Computer-assisted Discrepancy Management –
A Case Study in Research Transfer to Industry¹

Javier Andrade, Juan Ares, Santiago Rodríguez
Facultad de Informática
Universidade de A Coruña
Campus de Elviña s/n
15071 A Coruña, Spain

Francisco J. López
TCP Sistemas e Ingeniería
C/ Fernández Caro, 7
28027 Madrid, Spain
www.tcpsi.com

Juan Pazos, Andrés Silva²
Facultad de Informática
Universidad Politécnica de Madrid
28660 Boadilla del Monte, Madrid, Spain

This paper reports on the “repackaging” and transfer to industry of research results in Viewpoint-Based Requirements Engineering (VBRE). Our experience, as indicated here, is a particular case of the problems and solutions that small research groups can find in their transfer from theoretical work to practical application. In particular, starting with a formal approach for discrepancy management, we first developed a “light-weight version” of VBRE and empirically validated their most important claims. Then we developed a software tool (discRman) that encapsulates the main points of the approach and, finally, partner organizations used it in small projects. We think that this “theoretical-empirical-light-weight-tool” sequence can be applied in similar situations, especially by small research groups seeking for transfer of their results to industry.


1. INTRODUCTION
Requirements engineering (RE) is the branch of software engineering that aims to identify the functions, goals and constraints of software-based systems (Zave, 1997). RE aims to acquaint software engineers with the problem environment, analysing needs and specifying the behaviour of the system. But problems are seldom small enough for there to be a single source of requirements

¹ This paper is a result of work performed under research contract P021005056 between the Universidad Politécnica de Madrid and La Caixa bank. We are very grateful to COFAGA for its invaluable help and participation in the pilot project described in this paper.

² Referring author: asilva@fi.upm.es
and, even if this is the case, there is no guarantee against inconsistency. Some proposals focusing on the study and analysis of this multiplicity of viewpoints have emerged in the RE field (a survey has been published by Darke and Shanks, 1996). These proposals go by the name of Viewpoint-Based RE (VBRE).

VBRE recognizes that there is a multiplicity of stakeholders that take part in a requirements process, inevitably leading to discrepancies (conflicts and inconsistencies). In this paper, we use the term discrepancy to embrace both conflicts and inconsistencies. Briefly, a conflict is any interference of one party with the interests of another party (Easterbrook, 1991) and an inconsistency is any situation in which some rule has been broken (Easterbrook and Nuseibeh, 1996).

The two main points of VBRE are that (i) discrepancies are not undesirable, as they can be used as a means to improve requirements elicitation (Nuseibeh, Easterbrook and Russo, 2000; Nuseibeh, Kramer and Finkelstein, 1994) and (ii) discrepancies require (not necessarily immediate) actions to handle them. Systematic RE approaches for dealing with discrepancies need to properly diagnose each discrepancy found, before making any decision about what to do next. This diagnosis includes locating the discrepancy, identifying the cause of the discrepancy and classifying the discrepancy. Once diagnosed, discrepancies should be handled accordingly (Nuseibeh, Easterbrook and Russo, 2000; Spanoudakis and Kim, 2002).

In a previous proposal (Silva, 2002) we advocated that, before attempting to find discrepancies between two statements, their use should be clarified. We built our discrepancy-management approach over an RE model that actually advocates a multiplicity of uses. In this way, we can avoid the “monolithism” common to other approaches. One of the goals pursued by this process is to be able to deal with viewpoints expressed in a variety of formalisms, ranging from natural language to formal specification. This would free our framework from the idiosyncrasies of each particular modelling language, although at the cost of being more coarse-grained.

However, the aim of this paper is not to introduce that method. We are interested here in the fact that this method was not directly usable in its original, theoretical form. In particular, our interests are threefold:

1. To show empirical validation of the key assumptions of our approach.
2. To show how, in order to pave the way between theory and industry, we “light-weighted” our approach and subsequently built a software tool.
3. To report on the actual usage of the tool-supported approach in real projects.

We think that the process we followed for transferring research results in RE to industrial practice can be generalized into a “repackaging” model, which can be very useful for small research groups like ours. The word “repackaging” here is used with the meaning given by Baber (1997) who, in his parallel analysis of the evolution of electrical and software engineering showed that, before widespread application, theoretical work must be simplified, reorganized and reformulated. An example took place in electrical engineering, after Heaviside reformulated Maxwell’s theory some years after its publication. Before Heaviside, however, Maxwell’s theory required mathematical knowledge and abilities that few engineers had, making it almost unusable in practice. This reorganization of theoretical work provided real practical advantages, but not theoretical ones. Similar examples can be found in the history of other engineering disciplines, where theoretical work is normally restructured, repackaged and organized to facilitate, or enforce, its regular application (Baber, 1997). However, the importance of this “repackaging” step is frequently overlooked in software engineering research. A notable exception is David Parna’s approach to formal methods using tabular notations (Parnas, 2003). This tabular notation does not make a
difference, theoretically speaking, as nothing can be expressed with tables that can not be expressed without them. However, they are easier to read and write, and people make fewer mistakes, thus providing a very useful tool for documentation and during inspections.

This paper is structured as follows: In Section 2 we introduce our approach to discrepancy management, including the hypothesis on which it is based and the repackaging performed; in Section 3 we report on the empirical validation performed; Section 4 shows the tool support added and, finally, Section 5 presents some conclusions.

2. BACKGROUND

The three main tasks that a discrepancy management process should perform are discrepancy detection, diagnosis and handling. In the context of a standard RE process, composed of elicitation, analysis and negotiation, specification and validation (Sommerville and Sawyer, 1998), discrepancy detection is performed at the analysis and negotiation phase. Clearly, it should be done after the requirements were elicited and captured (in a tool, for example) but before they are validated, or as a previous step to validation. As for the resolution tasks, they can be seen as being performed in parallel, or as part of the requirements validation phase. Discrepancy management can benefit all projects. It is more useful, however, when a high number of stakeholders are involved.

