Describing Fractal Processes with UML

Overview

1 - Who is FJA? What do we do?
2 - Which are our requirements?
3 - What is our approach? Examples
4 - What are our experiences?
5 - Where do we go from here?
What does FJA do for a living?

- **Life Factory**
  comprehensive software suite for life insurance administration

- Application Service Providing
  so called „Riester-Rente“

- large integration projects
  with almost all German insurance companies

...
2 - Requirements: Observations about our Software Processes

- Different kinds of projects
  - standard software
  - integration
  - ASP

- Varying sizes & durations
  - up to 5k person-days for 2-3 years
  - 8 weeks . . . 4 years

- multi-party / multi-systems projects

- Process risks are spread unevenly

For instance: ASP Project Stakeholders

- Customer
- FJA
- Subcontractor (internal)
- Subcontractor (external)
Describing Fractal Processes with UML

For instance: ASP Project Overall Process

2-6 months

Init Analysis Realization Deploym. Production Use

for each component

the important issue: time to market

Describing Fractal Processes with UML

2 - Requirements: Observations about our Software Processes

- Different kinds of projects
  - standard software
  - integration
  - ASP

- Varying sizes & durations
  - up to 5k person-days for 2-3 years
  - 8 weeks . . . 4 years

- multi-party / multi-systems projects

- Process risks are spread unevenly
Describing Fractal Processes with UML

So:

What kind of Process Model should we use?
Discarded process model candidates

- **classical models** *(VM'97, RUP, ...)*
  - overweight & bureaucratic
  - hard to learn
  - slow reaction to environment changes

- **“lightweight” models** *(XP, ...)*
  - doesn’t scale, doesn’t fit to all kinds of projects
  - no handle for process improvement
  - customer acceptance (contracts)
  - only informal presentations available

- both: little provision for legacy integration / reengineering

---

3 - Our (current) approach: Process Patterns / UML

What is a pattern in the first place?

"A pattern is a practically proven solution to a recurring problem"

- Patterns apply at all scales.
  - So, structuring a process by patterns induces similar structures at all levels, resulting in a self similar (i.e. fractal) process.

- Patterns come in families.
Describing Fractal Processes with UML

Expected benefits

- more flexibility:
  - scale invariant (one size fits all)
  - highly adaptive (even on-the-fly)
  - little organisational impact (cf. RUP)

- cheaper:
  - selective precision
  - good cost / benefit ratio
  - graceful degradation

- easier:
  - soak up knowledge „as is“
  - learn a bit at a time
  - use only those parts you need
  - best practices oriented

How to do it?

- use standard descriptive scheme for patterns
  - (Go4, Go5)

- use Unified Modeling Language
  - process aspect replaces structure aspect
  - Activity Diagram replaces Class Diagram
  - fix RUP-additions by proper metamodel-embedding

- classical terminology & tools remain the same
  - (roles, artifacts, glossary, ...)

- generate different presentations from XML
Describing Fractal Processes with UML

Example - a toy language of process patterns

Realize
make-or-buy
refine
implement
buy
test

Refine
split
realize . . . realize
integrate
test

Implement
analyze
design
code
test

Example - applying process patterns

Realize
Product
Process
Example - applying process patterns

Realize

make-or-buy

refine

implement

buy

test
Describing Fractal Processes with UML

Example - applying process patterns

- **Realize**
  - make-or-buy
  - refine
  - implement
  - buy
  - test

- **Refine**
  - split
  - realize . . . realize
  - integrate
  - test

---

Example - applying process patterns

- **Realize**
  - make-or-buy
  - refine
  - implement
  - buy
  - test

- **Refine**
  - split
  - realize . . . realize
  - integrate
  - test
Example - applying process patterns

Consequences

- recursive application of patterns
- same patterns on all levels
- easier process implementation

recursive system structure
scale invariant (fractal) process
simpler, faster, cheaper
Describing Fractal Processes with UML

Example - describing a process pattern

- Name
- Synonyms
- Classification
- Intent
- Motivation
- Applicability
- Process
- Roles
- Consequences
- Sample Execution
- Implementation
- Known Uses
- Related Patterns

---

Describing Fractal Processes with UML

Example (1/3) - describing a process pattern

- Name
- Synonyms
- Classification
- Intent
- Motivation
- Applicability
- Process
- Roles
- Consequences
- Sample Execution
- Implementation
- Known Uses
- Related Patterns

- Abstraction level
  - Technique
  - Activity
  - Process

- Phase
  - Analysis
  - Design
  - ...

- Purpose
  - Engineering
  - Support
  - Control
  - ...

---
Describing Fractal Processes with UML

Example (1/3) - describing a process pattern

- Name
- Synonyms
- Classification
- Intent
- Motivation
- Applicability
- Process
- Roles
- Consequences
- Sample Execution
- Implementation
- Known Uses
- Related Patterns

\[ e.g.: \text{Refine} \]

\[ \text{System}_1 \]

\[ \text{System}_n \]

\[ \text{Integrate} \]

\[ \text{Realize} \]...

\[ \text{Realize} \]

\[ \text{Test} \]

\[ [\text{tested}] \]
Example - process pattern „waterfall“ (sketch)

<table>
<thead>
<tr>
<th>Name</th>
<th>waterfall model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synonyms</td>
<td>sequential development</td>
</tr>
<tr>
<td>Classification</td>
<td>phase: all / level: process / purpose: all</td>
</tr>
<tr>
<td>Intent</td>
<td>force development into rigid scheme, keep things simple</td>
</tr>
<tr>
<td>Motivation</td>
<td>increase controlability, improve process &amp; quality</td>
</tr>
<tr>
<td>Applicability</td>
<td>very large projects, mission critical software, fixed time/requirements/environment, E/RT-systems</td>
</tr>
<tr>
<td>Process</td>
<td>...</td>
</tr>
<tr>
<td>Roles</td>
<td>Customer, Analyst, Developer, Tester, ...</td>
</tr>
<tr>
<td>Consequences</td>
<td>can’t go back, late validation incurs high risk at roll-out, ...</td>
</tr>
<tr>
<td>Sample Execution</td>
<td>...</td>
</tr>
<tr>
<td>Implementation</td>
<td>organisational issues, ...</td>
</tr>
<tr>
<td>Known Uses</td>
<td>OS/360, ...</td>
</tr>
<tr>
<td>Related Patterns</td>
<td>iterative model, spiral model</td>
</tr>
</tbody>
</table>

4 - What are our experiences? Patterns we have defined yet

<table>
<thead>
<tr>
<th>Style level (process)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 ready</td>
</tr>
<tr>
<td>~5 in preparation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pattern level (subprocess, activity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 ready</td>
</tr>
<tr>
<td>~14 in preparation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Idiom level (technique)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 ready</td>
</tr>
<tr>
<td>~45 in preparation</td>
</tr>
</tbody>
</table>
4 - What are our experiences?

- Possible to integrate existing fragments of processes ("beg-borrow-steal" approach, e.g. "QuickCards")
- Easier to learn and use by (naive) developers but even so: needs "implementation by coaching"
- And, worst of all: extremely difficult to communicate to management, even with direct evidence of commercial value

5 - Where do we want to go from here?

- More complete description of approach
  - underlying theory & formalisms
  - how-to cookbook
- Collect and document much more patterns
- Suggestions for pattern candidates welcome!
- But empirical studies are currently unrealistic!
- Comments? Questions? Suggestions? => Harald.Stoerrle@fja.com