Evaluation Framework for Tools that Manage Requirements Inconsistency

Jyothi Iyer and Debbie Richards
Department of Computing, Division of ICS, Macquarie University, Sydney, Australia, 2109
{jiyer,richards}@ics.mq.edu.au

Abstract

Management of inconsistency in requirements is an important but effort intensive task. Automated tools promise greater reliability and less effort. However, after review of a number of surveys of current commercial requirements engineering (RE) tools we found that while some forms of inconsistency are considered in the evaluations, such as version control and spell checking, the surveys themselves do not address the deeper concerns that inconsistency raises. Hence there is a need for a framework to facilitate comparative evaluation of existing tools that could also serve as a requirements specification for a new or enhanced tool. Our framework and underlying reasoning are presented.

1. Introduction

Management of requirements through the software development life cycle is essential [1]. Management of inconsistency in requirements is a particular problem. Some of the factors that make the software development environment conducive to inconsistency include: diversity of stakeholders’ backgrounds, skills, knowledge, perceptions and goals; resource constraints limiting the time available for proper inconsistency checking; informal specification of requirements; and requirements evolution and volatility. However inconsistency is inevitable in real life and sometimes intentionally desired [2][3], therefore the task is not one of plain inconsistency detection and removal, but, one of managing inconsistency. Manual management is effort intensive and impractical. While complete automation of inconsistency management is unlikely, automated tools can be used to reduce the effort involved. The 2nd International Workshop on Living with Inconsistency [4] expressed the need for tools to support the management of inconsistency. A recent Standish Report [5] too pointed to a similar need.

Although research into inconsistency has been ongoing since the 1970s, automated tool support for inconsistency management in requirements is still in its infancy. We reviewed a number of surveys [6-9] of RE tools and found that while some forms of inconsistency are considered in the evaluations, such as version control and spell checking, the surveys themselves do not address the deeper concerns that inconsistency raises. Hence there is a need for a framework to facilitate comparative evaluation of existing tools that could also serve as a requirements specification to enhance or develop a tool. Based on the research literature into tool evaluation and inconsistency management, we have developed an evaluation framework that we present in this paper.

We commence in Section 2 with a discussion on what is meant by inconsistency. Section 3 details the evaluation framework and section 4 gives our conclusions and future work.

2. Inconsistency and its Management

A simple definition of inconsistency offered by Nuseibeh et al [1, 2] is any situation in which two descriptions do not obey some relationship that is prescribed to hold between them. A relationship between descriptions can be expressed as a consistency rule against which descriptions can be checked. Van Lamsweerde et al [10] point out that inconsistency is not necessarily a binary relationship. More than two descriptions can contribute to an inconsistency.

Finkelstein et al [11] looks at inconsistency from the view of overlaps between specifications and considers overlap and inconsistency to be at two different levels of interference between specifications, the first of which is a pre-requisite for the second. The existence of an overlap implies consistency rules between specifications[12], which if breached make them inconsistent.

Having reviewed a number of surveys of commercial RE tools, we found that inconsistency was being discussed in relation to traceability links between
requirements, version control, document quality checking and consistency of datasets exported and imported between databases. This is understandable as inconsistency is a term that can be used in different contexts. For example, in the context of a configuration management environment, Schwanke and Kaiser [13] classify inconsistency into six kinds. Van Lamsweerde [14] lists nine kinds of inconsistency, based on a goal-driven approach to RE. Robinson[15] gives two broad types as: syntactic and semantic inconsistency. Syntactic inconsistency are those caused by terminology inconsistency or improper grammar and semantic inconsistency concern conceptual meanings [15]. Inconsistency management has also been widely explored in the knowledge and database sciences [16-18]. However inconsistency in requirements is focused at a higher level of abstraction. For example, in a database we are concerned with a particular employee id being unique and consistently same in all relevant tables and records. At the requirements level we do not want to check the actual value but state generally that unique ids must be assigned. More closely related are concepts from meta modeling [19, 20] and deductive databases [21] which form a bridge between the high level abstractions found in requirements specifications and the practical implementation issues involved in maintaining consistent databases. As our scope is purely on supporting the requirements phase, and not implementation, we have not included Data or Database types of inconsistency.

