Fifth International Workshop on Managing Technical Debt, Baltimore 2013

Product Release Planning in Consideration of Technical Debt

Guenther Ruhe
(joint work with Jason Ho)
University of Calgary, Canada

Agenda

• Taxonomy of release decisions
• What-to-release under complex feature dependency constraints
• When-to-release
• Feature planning in consideration of technical debt
• Prototype implementation
• Initial evaluation
• Summary and outlook
Taxonomy of release planning approaches

- One versus multiple products
- Systematic/analytic versus agile
- What-to-release versus when-to-release
- One versus multiple releases
- Operational versus strategic
- With or without consideration of themes
  - With or without consideration of bug fixing
  - Consideration of functional versus quality requirements
  - With or without consideration of feature dependencies
  - With versus without consideration of technical debt

Release planning - Why it is difficult?

- Information is
  - Uncertain
  - Inconsistent
  - Incomplete
  - Fuzzy
- Multiple objectives
  - Usability
  - Value
  - Time-to-market
  - Maintainability
  - Risk
- Hard and soft constraints on
  - Time
  - Effort
  - Quality
  - Resources
  - Technical debt
- Decision space
  - Large size
  - High complexity
  - Dynamically changing

MTD 2013
Ad hoc product decisions are no longer good enough - Why?

Ginni Rometty, Chairman, President and CEO, IBM, 2013:

*Responding to change for gaining competitive advantage in the era of smart decisions will be based *not* on “gut instinct,” but on predictive analytics.*

---

**Agenda**

- Taxonomy of release decisions
- What-to-release under complex feature dependency constraints
- When-to-release
- Feature planning in consideration of technical debt
- Prototype implementation
- Initial evaluation
- Summary and outlook
The diversification principle

A single solution to a cognitive complex problem is less likely to reflect the actual problem when compared to a portfolio of qualified solutions being structurally diversified.

Diversified release plans
Why feature dependencies?

• Features cannot be treated in isolation, as only approximately 20% of features are singular (study [1]).
• One feature may affect other features when it
  – constraints when and how they are implemented,
  – affects their cost of implementation, or
  – changes their value.
• Ignoring the dependencies when assessing the impact of a change implies changes in process and higher costs.
• Challenge: Identifying, storing, and maintaining feature dependencies
• Technical debt: By addressing dependency, teams do not incur debt by broken constraints, or expensive silo implementation of functionality that could have been reused

Release planning under complex feature constraints

Application of Constraint Programming:
MiniZinc: Syntax to define decision variables, input variables, and constraints using various logical and arithmetic operators such as

\[ \land \text{ (logical AND)} \]
\[ \lor \text{ (logical OR)} \]
\[ \rightarrow \text{ (logical implication)} \]
\[ = \text{ (equality)} \]
\[ \neq \text{ (inequality)} \]
\[ + \text{ (addition)} \]
\[ \ast \text{ (multiplication)} \].
Hybrid approach [4]

Phase 1 (EVOLVE II)
If solution $x^*$ gained from EVOLVE II is feasible and there are no added value components in the objective function
then STOP else goto Phase 2

Phase 2 (CP versus Hybrid RP)
CP
2.1 Transformation of the problem using MiniZinc
2.2 Solution of the transformed problem using Gecode
Hybrid RP
2.1 Transformation of the problem using MiniZinc
2.2 Solution of the transformed problem using Gecode
2.3 Initiating solution process with $x^*$

Empirical evaluation (1/2) [4]

(1) Artificial data sets: Worst-performing project from the artificial dataset
- 150 features, 7 resources,
- 5 releases, 21 precedence constraints,
- 3 coupling constraints
- Progressively addition of 0, 5, 10, 15, 20, 25, 30, 35, up to 40 additional constraints.

