# Interpreting Procedures from Descriptive Guidelines

**Mor Peleg, Ph.D.[1], Lily A. Gutnik[2], Vincenza Snow, M.D.[3], and Vimla L. Patel, Ph.D., D.Sc.[2]**

**[1]Department of Management Information Systems, University of Haifa, Israel;**

**[2]Department of Biomedical Informatics, Columbia University, NY, USA;**

**[3]American College of Physicians, Philadelphia, PA, USA**

Mor Peleg

Department of Management Information Systems

University of Haifa, 31905, Israel

Email: morpeleg@mis.hevra.haifa.ac.il

Fax: 972-4-828-8522

## Abstract

Errors in clinical practice guidelines translate into errors in real world clinical practice. The best way to eliminate these errors is to understand how they are generated, thus enabling the future development of methods to catch errors made in creating the guideline before publication. We have examined the process by which a medical expert from the American College of Physicians (ACP) creates clinical algorithms from narrative guidelines, as a case study. We studied this process by looking at intermediate versions produced during the algorithm creation. We identified and analyzed errors that were generated at each stage, categorized them using Knuth's classification scheme, and studied patterns of errors that were made over the set of algorithm versions that were created. We then assessed possible explanations for the sources of these errors and provided recommendations for reducing the number of errors, based on cognitive theory and on experience drawn from software engineering methodologies.

Keywords: decision-support, clinical practice guidelines, case study, CIGs, Knuth's classification scheme, software engineering, review process, cognitive studies

# 1 Introduction

Clinical guidelines are systematically developed statements to assist practitioners and patient's decisions about appropriate healthcare for specific clinical circumstances [1]. While the recommendations aim to be based on evidence, they are often not constructed in a way that reflects the flow of actual patient encounters. In order to solve this problem, clinical guidelines are sometimes portrayed as algorithms to guide physicians about the recommended steps of data gathering, decision making, and actions (i.e., process flow) during patient encounters. Clinical guidelines aim to eliminate clinician errors, reduce practice variation, and promote the best medical practices. Yet, during the process of creating clinical algorithms out of the descriptive guideline text, errors are often introduced. By using cognitive psychological techniques for studying the ways by which people process descriptive information into procedural representations, it is possible to understand the reasons that cause people to err while they are processing descriptive information, and to help them in finding their errors. An additional domain in which processes for error detection have been developed, is software engineering. Following is a short review of relevant results from these two domains.

***Causes of errors during processing of descriptive information: results from cognitive studies***

In previous work [2], Anzai and Patel examined the process by which people study scientific narrative text and use their knowledge to create mental model representations that would assist them in problem solving tasks. These studies have shown that people learn how to solve a task by actually solving it (*learning by doing*). In fact, learning by doing is one of the dominant strategies of obtaining knowledge and skills. Learning by doing has also been studied in the context of nurturing a layperson in acquiring physics expertise by learning how to draw and use diagrams [3]. In this study, it was shown that by representing a particular problem with a set of diagrams in the domain of physics, physicists recognize the underlying structure of the problem and can organize computationally efficient inference procedures for solving it. It has been suggested that recognition, generation, and inference are three main components of the processes in medical problem solving and the development of expertise. However, learning by doing has disadvantages, as people can also *over learn*, repeating errors that they have made throughout the process.

An additional source of errors that may occur during problem solving was determined in research that investigated decision-making strategies used by nurses in telephone triage involving public emergency calls for medical help [4]. The researchers concluded that some of the errors that the triage nurses made occurred because they were *confused by dealing with similar situations* that should have been treated differently. Noticing the similarity and ignoring differences in pairs of situations, the nurses have erroneously applied strategies that they have used in one situation to another situation.

Another study investigated the impact of algorithmic-based and text-based practice guidelines on clinical decision making by physicians at varying levels of expertise, with and without the use of the guidelines [5]. The narrative guideline and the algorithm served as reminder tools for the expert, whereas they were more useful for the non-experts as a means to fill in gaps in knowledge and as an aid in organizing their knowledge. This suggests that the content and presentation of guidelines should be adapted to the user's depth of knowledge. Guidelines and clinical algorithms are generally written by a team of experts in the medical area covered by the guideline. They aim to improve quality of care and standardize care and are often targeted toward non-expert clinicians. Since experts approach the guideline with a highly organized knowledge base, as writers, they may inadvertently expect the reader to have the same knowledge, and be able to make the inferences required to fully comprehend the guideline. Therefore, they *may not represent all of the guideline's knowledge explicitly*. A non-expert may not be able to correctly make these inferences, leading to errors or frustration. This evidence led the authors to hypothesize that non-experts may use guidelines and algorithms inaccurately. This hypothesis was confirmed by the study, which showed that the non-experts seemed content to take things at face value, whereas the experts showed a lot more concern in checking before diagnosing.

