Usability Reporting with UsabML

Johannes Feiner, Keith Andrews

Usability Reporting with UsabML

Johannes Feiner[1] and Keith Andrews[2]

[1] FH JOANNEUM, Internet Technology,
Werk-VI-Straße 46, 8605 Kapfenberg, Austria
Johannes.Feiner@fh-joanneum.at,
http://www.fh-joanneum.at/itm
[2] Graz University of Technology, Inffeldgasse 16c, 8010 Graz, Austria
kandrews@iicm.edu,
http://www.iicm.tugraz.at/keith

Abstract. Usability practitioners conduct formative evaluations, such as heuristic evaluations and thinking aloud tests, to identify potential problems in a user interface as part of the iterative design cycle. The findings of a formative evaluation (in essence, a list of potential problems) are usually compiled into written reports and typically delivered as a PDF or Word document. A written report is convenient for reading, but makes it difficult to reuse the findings electronically. The usability markup language (UsabML) defines a structured reporting format for the results of usability evaluations. In agile software development the direct handover of usability findings to software engineers can speed up development cycles and improve software quality.
Usability managers can now enter the findings of formative evaluations into a new, web-based system called Usability Reporting Manager (URM). Findings can be exported in UsabML format, which in turn can easily be imported by software engineers into an issue-tracking system connected to a source code repository. UsabML can also be transformed into other formats such as HTML and PDF via stylesheets (XSL).

Keywords: formative evaluation, usability findings, exchange, XML, reporting format.

1 Introduction

Formative usability studies form an integral part of iterative software design and development. Heuristic evaluations or thinking aloud tests, for example, can help identify problems in a user interface. Usability practitioners typically compile the findings and deliver them as a written report, say in PDF format. This paper discusses the UsabML format and its applications to usability reporting, particularly with regard to reuse of usability data during software development.

The are many examples of written report styles for formative tests, for example Molich et al. [1] and the NIST IUSR Formative Project [2]. There is even an ISO standard [3] for written reports of summative tests (formal experiments). However, results on paper or in written reports are hard to reuse (see Figure 1).

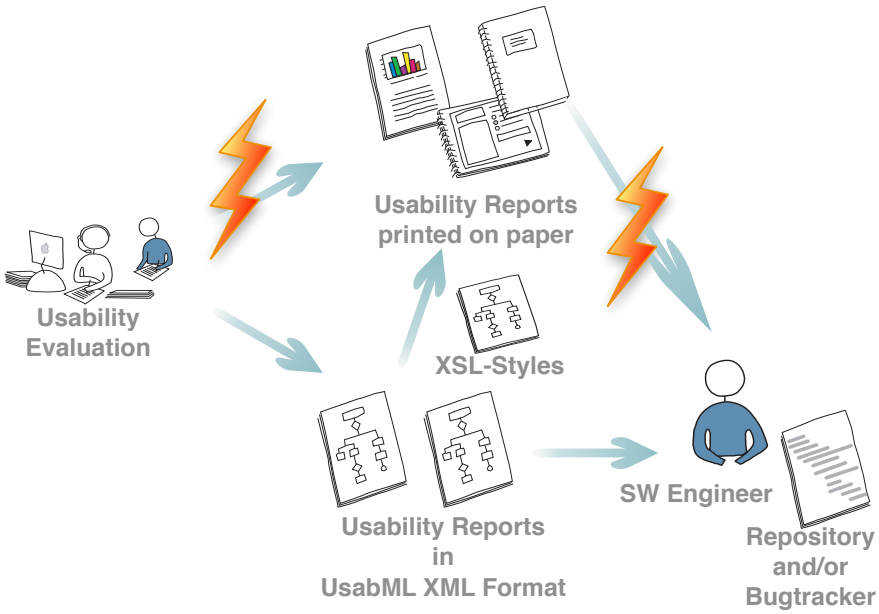

Fig. 1. Life-cycle of evaluation reports: The information exported via UsabML can be processed by other systems. For example, software engineers can import findings into a bug-tracking system.

It is difficult to import the findings into other systems, such as bug-tracking systems used by software engineers. In the age of agile development, fast feedback cycles are essential. The ability to automatically import usability findings into the issue-tracking systems associated with software code repositories would be extremely valuable.

