History and Future of Software Architecture

A CS 718 Lecture

R K Joshi
IIT Bombay

The Early Culprit

The GOTO Statement

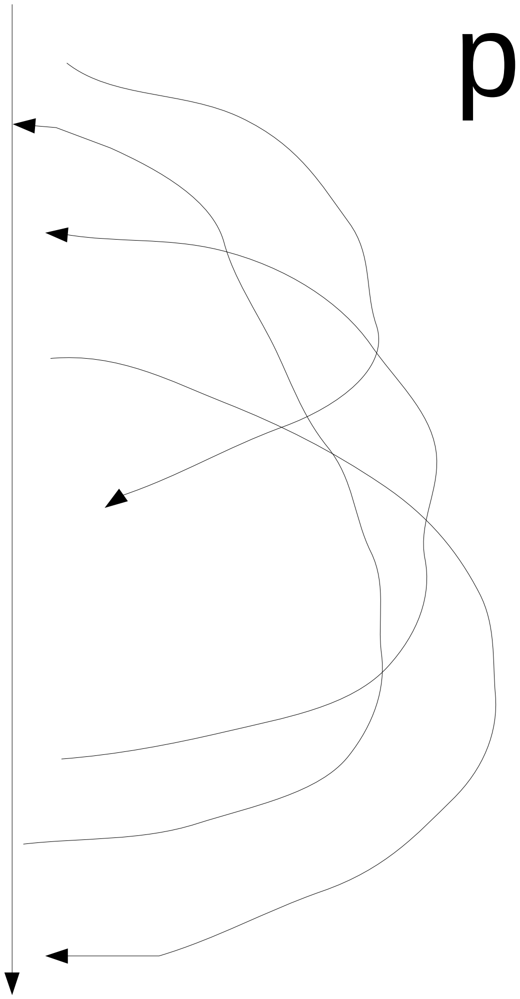
Indispensible Low Level Abstraction in Assembly Language or Machine Language