Our focus is on a practical solution that can be supported by a tool. Description of the types of inconsistencies in requirements to be addressed would make tool support more realistic and feasible. Drawing from knowledge sciences, a notable paper by Gaines and Shaw [22] brings out the problem of eliciting knowledge from several experts in a domain and details the two most important factors as terminology and concepts. We found these broad categories covered the key notions and thus in our work we define the two main types of inconsistency as:

- **Domain Term Inconsistency** is when a single domain concept is described using different terms or a single domain term designates different domain concepts. For example, a stakeholder in a restaurant business, might state that, at the end of a meal a customer is presented with a check. Another stakeholder in the same restaurant might call the 'check' a 'bill' and state that, at the end of a meal a customer is presented with a bill. This type of inconsistency could also be viewed as terminological inconsistency.

- **Logical Inconsistency** is the most popular form of inconsistency and the one usually implied when type is not specified. A logical inconsistency is when a concept contradicts another concept. Logical inconsistency is irrespective of domains used. The most simple form of logical inconsistency is A and ¬ A, however, one could also have A and (part of ¬ A) which still results in logical inconsistency. For example, with the same restaurant example, a stakeholder might state that a customer gets a 20% discount on the customer's 10th visit to the restaurant and other discounts do not apply. While another stakeholder might state that the customer gets 10% discount on the 10th visit in addition to other discounts such as discount vouchers, etc.

What are the activities that constitute managing these inconsistencies in requirements? Tool support is not simply working on the content or data, but must also support the activities that constitute the task of requirements inconsistency management. How do we evaluate that a tool supports requirements inconsistency management? Both these questions are answered in the evaluation framework presented in the next section.

3. Evaluation Framework

Evaluation of software tools is not a new endeavour. Kitchenham [23-26] and Basili [27] laid down some valuable foundations for evaluating software. Kitchenham uses the term Feature Analysis to describe comparing different packages or tools feature by feature [24]. Feature Analysis is based on identifying the requirements that users have for a particular task/activity and mapping those requirements to features that a method/tool aimed at supporting that task/activity should possess. Kitchenham [24] brings out an important point, that a comparative framework is required so as to perform feature analysis. The evaluation framework proposed here serves that purpose.

Research literature has three frameworks describing the activities that constitute the task of inconsistency management one by Finkelstein et al [12], the second by Nuseibeh et al [1] and the third by Spanoudakis et al [28], which is a modified combination of the other two approaches. All three frameworks share the same core activities of inconsistency detection, diagnosis, handling and tracking, although Nuseibeh et al [1] do not explicitly state anything about tracking there is an indirect assumption that this will be done in their activity named measurement of inconsistency and
analyzing impact and risk. In the case of Finkelstein et al. [12] handling is referred to as resolution. The evaluation framework presented in this paper builds on these three frameworks, drawing also from INCOSE’s [6] good list of criteria for RE tools, in addition to incorporating several additional requirements identified by Grundy et al. [29, 30] and Van Lamsweerde et al. [31], which include: presentation and querying of data related to inconsistencies; the need to support multiple users; tool integration with other software tools; incremental building of specifications either top down or bottom up; richer structuring mechanisms such as stakeholder viewpoints; non-functional properties such as performance, usability and separation of concerns such as descriptive and prescriptive properties.

As can be seen, the criteria for evaluation are many and the evaluation framework could suffer from the problems identified by Kitchenham [24] of being time consuming, unwieldy, and therefore not easily usable. Striking a balance between the depth of understanding required to achieve the desired level of confidence in an evaluation and the practical difficulties in handling a large number of criteria is required. This framework proposes to assist the user by breaking the evaluation criteria into categories. Budgen et al. [32-34] defines three categories, namely evaluation against the tool’s design principles, evaluation against the needs of the user and evaluation against software engineering problems and their domains. Vessey and Sravanapudi [34] argue for a fourth category, evaluation against the work environment, in order to support collaborative work groups. The IEEE Std 1209:1994 [35] (standard for evaluation and selection of CASE tools) has 7 categories, namely reliability, usability, efficiency, functionality, maintainability, profitability and general, with functionality being the core functionality of the tool. The rest are very generic and expected criteria in tools. As we are concerned with a generic tool rather than a domain specific one, we restrict our focus to the tools functionality and usability. Usability being selected only because functionality required for requirements inconsistency management could be quite complex, however keeping it simple for the user would be the hallmark of its success. We have partitioned the criteria into the following 4 broad categories:

- Evaluation of tool’s core functionality of requirements inconsistency management (which is the software engineering problem and domain we are focusing on).
- Evaluation of tool’s support for real world problems within the context of requirements inconsistency management (the users needs within this context).
- Evaluation of tool’s usability within the context of requirements inconsistency management (usability being the key design principle we have chosen).
- Evaluation of tool’s support for collaborative working within the context of requirements inconsistency management.

