Applications of the Essence Framework

GÖRKEM GİRAY, TURKEY
ERAY TÜZÜN, HAVELSAN, TURKEY
BEDİR TEKİNERDOĞAN, WAGENINGEN UNIVERSITY, NETHERLANDS
YAGUP MACİT, HAVELSAN, TURKEY
Contents

Operating Concept

Why do we essentialize a software Development methodology?

Systematic Approach for Essentializing

Examples
  • Essentializing Nexus
  • Essentializing SPLE

Tool Support for Essentializing
Operating Concept
Determine Current State

**Completed**
- Requirements
  - Conceived
    - Need for system agreed by initial stakeholders
  - Bounded
    - Theme, scope, success criteria of system is clear
  - Coherent
    - Described requirements provide coherent picture of the system
  - Sufficient
    - Requirements adequately describe solution and acceptable to stakeholders
  - Software System
    - Architecture Selected
      - Architecture selected that address key technical risk
  - Usable
    - System is usable and has desired quality characteristics
  - Ready
    - System as a whole has been accepted for deployment in operational environment
  - Operational
    - System in use in operational environment
  - Team
    - Seeded
      - Team’s mission is clear

**Pending**
- Requirements
  - Conceived
    - Need for system agreed by initial stakeholders
  - Bounded
    - Theme, scope, success criteria of system is clear
  - Coherent
    - Described requirements provide coherent picture of the system
  - Sufficient
    - Requirements adequately describe solution and acceptable to stakeholders
  - Software System
    - Architecture Selected
      - Architecture selected that address key technical risk
  - Usable
    - System is usable and has desired quality characteristics
  - Ready
    - System as a whole has been accepted for deployment in operational environment
  - Operational
    - System in use in operational environment
  - Team
    - Seeded
      - Team’s mission is clear

**Fulfilled**
- Requirements
  - Conceived
    - Need for system agreed by initial stakeholders
  - Bounded
    - Theme, scope, success criteria of system is clear
  - Coherent
    - Described requirements provide coherent picture of the system
  - Sufficient
    - Requirements adequately describe solution and acceptable to stakeholders
  - Software System
    - Architecture Selected
      - Architecture selected that address key technical risk
  - Usable
    - System is usable and has desired quality characteristics
  - Ready
    - System as a whole has been accepted for deployment in operational environment
  - Operational
    - System in use in operational environment
  - Team
    - Seeded
      - Team’s mission is clear
Determine Next State

- **Requirements**
  - Conceived
    - Need for system agreed by initial stakeholders
    - Users and customers identified
    - Expected benefit of system agreed
  - Bounded
    - Theme, scope, success criteria of system is clear
    - Maintenances for managing requirements in place
    - Constraints and assumptions considered
  - Coherent
    - Described requirements provide coherent picture of the system
    - Conflicting requirements separated
    - Important usage scenarios explained
    - Priority of requirements clear
  - Sufficient
    - Requirements adequately describe solution and acceptable to stakeholders
    - Rate of change to agreed requirements is low and under control

- **Software System**
  - Architecture
    - Architecture selected that address key technical risks
    - Criteria for selecting architecture agreed
    - Platforms, technologies, languages selected
    - Buy, build, reuse decisions made
  - Demonstrable
    - Executable version of system demonstrates architecture is fit for purpose
    - Supports functional and non-functional testing
    - Critical interface and system configurations exercised

- **Usable**
  - System is usable and has desired quality characteristics
  - System can be operated by users
  - Functionality and performance have been tested and accepted
  - Defect levels acceptable
  - Release content known

- **Ready**
  - System (as a whole) has been accepted for deployment in operational environment
  - Sponsors, users, stakeholders accept system as fit for purpose
  - Installation and other documents available
  - Operational support in place

- **Operational**
  - System in use in operational environment
  - System available to intended users
  - At least one example of system is fully operational
  - System supported to agreed service levels

- **Fulfilled**
  - System implementing requirements is worth making operational
  - Enough requirements are implemented
  - Stakeholders accept requirements as accurate