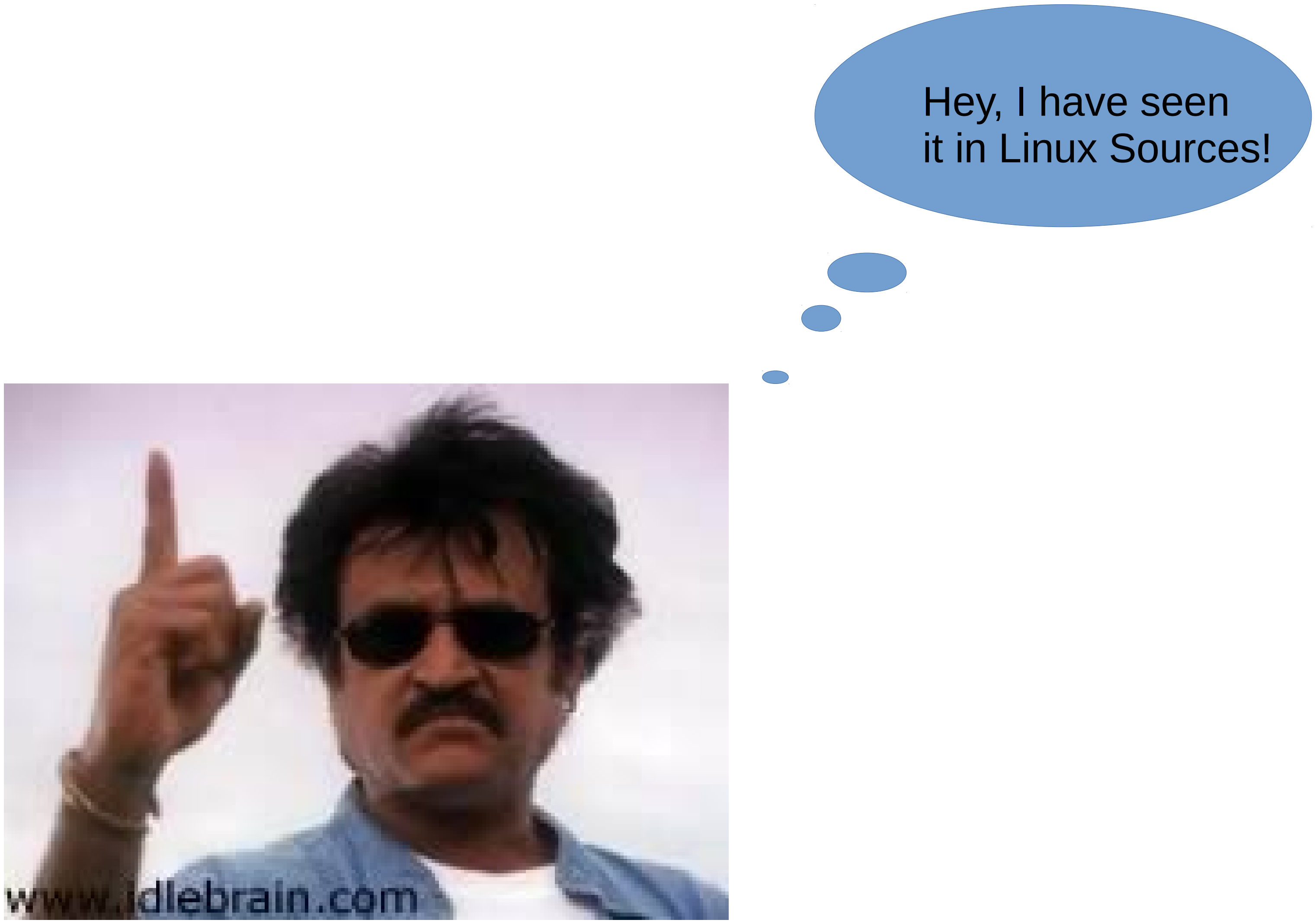
But Havoc

in

High Level Programming Languages

Ho w prog rams Lo ok ed




Hey, I have seen it in Linux Sources!
www.idlebrain.com

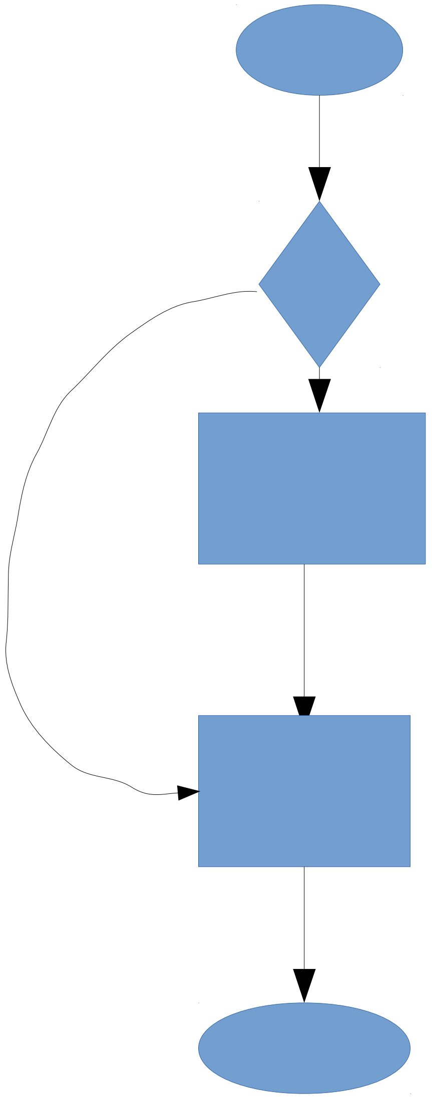
It's

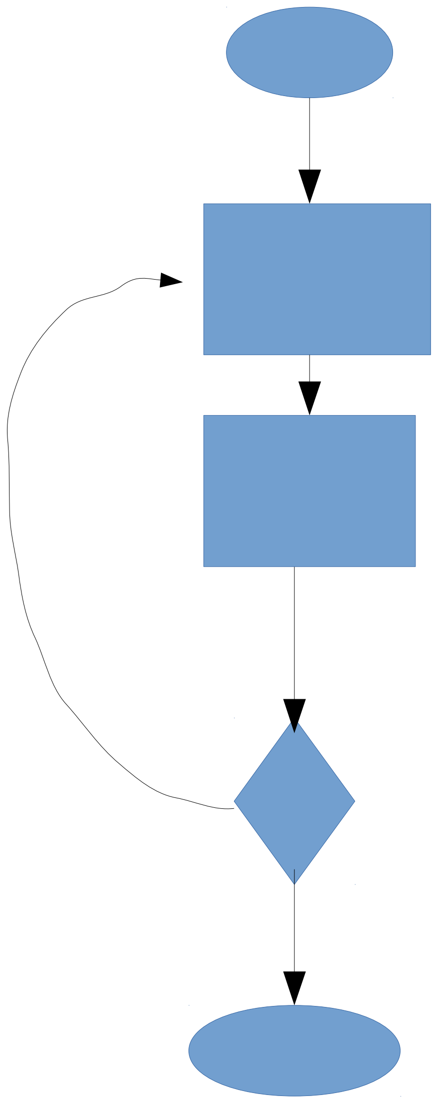
Solution

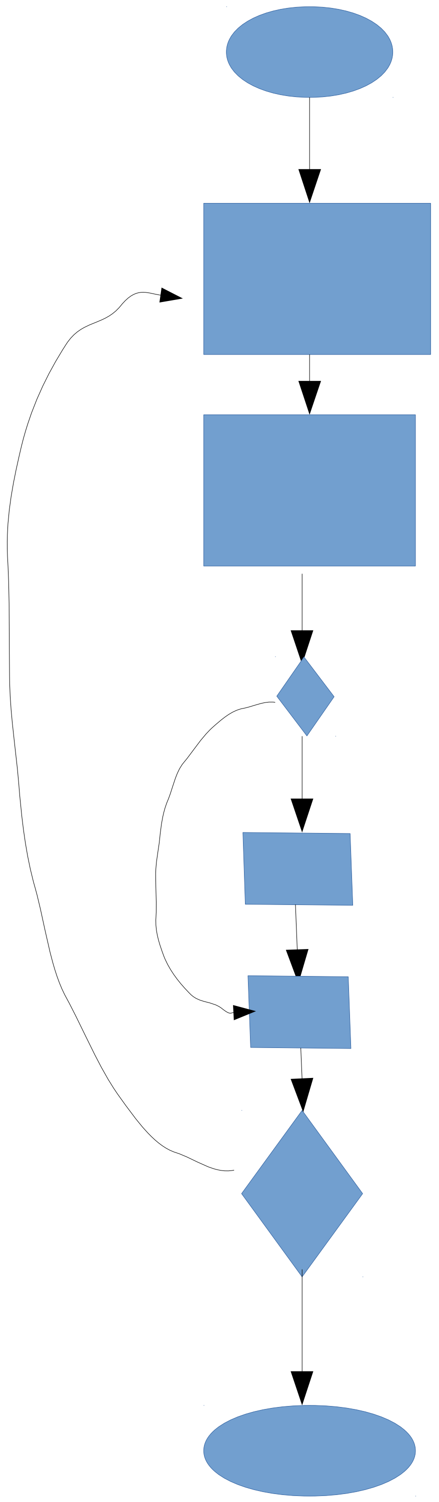
Was

Structured

Programming







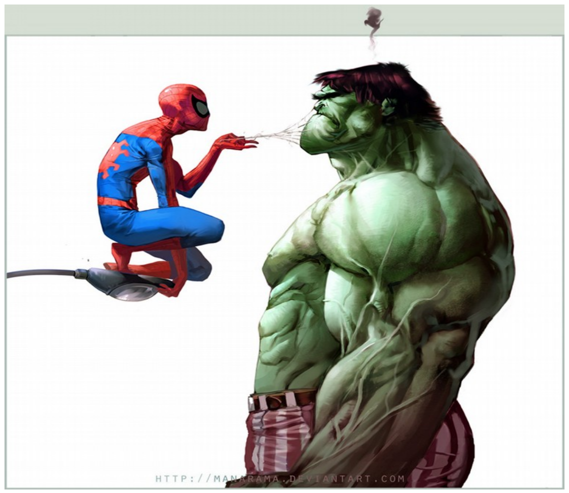
But the **Programs** got

bigger

and

bigger

New Abstractions were needed to Manage them



What Were they?

Functions, Procedures and Modules

Libraries

Classes and Objects

Interfaces and Components

Services

... and so on

The Birth and Growth of Methodologies

Structured Methodology
ER
DFD
Modularity
Cohesion
Coupling

and

the Waterfall Model

The Object Oriented Software Development Methodologies

Peter Coad - Patterns
Yourdon – OOA
Rebecca Wirfs Brock - OOD
Kent Beck and Cunningham's CRC
Booch- OOD
Jacobson- USE CASES
Rumbagh - OMT
...
The UML