Results:

<table>
<thead>
<tr>
<th># additional constraints</th>
<th>Value (CP)</th>
<th>Value (Hybrid-RP)</th>
<th>Hybrid-RP/CP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>20961</td>
<td>47843</td>
<td>2.28</td>
</tr>
<tr>
<td>5</td>
<td>21581</td>
<td>47874</td>
<td>2.22</td>
</tr>
<tr>
<td>10</td>
<td>23471</td>
<td>48173</td>
<td>2.05</td>
</tr>
<tr>
<td>15</td>
<td>22230</td>
<td>47477</td>
<td>2.13</td>
</tr>
<tr>
<td>20</td>
<td>22823</td>
<td>47802</td>
<td>2.09</td>
</tr>
<tr>
<td>25</td>
<td>21252</td>
<td>47148</td>
<td>2.21</td>
</tr>
<tr>
<td>30</td>
<td>21313</td>
<td>46533</td>
<td>2.18</td>
</tr>
<tr>
<td>35</td>
<td>22335</td>
<td>46533</td>
<td>2.08</td>
</tr>
<tr>
<td>40</td>
<td>24475</td>
<td>48299</td>
<td>1.97</td>
</tr>
</tbody>
</table>
Empirical evaluation (2/2) [4]

(2) Real world data sets with added artificial constraints:

<table>
<thead>
<tr>
<th>Project</th>
<th># feat.</th>
<th># res.</th>
<th># releases</th>
<th># dep.</th>
<th># dep. features</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25</td>
<td>7</td>
<td>3</td>
<td>23</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>75</td>
<td>5</td>
<td>2</td>
<td>81</td>
<td>75</td>
</tr>
<tr>
<td>3</td>
<td>633</td>
<td>1</td>
<td>5</td>
<td>19</td>
<td>67</td>
</tr>
<tr>
<td>4</td>
<td>914</td>
<td>1</td>
<td>4</td>
<td>85</td>
<td>232</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project</th>
<th>Hybrid-RP/CP after seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1.0132</td>
</tr>
<tr>
<td>3</td>
<td>*</td>
</tr>
<tr>
<td>4</td>
<td>*</td>
</tr>
</tbody>
</table>

Agenda

- Taxonomy of release decisions
- What-to-release under complex feature dependency constraints
- When-to-release
- Feature planning in consideration of technical debt
- Prototype implementation
- Initial evaluation
- Summary and outlook
When-to-release problem W2RP [2]

- Is defined by a sequence \( \{RPP_i\}_i \) of problems RPP.
- Each individual RPP\(_i\) has a different fixed release date RD\(_i\).
- W2RP means to determine operational release plans with varying feature sets \( F_i \) which represent trade-off solutions in terms of:
  - Maximize total release value \( TRV(F_i) \)
  - Maximize total release quality \( TRQ(F_i) \)
  - Minimize RD\(_i\)

Modeling (1/2)

- Time:
  - RD: Targeted time to be released by stakeholders (person days)
  - RD ± Δ: The duration in which the release date can be varied to find the optimized release time

- Values:
  - Measured by Customers’ weighted satisfaction score
    - For a release k with n features, the total release value is:
      \[ TRV(n, k) = \sum_{i=1}^{n} w_i \cdot sat(f_i) \]
  - As each feature consumes resources, values is affected by capacity of the resources assigned to that feature set.
Modeling (2/2)

- Quality can be measured by how many transactions of the final product is expected to succeed (consistently)
- By applying a reliability model we obtain the quality indicator TRQ for the whole release as

\[ TRQ = \sum_{i=1}^{n} S_i Q_i(f_i) \times Q_{int} \]

with

- \( n \) = number of features of the product,
- \( S_i \) is the complexity factor of feature \( f_i \)
- \( Q_i(f_i) \) is the probability that feature \( f_i \) performs correctly after release
- \( Q_{int} \) is the probability of the integrated functionality of the release performs correctly

Re-optimization

[Diagram showing the process of re-optimization with steps and decision points]
Agenda

• Taxonomy of release decisions
• What-to-release under complex feature dependency constraints
• When-to-release
• Feature planning in consideration of technical debt
• Prototype implementation
• Initial evaluation
• Summary and outlook

The modeling of technical debt

“The mere formulation of a problem is far more essential than its solution, which may be merely a matter of mathematical or experimental skills. To raise new questions (and), new possibilities, to regard old problems from a new angle, requires creative imagination and marks real advances in science.”