- **Work**
  - Initiated
    - Work initiator and client known
    - Work goals and constraints clear
    - Sponsorship and funding model clear
    - Priority of work clear
  - Prepared
    - Cost & effort understood
    - Funding in place
    - Resource availability and risk exposure understood
    - Governance model is clear
    - Integration and delivery points defined
  - Started
    - Development work has started
    - Work progress is monitored
    - Work broken down into actionable items with clear definition of done
    - Team members are accepting and progressing work items

- **Under Control**
  - Work going well, risks being managed, productively levels acceptable
  - Unplanned work & re-work under control
  - Work items completed within estimates
  - Measures tracked

- **Concluded**
  - Work to produce results have been finished
  - Work results are being achieved
  - The client has accepted the resulting software system

- **Closed**
  - All remaining housekeeping tasks completed, and work officially closed
  - Everything has been archived
  - Lessons learned and metrics made available

- **Team**
  - Seeded
    - Team’s mission is clear
    - Team knows how to achieve mission
    - Required competencies are identified
    - Team size is determined
  - Formed
    - Team has enough resources to start the mission
    - Team organization & individual responsibilities understood
    - Members know how to perform work
  - Collaborating
    - Members working as one unit
    - Communication is open and honest
    - Members focused on team mission
    - Success of team ahead of personal objectives
  - Performing
    - Team working efficiently and effectively
    - Adapts to changing context
    - Produces high quality output
    - Minimal backtracking and re-work
    - Waste continually eliminated
  - Adjourned
    - Team no longer accountable
    - Responsibilities handed over
    - Members available for other assignment
How to Achieve Next State

- **Requirements**
  - Satisfactory
    - System implementing requirements is worth making operational
    - Enough requirements are implemented
  - **5 / 6**

- **Software System**
  - Usable
    - System is usable and has desired quality characteristics
    - System can be operated by users
    - Functionality and performance have been tested and accepted
    - Defect levels acceptable
    - Release content known
  - **3 / 6**

- **Work**
  - Under Control
    - Work going well, risks being managed, productivity levels acceptable
    - Unplanned work & re-work under control
    - Work items completed within estimates
    - Measures tracked
  - **4 / 6**

- **Team**
  - Performing
    - Team working efficiently and effectively
    - Adapts to changing context
    - Produce high quality output
    - Minimal backtracking and re-work
    - Waste continually eliminated
  - **4 / 5**
Some context (HAVELSAN)

Figure: Scrum and CMMI/MPS:BR, A. Rouiller, B. Gloger, ScrumGathering, Sao Paulo, 2009.
Essentializing (Why?)

- To monitor projects with unified state model
- To tailor the method for company’s context
- To understand the practice
- To assess potential gaps
- To improve the process
Systematic Approach for Mapping


* http://www.omg.org/spec/Essence/1.1/
Systematic Approach for Mapping

1. Extract concepts from method specification
2. Classify extracted concepts according to Essence concepts
3. Specify concepts using Essence Language
4. Specify properties of concepts
5. Associate related concepts
6. Review essentialized method
Systematic Approach for Mapping

1. Extract concepts from method specification

Sprint Backlog
The Sprint Backlog is the set of Product Backlog items selected for the Sprint, plus a plan for delivering the product Increment and realizing the Sprint Goal. The Sprint Backlog is a forecast by the Development Team about what functionality will be in the next Increment and the work needed to deliver that functionality into a “Done” Increment.

The Sprint Backlog makes visible all of the work that the Development Team identifies as necessary to meet the Sprint Goal.

The Sprint Backlog is a plan with enough detail that changes in progress can be understood in the Daily Scrum. The Development Team modifies the Sprint Backlog throughout the Sprint, and the Sprint Backlog emerges during the Sprint. This emergence occurs as the Development Team works through the plan and learns more about the work needed to achieve the Sprint Goal.