The definition of the UsabML usability reporting exchange format is an important step towards formalising the reporting of usability results and will greatly simplify the handover of such findings to software development teams.

2 The UsabML Markup Language

For exchange of data, the extensible markup language (XML) is convenient, because XML can be readily parsed with standard tools and is extensible to further functionality without breaking existing code. Formulating UsabML in XML guarantees not only well-formed XML documents, but allows documents to be validated against the XSD schema provided. A tree hierarchy of tags defines the sections, subsections, and details for the various parts of typical usability evaluation reports. Tags hold their main information as XML text nodes. Tag attributes are used only for computer generated information or meta-information, such as unique identifiers, references to other tags, and sort-order.

```
1  ?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
       attributeFormDefault="qualified">
3    <xs:element name="project">
4      <xs:complexType>
5        <xs:sequence>
6          <xs:element name="title" />
7          <xs:element name="description" />
8          <xs:sequence id="reports" maxOccurs="unbounded">
9            <xs:element ref="report"/>
10         </xs:sequence>
11       </xs:sequence>
12        ...
```

Listing 1.1. Top elements of the schema definition file hold information about the project and one or more reports.

Separate UsabML schemas are defined for the two different kinds of reports currently supported: thinking aloud (TA) test and heuristic evaluation (HE). The overall outline of UsabML is shown in Figure 2, where the internal structures of heuristic evaluation and thinking aloud test reports are listed side-by-side, so differences within the schemas become evident. Within a schema, only very few sections are marked as `required` to give users of the UsabML standard more freedom to omit parts they do not require. The detailed schemas can be found online at the project web site [4]. Roughly speaking, the structure shown in Listing 1.1 is enforced by each schema. In particular:

- The root xml tag `project` (see Listing 1.1) holds general information (title, description) about a software project and one or more usability reports.
- The `report` xml tag specifies the kind of evaluation (for example "TA" or "HE"), further meta-information ("generated" timestamp), and the main report contents. This includes first the `title`, date/time of the report, `author`(s), `description` and `summary`, a general `introduction` and a description of the `methodology`. Some of these tags may hold pre-formatted text, which is used later as introductory text in the generated PDF or HTML reports.

For a HE report, the `heuristics`, `evaluators` their `environment` come next, followed by the `heuristicissues`:

- A `heuristic` section provides a fixed list of (about ten) heuristics. Many practitioners use those suggested by Jakob Nielsen [5]. More specialised sets of heuristics are sometimes used when evaluting specific kinds of interface, such as mobile devices or information dashboards.

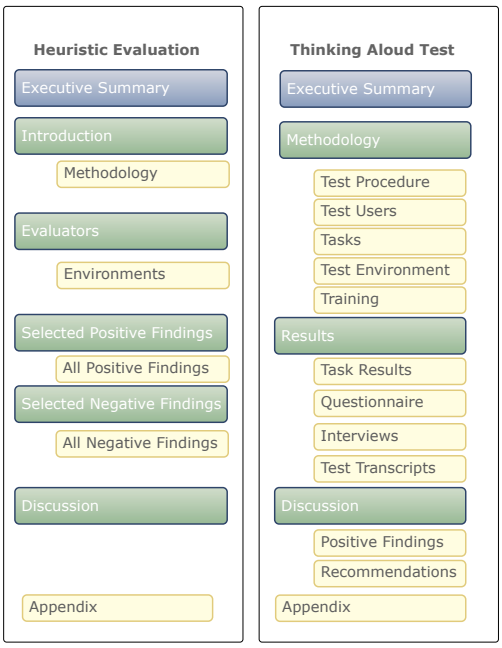

Fig. 2. The internal structure of heuristic evaluation and thinking aloud test reports.

– The `heuristicissues` section stores the positive and negative findings of the HE evaluators. Detailed information about findings including their `severity` and steps to `reproduce` them can be entered.

For a TA report, the `users`, `test environment`, `tasks` and `questionnaire` sections follow the introductory part. The findings are available in detail in the sections named `taskresults` and `questionnaireresults`:

– The `users` section holds several TA test `persons` with their profile (`gender`, `education`, `itexperience` and so forth). Information about test users is connected to the `video` clip(s) of their thinking aloud test.
– An `environment` section allows the specification of the environment used for the TA test, including the location and any hardware (computers, monitors, cameras) and software (screen capture) used.
– The section `tasks` denotes the tasks prepared in advance by the test team for the test users. A task requires a `title`, `description`, and `prerequisites` as well as `possilbesolutionpath`, `endingcriteria`, and `scheduledduration`.
– Details of the element `taskresult` are `start`, `end`, `actualduration`, and percent of `completion`.
– A `questionnaire` represents a list of `questions` and information about any rating scale (Very easy – Very hard) used. The answers to the questions are normalised values (say, points from 0 to 6 for a 7-point rating scale) stored for each answer per user.

