Abstract Machine for Software Process Models

- Finite State Machine for SPM
  - Waterfall
  - Incremental
  - Spiral
  - Extreme Programming (XP)
  - Scrum

- Generalized Abstract Machine for SPM
  - Problem set: objectives with colors
  - Operators customized for SPM
  - Cycles customized for each model
# Waterfall

<table>
<thead>
<tr>
<th>Current State</th>
<th>Input</th>
<th>Output</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement</td>
<td>User Requirements</td>
<td>System Requirement Doc</td>
<td>Design</td>
</tr>
<tr>
<td>Design</td>
<td>System Req Doc</td>
<td>Design &amp; Test Doc</td>
<td>Implementation</td>
</tr>
<tr>
<td>Implementation</td>
<td>Design &amp; Test Doc</td>
<td>Revised Design &amp; Test Doc Software</td>
<td>Verification</td>
</tr>
<tr>
<td>Verification</td>
<td>Revised Design &amp; Test Doc</td>
<td>Test Report Verified Software</td>
<td>Deployment</td>
</tr>
<tr>
<td>Deployment</td>
<td>Verified Software</td>
<td>Manual &amp; Deployed Software</td>
<td>Maintenance</td>
</tr>
</tbody>
</table>

![Waterfall Diagram]

- User requirements → Requirement doc → Design Doc
- Test Doc → Revised Design & Test Doc Software → Test Report Verified Software → Deployed Software
- Manual & Deployed Software
## Incremental

<table>
<thead>
<tr>
<th>Current State</th>
<th>Input</th>
<th>Output</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>User Communication Evaluation Report</td>
<td>Prototype Objectives</td>
<td>Design</td>
</tr>
<tr>
<td>Design</td>
<td>Prototype Objectives</td>
<td>Prototype Functionalities</td>
<td>Implementation</td>
</tr>
<tr>
<td>Implementation</td>
<td>Prototype Functionalities Previous Prototype</td>
<td>Executable Prototype</td>
<td>Evaluation</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Executable Prototype</td>
<td>Evaluation Report</td>
<td>Analysis</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Executable Prototype</td>
<td>Released Software</td>
<td>Delivery</td>
</tr>
</tbody>
</table>

**Diagram:**
- **User requirements**
- **Prototype Objectives**
- **Prototype Features**
- **Executable Prototype**

**Flowchart:**
1. **Ana** → **Des** → **Imp** → **Eva** → **Del**
2. Evaluation Report
## Spiral

<table>
<thead>
<tr>
<th>Current State</th>
<th>Input</th>
<th>Output</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>User Communication</td>
<td>Objectives &amp; Constraints &amp; Plan</td>
<td>Risk Analysis</td>
</tr>
<tr>
<td>Risk Analysis</td>
<td>Objectives &amp; Constraints &amp; Plan</td>
<td>Risk Control</td>
<td>Development</td>
</tr>
<tr>
<td>Risk Analysis</td>
<td>Objectives &amp; Constraints &amp; Plan</td>
<td>Risk Analysis</td>
<td>Planning</td>
</tr>
<tr>
<td>Development</td>
<td>Objectives &amp; Constraints &amp; Plan</td>
<td>Current-level Product</td>
<td>Evaluation</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Current-level Product</td>
<td>Final Product</td>
<td>Release</td>
</tr>
</tbody>
</table>

### Diagram

- **Communication**
- **Objectives & Constraints**
- **Plan**
- **Risk Control**
- **Product**
- **Final Product**
- **Risk Analysis**
- **Evaluation Report**
Extreme Programming