As new work is required, the Development Team adds it to the Sprint Backlog. As work is performed or completed, the estimated remaining work is updated. When elements of the plan are deemed unnecessary, they are removed. Only the Development Team can change its Sprint Backlog during a Sprint. The Sprint Backlog is a highly visible, real-time picture of the work that the Development Team plans to accomplish during the Sprint, and it belongs solely to the Development Team.
Systematic Approach for Mapping

2 Classify extracted concepts according to Essence concepts

Method Specification

Concept Categories

Essence Concepts

Categorized Concepts according to Essence Concepts
Systematic Approach for Mapping

3 Specify concepts using Essence Language

- Alpha 1
- Work Product 2
- Competency 3
Systematic Approach for Mapping

4 Specify properties of concepts
Systematic Approach for Mapping

5 Associate related concepts

- Role
- Activity
- WorkProduct
- Action
- Alpha

- Role participates in Activity
- Role is related to Action
- Activity has Action
- WorkProduct is responsible for Role
- WorkProduct creates, reads, updates or deletes Action
- Action affects Role
- Role is related to Alpha
Systematic Approach for Mapping

6 Review essentialized method

- Does each sub-alpha have proper states defined?
- Does each state of sub-alphas have associated checklists consisting of checkpoints?
- Does each WorkProduct have a list of LevelOfDetails defined?
- Is each Activity associated with an Alpha and/or WorkProduct?
- Does each Activity have at least one EntryCriterion and CompletionCriterion?
- Are all the concepts of the method described using Essence?
Multiple teams create integrated increments
More focus on dependencies and interoperation

Essentializing Nexus

1. Extract concepts from method specification

<table>
<thead>
<tr>
<th>Extracted Concept</th>
<th>Concept Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backlog Dependency</td>
<td>Dependency</td>
</tr>
<tr>
<td>Daily Scrum</td>
<td>Event</td>
</tr>
<tr>
<td>Definition of Done</td>
<td>Artifact</td>
</tr>
<tr>
<td>Development Work</td>
<td>Work</td>
</tr>
<tr>
<td>Integrated Increment</td>
<td>Increment</td>
</tr>
<tr>
<td>Integration Issue</td>
<td>Issue list</td>
</tr>
<tr>
<td>Nexus</td>
<td>Practice</td>
</tr>
<tr>
<td>Nexus Daily Scrum</td>
<td>Event</td>
</tr>
<tr>
<td>Nexus Integration Team</td>
<td>Team</td>
</tr>
<tr>
<td>Nexus Integration Team Member</td>
<td>Team member</td>
</tr>
<tr>
<td>Nexus Sprint Backlog</td>
<td>Task list</td>
</tr>
<tr>
<td>Nexus Sprint Goal</td>
<td>Artifact</td>
</tr>
<tr>
<td>Nexus Sprint Planning</td>
<td>Event</td>
</tr>
</tbody>
</table>
# Essentializing Nexus

## 2 Classify extracted concepts according to Essence concepts

<table>
<thead>
<tr>
<th>Extracted Concept</th>
<th>Concept Category</th>
<th>Essence Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backlog Dependency</td>
<td>Dependency</td>
<td>Pattern (Dependency)</td>
</tr>
<tr>
<td>Daily Scrum</td>
<td>Event</td>
<td>Activity</td>
</tr>
<tr>
<td>Definition of Done</td>
<td>Artifact</td>
<td>WorkProduct</td>
</tr>
<tr>
<td>Development Work</td>
<td>Work</td>
<td>Sub-Alpha of Work</td>
</tr>
<tr>
<td>Integrated Increment</td>
<td>Increment</td>
<td>Sub-Alpha of Software System</td>
</tr>
<tr>
<td>Integration Issue</td>
<td>Issue list</td>
<td>WorkProduct</td>
</tr>
<tr>
<td>Nexus</td>
<td>Practice</td>
<td>Practice</td>
</tr>
<tr>
<td>Nexus Daily Scrum</td>
<td>Event</td>
<td>Activity</td>
</tr>
<tr>
<td>Nexus Integration Team</td>
<td>Team</td>
<td>Sub-Alpha of Team</td>
</tr>
<tr>
<td>Nexus Integration Team Member</td>
<td>Team member</td>
<td>Pattern (Role)</td>
</tr>
<tr>
<td>Nexus Sprint Backlog</td>
<td>Task list</td>
<td>WorkProduct</td>
</tr>
<tr>
<td>Nexus Sprint Goal</td>
<td>Artifact</td>
<td>WorkProduct</td>
</tr>
<tr>
<td>Nexus Sprint Planning</td>
<td>Event</td>
<td>Activity</td>
</tr>
</tbody>
</table>
Essentializing Nexus

