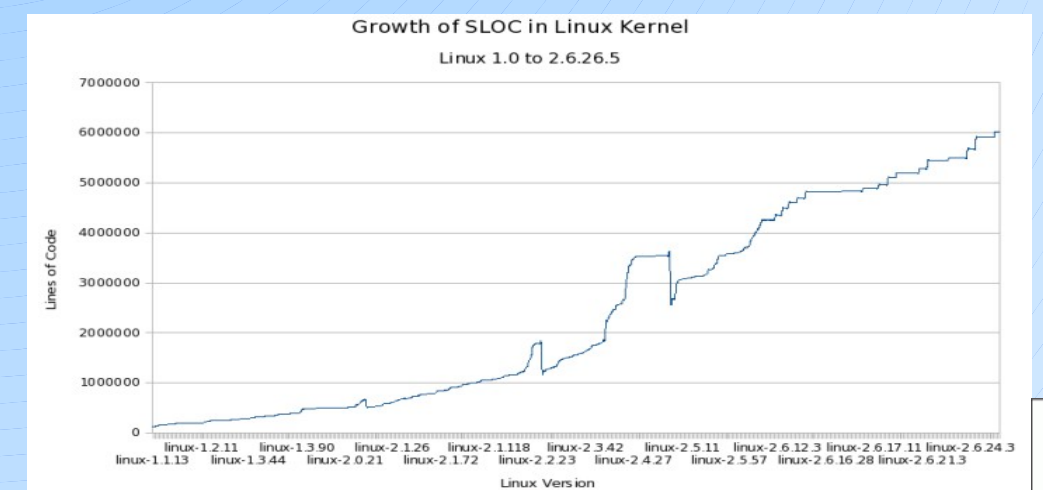Dr. Deepak D'Souza
Dr. K V Raghavan

Students:
Arnab De
Hari Shankar Gupta
Raghavendra K R
Ravichandran K M
Sudha Balodia
Vasanta K

Why is research in programming languages important?

Emerging areas with programmable systems

Increasing size and complexity of software

Need for reliable software

Adriane 5 explosion due to incorrect software
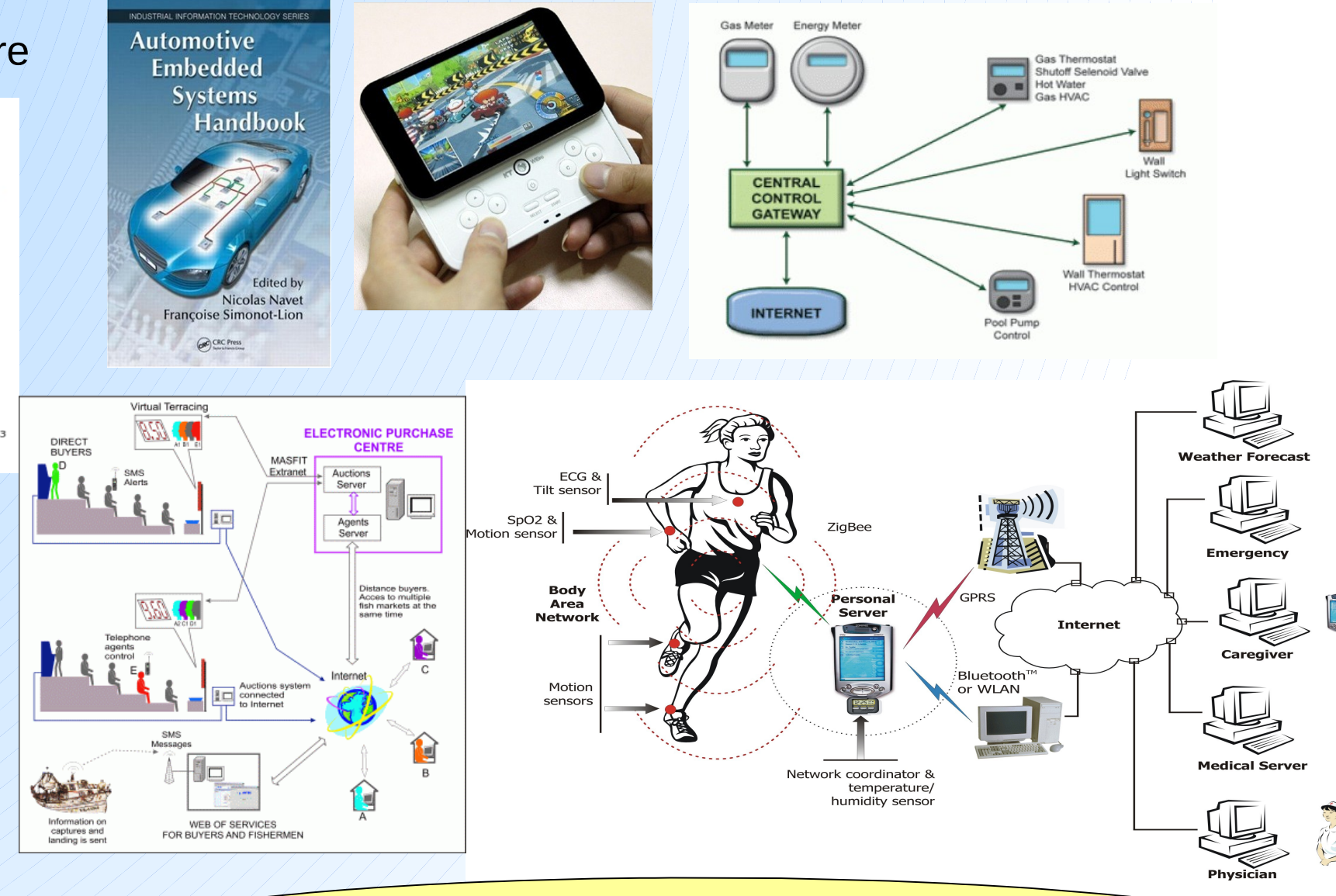
Five of last ten Turing Awards are for research in PL!