***Working in a team helps in error detection: experiences from software engineering***

The software engineering community has investigated methods for reducing the rate of programming errors. These methods often involve team work for finding errors. One of the successful methods is *inspection* of design and code [6]. When inspection is carried out systematically with well-defined roles for inspection participants, it can improve the quality of computer programs [7]. The team of participants include: (1) a moderator, who is a competent programmer but is not a technical expert on the program

being inspected. The moderator schedules the inspection, moderates it, reports its results promptly, and schedules follow up on rework, (2) the designer of the program, (3) the coder of the program, and (4) the tester, who is responsible for writing test cases and executing them[1]. Inspection sessions are planned for no longer than two hours because the error detection efficiency of most teams goes down after that period of time. The inspection session is highly structured. It begins in an overview in which the designer describes the design of the area to be examined and distributes design specification documents to the participants. (then what happens, says it begins, how does it end?)

A less successful method, which is often used because it is easier to apply, is Walkthroughs. A *walkthrough* is an informal review, in which the programmer describes the code that he has written to some colleagues and solicits comments. The author acts as the moderator, and there is no defined procedure for carrying out the walkthrough session [8].

Another team-process that has been quantitatively shown to improve software quality and reduce time to market is pair programming. *Pair programming* involves two programmers working side by side at one computer on the same design, algorithm, code, or test [9]. Collaboration improves the problem-solving process. As a result, far less time is spent in the chaotic, time-consuming compile and test phases. Programmers who have been working in pairs have reported the ability to solve difficult tasks together very quickly by putting their collective knowledge to work. Also, working as a team helps them stay focused on their task. Pair programming improves the programmers' job satisfaction and overall confidence in their work.

Arguing against pair programming, Parnas has remarked: "There is no doubt that four eyes can see more than two. I doubt that the four eyes have to look at the code at the same time in the same place?" [10]. Parnas defined a process for reviewing software designs for the purpose of finding errors in the design and its representation called Active Design Review [11]. The principle of this approach is that people use the design document, defend their own work, and cannot just watch passively. Instead of having one review process per project, teams conduct several types of reviews, each carried out with a different and

[1] For small-sized programs, the designer and coder may be the same person. It is best if the tester is a different individual, but if not, another individual fills the role of the tester during inspection

appropriate group of reviewers, designed to find different types of design errors. Instead of conducting the reviews as discussions, questionnaires are prepared for the reviewers to answer about the design documentation. Responding to the questionnaire requires the reviewer to study the documentation carefully, and to make assertions about design decisions. The issues raised by the reviewers as a result of answering the questionnaires are discussed in small meetings between designers and each reviewer. Instead of pairing programmers to produce codes together, pairs, or small teams, are created, where each person creates a document and the other checks it (this is a confusing sentence, who are these pairs or small teams?). In the following examples of such pairs, each member of the pair checks the other's work. The pairs could be: a person who writes the requirement document and a person producing the corresponding design document; a person who writes the design document and another person who writes code, designer and coder [10]. (so pairs of computer science professionals?)

# 2 Research Question

In this study, we have examined the process by which a medical expert from the ACP (VS) goes about creating clinical algorithms from narrative guidelines. Our hypothesis is that when an expert creates a clinical algorithm based on her processing of a narrative guideline, she learns how to improve the algorithm by iterating the process of algorithm creation. When she has new ideas for design changes (e.g., alphabetizing treatment options that are not ranked) she carries them out throughout the algorithm. From one iteration to the next, fewer changes are made until the expert is satisfied with her results. Getting feedback from other professionals, or reexamining the algorithm after a substantial period of time, results in making more changes.

# 3 Methodology

In a previous study [12] we compared the final version of the algorithm to the narrative guideline from which it was developed, analyzing the types of changes between them. By contrast, in this study we captured the versions of the algorithm, rather than focusing on the final versions only. In this way we could analyze the process and find reasons for changes made between consecutive versions. Our combined expertise in cognitive psychology and computer-science enabled us to produce different

explanations for the changes made, focusing on these different perspectives, and suggest recommendations for limiting errors in future processes of development of clinical guidelines, algorithms, and CIGs. These recommendations have implications for the conversion of narrative guidelines to computer-interpretable guidelines (CIGs), encoded in CIG formalisms such as GLIF3 [13], which allow automatic inference enabling decision support. In addition, studying the narrative guideline and algorithm, development process will assist us in refining our authoring tools and it will assure robustness in GLIF3.

### 3.1 Guideline and setting

Two members of our research team (MP, VP) observed the ACP expert, as she created flowchart versions of clinical algorithms based on a narrative guideline that ACP had created previously. The guideline studied was *Pharmacologic management of acute attacks of migraine and prevention of migraine headache* [14]. We recorded the expert as she "thought aloud" about what she was doing, and we also captured any conversations with the investigators and other guideline creators. We kept all the drafts of the algorithms (six drafts of the first algorithm and two of the second), which showed progressive changes in the algorithms. The Institutional Review Board reviewed and approved the study protocol.. A year after the creation of the clinical algorithms, the ACP expert went over the last version of the algorithm set, which she had created before and created a new version of it for the purpose of creating an html version that included hyperlinks.

### 3.2 The ACP process of creating algorithms

After the ACP team has created a narrative guideline, it constructs a clinical algorithm and publishes it at www.acponline.org. The algorithm is first created by a medical expert, who reads the narrative guideline and creates versions of the algorithm in an iterative manner until she is satisfied with the results. Computer-based tools are not used during this process. When finished, she delivers a final copy to the director of scientific policy (DSP) at the ACP. This person examines the algorithm and compares it to the narrative guideline. Then, in a review session with the expert, the DSP suggests clarifications and changes. The expert modifies the clinical algorithm and hands it over to a third member of the team, the flowchart designer, who uses software to generate flowcharts from the expert's hand-drawn algorithms. The expert then checks the flowcharts.