In most proposals, however, discrepancy management tasks are carried out without taking into account that an RE process manages different categories of information (Zave and Jackson, 1997; Parnas, 1995). The approaches to VBRE in the literature implicitly assume that the contents of each viewpoint belong to one, and only one, category (goals, descriptions, specifications, etc.). Some of these methods explicitly deal with goals (Robinson and Pawlowsky, 1999; Lamsweerde and Letier, 2000). Others deal with descriptions that have been represented in a particular modelling formalism (Spanoudakis and Finkelstein, 1997), or provide a framework for managing multiple viewpoints (Nuseibeh, Kramer and Finkelstein, 1994), but do not explicitly recognize the existence of different kinds of information as M. A. Jackson, for example, advocates (Jackson, 1995). So, from that point of view, with a focus on finding discrepancies between goals (when, according to Zave and Jackson, 1997, RE does not consist exclusively of goals) the fact that domain descriptions could also be discrepant is not considered. The possibility of having conflicting goals because there were discrepant visions of the domain is, consequently, not explored, further harming the analysis of, for instance, tracing the roots of inconsistencies between requirements back to inconsistencies between environment descriptions.

In RE, very often the use of a particular statement is not clear. For example, when developing a lift control system, we could elicit two paragraphs like:

1. “An available lift is always stationary and has its doors open. There should be always one available lift.”
2. “Available lifts are always stationary and at least one of them is available. If a lift is available then its doors are closed.”

These sentences mean nothing if we ignore their use. Are they descriptions? (i.e. statements that aim to describe a chunk of reality, whose truth value is unrelated to the existence of any future software system); are they requirements? (i.e. statements that express goals, whose truth value will depend on a future software controller). Is there a mix of requirements and descriptions in each paragraph? If, for instance, both of them are requirements, the relevant question is: how could a software controller enforce both? But the important question for VBRE is: should we compare one paragraph against the other if their use is not clear? For instance, both expressions could be
descriptions of actual facts (lifts could actually behave in one way, or the other). In this case, one of them must be wrong, and the way to elucidate which one is wrong is by means of refutation (checking them against the facts). On the other hand, both expressions could be goal statements. In that case, the sentences cannot be checked against reality. However, we could start a negotiation process to settle the conflict. If one sentence is descriptive and the other is a goal, we should not compare one against the other, as this comparison makes no sense. The fact that they are expressed in natural language is purely accidental. They could be translated to a more formal, or semiformal, notation and their use would still be unclear.

So, the main question is: Could detection, diagnosis, and handling tasks be enhanced by previously identifying the category of the potentially discrepant statements? With regard to checking for discrepancies, relevant viewpoint proposals, like those cited above and others (Hunter and Nuseibeh, 1998; Leite and Freeman, 1991; Easterbrook and Nuseibeh, 1996) focus primarily on the syntactic side, without paying attention to the different uses and/or, categories of the information involved. In our opinion, however, distinguishing between different categories of the information managed in a requirements process is a first step towards taking into consideration the semantic side, something that has been asked for in the literature (Leite, 1996). Thus, in our proposal we considered the use of each chunk of information managed in the requirements process.

2.1. The underlying hypothesis

To summarise the assumptions underlying our approach, now we will express them in the form of question and hypothesis. We will also present how we materialize this hypothesis into a VBRE setting by means of the KSR model (Zave and Jackson, 1997) and how we empirically tested it.

**Question:** Is discrepancy management enhanced by taking into account the different categories of statements managed during the requirements process, in order to compare one statement against another properly?

The hypotheses are:

**H0:** There is no significant difference between (A) the efficiency of subjects searching for discrepancies in a document that does not contain categorized statements and (B) the efficiency of subjects searching for discrepancies in a document where the statements are categorized according to KSR. Formally:

\[ H_0 : mean(Efficiency(A)) - mean(Efficiency(B)) = 0 \]

**H1:** There is a significant difference between (A) the efficiency of subjects searching for discrepancies in a document that does not contain categorized statements and (B) the efficiency of subjects searching for discrepancies in a document where the statements are categorized according to KSR. Formally:

\[ H_1 : mean(Efficiency(A)) - mean(Efficiency(B)) < 0 \]

If the hypothesis \( H_1 \) is certain, then a previous structuring of the viewpoints, according to those different categories, would be a solid basis for a discrepancy management process. With the roots in that hypothesis, we developed a complete method for discrepancy management (Silva, 2002). For the reader’s convenience, a summary of that method is presented in this section. Section 3 shows how we empirically validated the hypothesis.
2.2. Why the KSR model?
An incredible amount of information is managed in almost every RE effort. In the literature, there are some approaches to RE that explicitly recommend the classification of statements into different categories. Some of these approaches are semi-formal (Alexander and Stevens, 2002), while others are formal (Parnas, 1995). We have chosen the KSR model since it avoids some problems posed by the other approaches (Gunter, Gunter, Jackson and Zave, 2000) and its clarification potential is well-known in the RE community.

The KSR model, proposed by M. A. Jackson and others (Jackson, 1995; Gunter et al, 2000; Zave and Jackson, 1997), distinguishes between three categories of statements, contained in three sets:

- **K**: Contains descriptive sentences. These sentences are true irrespective of what the system will or will not do. For example “water boils at a temperature of 100 degrees, at pressure of 1 atm.”.
- **S**: Describes the external behaviour of the machine, that is, describes the interface between the machine and its environment. Defining the contents of S is a design task. Only shared phenomena can be part of the sentences contained in S. For example: “when the software reads the temperature sensor and detects a value above S, it will send a deactivation signal to the boiler”.
- **R**: Contains the requirements. Requirements, or goals, describe the desired effects of building and deploying the system in its intended environment. For example: “water in the boiler should always be below the boiling point of water at a pressure of 1 atm.”.

The logical/formal relationship that, ideally, should hold between these three sets is $K, S \models R$ so, given the domain $K$ and the specification $S$, the requirements $R$ will be met, i.e. they will become true in the environment.

The KSR model is more complex than the summary presented here (for a full description, see Gunter et al, 2000). However, the important point for our purposes is the distinction between $K$, $S$ and $R$. Also, despite its formalism, this model has been successfully applied to non-formal requirements documents, expressed in natural language (Kovitz, 1999).