Notable contributions of PL research

MP3 Encode on Atom 1.33 GHz

Languages

Optimizing compilers

Verification

Java PathFinder (Collaboration with NASA)

Bug detection

Refactoring
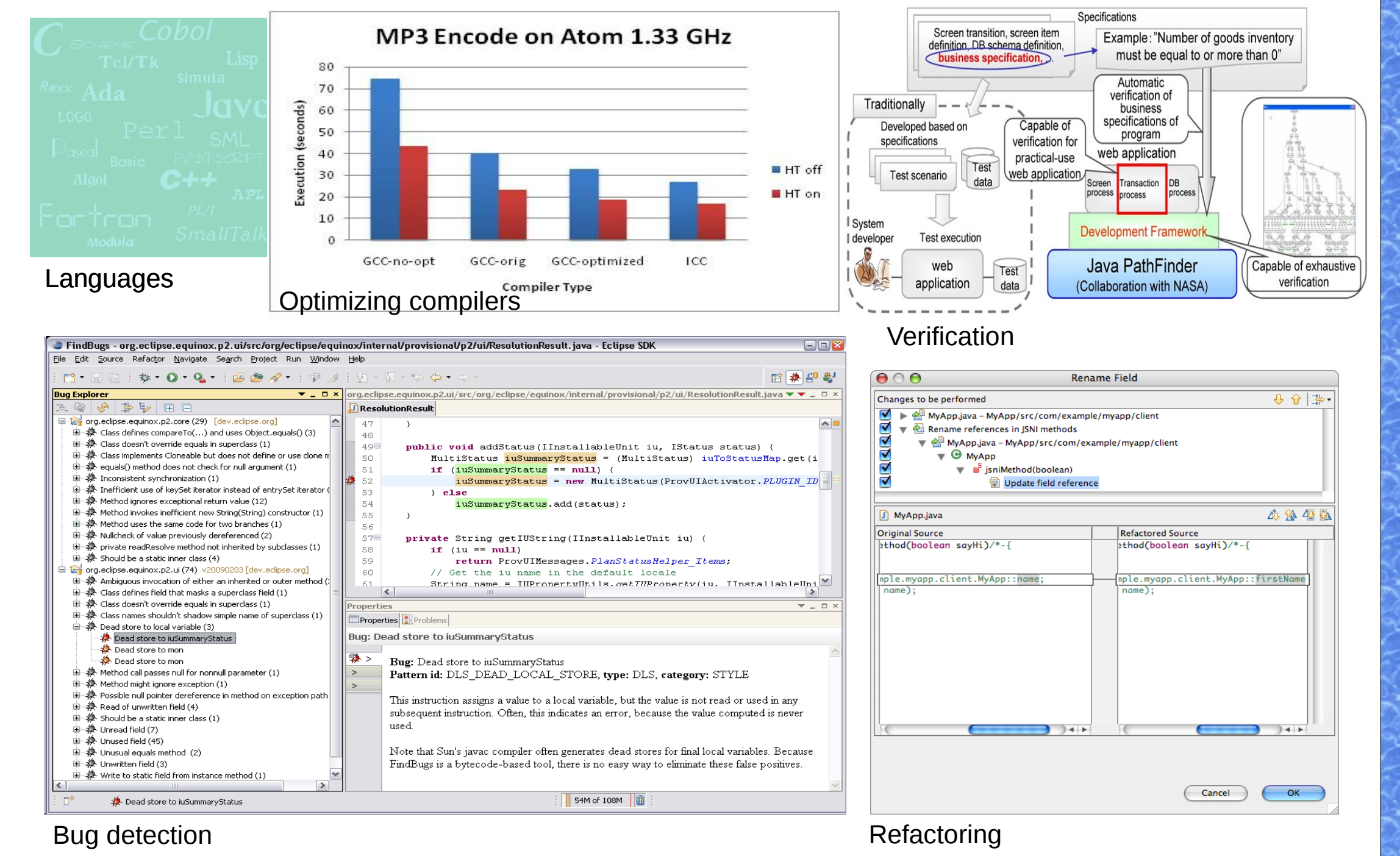
Analysis of Shared-memory Concurrent Programs

```
x = y = 0

x = 1;    ||    y = 1;
r1 = y;   ||    r2 = x;
```