Revolved around
Classes
Aggregation
Association
Inheritance
Membership

Then came a Major Contribution to Software Architecture

Design Patterns

Architecture Outgrew Products
into
Processes
and
Roles

Patterns in other fields of software design

Networking patterns

Distributed computing patterns

Parallelism patterns

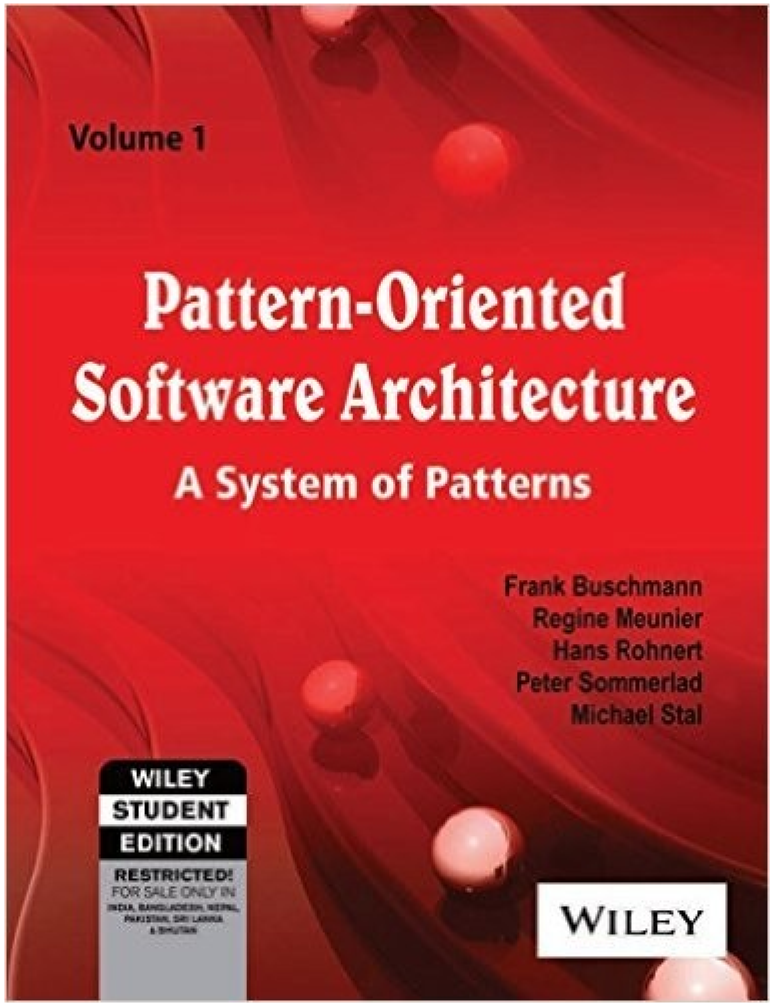
Documentation patterns

Coding patterns

Testing patterns

Requirements patterns

SA book from CMU, POSA



Volume 1

Pattern-Oriented Software Architecture

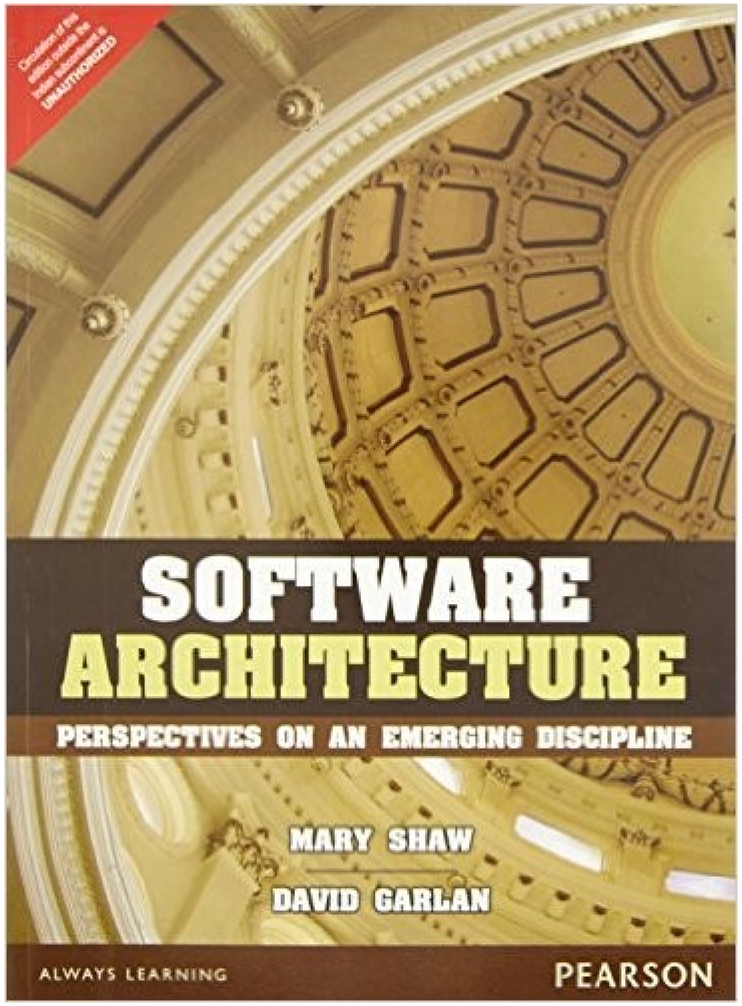
A System of Patterns

Frank Buschmann
Regine Meunier
Hans Rohnert
Peter Sommerlad
Michael Stal

WILEY STUDENT EDITION

RESTRICTED! FOR SALE ONLY IN INDIA, BANGLADESH, NEPAL, PAKISTAN, SRI LANKA & BHUTAN

WILEY



Circulation of this edition outside the Indian subcontinent is UNAUTHORIZED

SOFTWARE ARCHITECTURE

PERSPECTIVES ON AN EMERGING DISCIPLINE

MARY SHAW

DAVID GARLAN

ALWAYS LEARNING

PEARSON

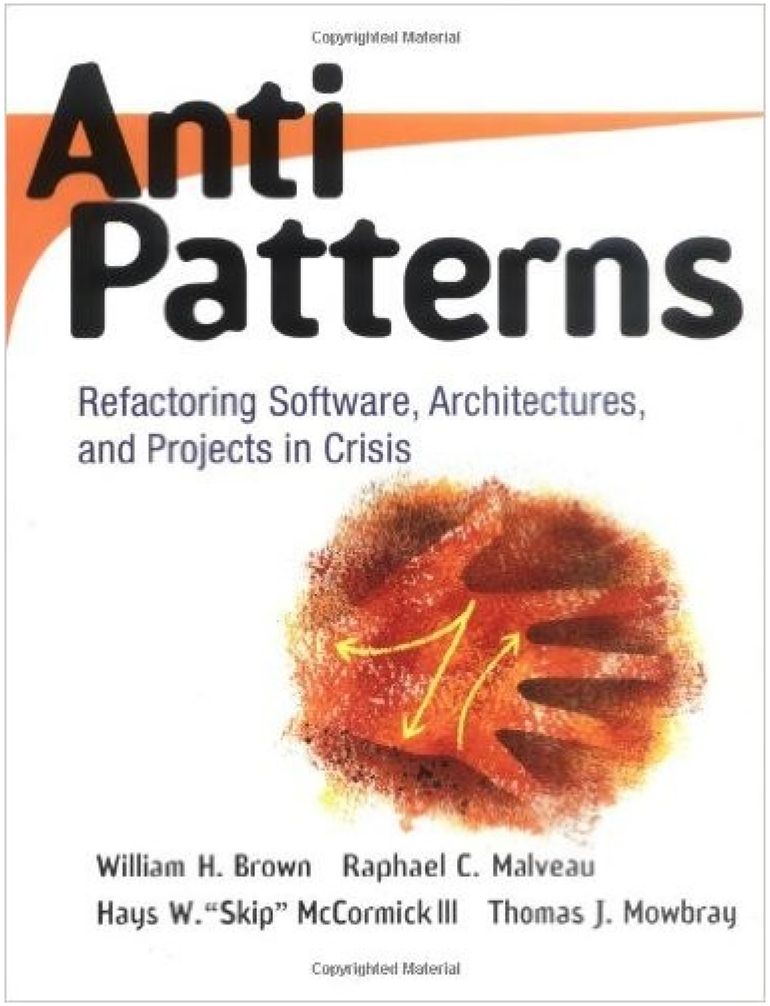
Anti-patterns

what is to be avoided

Refactoring patterns

the restructuring

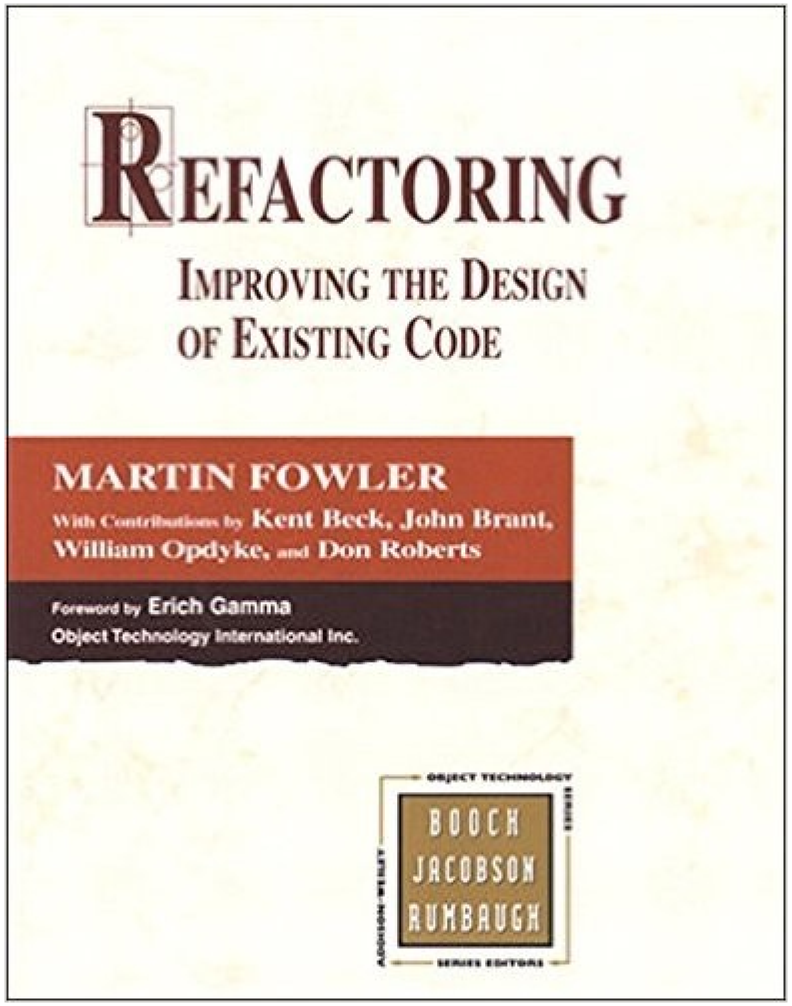
utility in on-the-fly software development (Agile methods)



AntiPatterns

Refactoring Software, Architectures, and Projects in Crisis

William H. Brown Raphael C. Malveau
Hays W. "Skip" McCormick III Thomas J. Mowbray


REFACTORING

IMPROVING THE DESIGN OF EXISTING CODE

MARTIN FOWLER
With Contributions by Kent Beck, John Brant, William Opdyke, and Don Roberts

Foreword by Erich Gamma
Object Technology International Inc.

