Interactive Search Approaches for Requirements Prioritization

Angelo Susi

In collaboration with Alessandro Marchetto, Francis Palma, Giuseppe Scanniello, Paolo Tonella

Fondazione Bruno Kessler
Software Engineering Research Unit
Trento, Italy
Outline

• The problem of requirements prioritization

• The purpose of interaction in prioritization methods

• Exploitation of Interactive Genetic Algorithm

• Applications of the approach to test cases prioritization (ongoing work)
The problem of (requirements) prioritization

• The activity of finding an order relation on the set of requirements under analysis

• Prioritize according to domain knowledge / constraints concerning
  • available budget
  • time constraints
  • business risks
  • stakeholder expectations
  • technical constraints
A simple Interaction Schema

Collect user input

Solution
The Interaction Schema

Collect initial user input

Constraints

Local SB optimization

Candidate solution

Determine critical user input needed
Why interaction in prioritization

• Increment the effectiveness of the prioritization process via the exploitation of knowledge by the decision makers
  – Decision makers have a lot of “hidden” information that can be “extracted” and exploited

• (Expected) effects:
  – Inclusion of “emerging” knowledge stimulated by the specific subproblem to solve
  – Decrease of the decision making effort
  – Increase of the precision of the result
Some key points

• Important ingredients for interaction are:
  – The process should be able to detect “critical” points (such as inconsistencies between constraints and inconsistencies in the feedback)
  – The process should be able to “express” the critical point and ask the decision maker(s) to solve it
  – The process should be able to exploit the feedback
Approaches

• **Incomplete Analytic Hierarchy Process (IAHP):** state-of-the-art pairwise comparison approach, considers only user feedback on the set of alternative requirements, and exploit it to drive the elicitation process.

• **CBRank:** pairwise approach based on Machine Learning techniques that take into account previous user feedback and the domain constraints to drive the process of elicitation of the feedback.

• **Interactive Genetic Algorithms:** use of genetic algorithms to find the solution and drive the elicitation process (that could be based on pairwise comparisons).
(One of) our approach(es)

- Based on *Interactive Genetic Algorithm (IGA)*
  - aims at *minimizing the disagreement* between a *total order of prioritized requirements* and the *various constraints that are either encoded with the requirements* or that are expressed *iteratively* by the user during the prioritization process
The Input

• Set of Requirements

• Requirements documentation (e.g., cost of the implementation, value for the stakeholders, dependencies between requirements) that can be converted into total or partial rankings of the requirements

• Evaluation from users in terms of orderings between pairs of requirements
The process

1. Acquisition and coding of set of Requirements and Documentation
2. Interactive Genetic Algorithm: computation of solutions (individuals), also exploiting evaluations from users
3. Output of the ranking (the most promising individual)
The IGA algorithm: step 2.

2.1. computation of a first set of solutions (individuals)
2.2. identification of conflicts (ties) between individuals and constraints
2.3. request of knowledge to users (to decide about conflicts)
2.4. computation of new solutions
   – via evolution rules
2.5. If max number of iterations
   – than exit
   – else 2.2
Req. documentation coding into graphs

Transform the **domain knowledge** into graphs

<table>
<thead>
<tr>
<th>Req</th>
<th>Priorities</th>
<th>Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>R₁</td>
<td>High</td>
<td>R₂, R₃</td>
</tr>
<tr>
<td>R₂</td>
<td>Low</td>
<td>R₃</td>
</tr>
<tr>
<td>R₃</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>R₄</td>
<td>Medium</td>
<td>R₃</td>
</tr>
<tr>
<td>R₅</td>
<td>Medium</td>
<td></td>
</tr>
</tbody>
</table>
Production of individuals and identification of conflicts

<table>
<thead>
<tr>
<th>Individual ID</th>
<th>Requirements rankings (Individual)</th>
<th>Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pr₁</td>
<td>&lt; R₃, R₂, R₁, R₄, R₅ &gt;</td>
<td></td>
</tr>
<tr>
<td>Pr₂</td>
<td>&lt; R₅, R₃, R₁, R₄ &gt;</td>
<td></td>
</tr>
<tr>
<td>Pr₃</td>
<td>&lt; R₅, R₁, R₃, R₂, R₄ &gt;</td>
<td></td>
</tr>
<tr>
<td>Pr₄</td>
<td>&lt; R₅, R₃, R₁, R₂, R₄ &gt;</td>
<td></td>
</tr>
<tr>
<td>Pr₅</td>
<td>&lt; R₅, R₃, R₄, R₁ &gt;</td>
<td></td>
</tr>
<tr>
<td>Pr₆</td>
<td>&lt; R₅, R₃, R₄, R₂ &gt;</td>
<td></td>
</tr>
</tbody>
</table>

Conflicts = \{(R₃, R₁), (R₃, R₄), (R₃, R₅)\}

\[
dis(pr₁, pr₂) = \{(r, s) \in pr₁^* | (r, s) \in pr₂^*\}
\]
Production of individuals and identification of conflicts

<table>
<thead>
<tr>
<th>Individual ID</th>
<th>Requirements rankings (Individual)</th>
<th>Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pr₁</td>
<td>&lt; R₃, R₂, R₁, R₄, R₅ &gt;</td>
<td>6</td>
</tr>
<tr>
<td>Pr₂</td>
<td>&lt; R₃, R₂, R₁, R₅, R₄ &gt;</td>
<td>6</td>
</tr>
<tr>
<td>Pr₃</td>
<td>&lt; R₁, R₃, R₂, R₄, R₅ &gt;</td>
<td>6</td>
</tr>
<tr>
<td>Pr₄</td>
<td>&lt; R₂, R₃, R₁, R₄, R₅ &gt;</td>
<td>7</td>
</tr>
<tr>
<td>Pr₅</td>
<td>&lt; R₂, R₃, R₄, R₅, R₁ &gt;</td>
<td>9</td>
</tr>
<tr>
<td>Pr₆</td>
<td>&lt; R₂, R₃, R₅, R₄, R₁ &gt;</td>
<td>9</td>
</tr>
</tbody>
</table>