Programmability!!
Efficiency!

Most processors may produce
r1 == r2 == 0!
❖ Reordering needed for performance.
❖ May produce unexpected behaviors!
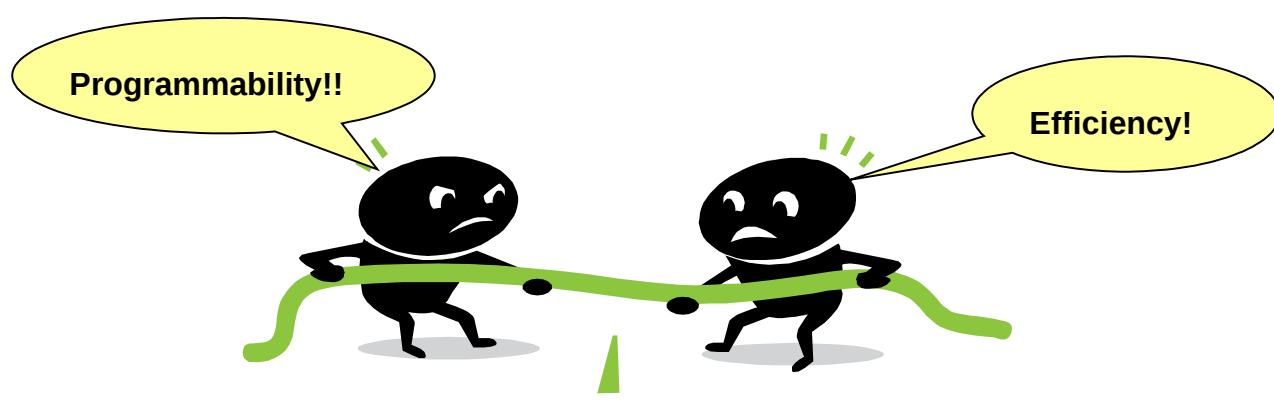
➤ What is the ideal semantics of shared-memory concurrent programs?
➤ How to verify properties of programs with weak semantics?
✓ Proposed an easy-to-understand operational semantics that also allows efficient implementation.
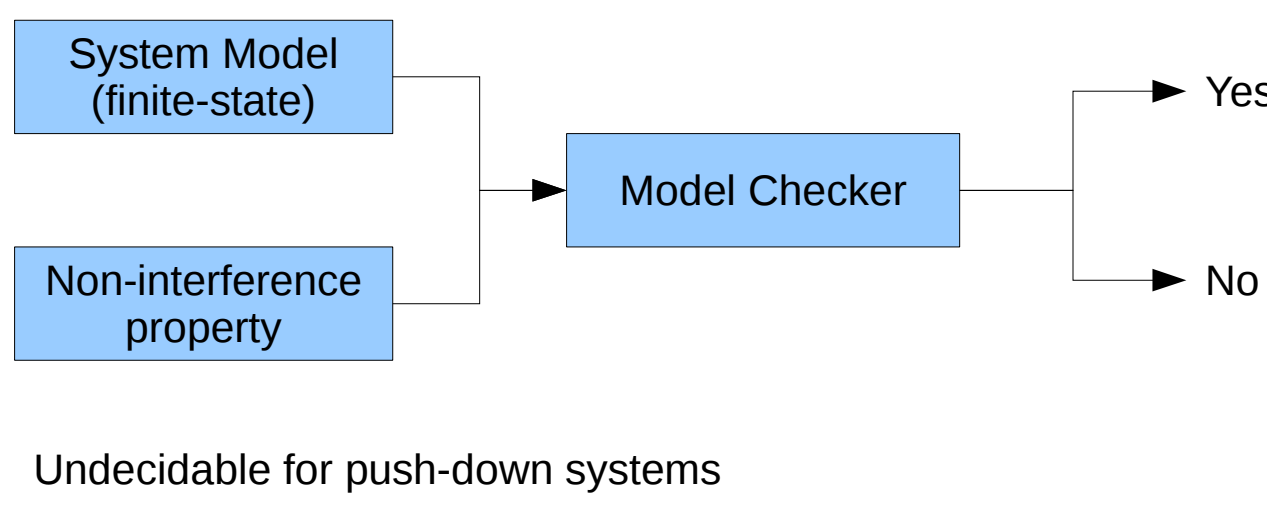✓ Developed a model-checker based on the semantics.

