Technical Debt Indexes provided by tools: a preliminary discussion

Francesca Arcelli Fontana     Riccardo Roveda     Marco Zanoni

ESSeRE Lab, DISCo, University of Milano-Bicocca

MTD - 4 Oct 2016, Raleigh
Tecnical Debt Indexes (TDI)

A single number used to control quality and/or debt. “Any kind of quality index computed by analysis tools”

A.k.a. Technical Quality Index, Technical Debt/Severity, Deficit Index

- **Q1** How are the quality indexes the tools provide exactly computed?
  - Which features do they take into account?
- **Q2** Which index does take more into account the architectural issues and in which way?
- **Q3** Which are the features not provided or taken into account by the indexes?
Tools

- CAST 7.3.2 (http://www.castsoftware.com/)
- inFusion v.1.8.5 (https://www.intooitus.com/)\(^1\)
- Sonargraph v.8.8.0 (https://www.hello2morrow.com/products/sonargraph)
- SonarQube v.5.2 (http://www.sonarqube.org)
- Structure101 v.4.2.10071 (http://structure101.com/products/)

\(^1\)its evolution at http://www.aireviewer.com
Other tools (examples)

- Massey Architecture Explorer: Antipatterns Score (J. Dietrich et al. 2012), Tangledness metric (S. M. A. Shah, Dietrich, and McCartin 2012);
- Lattix: Stability, Cyclicality, and Coupling metrics;
- STAN: different R. Martin’s metrics.
### Input information

<table>
<thead>
<tr>
<th>Information category</th>
<th>CAST</th>
<th>IF</th>
<th>SG</th>
<th>SQ</th>
<th>S101</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architectural Smells, e.g., (Lippert and Rook 2006, Garcia et al. 2009)</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Code Smells (Fowler 1999, Lanza and Marinescu 2006)</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Architecture/Design Metrics, e.g., (Martin 1995)</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Code Metrics, e.g., (Chidamber and Kemerer 1994, Lanza and Marinescu 2006)</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Architectural Violations$^\alpha$</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Coding Rule Violations$^\beta$</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

$^\alpha$: deviations from a reference architecture, i.e., unallowed dependencies  
$^\beta$: detected bad coding practices or excessive values of single metrics (some tools, e.g., SQ, internally refer to the latter as “smells”)
## Output information

<table>
<thead>
<tr>
<th>TDI name</th>
<th>CAST</th>
<th>inFusion</th>
<th>Sonargraph</th>
<th>SonarQube</th>
<th>Structure101</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution cost</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Keeping cost</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Unity Measure</td>
<td>US$</td>
<td>-</td>
<td>-</td>
<td>US$</td>
<td>Time</td>
</tr>
</tbody>
</table>
Q1: Used and generated information

As reported in the tables:

- Different TDI output approaches: abstract number, $ cost, time needed, LOC
- Extremely heterogeneuous;
- Different terminologies (e.g., smell, metric violation);

but,

- Similar aggregation/drill-down strategy: single indicators are composed in a linear combination, with different weighting schemes;
- Association of costs or weights to issues/indicators is arbitrary.
Q2: Architectural features

Observations:

- SonarQube ignores architectural information;
- SonarGraph, Structure 101, CAST are more oriented to architecture analysis and use it to integrate their TDIs.
Q3: Missing features

1. Keeping **and** Resolution costs (Principle and Interest);
2. Information unexploited: no single piece of information or analysis is supported by **all** tools;
3. Other existing information sources are not present, e.g., historical information
   - History is used only to track values over time
Thank you!
marco.zanoni@disco.unimib.it
References