OBJECT TECHNOLOGY SERIES
BOOCH JACOBSON RUMBAUGH
SERIES EDITORS
ADDISON-WESLEY

What really is
software architecture?

Then came
Zachmann's Model from IBM

Rule 1:

Columns have no order

Rule 2:

Each column has a simple, basic model

Rule 3:

Basic model of each column is unique

Rule 4:

Each row represents a distinct view

Rule 5:

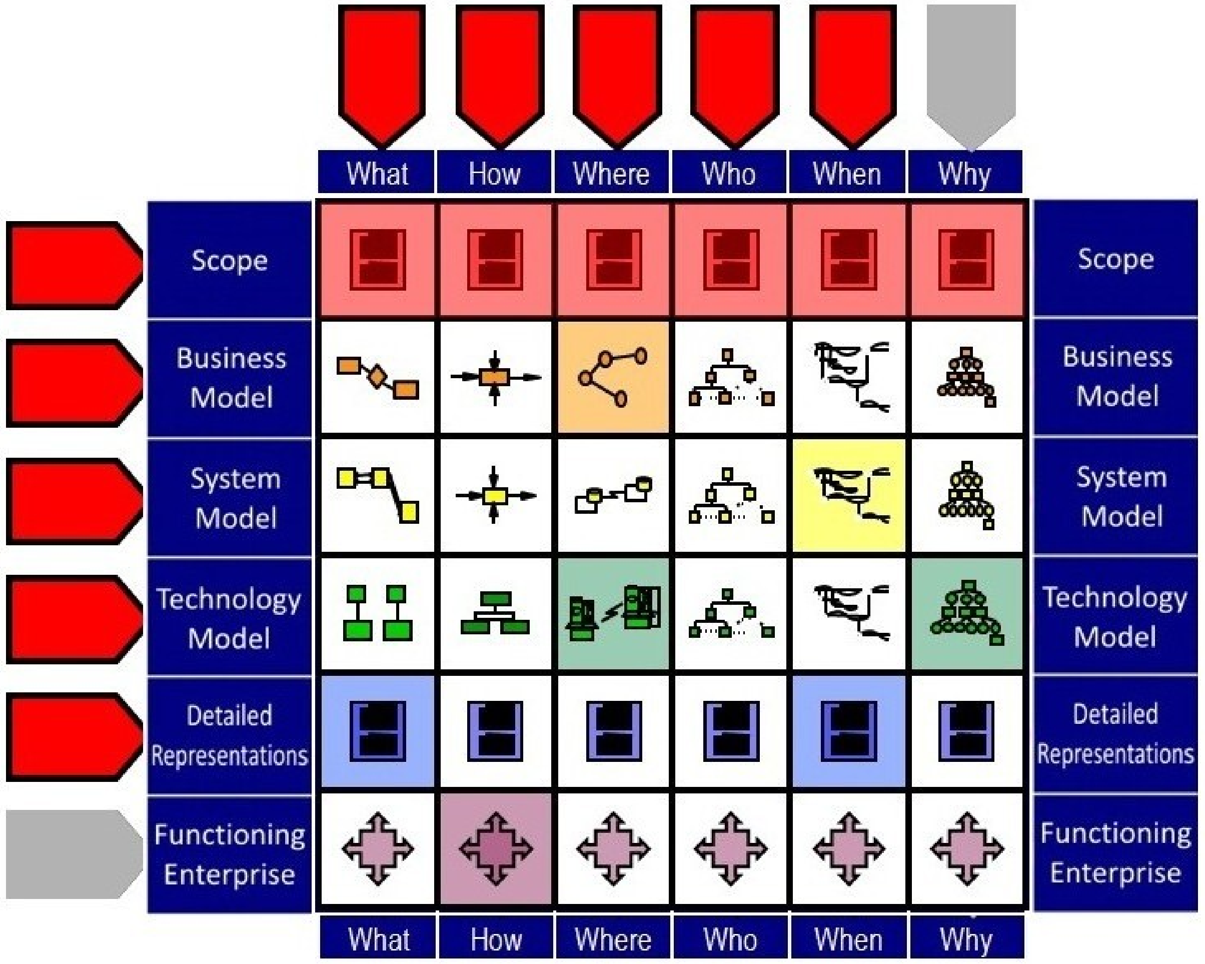
Each cell is unique

Rule 6:

Combining the cells in one row forms a complete description from that view

Rule 7:

The logic is recursive



Agents
Objetcts
Processes
Classes
Modules
Services
Components
Files
Functions
Calls

Processors
Machines
Networks
Connecting
Devices
Interfaces
Servers
Cloud
Databases

Distribution
Parallelism
Security
Availability
Fault
Tolerance
Response
Heterogneity
Speed
Look

Usability
Reusability
Adaptability
Configurability
Reconfigurability
Evolvability
Understadability
Tractability
Serviceability
...

Administrators
Programmers
Bug fixers
Owners
Developers
Bug Reporters..

Distribution
Parallelism
Security
Availability

How to you
conceptualize all this?

What models do you build?

Will it not soon go out of control if
you don't!

Some concerns addressed by

Architectural Patterns

Some concerns addressed by

Architectural Description Languages

Some concerns addressed by

Formal Specification Languages and Tools

Some concerns addressed by

Archiecture Evaluation Frameworks

Remember ..

What?

We are crossing the boundary of one a.out or one exe program

Programming in the Large

But the Bottom Line is ..

The Traditional Wisdom of Cohesion and Coupling

What shall we do in the next lecture?

A Formal Language called CCS (of Robin Milner) to express and build architectures

We learn next, how to express:

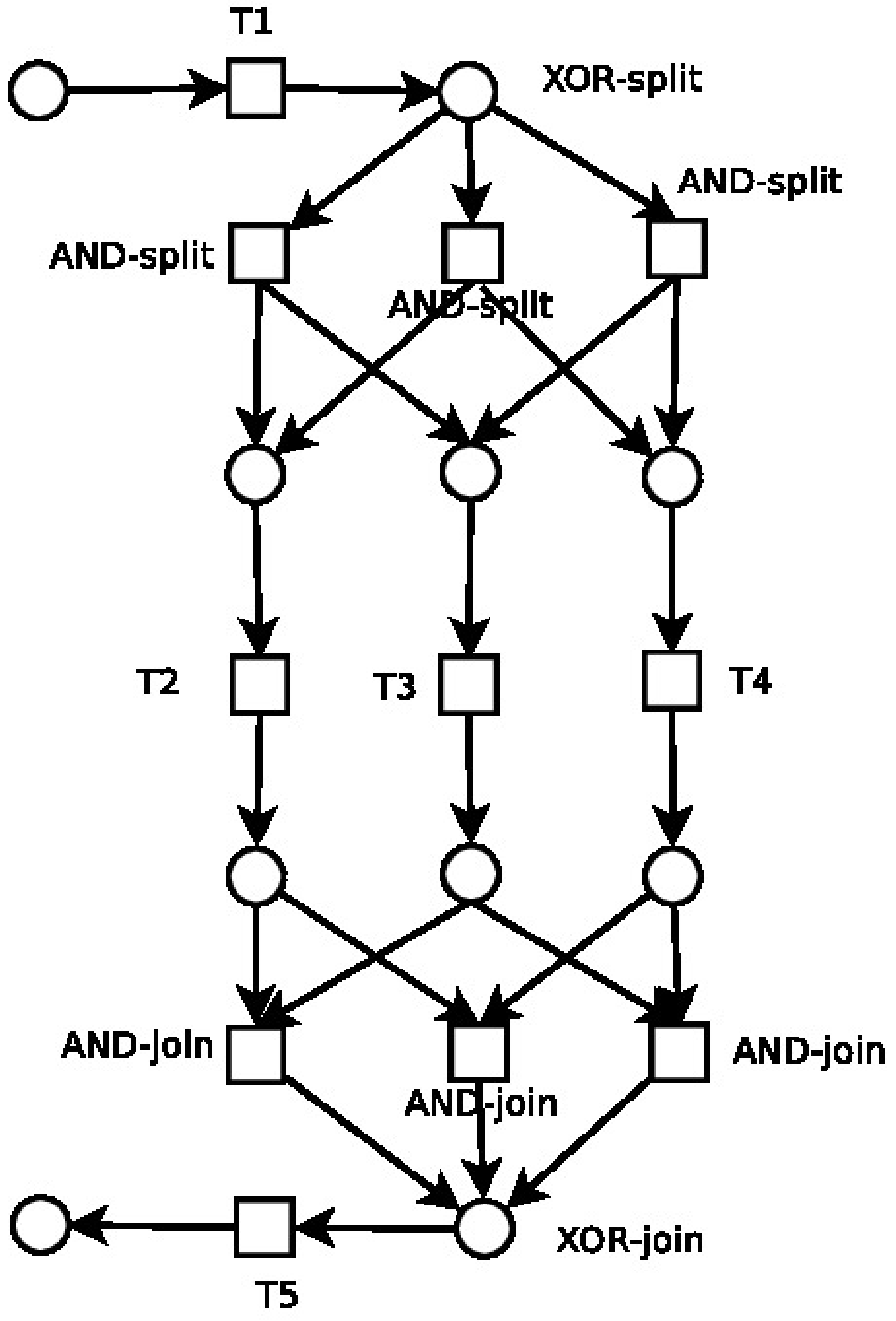
Component behavior

Components and Connectors

Non-determinism

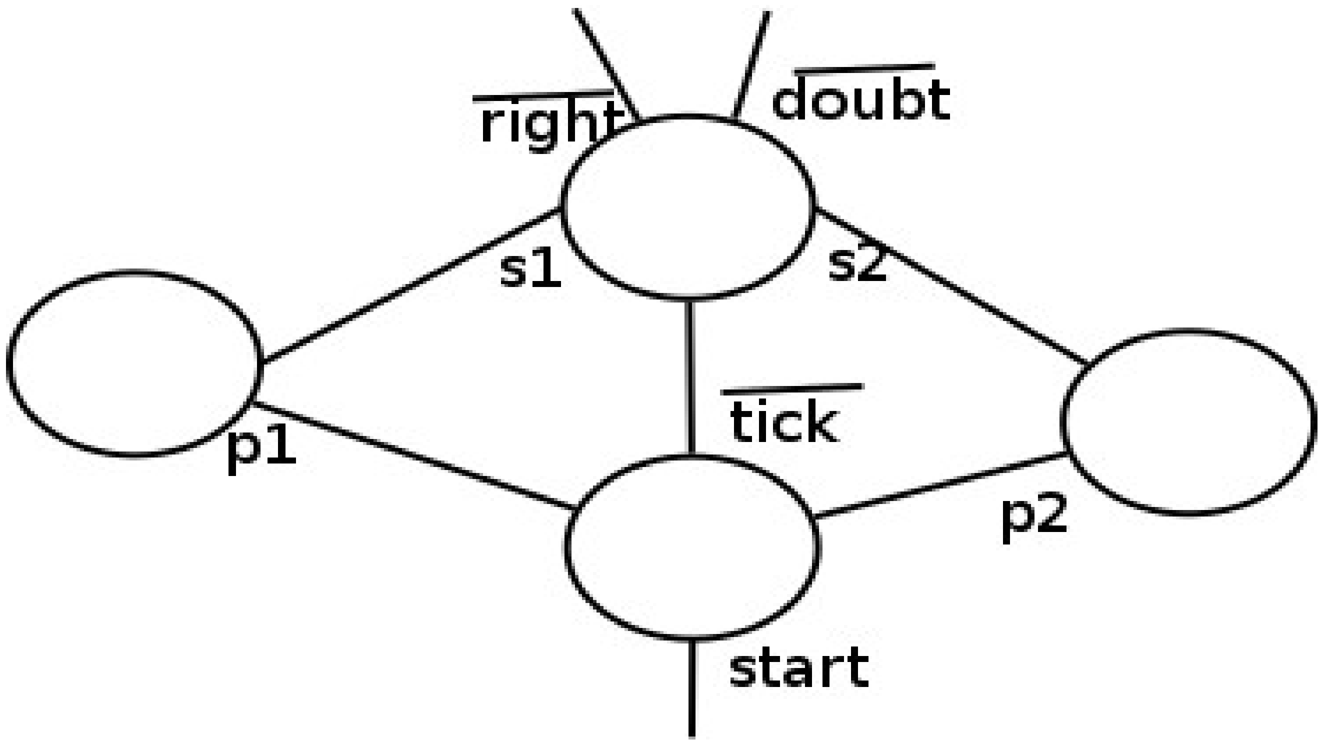
Composition of smaller components into bigger architectures