### 3.3 Identifying changes and classifying them

Our approach is to identify the types of errors introduced during the development process of narrative and CIGs. In addition to defining types of errors made, we are also interested in identifying their sources, and devising methods and tools for limiting them. We used a classification scheme proposed by Knuth [15] to classify changes between narrative guideline text and the clinical algorithm produced from it. Knuth classified discrepancies between the requirements document for TeX and the resulting software. The first twelve of the 15 change types that he suggested are applicable to the narrative guideline domain. The narrative guideline is analogous to the requirements document, whereas the clinical algorithm created from it is analogous to the software. We added specialization as a possible change type.

1. *Algorithm awry* (A): incorrect or inadequate specification of process execution sequence. For example, not specifying the processes that take place in the system in their correct order is an algorithm awry.
2. Forgotten function or *omission* error (F): forgetting to specify a function, or omitting a part of the problem statement from the specification. For example, excluding one of the processes or objects that were described in the problem statement from the model is a forgotten function or omission error.
3. *Language liability* (L): misusing or misunderstanding the (specification) language. For example, using an action step instead of a decision step is a language liability error.
4. *Mismatch between* modules, or, in our case, between *algorithms* (M): a clinical algorithm that is specified using several linked sub-algorithms that result in inconsistent recommendations is a mismatch between modules.
5. *Blunder* or botch (B): thinking the right thing but writing it in a wrong way using correct syntax (e.g., confusing 'before' with 'after').
6. Trivial *typo* (T): a typographical error.
7. Cleanup for consistency or *clarity* (C): making changes to make things more logical and/or easier to remember.
8. (a) *Generalization* or growth of ability (G): extending the original problem statement in order to make it more general or to increase its ability. (b) *Specialization* – making a definition narrower

9. *Interactive improvement* (I): improving the specification for better response to user needs.
10. *Promotion of portability* (P): changing the organization or documentation of the specification.
11. *Quest for quality* (Q): changing the original problem statement (narrative) to increase the specification (algorithm) quality.
12. *Surprising scenario* (S): changing the original problem statement (narrative) due to scenarios that were not considered in the problem statement.
13. Efficiency enhancement (E): changing the process to make it more efficient.
14. Data structure debacle (D): not updating properly the representation of information to preserve appropriate invariants.
15. Reinforcement of robustness (R): making the specification (algorithm) handle erroneous input.

### 3.3.1 Errors vs. improvements

In Knuth's classification, some of the changes can be improvements with positive qualities (e.g., quest for quality), some can have negative qualities (e.g., blunder). Others can be neither (e.g., generalization/specialization, which could include improving the algorithm where guideline eligibility was generalized from patients having a migraine attack to those experiencing headache or a specialization that is the result of an error, such as in the case of not listing all the possible criteria for being eligible for preventive treatment).. In our analysis, it was important for us to distinguish between improvements and errors and to see whether we could identify patterns of error occurrence during the iterative process of algorithm creation.

### 3.3.2 Important vs. unimportant modification

Not all of the modifications that the expert made during algorithm creation were equally important. We carried the analysis on the entire set of changes and on the subset of changes that we found to be important. Unimportant changes were changes that would not have affected the advice given by the algorithm (e.g., in addition to recommending efficacious drugs, specifying which drugs are not recommended) or changes that were made in order to clarify terms that were probably already known to clinician users (e.g., giving examples of anti-emetics). We believe that in order to learn how to improve the algorithm creation process, resulting in more accurate and robust algorithms that are easy to computerize, the focus should be on changes that are related to the *important* category. Taking

unimportant modifications into account may shift the results for the better, when the unimportant modifications were improvements, or for the worse, when the modifications were errors. In most cases, however, the results were not significantly affected by the exclusion of unimportant modifications. In this paper, we report the results of the important changes. In cases were the results differed when including or excluding unimportant changes, we report both of these results.

### 3.3.3 Grouping errors according to their location in the narrative guideline

When looking at the different statements from the guideline narrative that made it to the clinical algorithm, we distinguished between statements that originated in the: (I) recommendations, (II) paragraph below the recommendations, (III) background material contained in the guideline, and (IV) information added that was not in the original guideline. We wanted to see whether the section on recommendations was informative enough to create most of the algorithm and whether most of the important changes were done to particular parts of the narrative guideline. This may indicate on which sections of the guidelines we should focus our attention when analyzing errors and their causes.

# 4 Results

We recorded the changes that the expert made between each consecutive algorithm version. For each change we recorded (1) whether the change was positive or negative, (2) whether the change was important or unimportant, (3) the location of the change in the guideline narrative, and (4) the change's category according to Knuth's classification. To analyze the data, we looked for patterns in the distribution of errors over various versions of algorithms. We also searched for possible explanations for errors made by the expert. We present the patterns that emerged from the data and developed theoretical accounts as a set of hypotheses and described the extent to which these hypotheses were confirmed.