➤ Are well-synchronized programs easy to analyze?
✓ Goal: Developing a technique for dataflow analysis for well-synchronized programs.

```
p = &a;

lock l;    | lock l;
p = NULL;  | c = *p;
p = &b;    | unlock l;
unlock l;  |
```
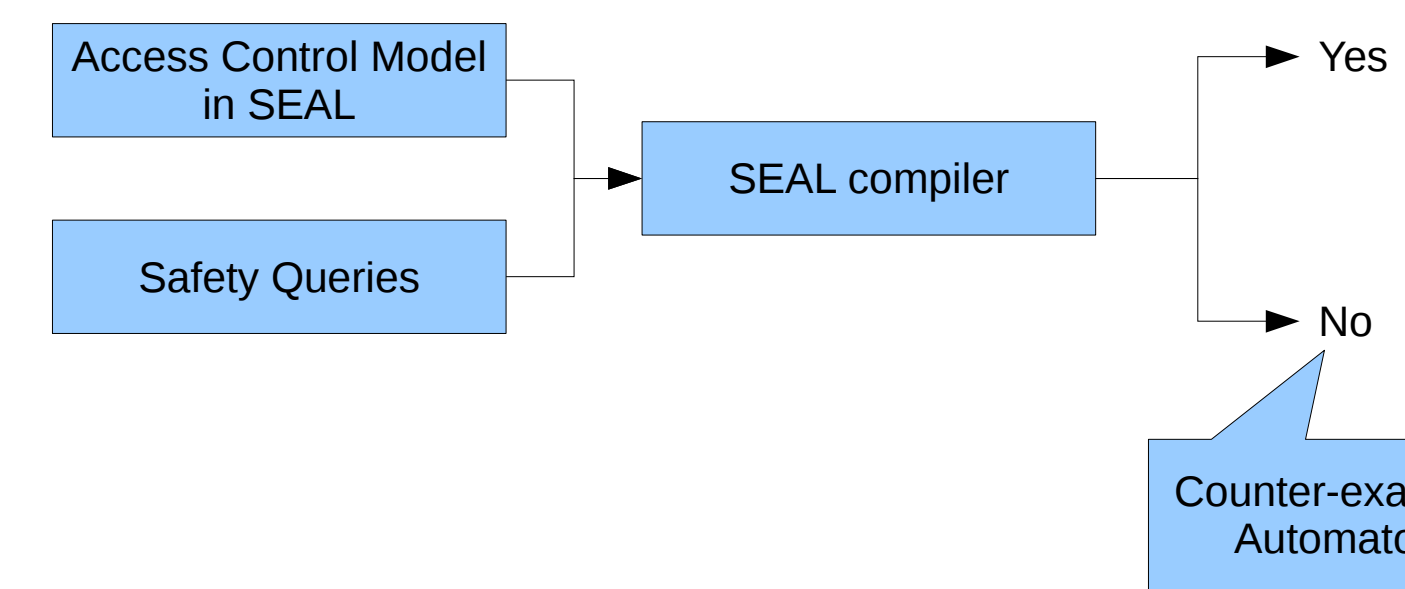
Given the program is well-synchronized, can we prove absence of null-deref efficiently?