<table>
<thead>
<tr>
<th>Current State</th>
<th>Input</th>
<th>Output</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement</td>
<td>Communication</td>
<td>User Stories</td>
<td>Planning</td>
</tr>
<tr>
<td>Planning</td>
<td>Communication &amp; User Stories</td>
<td>Tasks</td>
<td>Coding</td>
</tr>
<tr>
<td>Coding</td>
<td>Communication &amp; Timing Tasks &amp; Bug Report</td>
<td>Program Codes</td>
<td>Testing</td>
</tr>
<tr>
<td>Testing</td>
<td>Program Codes</td>
<td>Bug Report</td>
<td>Coding</td>
</tr>
<tr>
<td>Testing</td>
<td>Program Codes</td>
<td>Tested Software</td>
<td>Evaluation</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Communication &amp; Tested Software</td>
<td>Bugs from Users</td>
<td>Planning</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Communication &amp; Tested Software</td>
<td>New Requirements</td>
<td>Requirement</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Communication &amp; Tested Software</td>
<td>User Acceptance</td>
<td>Final Release</td>
</tr>
</tbody>
</table>

Communication: Req → Pla → Cod → Tes → Eva → Rel

User Stories: Req → Pla → Cod → Tes → Eva → Rel

Tasks: Req → Pla → Cod → Tes → Eva → Rel

Program Codes: Req → Pla → Cod → Tes → Eva → Rel

Communication: Req → Pla → Cod → Tes → Eva → Rel

Software:Req → Pla → Cod → Tes → Eva → Rel

Accepted Software: Eva → Rel

Bugs from Users: Req → Pla → Cod → Tes → Eva → Rel

Bug Report: Req → Pla → Cod → Tes → Eva → Rel

New Requirements: Req → Pla → Cod → Tes → Eva → Rel

Final Release
# Scrum

<table>
<thead>
<tr>
<th>Current State</th>
<th>Input</th>
<th>Output</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>Communication Product Backlog</td>
<td>Sprint Backlog</td>
<td>Sprint</td>
</tr>
<tr>
<td>Sprint</td>
<td>Sprint Backlog</td>
<td>Executable Version</td>
<td>Review</td>
</tr>
<tr>
<td>Sprint</td>
<td>Cancellation</td>
<td>Cancellation</td>
<td>Planning</td>
</tr>
<tr>
<td>Review</td>
<td>Executable Version Communication</td>
<td>Revised Product Backlog</td>
<td>Planning</td>
</tr>
<tr>
<td>Review</td>
<td>Executable Version Communication</td>
<td>Released Version</td>
<td>Final Release</td>
</tr>
</tbody>
</table>

![Scrum Process Diagram](chart.png)
Abstract Machine for Software Process Models

- **Generalized Abstract Machine**
  - \( M_{spm} = \{P, S, P_0, Cycles\} \), **starting from cycle_1**

- **Problem set**
  - A state that represents objectives in a certain style
    - Each objective can be assigned with a color. By default, color is NULL
    - Initial set usually includes real-world user-described requirements
    - Machine halts if **current set is empty** or **there’s no further operation**

- **Guard** [ \( a, b, \text{P\_checkpoint}, \text{constraint}, \text{P\_newInit} \) ]
  - At \( \text{P\_checkpoint} \) of cycle \( a \), if \( \text{constraint} \) is satisfied, transfer to cycle \( b \)
  - New cycle \( b \) will use \( \text{P\_newInit} \) as its initial set
Abstract Machine for Software Process Models

• Operators
  – Abstract
    • Translate non-green objectives into software-engineering requirements
    • Enumerate ( -abst< ): Color as white
    • Select ( =abst= ): **Generate new objectives from selected objectives.** Color as white
  – Design
    • Functionalize non-green objectives to module-level or function-level objectives
    • Enumerate ( -desi< ): Color as white
    • Select ( =desi= ): Color as white
  – Implement
    • Implement white/red objectives to real product-level objectives
    • Select ( =impl= ): Color as yellow
Abstract Machine for Software Process Models

- **Operators cont.**
  - **Test**
    - Validate yellow objectives
    - Select (=test=): Color as green/red
  - **Adjust**
    - Modify objectives
    - Delete + adapt + add (=+adju=): Delete, modify and add objectives. Color as white/red
  - **Deliver**
    - Delivery green objectives
    - Propagate (=deli+ )
Waterfall

- **Cycle_1**: guard[1, 2, P2, NULL, P2]
  - P₀  -**abst**< P₁ -**desi**< P₂
  - Requirement analysis and design for the whole software