If we classify viewpoint contents into the categories $K$, $S$ and $R$ before checking for discrepancies, we can make comparisons between each category (shown in Figure 1), and relate the

![Figure 1: KSR Comparison](image-url)
discrepancies found in one category to another (this relation will reveal powerful insights into the reasons behind each discrepancy, as we will show later), classify the discrepancies based on the categories affected and generate discrepancy settlement tasks fitted to this classification. We can also avoid comparing statements that belong to different categories. In order to find discrepancies, it is not sensible to compare statements in $K$ against those in $R$ or in $S$.

2.3. Discrepancy Management Process

For the approach presented in this paper to become useful in real-world requirements engineering, we need to pack everything together in a process that is able to identify discrepancies between two viewpoints, classify these discrepancies and generate resolution tasks. The process we propose for dealing with two discrepant viewpoints is represented in Figure 2. The three phases of the process are:

- **Phase 0**: Preliminary. In this phase, information is collected in viewpoints $VP_i$ and $VP_j$ and structured according to the categories $K$, $S$ and $R$. Each viewpoint should be internally consistent, verifying $K, S \cup R$. Overlap and discrepancy criteria are also defined in this phase.

- **Phase I**: Detection/Classification. This phase produces a list of all the discrepancies found, including the affected elements and their classification according to the criteria shown in Section 2.3.1. For a formal, detailed description of this phase, see Silva (2002), where Phase I is algorithmically specified.

- **Phase II**: Resolution. Taking as input the original viewpoints $VP_i$ and $VP_j$ and the list of discrepancies between them, this phase attempts to solve these discrepancies. The outputs of this phase are the modified viewpoints and a list of unresolved discrepancies, as not all discrepancies should be solved at this first stages (Nuseibeh et al., 1994). The proposed resolution process is based on a vocabulary of tasks that will be shown in Section 2.3.2.

Figure 2 shows that Phase II could be done as many times as needed until an adequate number of discrepancies have been solved. The modified viewpoints that result after Phase II could enter Phase I again, as the resolution of one discrepancy could lead to new discrepancies.
However, there are some points of flexibility in the method that can be conveniently tuned for making the method more usable and transferable than in its original, theoretical form. This will be the topic of Section 2.4.

2.3.1. Classification of Discrepancies

In our proposal, each viewpoint \( VPi \) is a triplet composed of three sets, each of which corresponds to one of the three categories \( K, S \) and \( R \). More formally:

\[
VPi = (Ki, Si, Ri) \\
Ki, Si \models Ri
\]

This is not an unrealistic way of approaching VBRE since each agent associated with a particular viewpoint has more than goals only. Those agents also possess domain and specification knowledge, which is potentially discrepant with another viewpoint. Hence, when comparing one viewpoint \( VPi \) against another viewpoint \( VPj \), statements classified into different categories should not be compared. For instance, it makes no sense to compare a descriptive statement (from \( Ki \)) against a requirement (from \( Rj \)), as this is tantamount to comparing “this is blue” with “paint it blue!”, as discussed.

Making comparisons between statements of the same category leads to a classification of discrepancies as shown in Tables 1 and 2. These tables show the different classes of discrepancies, depending on the sets concerned. In the notation used to name each discrepancy class, the existence of discrepant elements between sets is indicated by an apostrophe (‘). For example, if a discrepancy affects only the contents of sets \( Si \) and \( Sj \), this discrepancy is labelled \( KS'R' \). If the discrepancy affects \( Ki \) and \( Kj \) and \( Si \) and \( Sj \), it is labelled \( K'S'R \), and so on.

Table 1 shows the pure discrepancies: discrepancies that only affect one of the three information categories. Table 2 shows the mixed discrepancies, where two or three of the categories are affected. Pure discrepancies are less interesting than mixed ones, as their cause can be traced to some kind of spelling or typographical error. For example, it is not possible to have a discrepancy between elements of \( Si \) and elements of \( Sj \), with no discrepancies in \( K \) and \( R \) and, at the same time, verify \( Ki, Si \models Ri \) and \( Kj, Sj \models Rj \). Instead, when a mixed discrepancy appears, \( K, S \models R \) could still hold in both viewpoints, and no typographical errors are necessarily involved. The roots of a mixed discrepancy are always deeper than the roots of pure ones. Examples are given in Silva (2002).

<table>
<thead>
<tr>
<th>Notation</th>
<th>Discrepancy Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>K’S’R</td>
<td>Domain Discrepancy</td>
</tr>
<tr>
<td>KS’R</td>
<td>Specification Discrepancy</td>
</tr>
<tr>
<td>KSR’</td>
<td>Goal Discrepancy</td>
</tr>
</tbody>
</table>

Table 1: Pure discrepancies

<table>
<thead>
<tr>
<th>Notation</th>
<th>Discrepancy Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>K’S’R’</td>
<td>Domain-to-Specification Discrepancy</td>
</tr>
<tr>
<td>K’S’R’</td>
<td>Decontextualization Discrepancy</td>
</tr>
<tr>
<td>KS’R’</td>
<td>Requirements-to-Specification Discrepancy</td>
</tr>
<tr>
<td>K’S’R’</td>
<td>Total Discrepancy</td>
</tr>
</tbody>
</table>

Table 2: Mixed discrepancies
2.3.2. Resolution Tasks
In VBRE, there are studies regarding the “anatomy of a consistency check” (Easterbrook, Finkelstein, Kramer and Nuseibeh, 1994). We think that it is also very important to analyse the anatomy of a discrepancy resolution. With this objective in mind, and taking into account the categories of the KSR model, we make a distinction between the different atomic tasks that could be combined in a particular way to define a resolution process. A discrepancy between descriptions is settled differently to discrepancies between requirements or specifications. Similarly, the consequences of solving one or another will affect the future system in different ways.

Therefore, the categories affected by a discrepancy and the classification of discrepancies provide the basis for choosing an adequate combination of tasks aimed at a resolution. The tasks that can lead to discrepancy resolutions are shown below. These tasks were obtained after analyzing the resolution process for different examples of discrepancies, like those shown in Section 2.3.1. Note that these tasks are not excessively detailed and not even straightforward to apply (it would take a whole book to describe all the facets of a task like “negotiation”). However, it is useful to consider the differences between them when solving discrepancies:

• **Refutation.** This task compares discrepant sentences against reality to see which of them is true. In many cases, comparing a statement against reality (“the facts”) is not feasible, but the discrepancy can be solved by appealing to a well-respected source of information (for example, a book on the subject, a procedures manual, etc.). This task is applied when the discrepancy is between elements of $K$, and it makes no sense for it to be applied to $R$ or $S$, which do not refer to actual facts.