– For each user, an `interview` and a `test transcript` may optionally be provided.

The final part of a report are the `discussion` section, which contains an interpretation of the evaluation results, and an `appendix`:

– The `discussion` section holds information about findings, including a list of recommendations. For a TA as well as for a HE report, there is the possibility to store details like `video` clips and `codereference`s.
– In the `appendix` the material used is referenced. URLs to online resources are given.

Several tags, for example `heuristicissues`, `findings`, or `transcript-logs`, are designed to hold one or more (`codereference`) items, indicating a source code location (say, a potential origin of a bug).

With all this detailed data at hand, it is easy to automatically generate both comprehensive and aesthetically pleasing reports, in say HTML or PDF formats. Note, that not all elements hold numeric information such as numbers and dates, many hold descriptive free text passages. Reports are generated by applying a stylesheet (XSL) transformation to the UsabML data. Different XSL stylesheets might be provided to generate highly customised reports for different target groups. For example, managers might generate just a short report with executive summary and a main overview. Programmers might export prioritised bug lists. Other stylesheets allow the generation of comma-separated values (CSV) files of selected data for later statistical analysis. Stylesheets to generate HTML are available at the project web site [4].

3 Usability Reporting Manager

To create a report in UsabML, evaluation managers could hand-edit XML directly. A more comfortable alternative is to use web-based software called the Usability Reporting Manager (URM). URM provides a web interface to enter, manage and export data in UsabML and is shown in Figure 3. Optionally, the exported reports can be styled with stylesheets for rendering in browsers as HTML.

The main web page is shown in Figure 4a and the details for entering information about recommendations can be seen in Figure 4b. The equivalent information rendered in HTML can be seen in Figure 4c. Further formats can be created simply by adding further style sheets (XSL files).

The URM software is open source and can be downloaded from the GIT repository [6]. More detailed specifications and documentation is available from the project web site [7].

4 Reuse and Exchange of Usability Findings

UsabML is great for reuse and exchange, because it is standardised XML and can be processed accordingly. It is easy to parse the files and data points can be

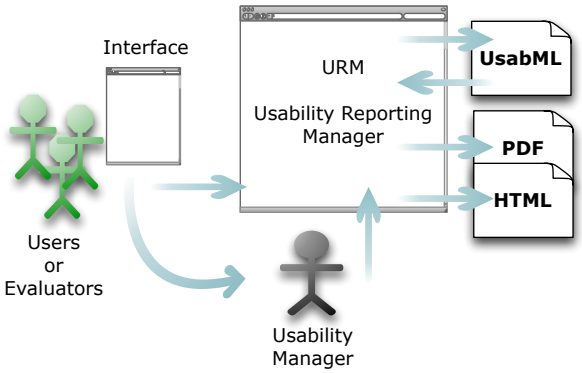

Fig. 3. URM enables evaluation managers to enter reporting data via a web interface.