Verification of Information Security

Model Checking Information Flow properties on abstract systems

System Model (finite-state)
Non-interference property
→ Model Checker → Yes / No

Undecidable for push-down systems

SEAL: Framework for Verifying Access Control Models

Access Control Model in SEAL
Safety Queries
→ SEAL compiler → Yes / No
Counter-example Automaton

Ongoing work: Verifying Security for Programs

Problem : Automatic Testing of "Firewall rule Configurations"

As Complexity of configuring firewalls has grown significantly, automatic testing is required to ensure conformance with the required policy.

Packet reaches firewall interface
Go to the first rule
Does it match the rule? → Next Rule
Take the specified action

Fig1. Filtering process

This is the policy

Packets from A machine should be allowed to go everywhere.
Oh.. But NOONE except C should be allowed to reach D.

Is this configuration correct?

Source.	Dest.	Action
C	D	Accept
A	Anywhere	Accept
Anywhere	D	Drop

Expected Results:
1. Detect violation1.
2. Detect violation2.
3. All other errors.

Violation 1 !!!!
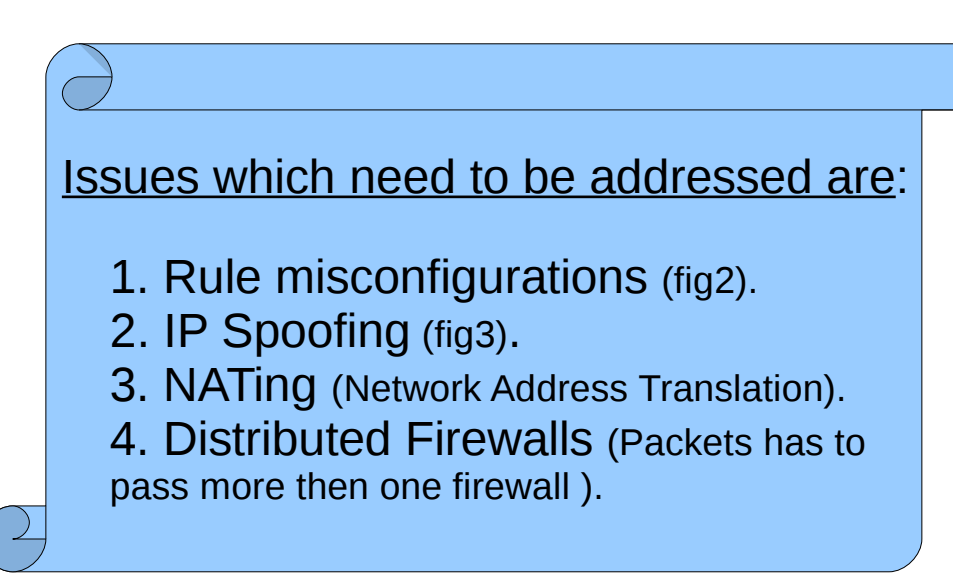Packets from A to data storage(D) will be allowed