The framework is open to customization as regards which categories to include and how to score. For example, not all projects involve collaborative working, research tools may ignore usability and prototypes may ignore real world issues such as scalability. Each question expects a ‘yes’ or ‘no’ answer, perhaps yielding a 1 or 0, respectively. Weights could be assigned to some or all questions. The higher the score the more suitable the tool.

### 3.1 Evaluation of the tool’s core functionality of requirements inconsistency management.

The criteria selected under this category relate to the core functionality required of the tool to support the RE analyst in doing the task of requirements inconsistency management. The activities that constitute the task of requirements inconsistency management is constructed from that found in research literature, specifically drawing from all the three inconsistency management frameworks found in [12] [1] [28] and other literature [31] [29, 30] as:

- Inconsistency Management Configuration,
- Inconsistency Detection,
- Inconsistency Diagnosis,
- Inconsistency Handling,
- Inconsistency Tracking,
- Inconsistency Measurement and Analysis and
- Modelling.

**Inconsistency Management Configuration** is an activity typically performed at the start of a project. Configuration options should control or direct the way inconsistency management is done with the possibility to change these options at any time during the lifecycle of the project. Configuration of the tool is required to specify:

- **Inconsistency Management Policies**, actions that will be taken by the tool when an inconsistency is detected. Two types of policies should be supported. Preventive Policy: Used when the RE analyst wishes to stop an anticipated inconsistency from ever occurring. The tool should reject any actions whose completion would cause the specific inconsistency to occur. Handling Policy: Used when the RE analyst wants to specify certain handling actions when an anticipated inconsistency occurs. See the Inconsistency Handling subsection.
• **Project Structure** and **Scope for Inconsistency Management** are both related. Most tools operate on a project basis i.e. an effective way of grouping all artefacts related to a project. Similarly dividing a project into sub-projects and hierarchically organising these projects is also required. Top down and bottom up approaches to problem solving can then be catered for. Local and global consistency checking can be performed on individual sub-projects and the entire project respectively. For example, initially the RE analyst may want to manage inconsistency only within a small subset of requirements, incrementally expanding this to include requirements in the entire project. Thus the need for a facility to control the **Scope for Inconsistency Management** where it should be possible to specify the individual subproject/s or project/s that constitute the scope of operation for inconsistency management at any given time.

• **Monitoring for Inconsistency** is specifying how often the inconsistency detection process is run, whether continuous, at specified intervals or times. For example, where a significant computational overhead is involved, monitoring need not be continuous, in which case there should be options available to the RE analyst.

**Inconsistency Detection** is picking up the violation of a consistency rule between two or more descriptions. Detection must be an automatic process, one that does not need any support from the user. However, the tools’ actual detection process should run as configured for monitoring for inconsistency. Detection of requirements inconsistency types such as ‘domain term’ and ‘logical inconsistency’ should be supported. Detection of inconsistencies specified in policies or those arising due to addition, deletion or modification of descriptions or those arising whilst handling other inconsistencies should also be supported.

**Inconsistency Diagnosis** is providing feedback on the inconsistency and facilitating the RE analyst do some investigation. Although diagnosis must essentially be an automatic process, one that does not need any support from the user, sometimes what the tool comes up with may not always make sense, hence the user should be permitted to do problem diagnosis too. The tool should be able to locate an inconsistency, identify the cause of an inconsistency and classify the type of inconsistency such as ‘logical’ or ‘domain term’.

