Domain Model Review

Problem  Domain  Program  Binary
Domain Model Review

• Observations / Issues
  – Models that are not Domain Models
  – Excessive details hide the concepts
  – Implementation, Roles, Attributes, Methods not discarded
  – Cardinality missing
  – Arrows instead of lines
• Go back the examples provided in prior lectures

Agile Planning
# CMMI for Development Model

## Maturity Levels

<table>
<thead>
<tr>
<th>Maturity</th>
<th>Process</th>
<th>Project</th>
<th>Engineering</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>• Organizational Performance Management</td>
<td>• Quantitative Project Management</td>
<td>• Causal Analysis and Resolution</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>• Organizational Process Performance</td>
<td>• Integrated Project Management</td>
<td>• Requirements Development</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Organizational Process Definition</td>
<td>• Risk Management</td>
<td>• Technical Solution</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Organizational Process Focus</td>
<td>• Requirements Development</td>
<td>• Product Integration</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Organizational Training</td>
<td>• Verification</td>
<td>• Decision Analysis and Resolution</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Requirements Management</td>
<td>• Validation</td>
<td>• Configuration Management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Project Planning</td>
<td>• Measurement and Analysis</td>
<td>• Process and Product Quality Assurance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Project Monitoring and Control</td>
<td>• Supplier Agreement Management</td>
<td>• Supplier Agreement Management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Supplier Agreement Management</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>• Requirements Management</td>
<td>• Project Planning</td>
<td>• Supplier Agreement Management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Project Monitoring and Control</td>
<td>• Verification</td>
<td>• Supplier Agreement Management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Supplier Agreement Management</td>
<td>• Validation</td>
<td>• Supplier Agreement Management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Configuration Management</td>
<td>• Process and Product Quality Assurance</td>
<td>• Supplier Agreement Management</td>
<td></td>
</tr>
</tbody>
</table>

---

### Scrum

Scrum is a framework for managing and planning the development of a product. It is based on a set of core concepts and techniques that are intended to provide a flexible and adaptable approach to software development. Scrum is an Agile methodology that promotes collaboration, accountability, and transparency.

1. **1 Scrum Team**
2. **Sprint Review**
3. **Daily Scrum**
4. **Sprint**
5. **Sprint Retrospective**

**Key Components of Scrum**

- **Sprint Backlog**: A list of tasks to be completed during the sprint.
- **Product Backlog**: A prioritized list of features or requirements that the product needs to be developed.
- **Scrum Team**: A cross-functional team that works together to deliver the product.

Scrum Framework © Scrum.org


[https://www.scrum.org/](https://www.scrum.org/)
Design fills the gap

- Problem
- Domain
- Design
- Program
- Binary

Design

A description of
- the structure of the software to be implemented,
- the data models and structures used by the system,
- the interfaces between the components, and
- the algorithms used.
Architectural Design

- Identifies the main structural components and the relationships between them.
- First stage of design, providing a link between the requirements and design.
- Can be done at the individual program level or large scale systems that include many programs.
- Can be heavily affected by the non-functional requirements.

Architectural Patterns

- Means of reusing knowledge about generic system architectures.
- Describes the architecture, when it may be used, its advantages and disadvantages.
- Stylized, abstract description of good practice which has been tried and tested in different systems and environments.
Some Architectural Patterns

- Blackboard
- Broker
- Client Server
- Data Warehouse
- Event Driven
- Extract-Transform-Load
- Layers
- Microkernel

- Microservices
- Model-View-Controller
- Online Transaction Processing
- Peer-to-Peer
- Pipe and Filter
- Repository
- Service Oriented

Model View Presenter

<table>
<thead>
<tr>
<th>Model</th>
<th>Presenter</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>updates model</td>
<td>updates view</td>
<td></td>
</tr>
<tr>
<td>change events</td>
<td>user events</td>
<td></td>
</tr>
</tbody>
</table>
MVP Hello World!

```
public class Model {
    private String message;
    public Model() {
        message = "Hello World!";
    }
    public String getMessage() {
        return message;
    }
}

public class View {
    public View() {
    }
    public void display(String text) {
        System.out.println(text);
    }
}

public class Presenter {
    private View view;
    private Model model;
    public Presenter(View view, Model model) {
        this.view = view;
        this.model = model;
    }
    public void start() {
        String result = model.getMessage();
        view.display(result);
    }
}

public class Main {
    public static void main(String[] args) {
        // argument handling?
        Model model = new Model();
        View view = new View();
        Presenter presenter = new Presenter(view, model);
        presenter.start();
    }
}
```
TripCo Rules

- Each team member is responsible for only one part of the system for Sprint 1 (your choice)
  - Model
  - View
  - Presenter/Main
- Changes to that class and its supporting classes must be made only by the team member that owns it.

Sprint 1 Test Data Activity

- Individual assignment on Canvas due Thursday noon.
  - Contribute a line of test data.
  - Practice good GitHub/Git etiquette.
- What happens when 88 people try to modify the same file in a relatively short period of time?
- How do you handle merge conflicts?
  - git pull before you commit/push on your local system
  - create pull request on GitHub after your push
  - https://help.github.com/articles/addressing-merge-conflicts/