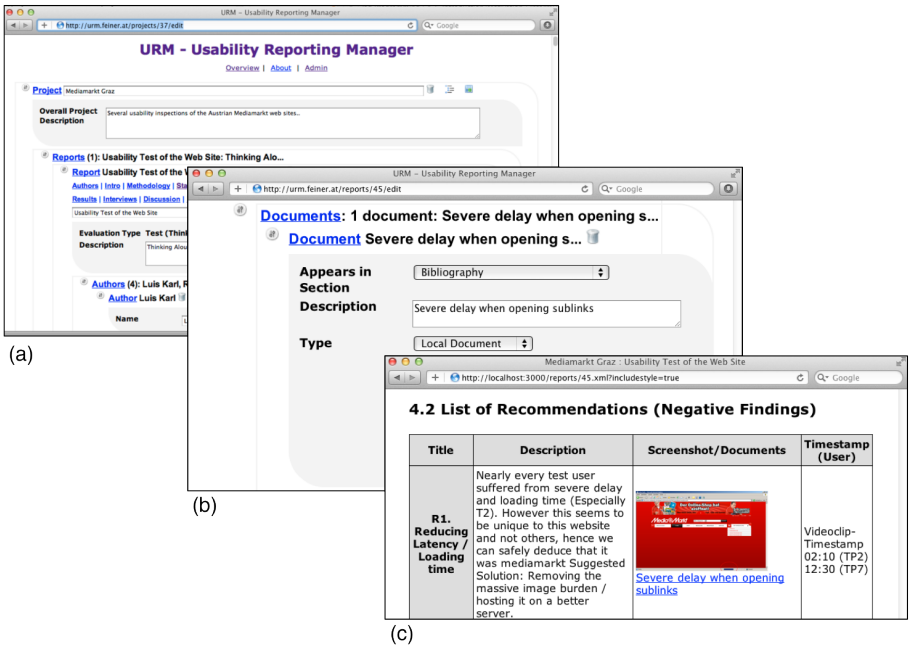

Fig. 4. (a) URM allows management of reports in a single, interactive AJAX and HTML5 enabled page. (b) Multimedia information, for example recommendations with screenshots and videos, can be entered. (c) The browser renders XML with selected XSL for pretty printing.

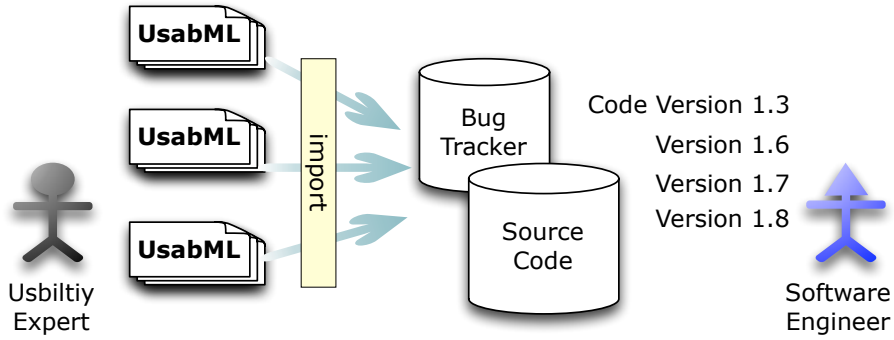

Fig. 5. Handover of evaluation results between different domains becomes possible by using standardised XML.

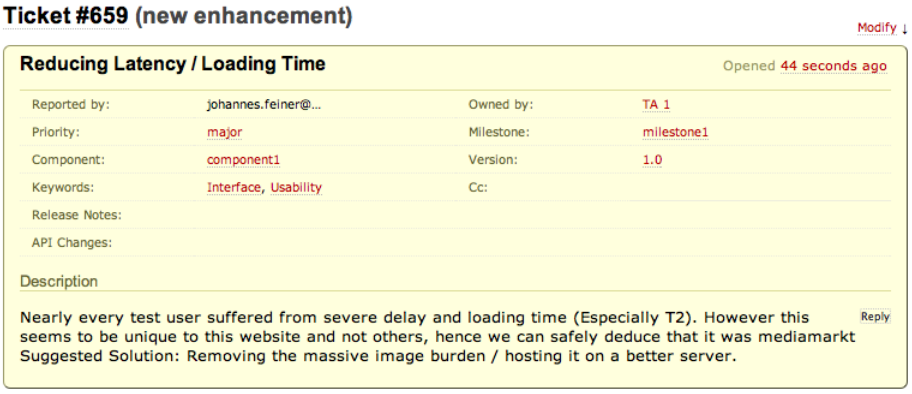

Fig. 6. Automated import into bug tracking systems becomes possible with scripts using web services to create new tickets out of UsabML.

extracted automatically at any given time. It is possible, for example, to imagine tools to compare evaluation results from different studies, like a `diff` on two data sets in respect of number of findings or severity of findings. This would allow a more automated approach to running studies like Rolf Molich's Comparative Usability Evaluation series [8].