• **Transference.** This task removes the incorrect knowledge $K$ after it has been recognised as incorrect, and substitutes this incorrect knowledge with a statement, acknowledged to be correct.

• **Internal Checking.** This task determines whether a viewpoint is internally consistent. In particular, this task should be enacted after transference, as new knowledge could lead to internal inconsistencies.

• **Negotiation.** This task is aimed at reaching an agreement with respect to the goals of the system. For this reason, it is applied only when there are discrepancies between the requirements $R$ of different viewpoints. It is important to differentiate this task from the refutation task. Refutation compares statements against reality and negotiation involves two viewpoints with conflicting views about the actions that the system should carry out.

• **Acceptance.** In this task, the requirements of one, or both, viewpoints are modified, after the negotiation task leads to an agreement with respect to the requirements. This is because every agreement should lead to a change in the requirements of at least one of the viewpoints.

• **Adaptation.** This task should be enacted when a discrepancy between requirements is due not to different, conflicting goals, but to a disagreement between different perceptions of the environment. It modifies the requirements to adapt to the new situation.

• **Redesign.** When descriptions of external phenomena change, this task should modify the external description of the system ($S$) to adapt to the new situation.

Resolution is based on matching tasks to discrepancies as shown in Table 3. For each discrepancy, the column should be read from top to bottom, and each cell that contains a bullet (●) indicates that the respective task should be performed. Thus each discrepancy class is related to a resolution process consisting on the sequential chaining of the tasks shown in its corresponding column.
2.4. A light-weight version of the approach

We recognized that, for real use in real projects, the method must first be “repackaged” and “light-weighted”. In its original form, the exposition of the method was quite formal and “academic”, and illustrated by its application to statements expressed in logic. However there are some aspects of the method that could be conveniently adapted to make it more practical, including tool support:

- First, a convincing argument was needed in order to “sell” the advantages of the approach. The results of our experiment, shown in next section, played an important role here.
- The criteria that establish when there is an overlap and an inconsistency can be weakly defined. Also, they can change during the process, being weaker in the first stages and stronger later, when more information is collected and formalized. In particular, when statements are recorded in natural language, as it is common practice in industry, the criteria are in the hands of the user. In this case, a discrepancy can be regarded as any relevant difference that requires further clarification actions.
- A proper software tool was needed, in order to help with the process. The tool should incorporate all the flexibility added to the method, in order to be easily accepted. Specifically, the algorithmic approach to Phase I discussed in Silva (2002) can be softened in order to be carried out by the user in collaboration with the software tool.
- A core requirement for our tool was to offer an interface similar to that of traditional Requirements Management tools like Doors or IRqA. With these goals in mind, we developed discRman (standing for discRepancy manager).
- The tool incorporated many of the elements of discrepancy management given by Nuseibeh et al (2000). Elements included were, mainly, for labelling the discrepancies with useful attributes, such as prioritization. This is a practice that, in our opinion, is easy to understand and accept by those requirements engineers used to the prioritizing and labelling requirements of their projects.
- In the method, classification of discrepancies leads to the tasks used to solve them. However, those tasks should not be enforced. On the contrary, they should be used as a suggestion or as a guide for monitoring the evolution of the discrepancy resolution process. In fact, the users wanted to add even more flexibility to the tasks, as we will show later.

It is also worthy of note that, at the same time, we also “repackaged” the KSR framework (Zave and Jackson, 1997) for use in a non-formal, discrepancy management, setting.
3. EMPIRICAL VALIDATION OF THE HYPOTHESIS

The main hypothesis of our approach, presented in Section 2.1 claims that, to improve discrepancy management, it is better to first categorize the statements to be compared. As stated, the hypothesis aims towards improving the effectiveness and the efficiency of discrepancy detection, as this is the basis (and, at the same time, the bottleneck) of discrepancy management. In this section we present the empirical validation we performed with regard to that hypothesis.

A serious problem in VBRE is the explosion of the number of comparisons that must be made to detect the discrepancies (Robinson and Pawlosky, 1998). When there are \( n \) statements to compare, the number of comparisons is \( n(n-1)/2 \). However, partitioning into three sets \((K, S \text{ and } R)\) can reduce the comparison effort. If we compare only between categories then, in the hypothetical case that there are \( n/3 \) elements in each set, the number of comparisons to be made is \( n((n/3)-1)/2 \). For example, if \( n=30 \), instead of 435, only 135 comparisons are needed.

However, this algebraic argument is not enough. If we want to go beyond the theoretical arguments given here in favour of classifying the statements before comparing them, we need to check if VBRE under the hypothesis proposed is, or is not, a more effective and efficient way of working for humans reviewing requirements expressed in natural language, as is usually the case.

In particular, as we intended to build a software tool for “computer-assisted discrepancy management” based on such assumptions, it should be better first to statistically test the efficiency and effectiveness of this “first classify, then compare” approach, in order to bring more evidence in favour of our proposal and convince potential adopters.

For testing the efficiency and effectiveness of non-automated discrepancy detection with KSR-structuring of viewpoints we performed an experiment to test the hypothesis using two groups, each consisting of 30 students. The efficiency of a “human discrepancy detector” was measured as the number of discrepancies found per minute. The effectiveness was measured as the percentage of actual discrepancies found. For the purpose of this experiment we used a weak discrepancy criteria, as the statements to be compared were expressed in natural language. We told students that a “discrepancy” is any pair of statements that differ in such a way that they deserve more exploration, like new elicitation steps, a negotiation step, checking some facts, etc. The artefacts used in the experiment, along with more information, can be downloaded from: http://www.ls.fi.upm.es/UDIS/miembros/asilva/Experiment_Artifacts.zip

From the point of view of correctness, one of our goals was to allow replication of this statistical experiment, as recommended by Kitchenham, Pfleeger, Pickard, Jones, Hoaglin, El Emam and Rosenberg (2002). With regard to the statistical software, we used StatGraphics Plus 5.0 to help us with the study. The steps we performed to carry out the experiment are shown below:

1. We collected data from 60 students of last year’s degree in computer science. All 60 individuals had neither previous experience nor knowledge about discrepancy management and VBRE. They had never been exposed to practical RE before. Their only contact with RE was four hours of a theoretical introduction to the subject, including: importance of RE, RE cycle (according to Sommerville and Sawyer, 1998), methods, techniques, etc.