## 4.1 Hypothesis 1: Most of the statements in the algorithm will be taken from the recommendations section of the guideline (I), or from the paragraphs below each recommendation (II). Statements from the introduction section (III) and statements not contained in the narrative guideline (IV) represent a small percentage

**[Insert figure 1]**

The data in figure one shows that the hypothesis was not confirmed. Much of the information relevant for creating an algorithm was not found in the Recommendations section (III). Looking at sequential versions, we couldn't find a temporal pattern of the distribution of errors grouped by the origin of statements (results not shown).

### 4.2 Hypothesis 2: The number of modifications made in each version reflects the process carried out by the expert

In the first version, the expert created many modifications to the original narrative. Then she made modifications looking only at the previous algorithm versions that she created until she found no more errors. Next, she moved on to the next stage: consulting the narrative guideline again, and kept consulting the narrative and making modifications until no more modifications were needed. She then consulted the DSP and according to her comments made additional modifications (version 6). In examining the guideline again a year later, she fond places where new modifications were required. She had forgotten the last algorithm that she had created, thus she examined this version with a critical eye.

**[Insert figure 2]**

Figure 2 describes the total number of modifications made in different algorithm versions. The results show that this hypothesis was confirmed. The expert created the first version based on the narrative, and created new algorithm versions by consulting only her previous versions until no more modifications were necessary (version3). Consulting another source – the narrative in version4 and the Director of Scientific Policy in version6 helped the expert make more relevant modifications. A year later (new version) she made more changes.

### 4.3 Hypothesis 3: More positive changes than negative changes are created in each version

**[Insert figure 3]**

Figure 3 gives the number of positive and negative modifications in the different algorithm versions. The hypothesis was not confirmed. More errors than positive modifications were generated when the clinical expert worked alone. Only following comments from the director of scientific policy (DSP-version 6), the clinical expert made more improvements than errors**.**

Figure 4 shows all the number of all modifications, including unimportant ones, which were made in the different algorithm versions. Examining all modifications, it seems as if more improvements were made than errors. However, most of the improvements were unimportant and most of the errors were highly important.

.

**[insert figure 4]**

### 4.4 Hypothesis 4: The distribution of error types should be the same in different versions. No apparent unjustified differences should be present.

**[Insert figure 5]**

Figure 5 shows the distribution of modification types made at each algorithm version. The hypothesis was not confirmed. In the last version, many more blunders (3) were made than in any previous versions (one blunder was made in version 4, one in version 6, and no blunders were made in versions 1,2,3, and 5).

However, versions 1 through 6 were done during one session. Comparing the number of *blunders* made during that session (2 blunders) to the number of blunders made during the creation of the new version, which was done a year later (3 blunders), shows that an almost equal number of blunders were made in each session. Comparing the sum of modifications made in the first session to the number of modifications made during the creation of the new version, we found that no important *interactive improvement* modifications were generated in the last version as compared to three done in the first version. *Correction* modifications could have only been made in the new version because we considered cases where the expert transiently made a blunder during the first session but caught it soon afterwards in the same session as artifacts that should not be considered.. Similarly, *Forgotten functions* were introduced in the first version. If they were corrected later in versions 1 through 6 of the first session, then they would have not been counted as forgotten functions, but as artifacts that should not be considered.

### 4.5 Hypothesis 5: The percentage of modifications made to guideline statements that originated at the recommendation section (level I), the paragraph below it (level II), or in the background section of the guideline (level III) will be equal. The percentage of modifications made to algorithm statements that were not part of the narrative guideline (level IV) will be smaller than the percentage of modifications in the other levels..

**[Insert figure 6]**

Figure 6 gives the percentage of modified statements, arranged by their location in the guideline, in the different algorithm versions. The first part of the hypothesis was not confirmed; a pattern in the distribution of modifications in different levels did not exist. The second part of the hypothesis was confirmed. Once the medical expert added information to the algorithm that was not part of the narrative guideline (level IV), she did not make modifications to that information. Perhaps, this is due to the fact that she was not able to consult with the guideline to see if her additions were correct. Also, during the algorithm creation and review process, she did not consult with other medical experts.

# 5 Discussion

In interpreting the results, we tried to account for our hypotheses about patterns of errors. These were not supported by the experimental results. Looking at the individual errors that were made, and following the expert's explanation of her rationale for making the modifications, we tried to find the sources of errors and suggest ways to limit them.

## 5.1 The different purposes of guidelines and clinical algorithms suggest differences in content

While we hypothesized that most of the algorithm could be created from material found in the recommendation section and in the paragraphs below the recommendations, we found that much of the information that is relevant for creating an algorithm has its origin in the background section of the guideline (level III in Figure 1). This suggests that the Recommendation section does not contain enough information for summarizing a process of care. We also found that a substantial part of the algorithm was not based on the guideline at all (IV). A possible explanation is that the narrative guideline aims to list recommendations that are based on evidence, but this is not enough to generate a clinical algorithm that describes a process of care, so information needs to be added, based on the expert's opinion.

