Visualising Architectural Dependencies

John Brondum, Liming Zhu
National ICT Australia (NICTA)
University of New South Wales (UNSW)
Liming.Zhu@nicta.com.au
Motivations

• Architectural dependency has TD potential
• Architectural dependency ≠ software dependency
  – New dependency concepts
    • understandable by all levels of stakeholders
    • not just aggregation of code-level dependency
    • architectural significant
  – “Implicit” dependencies & beyond code
    • Indirect; other factors (context, org structure, knowledge mgt..)
  – Automated analysis of code has limitations
    • large-scale system of systems with black-box components
    • Issues that are undetectable by code analysis
• Visualising them in architecture views is important
Contributions

• New dependency relationship types
  – Dimensions & degrees of dependency
  – Cross views & models
  – Relatedness of dependencies: implicit dependency

• Evaluated in three case studies
  – Online Production Systems
    • Implicit dependency → explicit design compromise → explicit debt
  – Learning & Teaching Portal
    • Synchronisation issues → highlighted as new dependency types with dimension → integration debt
  – Lending Valuation Systems
    • Upgrade problems → omitted dependency → design debt

Note: Analysis method and tool support (submitted to WICSA)
New Dependency Relationship Types

External

System E1 relatesTo System E2
System E1 limitedBy System E2
System E1 commWith System E2

Internal

System E1 influencedBy System E2
System E1 refersTo System E2

Behavioural  Structural
Dimensions & Degrees of Dependency

E \[\text{influencedBy} \] F

Properties:
- Name
- Responsibilities
- Interface visibility
- Implementation Info

Has no effect

Has an effect

Properties:
- Name
- Responsibilities
- Interface visibility
- Implementation Info

Dependency Dimensions: (Element Properties, Degree of Effect)
Case 1: Online Product System

Implicit dependency → explicit design compromise → explicit debt
Synchronisation issues → highlighted as new dependency types with dimension → integration mismatch debt
Case 3: Lending Valuation Systems

Upgrade problems → omitted dependency → design debt
## DSM Annotated with Dependency Types

<table>
<thead>
<tr>
<th></th>
<th>MJA</th>
<th>WP</th>
<th>PW1</th>
<th>eW1</th>
<th>JF</th>
<th>JA</th>
<th>JWP</th>
<th>JRP</th>
<th>BSP</th>
<th>SM</th>
<th>BOA</th>
<th>RE</th>
<th>EG</th>
<th>BS</th>
<th>SSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile Job App (MJA)</td>
<td>-</td>
<td>C</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R</td>
</tr>
<tr>
<td>Web Proxy (WP)</td>
<td></td>
<td>-</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R</td>
</tr>
<tr>
<td>pdaWeb1 (PW1)</td>
<td></td>
<td></td>
<td>-</td>
<td></td>
<td>C</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>endUserWeb1 (eW1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JobFulfilmentWS (JF)</td>
<td>C</td>
<td></td>
<td>L</td>
<td>C</td>
<td>I</td>
<td>I</td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JobAppointment (JA)</td>
<td>C</td>
<td>L</td>
<td></td>
<td></td>
<td>I</td>
<td>I</td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JobWorkerPortal (JWP)</td>
<td></td>
<td>C</td>
<td></td>
<td></td>
<td>I</td>
<td>I</td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JobRequesterPortal (JRP)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BizServiceProxy (BSP)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R</td>
</tr>
<tr>
<td>SalesManagement (SM)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C,R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BackOfficeAdmin (BOA)</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C,R</td>
<td></td>
<td></td>
<td>C,R</td>
</tr>
<tr>
<td>ReportingEngine (RE)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C,R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EmailGateway (EG)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C,R</td>
<td></td>
</tr>
<tr>
<td>BizService (BS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C,R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SqlServerDatastore (SSD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Conclusion

• **New architecture-level dependency types**
  – Top-down motivation
    • not just aggregation of code-level dependency
    • for stakeholders at different technical levels
  – For debt that not easily detectable in code
    • incorporating non-code factors as dimensions

• **Evaluated in real world case studies**
  – Retrospectively on projects by linking problems with new dependency and design debt
  – Proactively on projects by identifying new dependency and making the debt explicit

• **Future work**