2. We provided each student with a document containing two viewpoints. Each viewpoint consisted of 21 statements in natural language about the process of building computers from different parts, intended to develop a software system for a company that builds and sells computers. This medium-size problem was used two years ago as an assignment project to the students of Software Engineering at the Universidad Politécnica de Madrid.
3. In our experiment the task assigned to each student was to compare the two viewpoints and find pairs of discrepant statements (as shown in Section 2.4). However, two different treatments were considered and the students were randomly assigned to them: (i) a first group of 30 randomly

<table>
<thead>
<tr>
<th>Discr.</th>
<th>Minutes</th>
<th>Efficiency</th>
<th>% Effectiveness</th>
<th>Discr.</th>
<th>Minutes</th>
<th>Efficiency</th>
<th>% Effectiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>45</td>
<td>0.177</td>
<td>66.667</td>
<td>10</td>
<td>30</td>
<td>0.333</td>
<td>83.333</td>
</tr>
<tr>
<td>11</td>
<td>30</td>
<td>0.366</td>
<td>91.667</td>
<td>9</td>
<td>15</td>
<td>0.6</td>
<td>75</td>
</tr>
<tr>
<td>7</td>
<td>43</td>
<td>0.162</td>
<td>58.333</td>
<td>9</td>
<td>34</td>
<td>0.264</td>
<td>75</td>
</tr>
<tr>
<td>9</td>
<td>50</td>
<td>0.18</td>
<td>75</td>
<td>8</td>
<td>20</td>
<td>0.4</td>
<td>66.667</td>
</tr>
<tr>
<td>10</td>
<td>35</td>
<td>0.285</td>
<td>83.333</td>
<td>10</td>
<td>25</td>
<td>0.4</td>
<td>83.333</td>
</tr>
<tr>
<td>8</td>
<td>45</td>
<td>0.177</td>
<td>66.667</td>
<td>8</td>
<td>20</td>
<td>0.4</td>
<td>66.667</td>
</tr>
<tr>
<td>8</td>
<td>50</td>
<td>0.16</td>
<td>66.667</td>
<td>8</td>
<td>40</td>
<td>0.2</td>
<td>66.667</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td>0.3</td>
<td>50</td>
<td>10</td>
<td>20</td>
<td>0.5</td>
<td>83.333</td>
</tr>
<tr>
<td>5</td>
<td>35</td>
<td>0.142</td>
<td>41.667</td>
<td>5</td>
<td>18</td>
<td>0.277</td>
<td>41.667</td>
</tr>
<tr>
<td>10</td>
<td>25</td>
<td>0.4</td>
<td>83.333</td>
<td>10</td>
<td>20</td>
<td>0.5</td>
<td>83.333</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
<td>0.2</td>
<td>75</td>
<td>9</td>
<td>30</td>
<td>0.3</td>
<td>75</td>
</tr>
<tr>
<td>7</td>
<td>90</td>
<td>0.077</td>
<td>58.333</td>
<td>7</td>
<td>45</td>
<td>0.155</td>
<td>58.333</td>
</tr>
<tr>
<td>10</td>
<td>40</td>
<td>0.25</td>
<td>83.333</td>
<td>11</td>
<td>25</td>
<td>0.4</td>
<td>91.667</td>
</tr>
<tr>
<td>5</td>
<td>18</td>
<td>0.277</td>
<td>41.667</td>
<td>9</td>
<td>13</td>
<td>0.692</td>
<td>75</td>
</tr>
<tr>
<td>6</td>
<td>70</td>
<td>0.085</td>
<td>50</td>
<td>6</td>
<td>45</td>
<td>0.133</td>
<td>50</td>
</tr>
<tr>
<td>10</td>
<td>70</td>
<td>0.142</td>
<td>83.333</td>
<td>10</td>
<td>40</td>
<td>0.25</td>
<td>83.333</td>
</tr>
<tr>
<td>6</td>
<td>25</td>
<td>0.24</td>
<td>50</td>
<td>7</td>
<td>15</td>
<td>0.466</td>
<td>58.333</td>
</tr>
<tr>
<td>7</td>
<td>30</td>
<td>0.233</td>
<td>83.333</td>
<td>8</td>
<td>23</td>
<td>0.347</td>
<td>66.667</td>
</tr>
<tr>
<td>8</td>
<td>30</td>
<td>0.266</td>
<td>66.667</td>
<td>8</td>
<td>15</td>
<td>0.533</td>
<td>66.667</td>
</tr>
<tr>
<td>7</td>
<td>30</td>
<td>0.233</td>
<td>58.333</td>
<td>7</td>
<td>15</td>
<td>0.466</td>
<td>58.333</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
<td>0.2</td>
<td>75</td>
<td>11</td>
<td>25</td>
<td>0.44</td>
<td>91.667</td>
</tr>
<tr>
<td>6</td>
<td>25</td>
<td>0.24</td>
<td>50</td>
<td>10</td>
<td>20</td>
<td>0.5</td>
<td>83.333</td>
</tr>
<tr>
<td>7</td>
<td>50</td>
<td>0.14</td>
<td>58.333</td>
<td>10</td>
<td>35</td>
<td>0.285</td>
<td>83.333</td>
</tr>
<tr>
<td>9</td>
<td>27</td>
<td>0.333</td>
<td>75</td>
<td>12</td>
<td>21</td>
<td>0.571</td>
<td>100</td>
</tr>
<tr>
<td>8</td>
<td>50</td>
<td>0.16</td>
<td>66.667</td>
<td>8</td>
<td>40</td>
<td>0.2</td>
<td>66.667</td>
</tr>
<tr>
<td>9</td>
<td>75</td>
<td>0.12</td>
<td>75</td>
<td>10</td>
<td>45</td>
<td>0.222</td>
<td>83.333</td>
</tr>
<tr>
<td>7</td>
<td>70</td>
<td>0.1</td>
<td>58.333</td>
<td>11</td>
<td>55</td>
<td>0.2</td>
<td>91.667</td>
</tr>
<tr>
<td>8</td>
<td>60</td>
<td>0.133</td>
<td>66.667</td>
<td>7</td>
<td>20</td>
<td>0.35</td>
<td>58.333</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>0.2</td>
<td>33.333</td>
<td>4</td>
<td>16</td>
<td>0.25</td>
<td>33.333</td>
</tr>
<tr>
<td>8</td>
<td>35</td>
<td>0.228</td>
<td>66.667</td>
<td>11</td>
<td>20</td>
<td>0.55</td>
<td>91.667</td>
</tr>
</tbody>
</table>