- **Cycle_2**: guard[2, 2, P2, has_non-green, P2]
  - P₀  =**impl**= P₁  =**test**= P₂  =**deli**+ P₃
  - Implementation, verification and deployment

*Halt when cycle_2 is done*
Incremental

- **Cycle_1**: guard\([1, 2, P_2, NULL, P_2]\)
  - \(P_0 = \text{abst} = P_1 \ - \text{desi} < P_2\)
  - Define current increment: a subset of the whole software

- **Cycle_2**: guard\([2, 2, P_2, \text{one\_non\_green}, P_2]\), guard\([2, 1, P_3, \text{all\_green}, NULL]\)
  - \(P_0 = \text{impl} = P_1 = \text{test} = P_2 = \text{deli} + P_3\)
  - Go to next increment after finish current one

Halt when no more new requirement: \(P_0\) in cycle_1 is empty.
Spiral

- **Cycle_1**: guard[1, 1, P3, one_red, NULL], guard[1, 2, P3, no_red, P1]
  - P0 =\texttt{abst}\ = P1 =\texttt{impl}\ = P2 >\texttt{adju}\ = P3
  - Evaluate risk based on a simple prototype

- **Cycle_2**: guard[2, 1, P5, all_green, NULL]
  - P0 =\texttt{abst}\ < P1 =\texttt{desi}\ < P2 =\texttt{impl}\ = P3 =\texttt{test}\ = P4 =\texttt{deli}\ + P5
  - Detailed design and implementation

Halt when no more new requirement: P0 in cycle_1 is empty.
XP

• **Cycle_1**: guard[1, 2, P2, NULL, P2]
  - $P_0 = \text{abst}= P_1 = \text{desi}= P_2$
  - Choose user stories and tasks. Simplify design

• **Cycle_2**: guard[2, 2, P2, **hours**, P2], guard[2, 3, P3, **days**, P3],
  - $P_0 >+ \text{adju}= P_1 = \text{impl}= P_2 = \text{test}= P_3$
  - Implement the chosen user story with time constraint. Embrace change

• **Cycle_3**: guard[3, 1, P2, non-empty, P2]
  - $P_0 = \text{deli}+ P_1 >+ \text{adju}= P_2$
  - Release current increment. Adjust user stories.

Halt when no more new user stories: $P_2$ in cycle_3 is empty.
Scrum

- **Cycle_1**: guard[1, 2, P2, NULL, P2]
  - \( P_0 = \text{abst} = P_1 = \text{ desi} = P_2 \)
  - Define sprint backlog

- **Cycle_2**: guard[2, 2, P2, \text{day}, P2], guard[2, 3, P2, \text{Weeks}, P2]
  - \( P_0 = \text{impl} = P_1 = \text{ test} = P_2 \)
  - Perform a sprint. No requirement change inside a sprint.

- **Cycle_3**: guard[3, 1, P3, non-empty, P3]
  - \( P_0 >+ \text{adju} = P_1 = \text{deli} + P_2 >+ \text{adju} = P_3 \)
  - Review, retrospective, and release. Adjust for next sprint.

Halt when no more new backlogs: P3 in cycle_3 is empty.
Abstract Machine for Software Process Models

- Finite State Machine for SPM
  - Waterfall
  - Incremental
  - Spiral
  - Extreme Programming (XP)
  - Scrum

- Generalized Abstract Machine for SPM
  - Problem set: objectives with colors
  - Operators customized for SPM
  - Cycles customized for each model

\[ M_{spm} = \{P, S, P_0, Cycles\} \]

Cycle_1: guard[1, 1, P3, one_red, NULL], guard[1, 2, P3, no_red, P1]
- \( P_0 = \text{abst} = P_1 = \text{impl} = P_2 = \text{adj = } P_3 \)
- Evaluate risk based on a simple prototype

Cycle_2: guard[2, 1, P5, all_green, NULL]
- \( P_0 = \text{abst} < P_3 = \text{desi} < P_2 = \text{impl} = P_3 = \text{test} = P_4 = \text{deli} = P_5 \)
- Detailed design and implementation