UsabML supports the exchange of information between different domains. Usability testers can hand over their findings to software engineers, who then integrate the findings automatically into their bug tracking system. That is a huge improvement over reports on paper, where bugs must be extracted and entered into bug tracking systems manually. We are currently working on import tools (see Figure 5) for well-known open source bug tracking systems such as Trac, Bugzilla, and Mantis. Automated import can be accomplished by con-

verter scripts which parse UsabML and submit new issues via XML-RPC (Trac, Bugzilla), ReST (Bugzilla), or a SOAP web service (Mantis) as appropriate. The finding (problem report) shown in Figure 4c is converted into a corresponding Trac ticket in Figure 6.

5 Related Work

The importance of usability evaluation in general was promoted by Nielsen [9]. The use of usability evaluation to increase the quality of software is discussed in many papers such as Komiyama or Law [10,11]. Many books [12,13,14,15] advocate the use of usability methods. Howarth et al [16] discuss the need for tool support, especially for novice usability practitioners.

A collection of the most common usability evaluation methods is described in detail in Andrews [17]. For formative usability evaluation, the two classical methods are heuristic evaluation (HE) and thinking aloud (TA) testing. Both are supported by UsabML to manage and generate reports. A standard written report structure for reporting the results of summative evaluations (formal experiments) was developed by NIST as the Common Industry Format (CIF) [18] and is now an ISO standard [3]. A similar effort to standardise written reports of formative evaluations was started under the name IUSR Formative Project [2], but has yet to produce a draft standard (Theofanos [19]).

The Extended Structured Problem Report Format (ESPRF) for capturing problem reports during a usability inspection (heuristic evaluation) was introduced by Cockton et al. [20]. However, each ESPRF covers the reporting of a single problem rather than an entire structured report, and furthermore is limited to the case of heuristic evaluation.

The general idea of a markup language for usability findings to promote their electronic reuse was proposed in previous work by Feiner, Andrews and Krajnc [21].

As shown with the Usability Reporting Manager URM, evaluation managers can automate parts of their workflows using software tools. Loitzl [22] presents a web based software tool for data collection and reporting when performing heuristic evaluations.

UsabML generated out of URM focuses on formative evaluation and differs therefore from Wilson et al. [23] and Spacco et al. [24] who view issues more or less as standard software bugs.

Wilson et al. compare the approach of storing usability issues directly in the standard bug tracking system with the technique of creating a dedicated usability issues database. With URM we provide a standalone usability issues database with the possibility to export into bug databases, hence both approaches are supported.

Spacco implemented the idea of comparing issues from one software version to another by creating extensions for code analysis tools. In contrast, the UsabML format prepares for comparison between versions of usability evaluation data in a tool agnostic way, requiring XML comparison feature only.

6 Concluding Remarks

UsabML defines a new standard file format for formalised usability reporting. The XML format allows easy reuse of formative evaluation reports. It provides advantages for structuring a report and for transforming usability reports into different formats. Furthermore, it has the built-in possibility to store cross-references from usability findings to corresponding source code locations (these cross-references must be established manually by project managers). UsabML has the potential to bridge part of the gap between usability specialists and software developers and improve overall software quality.

With the URM web application, UsabML can be created on a server by entering data through the web. The information is stored in a central database and can be modified or appended any time. On demand, one or several reports in different formats can be generated. So, in future, a cycle of creation and reuse is feasible. The electronic handover (see Figure 5) of usability findings to software engineers is now becoming possible.

Although UsabML as described currently only supports the generation of HE and TA reports, it can of course in future be extended analogously to cover other forms of formative evaluation, such as cognitive walkthrough.

UsabML and URM will be evaluated in the near future during an upcoming project with industry partners. A case study will demonstrate the potential advantages of reusing usability data for development of real world software applications.

The UsabML specification, schema files, and example reports are available at the project web site [4].

References

1. Molich, R., Chattratichart, J., Hinkle, V., Jensen, J.J., Kirakowski, J., Sauro, J., Sharon, T., Traynor, B.: Rent a Car in Just 0, 60, 240 or 1,217 Seconds? — Comparative Usability Measurement, CUE-8. Journal of Usability Studies volume 6(1), pages 8–24 (2010), http://www.upassoc.org/upa_publications/jus/2010november/JUS_Molich_November_2010.pdf
2. NIST: Common Industry Format (CIF) IUSR Formative Project. National Institute of Standards and Technology (2010), http://zing.ncsl.nist.gov/iusr/formative/
3. ISO: SO/IEC 25062:2006 Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Common Industry Format (CIF) for Usability Test Reports. International Organization for Standardization (2006), http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43046
4. Feiner, J., Andrews, K.: UsabML: The Usability Markup Language (2010), http://usabml.fh-joanneum.at
5. Nielsen, J.: Ten Usability Heuristics (1994), http://www.useit.com/papers/heuristic/heuristic_list.html
6. Feiner, J., Andrews, K.: github - Usability Reporting Manager (2012), https://github.com/internettechnik/urm