**Average:** 0.207 64.444 0.374 73.0555

**Std.dev.:** 0.079 14.171 0.145 15.733

Table 4: Raw data gathered from the experiment.
chosen students (Sample A) received a copy of the exercise with no classified statements; (ii) the other 30 students (Sample B) received the same exercise but all the statements were adequately classified into $K$, $S$, and $R$. The actual number of discrepancies in the exercise was 12. Of course, previously to the experiment, the students had no idea of how many discrepancies were in the document.

4. We gathered the number of discrepancies found and the time that each student needed for discovering them. Basically, at the end of the exercise, each student gave us a paper with all the discrepancies found (from this, we know the number and a description of each one). They also reported the time required to find the discrepancies, as at the beginning of the exercise they were instructed to measure it. The data are shown in Table 4 (we performed a data screening to check that there were no incorrect values or inappropriate data). The column labelled “efficiency” is the quotient of the first and the second columns. The column labelled “effectiveness” shows the percentage of real discrepancies found.

5. Efficiency: Using StatGraphics, we found that the distribution of the efficiency data does not show a significant departure from normality. According to the F-test we performed, the standard deviations were different for both samples. Then, we performed a t-test for comparing the means of efficiency for sample A and B. We found that, according to the P-value (0.00000106059 in this case), we can confidently say that the efficiency of the group of students who worked with $KSR$-classified statements is better than the efficiency of the students in the other group. More formally, the alternative hypothesis ($\text{mean(Efficiency}(A)) - \text{mean(Efficiency}(B)) < 0$) is supported with a 95% confidence interval $[-0.167153,-0.116342]$. So, on average, the data show an improvement of 68.4% in the efficiency of sample B with respect to A.

6. Effectiveness: Using StatGraphics, we found that the effectiveness data did not show a significant departure from normality. According to the F-test, the standard deviations are the same for both groups of data. Then, we performed a t-test for comparing the means of effectiveness for sample A and B. We found that, according to the P-value (0.0149059 in this case), we can confidently say that the effectiveness of those individuals who worked with $KSR$-classified statements is better than the effectiveness of the individuals in the other group. More formally, the alternative hypothesis ($\text{mean(Efficiency}(A)) - \text{mean(Efficiency}(B)) < 0$) is supported with a 95% confidence interval $[-8.6111,-2.14903]$. Given that the middle point of the interval is -5.38, the data show that the effectiveness with $KSR$-classified statements adds 5.38% to the average effectiveness. In the better case, it could add 13.36% to the effectiveness of sample B with respect to A.

7. Later, we organized the students in pairs (one student from sample A and the other from sample B) and let them compare their results, freely discuss their findings and report their comments on the experience. All 30 pairs of students agreed that, not only the $KSR$ approach was faster, but the discrepancy finding process was clearer, it was easier to understand the nature of each discrepancy and they looked easier to solve (it was clearer when to negotiate, when to use refutation, etc.). We did not perform any statistical analysis to validate those informally gathered result, but their opinions were unanimously favourable to the $KSR$ approach.

So, clearly, the approach based on a previous classification of the statements to be compared worked better than the other. Additionally, this approach seems to be very easy to understand and a natural way for detecting discrepancies, as shown from the opinions gathered. The underlying
hypothesis of our method has been validated and, from this result, we can now confront the task of building a software tool to support the method. Before that, however, following the recommendations of Kitchenham et al (2002), we present a brief discussion of the validity problems that affect our experiment.

3.1. Brief discussion of threats to validity.

The main threat to internal validity is selection bias: Some of the students (20%) also work in industry or at University labs. Although none of them work in Requirements Engineering, perhaps there is a chance that some of them are somewhat familiarized with the subject domain of the experiment.

The main threats to external validity are two: (i) Population validity: The use of students issues a warning, as perhaps mature analysts could perform the task more efficiently. However, performance of senior analysts could also be better for both treatments (KSR and not-KSR). (ii) Ecological validity: During the process, the students did not have interruptions and the environment was not noisy. In real office work this could not be the case. Also, the students worked with paper documents, not with a software tool. It is possible, however, that a software tool could enhance subject performance for both treatments.

We also considered an important threat to construct validity related to the demand characteristics in the research setting: even when we told the students that we will not evaluate their performance, the anxiety about being evaluated is always an issue with subject students.

4. TOOL SUPPORT

In this section we present the details of the software tool developed by one of the authors (F. J. López) at “TCP Sistemas e Ingeniería”, the company behind IRqA, the requirements management tool. DiscRman is a Windows based tool developed with C# using Visual C# .NET as development environment and MS-Access as the data repository. This environment will allow us, in the future, to create Web Services that provide a mechanism for web communication between discRman and other applications.

We will present how the tool supports Phase I, how it integrates with Phase II and, finally, some conclusions derived from its use.

4.1. Integration of the tool in Phase I

The light-weighting of Phase I, as reported in Section 2.4, took into account the overlap and discrepancy criteria. The tool based on such suppositions supports Phase I in the following way:

- **Viewpoint management:** Inside each viewpoint, it allows the user to make blocks of related K, S and R statements. This is a non-formal way of considering the \( K,S \Rightarrow R \) relation given by Zave and Jackson (1997). For instance, all the interrelated K, S and R statements about “booking” can be grouped in the same “booking” block. See the screenshot in Figure 3.