A Few Examples

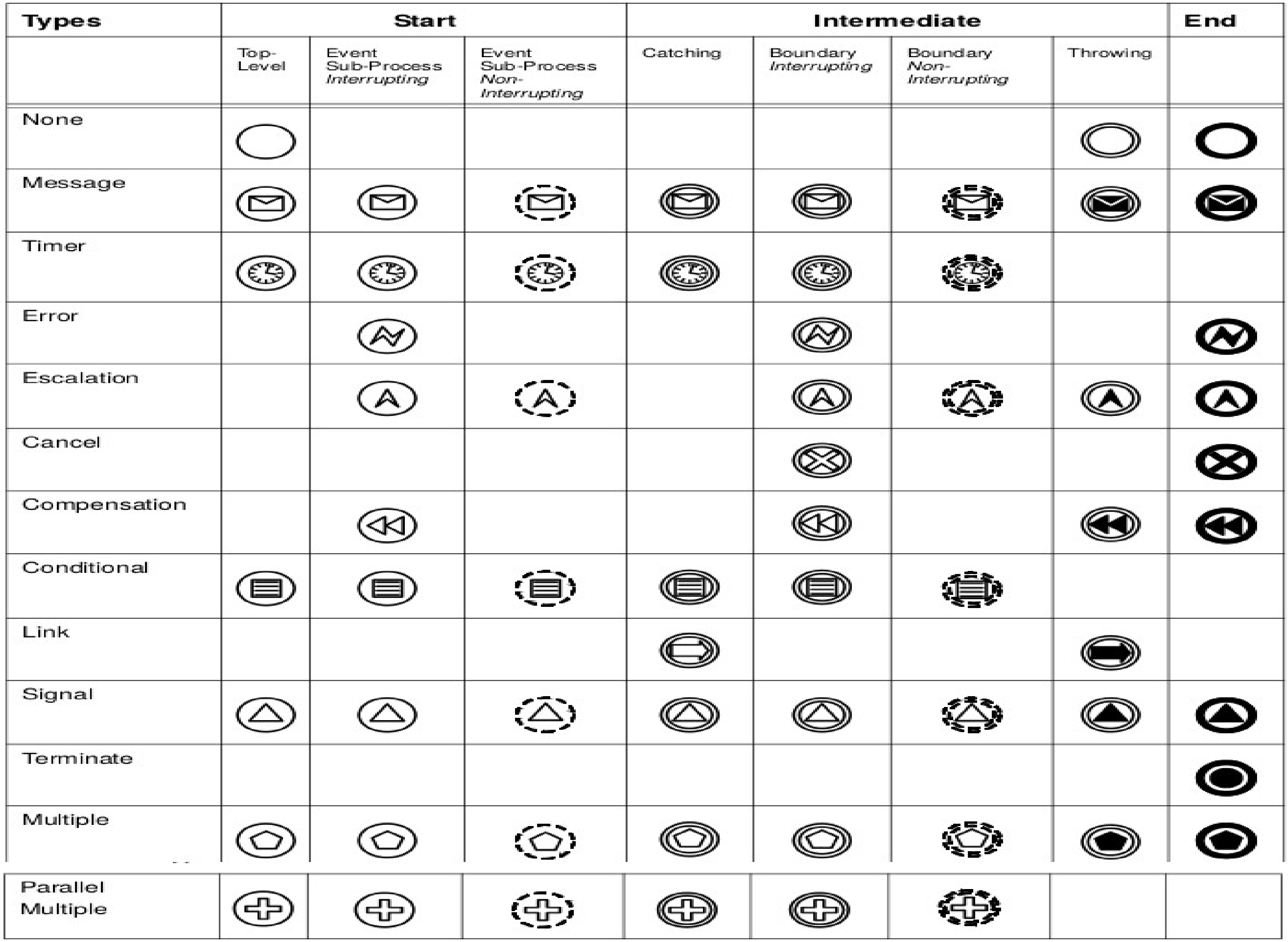


2/3 split-join

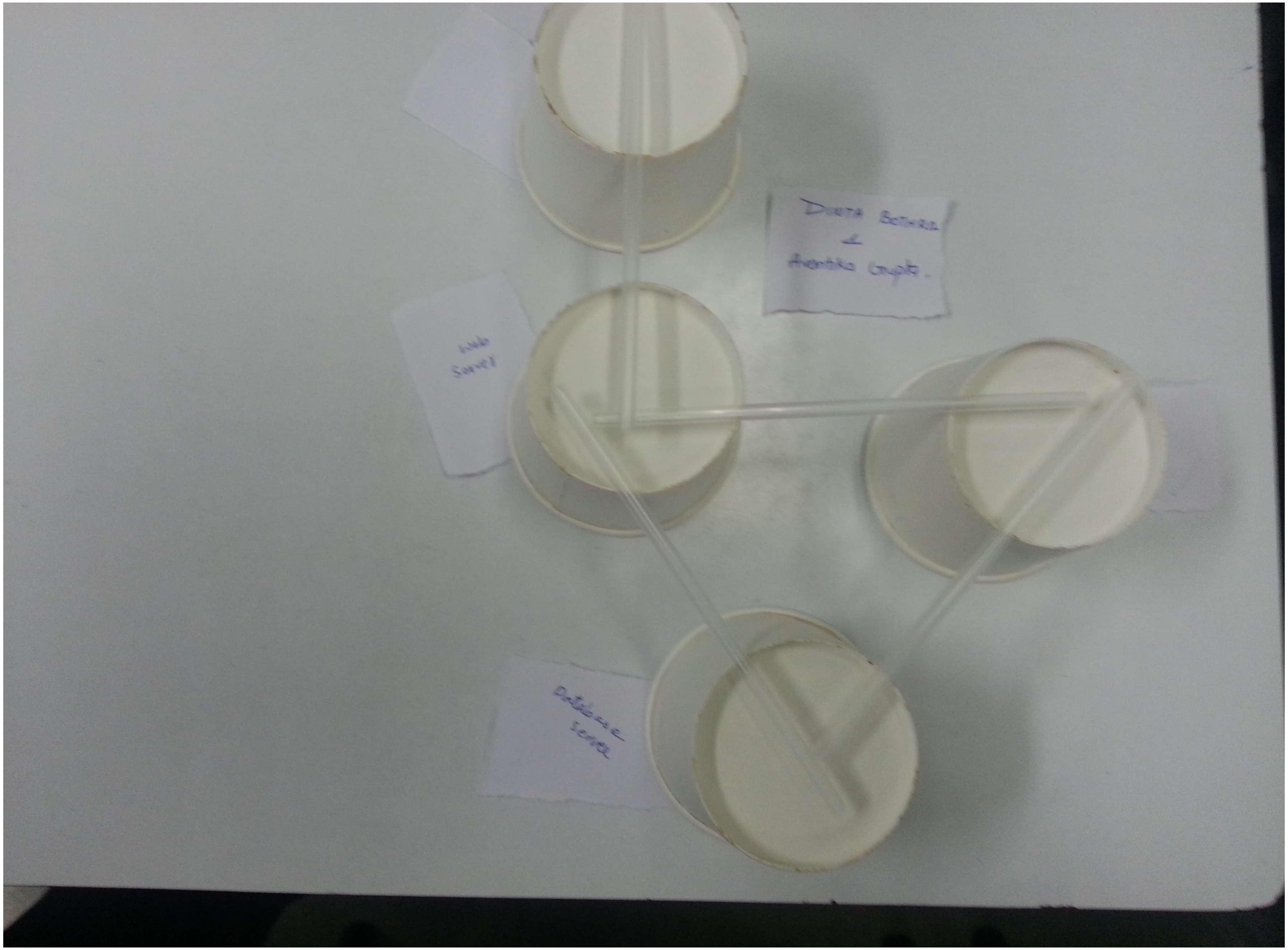
When the start signal arrives, the bottom agent sends two instances of the same problem to two agents. The left agent and the right agent solve the problem that they receive, and output their solutions to the top agent. The top agent compares the two solutions received from the two solvers and then outputs on either of two ports (right, doubt) indicating whether the solutions compared. However, the top agent acts only after it receives a tick from the bottom agent. The bottom agent ticks the top agent as soon as it sends the problem instances to left and right agents.