3 Specify concepts using Essence Language
Essentializing Nexus

4 Specify properties of concepts

alpha IntegratedIncrement:
“represents sum of all integrated work completed by a Nexus”
with states {
state usable and potentially releasable {
checks {
item c1 {“Definition of Done is met.”} }
}
}

}
Essentializing Nexus

5  Associate related concepts
Software Product Line Engineering

- Products
  - pertain to
  - share an
  - are built from

- Market Strategy / Application Domain
  - is satisfied by
  - Architecture
    - used to structure
  - Components
    - CORE ASSETS
What is SPLE

[Diagram showing the process of Domain Engineering and Application Engineering with emphasis on Domain Artefacts and Variability Model.]
Essentializing SPLE

1. Extract concepts from method specification

<table>
<thead>
<tr>
<th>Extracted Concept</th>
<th>Concept Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binding the Variability</td>
<td>Activity</td>
</tr>
<tr>
<td>Domain Requirements</td>
<td>Activity</td>
</tr>
<tr>
<td>Elicitation</td>
<td>Activity</td>
</tr>
<tr>
<td>Major common and variable features</td>
<td>Vision</td>
</tr>
<tr>
<td>Scope</td>
<td>Vision</td>
</tr>
<tr>
<td>Traceability links</td>
<td>Dependency</td>
</tr>
<tr>
<td>Domain engineer</td>
<td>Role</td>
</tr>
<tr>
<td>Application engineer</td>
<td>Role</td>
</tr>
<tr>
<td>Inspections</td>
<td>Practice</td>
</tr>
<tr>
<td>Reviews</td>
<td>Practice</td>
</tr>
<tr>
<td>Walkthroughs</td>
<td>Practice</td>
</tr>
<tr>
<td>Running Application</td>
<td>Working Software</td>
</tr>
<tr>
<td>Domain Requirements</td>
<td>Requirements</td>
</tr>
<tr>
<td>Application Requirements</td>
<td>Requirements</td>
</tr>
</tbody>
</table>
## Essentializing SPLE

### 2. Classify extracted concepts according to Essence concepts

<table>
<thead>
<tr>
<th>Extracted Concept</th>
<th>Concept Category</th>
<th>Essence Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binding the Variability</td>
<td>Action</td>
<td>Action</td>
</tr>
<tr>
<td>Domain Requirements elicitation</td>
<td>Activity</td>
<td>Activity</td>
</tr>
<tr>
<td>Major common and variable features</td>
<td>Vision</td>
<td>Opportunity</td>
</tr>
<tr>
<td>Scope</td>
<td>Vision</td>
<td>Opportunity</td>
</tr>
<tr>
<td>Traceability links</td>
<td>Dependency</td>
<td>Pattern</td>
</tr>
<tr>
<td>Domain engineer</td>
<td>Role</td>
<td>Pattern</td>
</tr>
<tr>
<td>Application engineer</td>
<td>Role</td>
<td>Pattern</td>
</tr>
<tr>
<td>inspections</td>
<td>Practice</td>
<td>Practice</td>
</tr>
<tr>
<td>reviews</td>
<td>Practice</td>
<td>Practice</td>
</tr>
<tr>
<td>walkthroughs</td>
<td>Practice</td>
<td>Practice</td>
</tr>
<tr>
<td>Running Application</td>
<td>Working Software</td>
<td>Software System</td>
</tr>
<tr>
<td>Domain Requirements</td>
<td>Requirements</td>
<td>Sub Alpha of Requirements</td>
</tr>
<tr>
<td>Application Requirements</td>
<td>Requirements</td>
<td>Sub Alpha of Requirements</td>
</tr>
</tbody>
</table>
Essentializing SPLE

3 Specify concepts using Essence Language

- SPLE Practice
  - Stakeholders
    - Domain Stakeholders
    - Application Stakeholders
    - Product Line Stakeholders
  - Opportunity
    - Domain Opportunity
    - Application Opportunity
  - Requirements
    - Domain Requirements
    - Application Requirements
  - Software System
    - Platform
    - Customer-specific Applications
Develop the Domain Requirements to provide a consistent description of the essential characteristics of the platform.