## 5.2 Team-work is crucial for detecting errors

While we expected the expert to make more improvements than errors while she iteratively created the algorithm, we found that this was not the case (Figure 3). Looking at the total number of modifications made by the end of the first phase of algorithm creation process (total modifications v1-v6), we found that more errors were made relative to positive modifications. Correlating the errors with the phases of work that the expert made while she was working alone versus errors she made while she was doing collaborative work, we saw that more errors than positive modifications were made when the clinical expert worked alone. Only upon following comments from the director of scientific policy (DSP), the clinical expert made more improvements than errors**.**

As explained in the Introduction, team work has been advocated as a means for detecting errors in software design and coding. Controlled experiments have shown that working in pairs reduces the number of errors that are generated [9]. The review process that the DSP and the clinical expert were involved in was similar to the inspection or design review used in the software engineering area, though it was less formally defined. In the ACP's review process, the expert gave the DSP the algorithms for review. After reviewing the algorithms and the original guideline, the DSP and the clinical expert met and the DSP asked for clarifications, pointed out terms that were not defined, and suggested a different design that combined the prevention and treatment algorithms. Perhaps the ACP's process could be improved by assimilating the process used during software inspection and review processes. First, as in the inspection process, the review meeting may involve another clinician, knowledgeable in medicine, who could act as a moderator as well as a tester, such as an informatician who can think of test cases for checking the completeness and correctness of the algorithm. Second, as in the design review process, the expert can have one-on-one meetings with another clinician, and with an informatician to review the algorithms that she had produced, using questionnaires that were prepared in advance. Third, similarly to the pair programming process, two clinicians could produce the algorithm. It would be interesting to compare the number of errors produced in algorithms when using the standard process carried out by the ACP with the three alternative processes suggested here.

## 5.3 The need for an informatician on the team

Despite the rigorous process of algorithm development, the informatician on our team (MP) still found places in the algorithm requiring modifications, which were not found by the ACP team. These modifications reflect a formal computer-science perspective of algorithm creation, which is not clinical. The ACP team agreed with these modifications, which included:

### 5.3.1 Missing definitions of branching points and interaction among guidelines

Guideline authors usually do not specify the interaction among related guidelines that are applicable to the same patient (in our case, the prevention and the acute algorithms). The guideline noted that while a patient is on preventive therapy, care should be taken to avoid overuse of acute medication. However, the guideline did not clarify what *overuse* means. The expert first specified the interaction

by introducing a branching point, so that a patient being treated by means of the prevention algorithm would receive treatment according to the acute algorithm, thus he would not be treated for acute attacks at all. Later, the expert reversed her specification, by removing this branching point. This allowed her to make it possible for patients who are on the prevention algorithm to enter the acute algorithm. However, instead of restricting the use of acute medications, the specification gave acute treatment to patients in each of their acute attacks regardless of whether they were being treated in accordance to the prevention algorithm. The expert did not realize that neither of the specifications correctly addressed the situation of avoiding overuse of acute medications. Had a clear definition of overuse been given, the expert would have likely specified a branching point that decides when a patient being treated on the prevention algorithm should be routed to the acute algorithm. A computer scientist would have been suspicious of a situation where a branching point is removed. If there had been a problem with the course of action recommended to one of the branches, then that branch should have been fixed. The branching point itself should not have been removed because the reason for dividing the possible cases into two groups usually would have remained. In fact, another branching point should have been added, which would divide the population of patients who are receiving treatment via the prevention algorithm and are experiencing an acute attack into those who should be treated for their acute attack and those who should not be treated.. The expert wanted to allow a patient who should be treated for his acute attack to enter the acute algorithm, but she did not realize that by removing the branching point she also allowed patients who should not be treated for their acute attacks (thus avoiding overuse) to enter the acute algorithm.

### 5.3.2 Problem with negation and implication

The guideline specified that if a patient is vomiting then he should be given non-oral medication (Vomiting implies non-oral). If we know that the patient is not vomiting, then what kind of medication should be given to him? The answer is *any route of medication*. We can see this by looking at the truth table for the implication statement "vomiting implies non-oral". For the value of vomiting = false the implication is true when "non-oral" is true or false. However, the expert specified that if the patient

is not vomiting then only "oral" medication should be given. Due to her confusion, she simply negated "non-oral".

### 5.3.3 Confusing AND with OR

*And* and *Or* are used informally in the English language. The expert used AND/OR to replace the word "with". In fact, in the sentence "two attacks a month *with* disability lasting more than 3 days", the word *with* should be translated into "AND". Formally, AND/OR equals OR. While OR makes a criterion more general, AND makes a criterion more specific.

### 5.3.4 Altering the control flow as a result of considering patient situations that were not addressed by the guideline

The guideline distinguishes between migraine-specific medications and non-steroidal anti-inflammatory drugs (NSAIDs). For patients who had taken NSAIDs with good response, the guideline recommends taking the same medication in later attacks. The symmetric situation, of patients who had taken migraine-specific medication with good response was not considered. Computer scientists are trained to think about decision tables [16] that list the actions that should be taken for every possible variable assignment. They are taught to systematically list all decision variables and all their possible values and check that actions are defined for every possible combination of values for all of the variables that are important in making a decision. If the decision is on what medication to prescribe for an acute attack, and the variables are the type of medication used before and the type of response achieved, then the table should specify the course of action for all four combinations of these two binary data items.