- **Viewpoint comparison in order to find mini-discrepancies:** The statements of one viewpoint can be compared against the statements of the other viewpoint, in order to find “mini-discrepancies” between them (see Figure 4). In the context of discRman, a mini-discrepancy is a discrepancy between elements of the same category. In this way, the notation \( K'SR' \), for example, denotes one discrepancy composed of two mini-discrepancies between K and R statements, as shown in Figure 5.

- **Full discrepancy identification:** For moving from mini-discrepancies to full discrepancies, the tool allows the developer to trace back the statements affected by each mini-discrepancy, in
order to facilitate the comparisons of related statements. For instance, if a discrepancy is found between the $K$ statements $K11_{VP1}$ and $K09_{VP2}$, the tool can show the $R$ and $S$ statements linked to them, so they can be analyzed and marked as discrepant, if needed.

In this way, the output of this phase is exactly as shown in Figure 2: a list of discrepancies properly classified, including the statements affected.

### 4.2. Integration of the tool in Phase II

In this phase, tool support is, of course, not aimed at an automatic discrepancy resolution, but to aid in the management of the resolution process. A screenshot related to the activities of Phase II is shown in Figure 5. Tool-supported activities include:

- **Discrepancy evaluation:** Discrepancies are shown by type, so the user can check the details, modify those details, add comments, record the status of the resolution process, prioritize it, rate the risk of not solving it, evaluate the cost and effort of resolution, etc.
- **Discrepancy resolution:** The tool suggests the tasks needed to solve the discrepancy, according to their type. It can also record the outcome of each resolution step, so the resolution process can be properly monitored. The traceability from the discrepancies to how they were solved is, thus, guaranteed.
Figure 4: discRman screenshot showing the comparison windows.
Impact: The tool we developed also included simple metrics for measuring the impact of each discrepancy, by taking into account the number of statements affected.

In this way, Phase II of the process is now adequately supported by the discRman tool. As shown in Figure 2, the output of this phase would lead to modified viewpoints, but not all discrepancies need to be solved at this point, so Phase II can be undertaken again, if desired.

4.3. Report of actual usage of the discRman tool
At this time we have transferred the tool to two pilot projects: one in COFAGA (www.cofaga.org) a pharmaceutical distribution organization, located in A Coruña and the other in La Caixa (www.lacaixa.es) a major Spanish bank, with offices located in Madrid. A discussion of both projects follows, along with major findings.

In both cases, there was no explicit viewpoint management process although, of course, discrepancies were always there, as a fact of life. Both development groups considered it important to deal with these discrepancies during the initial phases of the project. Hence, their initial attitude towards a tool that helped them to manage discrepancy-related issues was positive or, at least, not negative. At the time of writing this paper, the impression of both development groups remains positive, even when some issues have been raised in order to improve tool support, as we will show below.

La Caixa helped to finance this research, under a research contract with Univ. Politecnica de Madrid.
Before starting, the project managers were adequately convinced of the advantages of our approach by the experiment we performed. They also considered the KSR model to be a very clear and useful way of dealing with requirements. Their main concern was the effort required to classify each statement as K, S or R. However, the classification effort is not so great if statements are classified as they are recorded, instead of recording all of them and classifying them later. This means that, in Phase 0, categorization of statements can be done in parallel with other requirements management tasks (prioritization, for example), normally carried out with the help of requirements management tools. This categorization is not an overwhelming task, as in the RE process there is a need to distinguish between information of different nature, as the KSR model recommends. At the end, labelling a statement with K, S or R requires a similar effort as labelling it with traditional requirements attributes, such as priority, risk or status.

**Pharmaceutical Company**
The pilot project was a new module for their warehouse management system. It was a small-medium size project, of about 14 high-level features. Those features, in terms of discRman, were dealt with as different KSR “blocks”. An acceptable requirements management process was in place. This company has had close ties to some of the authors of this paper, as we helped them as RE consultants in the past. In particular, this company had implemented 7 of the top 10 guidelines recommended in the Good Practice Guide (GPG) for RE (Sommerville and Sawyer, 1998). The software developers were used to working with requirements management tools. It is clear to us that, if this were not the case, there would have been more resistance to the introduction of our tool.

Two viewpoints were considered and relevant stakeholders were (separately) interviewed by the requirements engineer: warehouse management people and sales-management people, whose job is directly related to the departure and entrance of goods in the portion of the warehouse affected by the new system. Each group was kept ignorant of the views of the other group, and no discrepancy identification or resolution was attempted during the elicitation process. In this way, stakeholders could communicate their thoughts freely without being biased by the views held by the other group.

From the experience it can be reported that:

- The viewpoints were collected using the tool without problems, as if they were collected in a traditional requirements management tool.
- For comparing statements, however, it clearly emerged the need to write statements as atomically as possible, as in some cases one statement in one viewpoint corresponded to two statements in the other viewpoint. In this case, the first statement should be broken into two. However, this helped to manage the “atomicity” of the statements.
- The comparison process was done smoothly for overlapped statements. The discRman feature for showing statements related to the discrepant statements already found was particularly useful, as reported, as it helped to focus the detection process and to group mini-discrepancies into “full” discrepancies. In most cases, the user felt that there was a strong relation between the mini-discrepancies affected, and that solving one would help to solve the other. It is true, however, that in three situations, where the statements related to a mini-discrepancy were misclassified, it was difficult for the user to understand if there was or was not a discrepancy that affected them. This indicates to us that for perfect discrepancy management the classification and categorization of statements should be done carefully. However, it is also true that in a discrepancy management context it is easier to pinpoint misclassified statements.
Each discrepancy, properly classified by the tool according to the mini-discrepancies affected, was considered by the tool as the “unit” of discrepancy management and dealt with accordingly. The resolution tasks suggested by the tool were taken by the users as a first step, or a suggestion, on how to proceed. Their main complaint at this point was that our tool lacked a feature for adding, deleting and grouping tasks, as some of them could be carried out in one step (for instance, in the same meeting).