This activity is started when:
- Domain Stakeholders: Represented
- Domain Opportunity: Solution Needed
- Domain Requirements: Conceived
- Product Line Scope: Approved
- Product Roadmap: Approved

Active Roles:
- Main Role: Domain Requirements Engineer
- Related Roles: Product Manager, Domain Architect

This activity is completed when:
- Domain Requirements: Coherent
- Domain Stakeholders: In Agreement
- Domain Opportunity: Viable
- Domain Requirements Specification: Finished
- Domain Variability Model: Finished
Five Essentializing SPLE Practice

1. Extract concepts from method specification
2. Associate related concepts

- Stakeholders
  - Domain Stakeholders
  - Application Stakeholders
  - Product Line Stakeholders

- Opportunity
  - Domain Opportunity
  - Application Opportunity

- Requirements
  - Domain Requirements
  - Application Requirements

- Software System
  - Domain Artefacts
  - Application Artefacts

- Customer-specific Applications
  - Existing Products
  - Market Strategy
  - Product Line Scope
  - Company Goals
  - Product Roadmap
  - Domain Variability Model
  - Domain Requirements Specification
  - Application Variability Model
  - Application Requirements Specification
  - Domain Architecture
  - Domain Artefacts
  - Application Architecture
  - Application Artefacts
The stakeholder requirements are classified as application requirements or domain requirements.

The deltas between domain and application requirements are identified.

The traces between application requirements and the corresponding domain requirements are identified.

The application variability model is defined.

The application requirements are shared with the team and the stakeholders.

The rationale behind the application requirements is clear.

The rationale behind the application variability model is clear.

The application requirements communicate the essential characteristics of the system to be delivered.

The most important usage scenarios for the system can be explained.

The priority of the application requirements is clear.

Constraints are identified and considered.

Assumptions are clearly stated.

The application requirements are documented in detail.

Finding the Gaps

Customer
- Explore possibilities
- Understand Stakeholder Needs
- Ensure Stakeholder Satisfaction
- Use the System

Solution
- Define the Scope
- SPLE Feasibility Analysis
- Domain Requirements Elicitation
- Application Requirements Elicitation
- Define Product Features
- Feedback for Domain Artefacts

Endeavor
- Prepare to do the Work
- Coordinate Activity
- Support the Team
- Track Progress
- Stop the Work

DevOps Strategy
- Select the Transition Strategy
- Define the Organization Structure
- Coordinate the Interrelated Activities
- Product Elimination

Understand the Requirements
- Develop the Domain Requirements
- Develop the Application Requirements

Shape the System
- Defining the Platform Architecture
- Defining the Application Architecture

Implement the System
- Platform Realization
- Application Realization

Test the System
- Platform Testing
- Application Testing

Deploy the System
- Platform Realization
- Application Realization

Operate the System
- Platform Testing
- Application Testing

Prepare to do the Work
- Product Elimination
Essence Practice Workbench
The Practice Explorer shows Practice Workbench projects.

The Essence Kernel project contains the elements defined in the OMG Essence specification.

Alphas that represent the essential things to work with.

Activity Spaces that represent the essential things to do.
When selecting an element in the Practice Explorer you can switch between different views.

- **ETextile Source view**: The main editor for authoring the practice using plain text and annotations.
- **Guideline Preview**: Renders how the guideline will be presented in HTML.
- **Overview Card Preview**: Renders the card presentation.
Scrum Roles

Scrum roles are represented as Patterns.

Product Owner (Guideline Preview)

Product Owner (Card Preview)
Scrum Sprint

Sprint is represented as a sub-alpha of Work

The Sprint has States with Checkpoints

Sprint in Under Control State (Card Preview)

The Sprint has associated the Work Product Sprint Backlog that contains the set of Product Backlog items selected for the Sprint, and the plan for delivering the product Increment

Under Control (State Card Preview)
Tool Support

- Easy creation and maintenance of practice elements
- Straightforward to share practice results between users
- Supports composition of practices into useful methods
- Practice template creation and deployment
- Innovative card-based representation
- Publication of methods, practices and kernels as browsable HTML web-sites

www.ivar Jacobson.com/esswork-practice-workbench
Essentialized Practices / Methods
Scrum, Nexus
SPLE
Synthesis-based Software Architecture Design
The Unified Process
The Waterfall ...
Thank you