Fig2. Rule misconfiguration

Issues which need to be addressed are:
1. Rule misconfigurations (fig2).
2. IP Spoofing (fig3).
3. NATing (Network Address Translation).
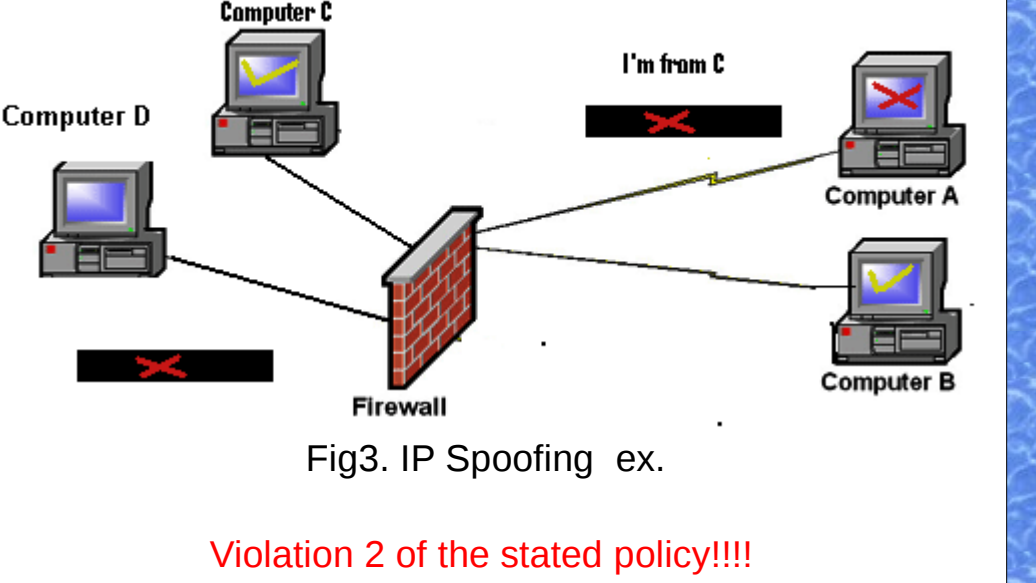4. Distributed Firewalls (Packets has to pass more then one firewall).

Computer C
Computer D
I'm from C
Firewall
Computer A
Computer B

Fig3. IP Spoofing ex.

Violation 2 of the stated policy!!!!

Testing of Transaction Based Systems
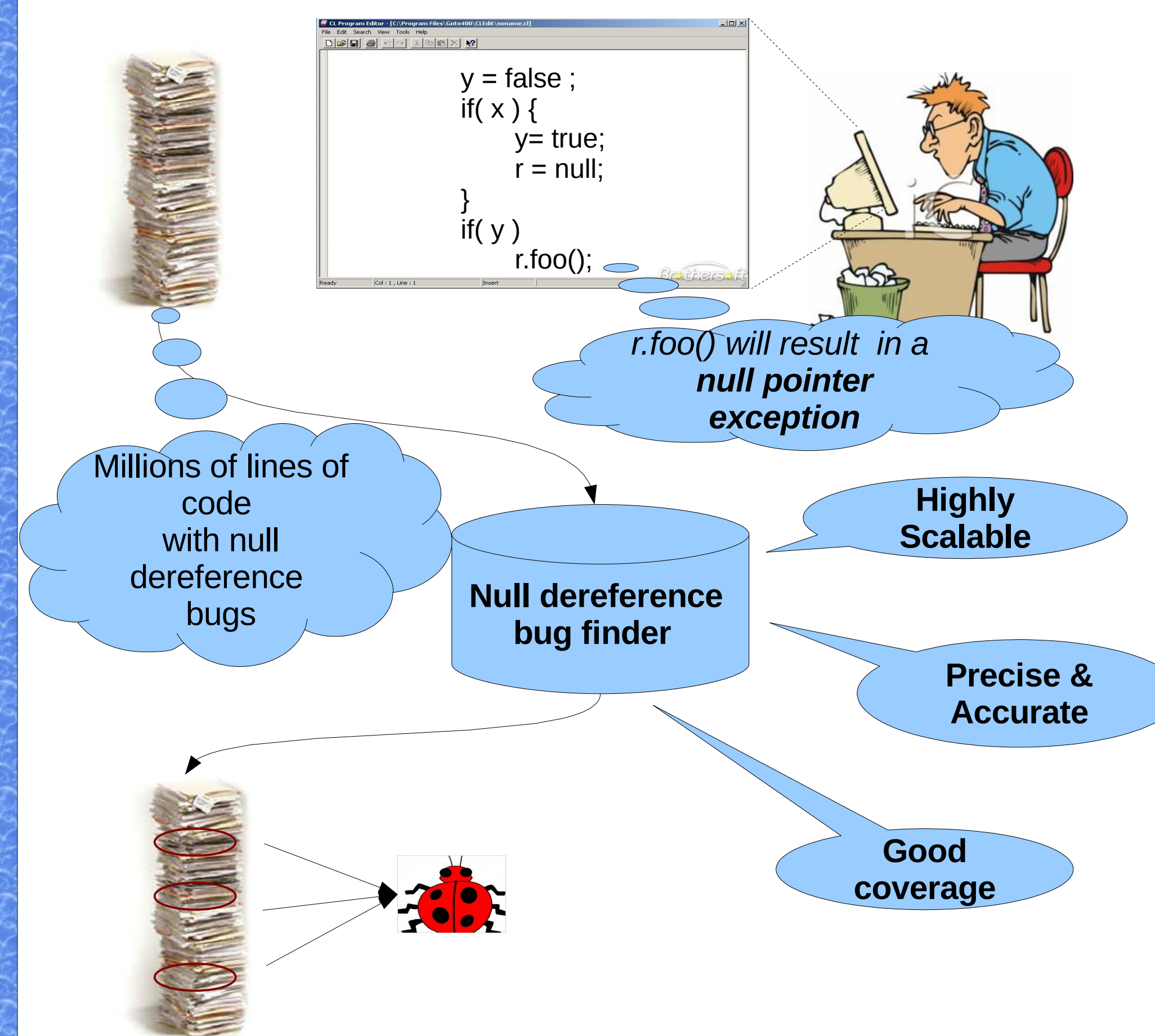
Input:
• Withdrawal/Deposit
• Amount
→ Reject/ Accept