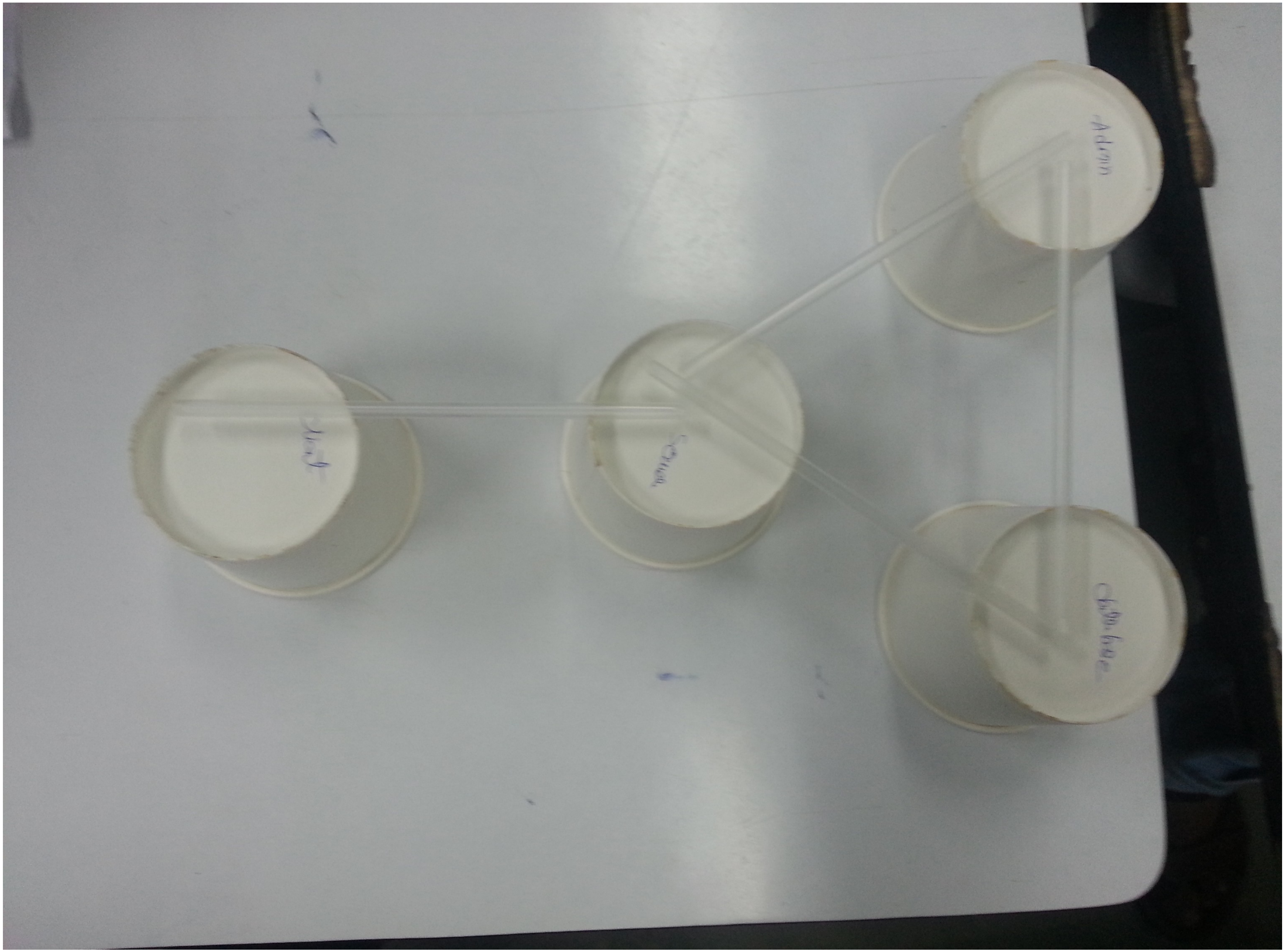


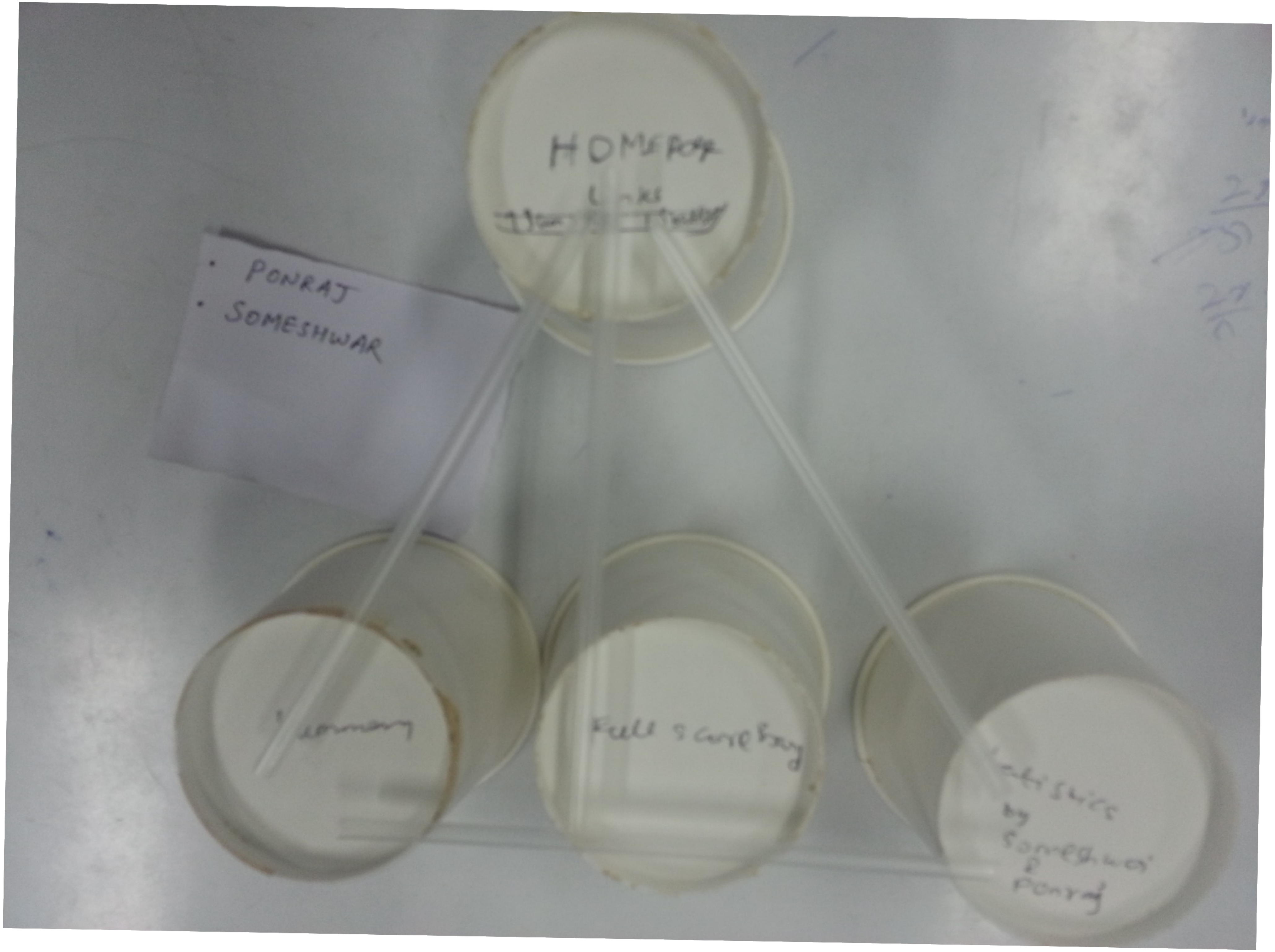
Events in BPMN - A Summary Sheet

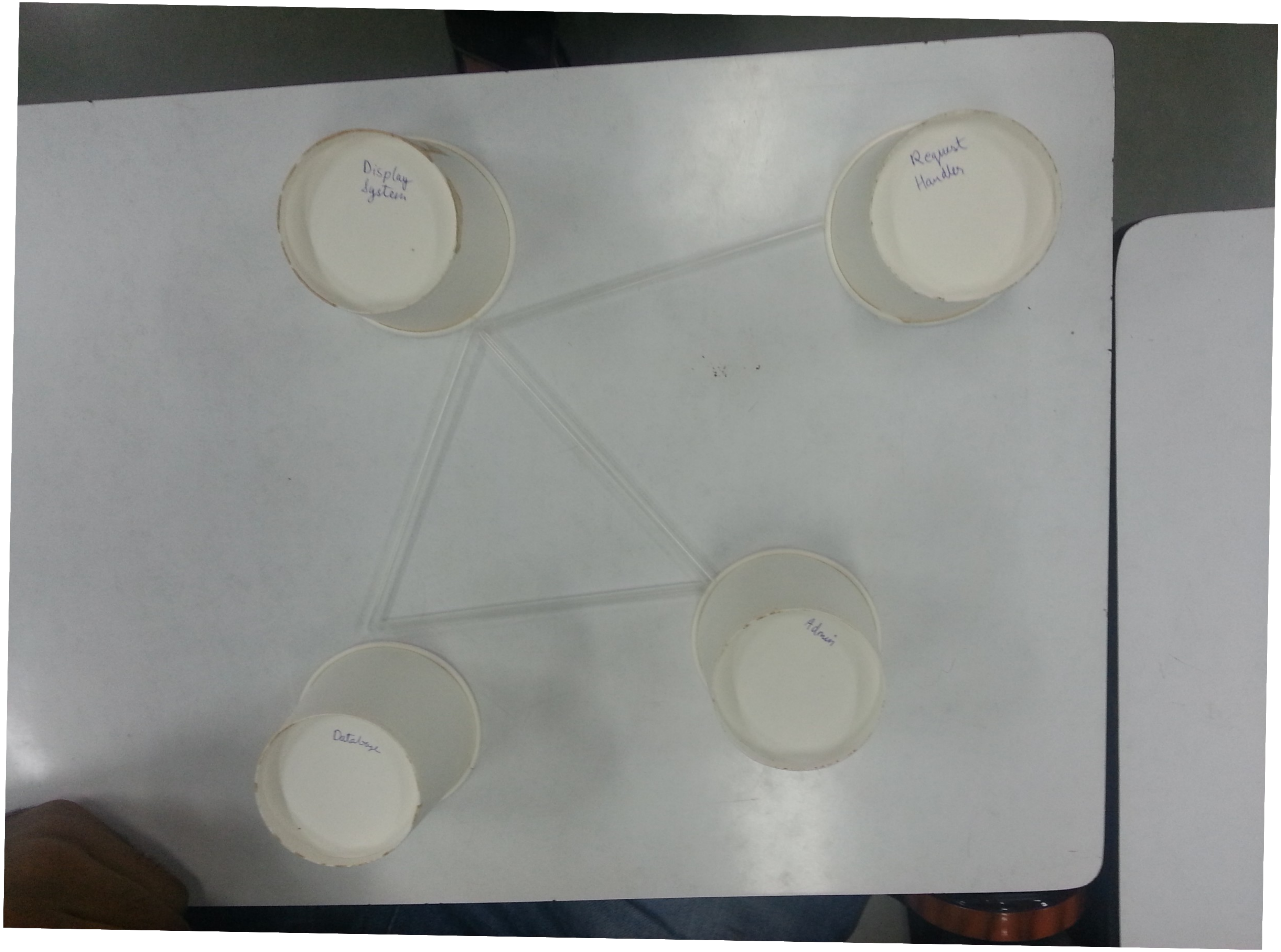


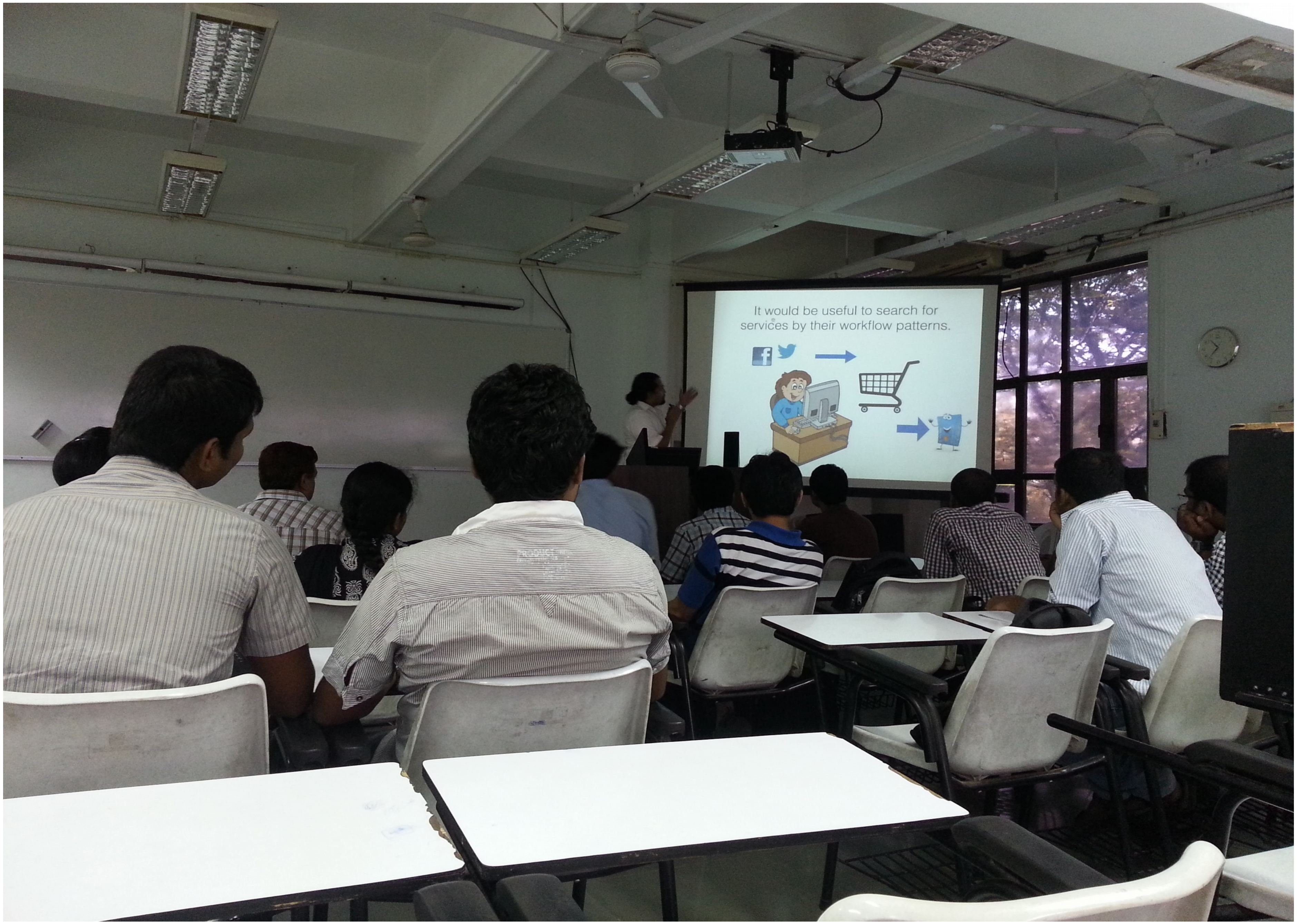
Types	**Start**			**Intermediate**				**End**
	Top-Level	Event Sub-Process *Interrupting*	Event Sub-Process *Non-Interrupting*	Catching	Boundary *Interrupting*	Boundary *Non-Interrupting*	Throwing	
None	○						◎	●
Message	✉	✉	✉	✉	✉	✉	✉	✉
Timer	🕒	🕒	🕒	🕒	🕒	🕒		
Error		⚡			⚡			⚡
Escalation		⌃	⌃		⌃	⌃	▲	▲
Cancel					⊗			⊗
Compensation		⏪			⏪		⏪	⏪
Conditional	☰	☰	☰	☰	☰	☰		
Link				⇨			➡	
Signal	△	△	△	△	△	△	▲	▲
Terminate								●
Multiple	⬠	⬠	⬠	⬠	⬠	⬠	⬟	⬟
Parallel Multiple	✚	✚	✚	✚	✚	✚		


Dinta Bothra
&
Avantika Gupta.
Web Server
Database Server


Adam


HOMEPAGE
Links
• PONRAJ
• SOMESHWAR
Summary
Feed & Category
Statistics
by
Someshwar
&
Ponraj


Display System
Request Handler
Admin
Database


It would be useful to search for services by their workflow patterns.

Base of Semantic Interpretation: Ontologies

- Means to formally model the structure of a system.
  - The relevant entities
  - Relations that emerge from its observation, and which are useful to our purposes.
- An example of such a system can be a company with all its employees and their interrelationships.
- The ontology engineer
  - analyzes relevant entities
  - organizes them into concepts and relations using predicates.

Dia
★ Diagram creation Program
★ Free
★ cross platform
★ Open source
★ Extensible
★ Intuitive UI
★ Sheets
★ (reasonably) stable
MESSI
10

What is IMS?

Instructional Management System (IMS) facilitates the creation of blended learning environments and streamlines online learning by enabling teachers, administrators, district staff and stakeholders to devise and share educational resources.

IMS provides a unified learning portal that enables schools and district to communicate, collaborate and share all of these crucial learning resources and more.