(Albert Einstein, 1879-1955)
Technical debt – Some assumptions

- Context-specific
- Technical depth can be formulated as cross-cutting concern (integrated functionality).
- We can estimate the effort and the (time-dependent) impact of this integrated functionality(s).
- Integrated functionalities are forms of dependencies and this can be handled by Hybrid-RP.
- Scenario-playing and learning is the key to support actual decision-making.

The many facets of technical debt

[3]
TD formulations by feature dependencies

- Feature set F with two sample individual (cross cutting) features f(n') and f(n''),
- respective solution vector components x(n') and x(n'')
- K being the total number of releases planned for, and
- x(n') = K+1 if f(n') is postponed

**Dependency** | **Constraint**
--- | ---
Weak precedence | For x(n'') ≤ K: x(n') ≤ x(n'')
Strict precedence | For x(n'') ≤ K: x(n') < x(n'')
Coupling | x(n') = x(n'')
Inclusive OR | x(n') ≤ K OR x(n'') ≤ K
Exclusive XOR | x(n') ≤ K XOR x(n'') ≤ K
Boolean | Boolean expression of set consisting of any subset F1 of F

At least one | At least one of p given features F1 must not be postponed (p > 1)
At most one | At most one of p given features F1 must not be postponed (p > 1)
Exactly one | Exactly one of p given features F1 must not be postponed (p > 1)
Multiplicative ICOST | If x(n') ≥ x(n'') then cost(n') := cost(n') * c(n',n'')
Multiplicative CVALUE | If x(n') = x(n'') < K+1 then value(n') := value(n') * v(n',n'')

---

**Agenda**

- Taxonomy of release decisions
- What-to-release under complex feature dependency constraints
- When-to-release
- Feature planning in consideration of technical debt
- Prototype implementation
- Initial evaluation
- Summary and outlook
Use Case: API to JIRA and TFS

JIRA API Implementation

TFS Integration Implementation
Use Case: Interactive planning

Agenda

• Taxonomy of release decisions
• What-to-release under complex feature dependency constraints
• When-to-release
• Feature planning in consideration of technical debt
• Prototype implementation

• Initial evaluation
• Summary and outlook
Technical debt trade-off – Sample results

1. Planning without TD concerns
2. Planning with Design TD in mind
3. Planning with Design TD, and Refactoring from efforts freed up by reducing features.
4. Planning with Design TD, and Refactoring from efforts freed up by reducing test cases.

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Bronco Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario</td>
<td>1   2  3  4</td>
</tr>
<tr>
<td>Total No of features</td>
<td>51  51  51  51</td>
</tr>
<tr>
<td>Features for next release</td>
<td>44  41  40  41</td>
</tr>
<tr>
<td>Total Business Values</td>
<td>201  216.25  212.75  216.25</td>
</tr>
<tr>
<td>Total Test Cases Executed</td>
<td>100% 100% 100% 99%</td>
</tr>
</tbody>
</table>

Generating trade-off solutions – Case study

• We evaluate the approach using a case study from a real life technical project
• Objectives:
  – Evaluate optimization approach
  – Collect data on potential trade-off solutions
• Case set up:

<table>
<thead>
<tr>
<th>Project name</th>
<th>Bronco Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total No of features</td>
<td>51</td>
</tr>
<tr>
<td>Features for next release</td>
<td>40</td>
</tr>
<tr>
<td>Original release date $RD_0$</td>
<td>472</td>
</tr>
<tr>
<td>Technical Design Debt: If Features 2, 8, 12, 13, 47 are in the same release, the value and revenue of the release increased by</td>
<td>25%</td>
</tr>
<tr>
<td>Technical Refactoring Debt: A refactoring will prevent future errors, but takes time</td>
<td>15</td>
</tr>
</tbody>
</table>
Scenarios playing of trade-off solutions

Agenda

- Taxonomy of release decisions
- What-to-release under complex feature dependency constraints
- When-to-release
- Feature planning in consideration of technical debt
- Prototype implementation
- Initial evaluation
- Summary and outlook
Summary and conclusions

- Attempt to model TD by mapping to feature dependencies
- Combined with other release planning capabilities → aspects of TD can be addressed
- Human expert involvement is key → interactive optimization
- Future work on enhancing the range of TD constraints
- Need for prediction of value and system complexity in dependence of TD operations.
- Learn from practitioners and help them to make things easier or faster.

References