7. Feiner, J., Andrews, K.: URM - Usability Reporting Manager (2012), http://itm.fh-joanneum.at/usabml/urm
8. Molich, R.: CUE - Comparative Usability Evaluation (2012), http://www.dialogdesign.dk/CUE.html
9. Nielsen, J.: Usability Engineering. Morgan Kaufmann (1993), ISBN 0125184069
10. Komiyama, T.: Usability Evaluation Based on International Standards for Software Quality Evaluation. Technical Report 2, NEC (2008), http://www.nec.co.jp/techrep/en/journal/g08/n02/080207.pdf
11. Law, E.L.C., Hvannberg, E., Cockton, G.: Maturing Usability: Quality in Software, Interaction and Value. Springer (2007), ISBN 1846289408
12. Barnum, C.M.: Usability Testing Essentials: Ready, Set...test. Morgan Kaufmann (2010), ISBN 012375092X, http://booksite.mkp.com/barnum/testingessentials/
13. Krug, S.: Don't Make Me Think!: A Common Sense Approach to Web Usability. New Riders, Second Edition (2005), ISBN 0321344758, http://www.sensible.com/dmmt.html
14. Rubin, J.B., Chisnell, D.: Handbook of Usability Testing: Howto Plan, Design, and Conduct Effective Tests. John Wiley & Sons, Second Edition (2008), ISBN 0470185481
15. Stone, D.L., Jarrett, C., Woodroffe, M., Minocha, S.: User Interface Design And Evaluation. Morgan Kaufmann (2005), ISBN 0120884364
16. Howarth, J., Smith-Jackson, T., Hartson, R.: Supporting Novice Usability Practitioners With Usability Engineering Tools. Int. J. Hum.-Comput. Stud. volume 67(6), pages 533–549 (2009), doi:10.1016/j.ijhcs.2009.02.003
17. Andrews, K.: Evaluation Comes in Many Guises. CHI 2008 Workshop on BEyond time and errors: novel evaLuation methods for Information Visualization (BELIV'08), http://www.dis.uniroma1.it/beliv08/pospap/andrews.pdf
18. NIST: Common Industry Format for Usability Test Reports. National Institute of Standards and Technology (1999), http://zing.ncsl.nist.gov/iusr/documents/cifv1.1b.htm
19. Theofanos, M., Quesenbery, W.: Towards the Design of Effective Formative Test Reports. Journal of Usability Studies volume 1(1), pages 28–45 (2005), http://www.usabilityprofessionals.org/upa_publications/jus/2005_november/formative.pdf
20. Cockton, G., Woolrych, A., Hindmarch, M.: Reconditioned Merchandise: Extended Structured Report Formats in Usability Inspection. In: Extended Abstracts on Human Factors in Computing Systems (CHI 2004), pages 1433–1436, ACM (2004), ISBN 1581137036, doi:10.1145/985921.986083
21. Feiner, J., Andrews, K., Krajnc, E.: UsabML - The Usability Report Markup Language. In: Proc. 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2010), pages 297–302, ACM (2010), ISBN 1450300839, doi:10.1145/1822018.1822065
22. Loitzl, M.: The Heuristic Evaluation Manager (HEM). Master's Thesis, Institute for Information Systems and Computer Media (IICM), Graz University of Technology (2006), http://www.iicm.tugraz.at/thesis/mloitzl.pdf
23. Wilson, C.E., Coyne, K.P.: The Whiteboard: Tracking Usability Issues: To Bug Or Not To Bug? Interactions volume 8(3), pages 15–19 (2001), doi:10.1145/369825.369828
24. Spacco, J., Hovemeyer, D., Pugh, W.: Tracking Defect Warnings Across Versions. In: Proc. International Workshop on Mining Software Repositories (MSR 2006), pages 133–136, ACM (2006), ISBN 1595933972, doi:10.1145/1137983.1138014