Banking organization
The development team in the bank (La Caixa) had in place six of the key practices of the GPG guide (Sommerville and Sawyer, 1998). Developers were used to requirements management tools. The pilot project was an intranet bug-tracking and issue-tracking system, of about 21 high-priority features and 12 medium and low-priority ones. In this bank, a bug tracking system was already in place for the maintenance of legacy COBOL programs, but it was not well suited to their requirements, as it was not web-enabled and did not integrate with other tools. The requirements process started although, at the beginning, it was not clear if the bug-tracking system should be bought or built in-house. After some initial requirements gathering, the information collected helped them to make this decision and it was finally decided to develop the system in-house. In this case, developers and users were almost the same people and all of them were software-engineering literate. The two viewpoint groups identified were managers and developers. However, unlike the pharmaceutical company, in this case one group was not completely ignorant of the opinions of the other. Some conclusions follow:

- The viewpoints were collected and analyzed using the tool, without any particular problems, with the exception of some usability glitches that were addressed immediately.
- Again, the problem of the atomicity of the statements appeared, forcing the users to write them at a low granularity. From a positive point of view, this forced them to be more concise which helped with traceability.
- With regard to architectural issues, the users considered it important to collect them using the tool, so they were classified as $S$ statements, even when properly speaking, this is not the case (statements related to architectural issues should be subordinated to $S$-statements, but they are not $S$-statements in itself). This suggests a further enhancement for future versions of the tool.
- For entering the cycle again, as shown in Figure 2, the users suggested that perhaps it would be useful some visual clue to warn them that some statements were already changed as the result of a previous iteration. This would facilitate the process of re-checking for new discrepancies, as those already solved should have a lower priority. This suggests the need for (dynamically) prioritizing the statements according to its likelihood of being discrepant or not.
- The users complained about the lack of streamdown integration of our tool with other tools that they used, in particular with testing tools and modelling tools. Even when the information stored in MS-Access could be exported and imported (ad-hoc) by other tools, we recognize that this issue requires more work. However, many commercial tools also are lacking in this aspect.

For illustrative purposes, we will show an instance of a discrepancy of the $KS'R'$ type, related to timing: One of the stakeholders wanted an immediate warning (implemented through an interface with Lotus Notes) when a particular type of bug changed its state. However, another stakeholder preferred a daily digest by e-mail. After a brief discussion in a meeting (negotiation), the stakeholder that proposed the immediate warning realized that the other position was better (acceptance), and this lead to a change in the specification related to the discrepant requirement (redesign, changing the “sending a message” option by a “sending a daily e-mail” option).
Computer-assisted Discrepancy Management – A Case Study in Research Transfer to Industry

5. CONCLUSIONS
In this paper, we have reported on the application of our research within VBRE by means of a combination of empirical methods and actual application in real projects. On the one hand, the empirical work was aimed at testing the main hypothesis underlying the approach in a practical setting. On the other hand, tool support greatly contributed to soften the hardest, less usable, parts of the method, making it more acceptable in an industrial setting.

This way of working, in our opinion, could be a basis for proper transfer of research results to industrial settings (this oversimplified, is termed “repackaging”) and is particularly well suited to small research groups. Repackaging has the cost of oversimplifying and reducing the theory. However, if proper empirical data has been provided about the repackaged approach, then a proper basis has been established. Also, in our case, the discRman tool provided users with a solid basis for managing discrepancies in projects, although not without problems, as reported. However, the advantages of “light-weight” discrepancy management with discRman were evident, as discrepancies were easier to find and the discrepancy types provided the users with good guidelines on how to settle them.

In the near future we aim to improve the discRman tool according to the feedback received and integrate it with RME (Requirements Management Engineering) tools by means of XML-based interchange mechanisms.

REFERENCES

**BIOGRAPHICAL NOTES**

**Dr Javier Andrade** is assistant professor in the Information and Communications Technologies Department at the University of A Coruña, Spain. His research interests in computer science include conceptual modelling, knowledge management and natural language processing. He has worked as a software engineering and technological solutions consultant at IAL Software Engineering and Norcontrol Soluziona (Quality and Environment Department). He has a BSc and PhD in computer science. He is author of several book chapters and publications in software engineering.

**Dr Juan Ares** is head of the Information and Communications Technologies Department at the University of A Coruña, Spain. He is associate professor and co-director of the Software Engineering Laboratory at this University. His research interests in computer science include conceptual modelling, knowledge management and software process assessment. He has worked as director and consultant in several organisations, including Norcontrol, Soluziona and Arthur Andersen. He has a BSc and PhD in computer science. He is editor of several books and author of numerous chapters and publications in software engineering.

**Dr Santiago Rodríguez** is assistant professor in the Information and Communications Technologies Department at the University of A Coruña, Spain. His research interests in computer science include conceptual modelling, knowledge management and distributed systems development. He has been a project leader in several Spanish organisations. He has a BSc and PhD in computer science. He is author of several book chapters and publications in software engineering.
Francisco J. López is a software engineer specialized in Requirements Engineering. He is part of the consultancy team of TCP Sistemas e Ingeniería (www.tcpsi.com), the company behind the RME tool IRqA. His research interests include viewpoint-based requirements engineering and the creation of a standard metamodel for requirements engineering data representation and exchange. He received his degree in Computer Science and his Master’s degree in Software and Knowledge Engineering from the Universidad Politécnica de Madrid.

Dr Juan Pazos received the first Spanish doctorate in Computer Science from the Technical University of Madrid, where he is currently full professor at the Department of Artificial Intelligence. He set up the first Spanish Artificial Intelligence Laboratory, and was a visiting professor at Carnegie Mellon University and Sunderland University, among others. He has been/is a member of the editorial board of the following journals: AI Magazine, Heuristics, Expert Systems with Applications and Failure and Lessons Learned in Information Technology Management, among others. He is author or co-author of ten books on computer science and of over 100 publications. His current research is on the construction of an Information Theory that integrates Computing Science, DNA and the brain.

Dr Andrés Silva received his PhD from the Universidad Politécnica de Madrid (UPM) in 2001. He has three years of industrial experience in Software Engineering and worked at the Joint Research Centre (JRC) of the European Commission in Ispra (Italy). He is lecturer in the Computer Science School at the UPM. His research interests include Requirements Engineering and Knowledge Management. He is author of several conference and journal papers in Requirements Engineering.