## 5.4 Overlearning may lead to errors

*Overlearning* refers to a process were we study material until we know it perfectly, and then continue to study it some more, until we master it. However, overlearning may some times lead to unnecessary modifications or even lead to errors. We found several examples where the clinical expert exhibited this kind of behavior:

- The expert added to the phrase "NSAIDS" the phrase "with proven efficacy" because she remembered that she had made this modification before. Only she did not notice that the addition was not necessary, as the list of NSAIDS contained only NSAIDS with proven efficacy.
- The expert made a modification to the algorithm for treatment of acute migraine attacks by combining a second decision criterion with a criterion that she had already seen in that algorithm. The two criteria should have been combined with *AND*, requiring that both of them be present (e.g., "two or more attacks a month" *AND* "disability lasting 3 days"). However, the expert combined them with *OR*, mistakenly. The expert later found the same criterion in the prevention of migraine attacks, but there, the two criteria were already combined correctly by the work "with" that implied an *AND* relationship between the two criteria. Erroneously, she replicated the mistake that she had introduced in the treatment algorithm into the prevention algorithm.

### 5.5 Confusing different situations may lead to errors

If we encounter a new situation and mistakenly identify it as being identical to a previously encountered situation, we may erroneously apply the technique that is appropriate for the original situation to the new situation. We found several examples of this kind:

- The expert remembered the comment made by the DSP about alphabetizing drug options when no ranking of options is given so as not to imply an order. However, she applied this technique to alphabetize a list of drugs that had been ordered by drug side-effects and availability.
- The expert read the guideline text that discussed generally accepted indications for migraine prevention. One of the lines in that section referred to "two or more attacks a month" while two lines down in the same section, a reference was made to "twice a week". When the expert specified the first line, she mistakenly wrote "two or more attacks a *week*", confusing month with week because of the two similar phrases found in the same guideline section.

### 5.6 Implications to GLIF3

The ACP team that develops clinical algorithms is already considering issues that are important for producing high-quality guidelines, as defined in the 1992 IOM report on the development of clinical guidelines [17]. These issues include: (1) validity of the algorithm, (2) reproducibility (i.e., the algorithm would always produce the same behavior for the same patient situations), (3) clinical applicability (i.e., eligibility criteria), (4) clinical flexibility (i.e., considering different patient scenarios that occur during a patient encounter), (5) a clear definition of clinical terms, decision points, and medical actions, (6) a clear definition of control flow, (7) logical and easy-to-follow modes of presentation, and (8) distinction among clinical decisions, actions, patient states, and entry and exit points of the algorithm. The ACP team uses a process of algorithm development that involves several people and several stages. This process includes face-to-face meetings and discussions of the algorithms that help spot errors or lack of clarity.

Unfortunately, not all of the considerations that are important for automating guidelines are explicitly considered when the narrative guidelines are being developed. This situation creates difficulties when the guidelines are to be encoded in a CIG formalism, such as GLIF3 [13]. We suggest that guideline developers should be given instructions which explain the cognitive sources of errors as well as the software engineering methodologies of review and inspection to limit errors. Another recommendation is to provide guideline authors with tools for creating clinical algorithms, thus being able to create clinical algorithms and guidelines that would be more complete and consistent, allowing easier development into CIGs. When we created authoring tools for GLIF3, we had considered tools that would ease the generation of what we call "the conceptual algorithm specification" [18]. This specification is created by clinical experts and does not formally specify decision criteria, but does specify the structure of the clinical algorithm and the patient data used to make clinical decisions. The GLIF3 authoring tool that we had developed [12] allows an author to create a graphical clinical algorithm, validate the algorithm for completeness and consistency, enforce definition of terms according to controlled terminologies, define rules for ranking alternative treatment options written in natural language, introduce links to support material, and specify other documentation attributes, such as the target audience of the guideline, and strength of evidence associated with each guideline step.

At the ACP, the clinical expert does not use computerized tools to aid in the creation of the clinical algorithm. . Once the algorithm is complete and validated, a flowchart designer creates a formal flowchart

out of the boxes and arrows specification that the medical expert created. The GLIF3 authoring tools can be introduced at this stage of algorithm creation by the expert or at the last stage, as it is now done at the ACP. Although there are benefits to using the tools from the start, it may also be too time-consuming for the expert. In addition, some experts feel more comfortable in using paper and pencil to draft the algorithms. In these cases, the informatics-considerations and GLIF3 authoring tools should be introduced at a later stage of the algorithm creation. The flowchart designer should be a person trained in informatics, who has training in identifying logical errors or incompleteness of specification. She would use GLIF3 tools to formalize the algorithms created by the expert and approved by the DSP. She would be more involved in impacting the process of algorithm creation. This will involve more rounds of iterations of algorithm creation, where the informatician and clinical expert would sit together while the informatician translates the algorithm into a more formal GLIF3 conceptual algorithm specification. We also think that adding another person to the team will result in improving the quality of the algorithm created. The modifications that were made by the clinical expert based on the review of her work by the DSP resulted in many improvements, and in only one error, which could have been detected by an informatician.

Other tools might be used to limit errors that result from forgetting to represent part of the narrative guideline or to copy part of the clinical algorithm or sidebars to the next version. A tool like GEM-Cutter [19] could be used to mark-up narrative guidelines using GEM elements. The tool could be used to view unmarked parts of the guideline, thus aiding in limiting omission errors.

