A Proposal for Enhancing User-Developer Communication in Large IT Projects

Ulrike Abelein, Barbara Paech
Institute of Computer Science, University of Heidelberg, Germany
{abelein, paech}@informatik.uni-heidelberg.de

Problem

User Requirement

An invoice must be delivered to the customer via email

End User

System Requirement

Should emails always be sent as the last step of a workflow-based system or should it be possible to send an email after an invoice generation?

Developer

Solution Ideas

We want to extend software development and project management methods by enhancing communication between users and developers to enable a better project integration and improve system success. Therefore we identified four aspects: granularity level on which to communicate with the users, trigger points when to start communication, representations of changes and means of communications.

1. Granularity Level
   - Communication with users is structured by the abstraction levels of Task oriented Requirement Engineering (TORE)
   - Most discussions will be on the domain level (e.g. changes on features)

2. Trigger Points
   - To start communication decisions taken in the refinement of agreed user requirements should be used as trigger points

3. Representation of Changes
   - Existing documentation for content representation and highlighting of occurring changes should be used

   - Changes to be approved by the management
   - Informing end users or management about changes
   - Changes to be approved or consulted by end users
   - All captured rationale of decisions should be available to all project members

Responsibility Matrix

<table>
<thead>
<tr>
<th>Abstraction level - based on [2]</th>
<th>Project level - definition of scope and resources of the project</th>
<th>Business process level - composition of activities in business processes</th>
<th>Task level - understanding of user responsibilities</th>
<th>Domain level - definition of system scope</th>
<th>Interaction level - distribution of activities between humans and system(s)</th>
<th>System level - internals of the application core and of the GUI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Changes / decisions in...</td>
<td>Cost allocation</td>
<td>Business processes</td>
<td>Responsibilities of users (roles and tasks)</td>
<td>To-be activities</td>
<td>Workflow of the system</td>
<td>Technology</td>
</tr>
<tr>
<td></td>
<td>Timing (go-live dates)</td>
<td></td>
<td></td>
<td>Features</td>
<td>User Interface (incl. Input/Output)</td>
<td></td>
</tr>
</tbody>
</table>

Mgmt. of users: R = Responsible, A = Approver, C = Consulted, I = Informed

Further Research

Open questions
- How can we represent the rationale of decisions?
- How can we represent changes in detail to draw comparisons to previous versions?
- How can we integrate changes or decisions of non-functional requirements?

Next steps
- Refinement of our method (e.g. when should which decisions be communicated to which user group?)
- Validation of our approach in case studies to ensure feasibility, if possible in real life IT projects.

References

This work is presented at the 5th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2012) at the ICSE 2012 Zurich, June 2nd, 2012