**Inconsistency Handling** involves actions taken in response to the inconsistency encountered. Handling actions can range from simple trivial actions, such as modifications to descriptions, to complex handling actions, as suggested by Nuseibeh et al [1], which are:

- Ignore – When the cost or effort in fixing an inconsistency is too great relative to the adverse impact of the inconsistency, the RE Analyst may choose to ignore the inconsistency. In some cases, the inconsistency may be intentional or desired, ignoring the inconsistency provides the RE analyst with the facility to cater to that.
- Defer – More time may be required to attend to the inconsistency however in the interim the inconsistency should be tolerated so that it does not adversely impact the progress of the project.
- Circumvent – The tool may detect an inconsistency which the RE Analyst does not really regard as an inconsistency. This may be because the rules being broken are wrong in the first place or the inconsistency represents an exception to the rule that has not been captured. The RE Analyst may choose to circumvent the inconsistency by modifying the rules or disabling them for a specific scope of operation.
- Ameliorate – The RE Analyst may choose to improve a description containing inconsistencies without necessarily resolving them all. This may include adding information to the description that reduces the adverse effects of the inconsistency and/or resolves other inconsistencies as a side effect.

The ability to simulate the results of certain parameter values will assist with exploring and selecting the various options.

**Inconsistency Tracking** is the recording of the events and actions, and information associated with these, through the life span of an inconsistency. This is different from tracking requirements, which is discussed in section 3.4 under change management. Details to be recorded about inconsistency are (a) the reasoning underpinning the detection of an inconsistency (b) the source and cause of inconsistency (c) the handling actions that were considered and (d) the arguments underpinning the decision to select one of these options and reject the other.

**Inconsistency Measurement** is needed to ensure efficient and effective inconsistency management. Measures are needed such as the cost, benefit and risks associated with the various handling actions and the number and severity of inconsistencies. Measures relating to make a decision on which choice is “more consistent”, which inconsistencies are urgent are factors that are of interest. Being able to query and
report on this data is another feature that is important in tools. Modelling of the domain involving a description of the problem is a precursor to management of inconsistency. The tool should support the representation of a wide range of descriptions, whether diagrammatic or language oriented, descriptive or prescriptive. The tool should support describing the requirement, giving it a unique identifier, associating properties with requirements, such as priority, establishing links between requirements, hierarchical organization of requirements and structuring requirements into project and sub-projects. The tool should also support relatively new RE approaches such as viewpoints, goals and scenarios. In addition to building a requirements model, the tool should assist in building a domain model defining the domain terms used.

3.2 Evaluation of the tool’s support to real world problems.

A second part of the framework is consideration of whether the tool supports usage for real projects rather than just toy problems. Criteria here could be numerous, however limiting criteria to the most essential in this category is of importance, keeping in mind that this is an evaluation framework for an RE tool specifically concerned with requirements inconsistency management.

Checking for inconsistency will involve some computational overhead and as the number of requirements grows so too will the overheads, hence performance scalability is definitely a criteria. Of great importance is the data within the tool i.e. the requirements, the RE analyst may want to query this data for analysis or print reports or export/import data from/to the tool. A variety of software tools are typically used on modern day software projects and therefore tool integration is another important criteria. These are all real world problems. Thus, the specific criteria to be evaluated are,

- Performance Scalability
- Integration with other software tools
- Data Import/Export
- Presentation Output

3.3 Evaluation of the tool’s usability

Several authors have their own list of evaluation criteria for usability, for example, Nielsen’s 10 heuristics for assessors of HCI’s [36], Norman’s ‘Seven Principles for Transforming Difficult Tasks into Simple Once’[37] and Schneiderman’s 8 Golden Rules[38]. Bobeva et al [39] have quite concisely integrated the three sets of heuristic evaluation criteria namely Nielsen’s [36], Norman’s [37] and Schneiderman’s [38] into eight criteria. Dix et al [40] has provided a simplistic categorisation of usability principles into three categories, namely learnability, robustness and flexibility. ‘Usability Engineering’ can be quite detailed in its own right, however the focus of this project is not a detailed analysis of the usability of the tool, the focus of this project is the tool’s usability to a RE analyst in managing inconsistency of requirements. The tool should be easy to learn, use and flexible not just for the RE analyst but also for common lay users, such as the stakeholders who, for example, could have read-only access. The tool should provide good error handling and support with the ability to reverse actions when required, for example, on performing a mistake while handling inconsistency the RE analyst may need to rewind to a previous state. The authors propose to use the criteria suggested by Bobeva et al [39], however categorised as suggested by Dix et al [40], grouping some criteria together as also adding multi-tasking and access flexibility (web access, etc.) which are thought necessary for current generation tools. The specific criteria and sub-criteria to be evaluated are:

- Leanability
  - Documentation
  - Feedback
  - Consistency
- Robustness
  - Error Handling
  - Error Messaging
  - Reversal Actions
- Flexibility
  - Multi-tasking
  - Access Flexibility

3.4 Evaluation of the tool’s support for collaborative working

Typically a project team consists of more than one person and projects in today’s world are increasingly being done in multiple locations around the world. Tools generally store the requirements or data in an in-built repository of some sort, which is most usually a database. Thus the important factors are the number of people working on a project, the number of locations and the number of databases used in recording requirements. The RE analyst should be able to control access to data for different users and manage change
requests. Version control too should be supported. Recording 'who' has done ‘what’, ‘when’ and ‘where’ is important as this information will support the RE analyst in handling requirements inconsistency. Thus the tool should support the RE analyst in managing requirements inconsistency across people, locations and databases. The specific criteria and sub-criteria to be evaluated are:

- **Team usage**
  - Multi-User
  - Single-Site
  - Multi-Site

- **Configuration Management**
  - Change Requests
  - Change Logging
  - Access Control
  - Version Control

### Table 2: The Framework At A Glance.

<table>
<thead>
<tr>
<th>Category</th>
<th>Evaluation Criteria</th>
<th>Evaluation Sub-Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation of tool’s core functionality of requirements inconsistency management.</td>
<td>Inconsistency Management</td>
<td>Inconsistency Management Policies</td>
</tr>
<tr>
<td></td>
<td>Configuration</td>
<td>Project Structure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Scope for Inconsistency Management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Monitoring Setup</td>
</tr>
<tr>
<td></td>
<td>Inconsistency Detection</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Inconsistency Diagnosis</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Inconsistency Handling</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Inconsistency Tracking</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Inconsistency Measurement</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modelling</td>
<td></td>
</tr>
<tr>
<td>Evaluation of tool’s support for real world problems within the context of requirements inconsistency management.</td>
<td>Performance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Scalability</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Integration with other software tools</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Data Import/Export</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Presentation Output</td>
<td></td>
</tr>
<tr>
<td>Evaluation of tool’s support for collaborative working within the context of</td>
<td>Team Usage</td>
<td>Multi-User</td>
</tr>
<tr>
<td></td>
<td>Change Management</td>
<td>Single-Site</td>
</tr>
<tr>
<td></td>
<td>Change Requests</td>
<td>Multi-Site</td>
</tr>
<tr>
<td></td>
<td>Change Logging</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Learnability</td>
<td>Help and Documentation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Feedback</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Consistency</td>
</tr>
<tr>
<td></td>
<td>Robustness</td>
<td>Errors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reversal Actions</td>
</tr>
<tr>
<td></td>
<td>Flexibility</td>
<td>User Multi-tasking</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Access Flexibility</td>
</tr>
</tbody>
</table>

### 4. Conclusions and Future Work.

The framework we have presented is based on an in-depth review of literature both into requirements inconsistency management and tool evaluation. Viewed another way and turned around appropriately, the evaluation framework can be transformed to a set of requirements that can facilitate RE tool vendor’s develop/enhance their tool products. To determine whether the set of requirements are indeed representative of the needs of potential users, a detailed online questionnaire survey has been developed to be completed by RE practitioners and academics subscribed to an RE mailing list at re-online@it.uts.edu.au (around 750 subscribers). The questionnaire is structured along the lines of the evaluation framework with a set of proposed requirements for each criteria/sub-criteria of the evaluation framework. The survey begins by eliciting the background of the participant and requests scoring each proposed requirement using the scale “Mandatory, Optional, Delete or Do not care”. The survey can be found at http://kakadu.ics.mq.edu.au:8080/quiz/. Initial results show that we have found support for most of the features in our framework, but we have encountered some interesting findings such as little interest in an RE tool supporting measures on cost, benefits and risks with handling the inconsistencies. Our complete results will be published in a future paper.

### 5. References


[28] G. Spanoudakis and A. Zisman, "Inconsistency Management in Software Engineering: Survey and