Given system specifications:

•Individual withdrawals have an upper limit of Rs. 20,000.
•Overdrafts (i.e., withdrawing more than the current balance) are allowed.
•After six overdrafts, no more overdrafts allowed until 12 normal withdrawals are completed. Overdraft not allowed:
  When balance is below negative Rs. 40,000,
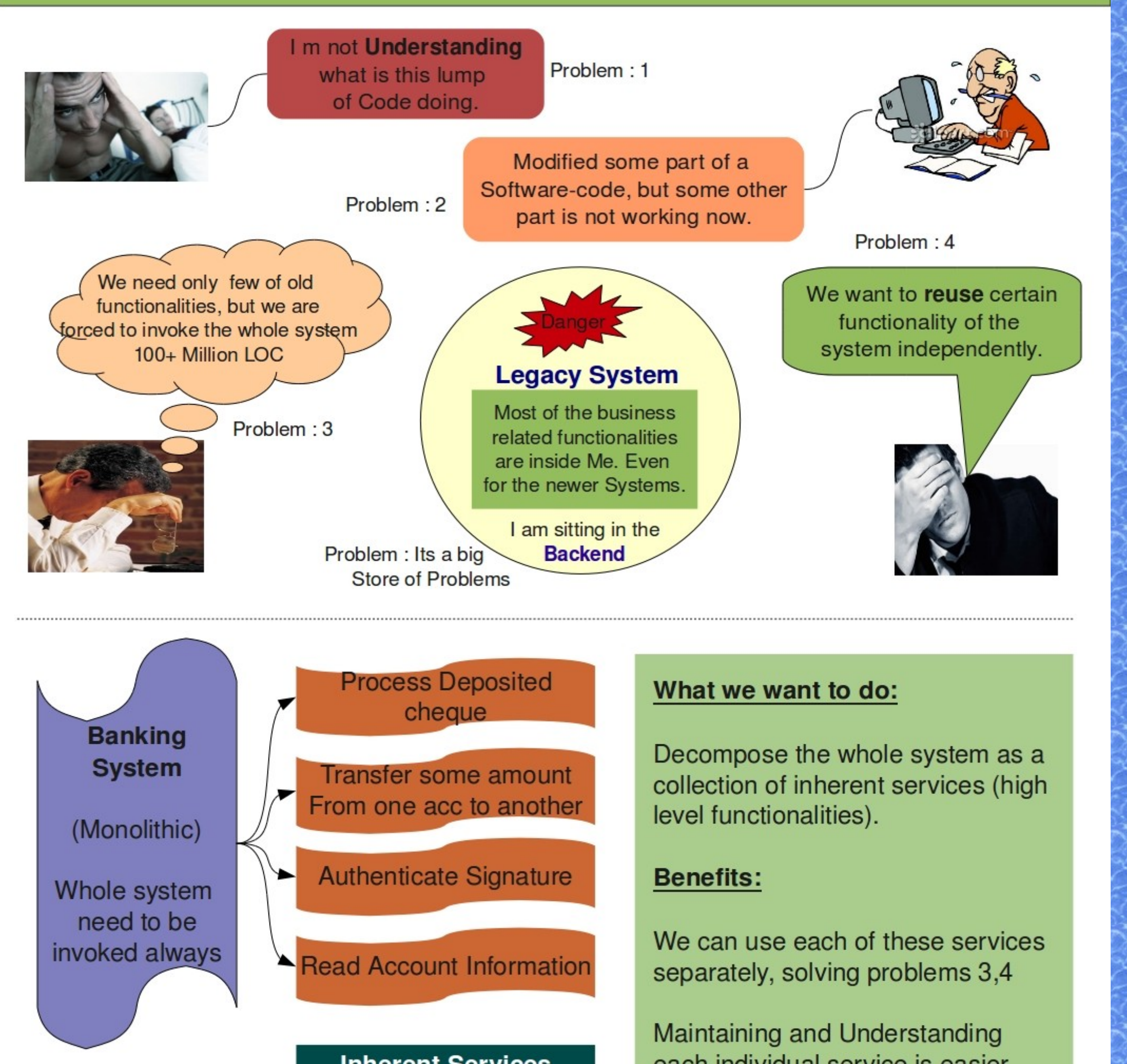 When previous three withdrawals were overdrafts.

