EXTENDING ESSENCE KERNEL TO ENACT PRACTICES AT THE LEVEL OF SOFTWARE MODULES

Alper Tolga Kocatas and Cengiz Erbas
ASELSAN Inc.
Ankara, Turkey
kocatas@aselsan.com.tr, cerbas@aselsan.com.tr
Motivation

- Software development: A quite tangled problem which depends on many parameters.
- It is hard to analyze aggregate effect of many parameters.
- Development of software using modules helps localize problems.
- Development of certain modules may require more discipline,
- Development of other modules may require more agility.
- This is more evident in large-scale software development.
**Motivation**

- If for different modules of a software project, different processes are followed, we call this mode of development as "*heterogeneous development*".

- Main point of this study is to extend Essence kernel:
  - To support development of software in modules
  - To be able to use different development practices for each module.
Outline

- The Need for Heterogeneous Development
- Extending Essence Kernel to Support Modularization
- Extending Essence Kernel to Support Heterogeneous Development
- Future Work and Discussion
The Need For Heterogeneous Development

- Erbas and Erbas (GTSE 2013) leveraged transaction cost economics to lay foundations for a theory of software engineering.
- Study mapped various concepts from microeconomics to software engineering.
# The Need For Heterogeneous Development

<table>
<thead>
<tr>
<th>Economics Term</th>
<th>Software Engineering Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>Governance Structure: Top down vs Bottom up</td>
<td>Development approach: Plan driven vs agile</td>
</tr>
<tr>
<td>Asset specificity: High vs Low</td>
<td>High vs low Dependence to: platform, domain, tool, language or interfaces.</td>
</tr>
<tr>
<td>Uncertainty: High vs Low</td>
<td>Low: Projects where requirements can be specified up front, High: Projects where requirements are uncertain</td>
</tr>
</tbody>
</table>
The Need For Heterogeneous Development

- Result: Low uncertainty and asset specificity imply plan driven development. High values for both parameters imply agile development.

<table>
<thead>
<tr>
<th>Asset Specificity</th>
<th>Uncertainty</th>
<th>(Governance Structure) Development Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>High</td>
<td>Bottom-Up (Agile)</td>
</tr>
<tr>
<td>Low</td>
<td>Low</td>
<td>Top-down (Plan driven)</td>
</tr>
</tbody>
</table>

- Different modules in a software project may have different asset specificity and uncertainty values.
- Therefore, different modules may need different development approaches.
Outline

- The Need for Heterogeneous Development
- Extending Essence Kernel to Support Modularization
- Extending Essence Kernel to Support Heterogeneous Development
- Future Work and Discussion
EXtending TO Support MODularization

- Affected Essence alphas:
  - Solution area of concern:
    - Software System,
    - Requirements,
  - Endeavor area of concern:
    - Team,
    - Work
Extending TO Support MOdularization

- In order to add monitoring and tracking support for modules:
  - We add sub-alpha called *software module*.
  - We add an alpha-containment relation between the software system alpha and the software module sub-alpha.
EXTending TO Support MOdularization

- Newly added software module sub-alpha will have its own states,
- States of software module sub alpha will drive state of software alpha.
- For example: state of software system alpha can only make a transition from “usable” to “ready” only if the states of its module sub-alphas are past beyond a particular state.
EXtending TO Support MOdularization

- Same way of extension is also applied to requirements alpha and other affected alphas (team and work):
Outline

- The Need for Heterogeneous Development
- Extending Essence Kernel to Support Modularization
- Extending Essence Kernel to Support Heterogeneous Development
- Future Work and Discussion
EXtending TO Support HEterogeneous DEvelopment

- Additionally, we also want to apply different development approaches for different software modules.
- Application of a plan-driven vs agile methodology would need different support in the kernel.
EXtending TO Support HETerogeneous DEvelopment

- Extension mechanism: Using “extension item” which is specified in the Essence kernel and language specification.

- We add two additional sub-alphas to each affected alphas for top-down and bottom-up development approaches.
Extending TO Support HETerogeneous Development

- Top-down governance and bottom-up governance sub-alphas will also have their own states
- Just like the module sub-alphas which they extend.
- Same pattern is applied to alphas:
  - Requirements,
  - Team,
  - Work


**COnsistency CHecking MEchanism**

- We also add constraints to support the following facts from the previous work:

<table>
<thead>
<tr>
<th>Asset Specificity</th>
<th>Uncertainty</th>
<th>(Governance Structure) Development Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>High</td>
<td>Bottom-Up (Agile)</td>
</tr>
<tr>
<td>Low</td>
<td>Low</td>
<td>Top-down (Plan driven)</td>
</tr>
</tbody>
</table>

- Tags “AssetSpecificity” and “Uncertainty” are added to module sub-alphas (i.e. Software module, module requirements, etc.)
**COnsistency CHecking MEchanism**

- We use Object constraint language (OCL) to check if chosen development approaches are compatible with modules:
- We add the following OCL statements to the module sub-alphas:

  **Sub-alpha: Top-Down Governance**
  **Invariant:**
  
  \[
  (\text{self.Uncertainty} = \text{Low} \land \text{self.AssetSpecificity} = \text{Low})
  \]

  **Sub-alpha: Bottom-Up Governance**
  **Invariant:**
  
  \[
  \text{not (self.Uncertainty} = \text{Low} \land \text{self.AssetSpecificity} = \text{Low})
  \]
**COnsistency CHecking MEkhanism**

- Added OCL statements will ensure that incompatible development approaches will not be selected for modules having specific asset specificity and uncertainty tag values.

**Sub-alpha: Top-Down Governance**

**Invariant:**

\[(self.Uncertainty = \text{Low} \land self.AssetSpecificity = \text{Low})\]

**Sub-alpha: Bottom-Up Governance**

**Invariant:**

\[\neg (self.Uncertainty = \text{Low} \land self.AssetSpecificity = \text{Low})\]
After adding support for modules:

After adding support for heterogeneous development + OCL Statements
Outline

- The Need for Heterogeneous Development
- Extending Essence Kernel to Support Modularization
- Extending Essence Kernel to Support Heterogeneous Development
- Future Work and Discussion
FUTURE WORK AND DISCUSSION

- Presented an extension to the Essence kernel, for adding support for development of software, where:
  - Each module can be tracked by different alphas,
  - Each module can follow different type of development approaches (i.e. plan driven or agile)

- Adding these extensions solely will not result in an actionable kernel, so as a follow up work:
  - States and checkpoints of the added sub-alphas should be determined,
  - Formal guidance statements should be added to determine how states of sub-alphas will drive states of their parent alphas.
**Future Work and Discussion**

- Extension presented here is quite *invasive* and *cross-cutting*.
- Same extension pattern is applied to all of the affected alphas.
- Since the kernel is small, not a big concern,
- But it would be nice if Essence language had some other extension mechanism which would enable the same extension without this much repetition.
Thanks for listening..