In this study, we used Knuth's classification scheme to categorize modifications between a narrative guideline and the final version of its clinical algorithm. Although the classification scheme was developed for modifications between requirements, documents, and software products; we found it appropriate for categorizing modifications between narrative guidelines and clinical algorithms, as well as errors in CIG specifications. A different classification scheme was developed by Tierney and coauthors, who described the problems encountered while they encoded a heart failure guideline [20]. They found that the guideline often lacked definitions of terms and branch points, did not focus on errors of commission, and did not account for comorbid conditions, concurrent drug therapy, or the timing of most interventions and follow-ups. Because our study did not examine implementation issues or the development of the narrative

guideline prior to clinical algorithm generation, we only considered the first of these problem types, although the implementation of two other guidelines developed by the ACP team has been reported by Patel et al [5].

## Conclusion

We have studied the process of creating clinical algorithms by the ACP by looking at intermediate versions produced during the algorithm creation, rather than examining only the final products. We identified errors that were made at each stage, categorized them, and studied patterns of errors that were made over the set of algorithm versions that were created. Analyzing patterns of errors as well as following individual errors, we found possible explanations for the sources of these errors.

Based on our analysis, and on the methods used in the software engineering community for reducing errors in software code, we suggest developing recommendations for authors who create clinical algorithm that go beyond the recommendations reported by Tierney and coauthors [20]. These recommendations include: (1) verification that all relevant information is carried from the narrative guideline to all versions of the clinical algorithm, (2) providing all the information necessary to rank treatment options, and (3) considering different patient scenarios. In addition, we recommend changing the process for creating clinical algorithms in two main ways: (1) by introducing an informatician to the team who would review the guideline and (2) by defining a more formal review/inspection process, as done in software development. We also suggest that guideline developers would be given instruction that would explain the cognitive sources of errors. We believe that taking such measures may reduce the number of errors introduced into the algorithms. Future studies will examine the impact of the new process that we suggested.

## Acknowledgement:

This study was conducted while the first author was on leave at Columbia University's Department of Biomedical Informatics. We thank Dr. Christel Mottur-Pilson, the director of Scientific Policy at the American College of Physicians, Philadelphia, PA for her help and her unconditional support of the study.

## References


1. Field MJ, Lohr KN. Guidelines for Clinical Practice: Directions for a New Program. Washington DC: Institute of Medicine, National Academy Press; 1990.
2. Anzai Y, Patel VL. Learning Graph-Reading Skills for Solving Problems. In: (Eds.) DAEaVLP, editor. Advanced Models of Cognition for Medical Training and Practice, NATO ASI Series F: Computer and Systems Sciences. Heidelberg, Germany: Springer-Verlag GmbH & Co. Kg.; 1992. p. 285-306.
3. Anzai Y. Learning and use of representations for physics expertise. In: Ericsson K.A. SJe, editor. Toward a General Theory of Expertise: Prospects and Limits. Cambridge, MA: MIT Press; 1991. p. 64-92.
4. Leprohon J, Patel VL. Decision-making strategies for telephone triage in emergency medical services. Medical Decision Making 1995;15(3):240-53.
5. Patel VL, Arocha JF, Diermeier M, How J, Mottur-Pilson C. Cognitive Psychological Studies of Representation and Use of Clinical Practice Guidelines. International Journal of Medical Informatics 2001;63(3):147-168.
6. Gilb T, Graham D. Software Inspection. Wokingham, UK: Addison Wesley; 1993.
7. Fagan ME. Design and Code Inspections to Reduce Errors in Program Development. IBM Systems Journal 1976;15(3):182 -211.
8. Wiegers KE. Seven Truths About Peer Reviews. Cutter IT Journal 2002;15(7).
9. Williams L, Kessler RR, Cunningham W, Jeffries R. Strengthening the Case for Pair-Programming. IEEE Software 2000;July-August:19-25.
10. Parnas DL. XP Versus MTP: The Limits of Extreme Programming. Slides of a presentation given at: eXtreme Programing 2003, www.xp.2003org/xp/2002talksinfo/parnas.pdf
11. Parnas DL. Active design reviews: principles and practice. J of Systems and software 1987;7:259-265.

12. Peleg M, Patel VL, Snow V, Tu S, Mottur-Pilson C, Shortliffe EH, et al. Support for Guideline Development through Error Classification and Constraint Checking. In: Proc AMIA Symp.; 2002. p. 607-11.

13. Boxwala AA, Peleg M, Tu S, Ogunyemi O, Zeng Q, Wang D, et al. GLIF3: a representation format for sharable computer-interpretable clinical practice guidelines. Journal of Biomedical Informatics 2004;37(3):147-61.

14. Snow V, Weiss K, Wall EM, Mottur-Pilson C. Pharmacologic management of acute attacks of migraine and prevention of migraine headache. American Academy of Family Physicians - Medical Specialty Society American College of Physicians - Medical Specialty Society. Ann Intern Med 2001;137(10):840-852.

15. Knuth DA. The Errors of TEX. Software - Practice and Experience 1989;19(7):607-685.

16. Shiffman RN. Representation of clinical practice guidelines in conventional and augmented decision tables. J Am Med Inform Assoc 1997;4(5):382-93.