Tragets for Testing Tool – Anomalies of the system:
- Withdrawal with amount = 0 is counted as normal withdrawal. So 6 overdrafts followed by 12 such withdrawals will allow further overdrafts.
- Balance of - Rs 39,000 will allow further overdrafts and balance can go below - Rs 40,000.

Finding null dereference bugs in Java Programs

```
y = false ;
if( x ) {
    y= true;
    r = null;
}
if( y )
    r.foo();
```

r.foo() will result in a null pointer exception

Millions of lines of code with null dereference bugs

Null dereference bug finder

Highly Scalable

Precise & Accurate

Good coverage

Analysis of Software Systems

I m not Understanding what is this lump of Code doing.
Problem : 1

Modified some part of a Software-code, but some other part is not working now.
Problem : 2

Problem : 4

We need only few of old functionalities, but we are forced to invoke the whole system 100+ Million LOC
Problem : 3

Legacy System
Most of the business related functionalities are inside Me. Even for the newer Systems.
I am sitting in the Backend

We want to reuse certain functionality of the system independently.

Problem : Its a big Store of Problems

Banking System
(Monolithic)
Whole system need to be invoked always

Process Deposited cheque
Transfer some amount From one acc to another.
Authenticate Signature
Read Account Information

Inherent Services

What we want to do:
Decompose the whole system as a collection of inherent services (high level functionalities).

Benefits:
We can use each of these services separately, solving problems 3,4

Maintaining and Understanding each individual service is easier.

Collaborations and Selected Projects

∗ A Theory of Conflict-Tolerant Features (with General Motors ISL, Bangalore)

∗ Constraint Temporal Logics (Indo-France project Timed-DISCOVERI)

∗ Model-Checking Information flow properties

∗ Java Memory Model aware Software Validation
(with Abhik Roychoudhury, National University of Singapore)

∗ Mining services from monolithic code (With Infosys)

∗ Converting batch programs to online programs (With IBM Research India)

∗ Inferring logical data models from legacy applications (With Microsoft Research India)

Selected Publications

∗ Conflict-Tolerant Features, *Deepak D'Souza and Madhu Gopinathan*, Int. Conference on Computer Aided Verification (CAV) 2008

∗ Conflict-Tolerant Real-Time Features, *Deepak D'Souza, Madhu Gopinathan, S. Ramesh, and Prahladavaradan Sampath*, Int. Conference on the Quantitative Evaluation of Systems (QEST) 2008

∗ Conflict-Tolerant Specifications in Temporal Logic, *Sumesh Divakaran, Deepak D'Souza and Raj Mohan M.*, India Software Engineering Conference (ISEC) 2010

∗ An Automata-Theoretic Approach to Constraint LTL, *Stéphane Demri and Deepak D'Souza*, Information and Computation (2007)

∗ Model-Checking Information Flow Properties, *Deepak D' Souza, Raveendra Holla, Raghavendra K. Ramesh and Barbara Sprick*, Journal of Computer Security (2010)

∗ Java Memory Model aware Software Validation, *Arnab De, Abhik Roychoudhury and Deepak D'Souza*, PASTE 2008 Workshop.

∗ Deep Packet Inspection Using Message Passing Networks, *Divya Jain, K.Vasanta Lakshmi and Priti Shankar*, Recent Advances in Intrusion Detection (RAID) 2008