Conflicts = \{ (R₂, R₁), (R₂, R₄), (R₂, R₅), (R₃, R₁), (R₃, R₄), (R₃, R₅) \}

\[ dis(pr₁, pr₂) = \{ (r,s) \in pr₁^* | (r,s) \in pr₂^* \} \]
**Pairs to be evaluated to choose the individuals for feedback**

<table>
<thead>
<tr>
<th>Individual ID</th>
<th>Requirements rankings (Individual)</th>
<th>Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pr₁</td>
<td>&lt; R₃, R₂, R₁, R₄, R₅ &gt;</td>
<td>6</td>
</tr>
<tr>
<td>Pr₂</td>
<td>&lt; R₃, R₂, R₁, R₅, R₄ &gt;</td>
<td>6</td>
</tr>
<tr>
<td>Pr₃</td>
<td>&lt; R₁, R₃, R₂, R₄, R₅ &gt;</td>
<td>6</td>
</tr>
<tr>
<td>Pr₄</td>
<td>&lt; R₂, R₃, R₁, R₄, R₅ &gt;</td>
<td>7</td>
</tr>
<tr>
<td>Pr₅</td>
<td>&lt; R₂, R₃, R₄, R₅, R₁ &gt;</td>
<td>9</td>
</tr>
<tr>
<td>Pr₆</td>
<td>&lt; R₂, R₃, R₅, R₄, R₁ &gt;</td>
<td>9</td>
</tr>
</tbody>
</table>

**Ranked individuals with respect to disagreement**

Candidate pairs to be asked to decision maker

<table>
<thead>
<tr>
<th>Indiv. ID</th>
<th>PAIRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pr₁, Pr₂, Pr₃</td>
<td>(R₄, R₅), (R₁, R₂), (R₁, R₃)</td>
</tr>
<tr>
<td>Pr₅, Pr₆</td>
<td>(R₄, R₅)</td>
</tr>
</tbody>
</table>

PR₁ = < R₃, R₂, R₁, R₄, R₅ >

vs

PR₂ = < R₃, R₂, R₁, R₅, R₄ >< R₄, R₅ >

London, February 12th, 2013
User feedback

Why \((R_4, R_3)\) ?
Nothing is said about \((R_4, R_5)\) in the Priorities and Dependencies graphs

Why \((R_1, R_2)\) and \((R_1, R_3)\) ?

User Preference Graph \(eliOrd\)
New round of the algorithm

- The **new evolved population**

  - Pr1’
    - R3 → R2 → R4 → R1 → R5
  - Pr2’
    - R3 → R2 → R1 → R4 → R5
  - Pr3’
    - R1 → R3 → R2 → R5 → R4

- is compared against the new set of constraints graphs

- Priorities
- Dependencies
- User Preference Graph **eliOrd**
Case Study

- Prioritize requirements for a real software system, as part of the project ACube (Ambient Aware Assistance)
  - designing a highly technological monitoring environment to be deployed in nursing homes to support medical and assistance staff
- After user requirements analysis phase,
  - 60 user requirements and 49 technical requirements
  - Four macro-scenarios have been identified
- A Gold standard from the software architect

<table>
<thead>
<tr>
<th>Id</th>
<th>Macro-scenario</th>
<th># of requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALL</td>
<td>Monitoring falls</td>
<td>26</td>
</tr>
<tr>
<td>ESC</td>
<td>Monitoring escapes</td>
<td>23</td>
</tr>
<tr>
<td>MON</td>
<td>Monitoring dangerous behavior</td>
<td>21</td>
</tr>
<tr>
<td>ALL</td>
<td>The three scenarios</td>
<td>49</td>
</tr>
</tbody>
</table>
Evaluation: Role of interaction

Role of interaction: Does IGA produce improved prioritizations Compared to non-interactive requirement ordering?

IGA outperforms GA (and RAND), especially when a higher number of pairwise comparisons can be carried out
Apply to test case prioritization

- Several objectives to increment the “value for the user” (remind the discussion yesterday)
  - Maximize Code Coverage (low level artifact)
  - Maximize “Most Important Requirements” Coverage (high level artifact)
  - Minimize Execution Cost

- Advantages
  - Explicitly considers both structural (code) and functional (requirements) dimension at the same time
  - Identifies both technical and business critical faults early
  - Fills gap between low level and high level artifacts by means of traceability
The most important requirements

- Requirements
- Requirements Impl. costs
- Requirements dependencies

Interactive GA

Prioritized Requirements

user input

London, February 12th, 2013
Test cases prioritization

Collect information about objectives
- Discover traceability links between Test and Source
- Discover Traceability links between Test and Requirements
- Measure execution time of test case

Prioritize
- Test case estimation (Fitness) using Objective Function

London, February 12th, 2013
Conclusions

• SSBSE is a point of contact between requirements and testing

• Not only, also the exploitation of user knowledge is important in both cases
In the future

TO BE EXPLORED - SSBSE in:

• Risk analysis and mitigation strategies selection
• Normative requirements, were we have to choose among different ways of being compliant with a given law
Thank you