17. Field MJ, Lohr KN. Guidelines for Clinical Practice: From development to use. Washington DC: Institute of Medicine, National Academy Press; 1992.

18. Greenes RA, Peleg M, Boxwala AA, Tu SW, Patel VL, Shortliffe EH. Sharable Computer-Based Clinical Practice Guidelines: Rationale, Obstacles, Approaches, and Prospects. In: MedInfo; 2001; London, UK; p. 201-205.

19. Argawal A, Shiffman RN. Evaluation of Guideline Quality Using GEM-Q. In: Medinfo; 2001; London, UK;p. 1097-1101.

20. Tierney WM, Overhage JM, Takesue BY, Harris LE, Murray MD, Varg DL, et al. Computerizing guidelines to improve care and patient outcomes: the example of heart failure. J Am Med Inform Assoc 1995;2(5):316-22.

.Figure 1. Total number of modifications made in different algorithm versions. There were seven versions: versions 1-6 were created a year before the "New Version" was created. The column marked as “1-6" indicates the total number of modifications made during the first sessions of algorithm creation, which included the first six iterations.

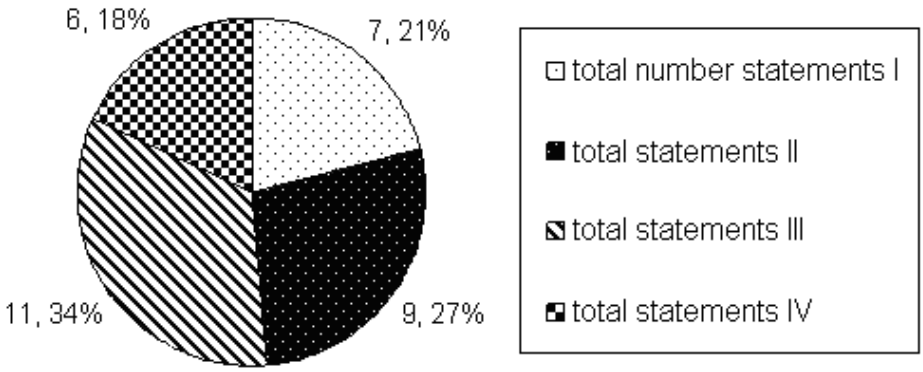
Figure 2. Number of positive and negative modifications in the different algorithm versions. The version numbers are as explained in the legend in Figure 2.

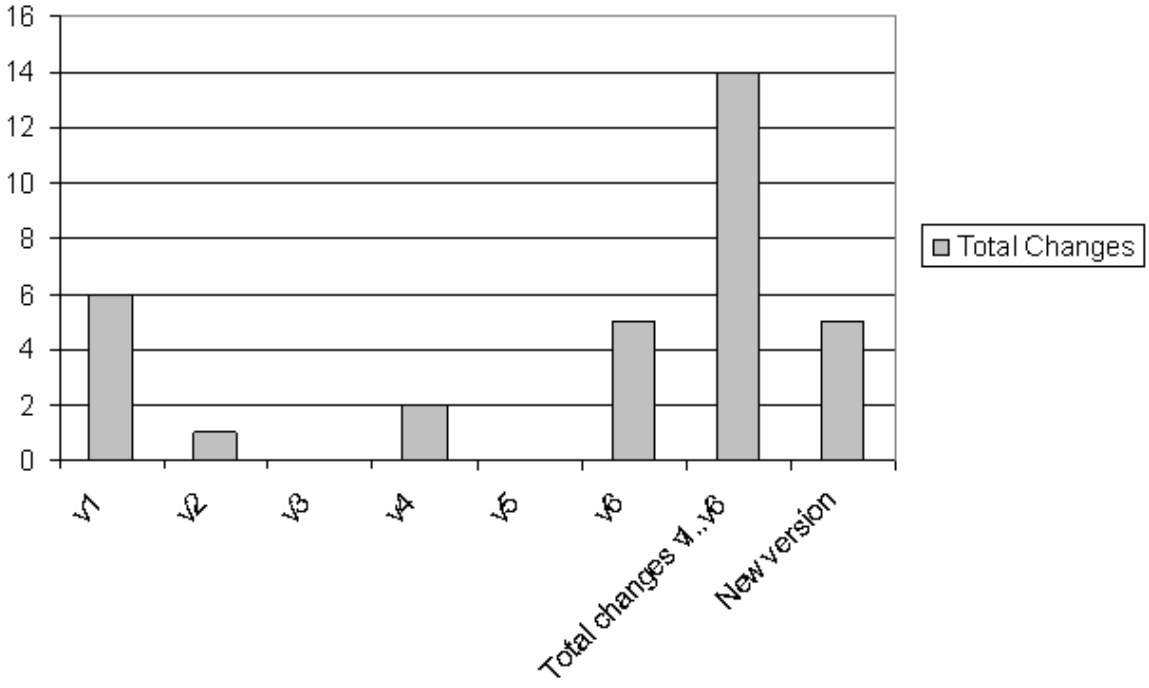
Figure 3. Number of positive and negative modifications made in the different algorithm versions, including unimportant modifications. The version numbers are as explained in the legend in Figure 2.

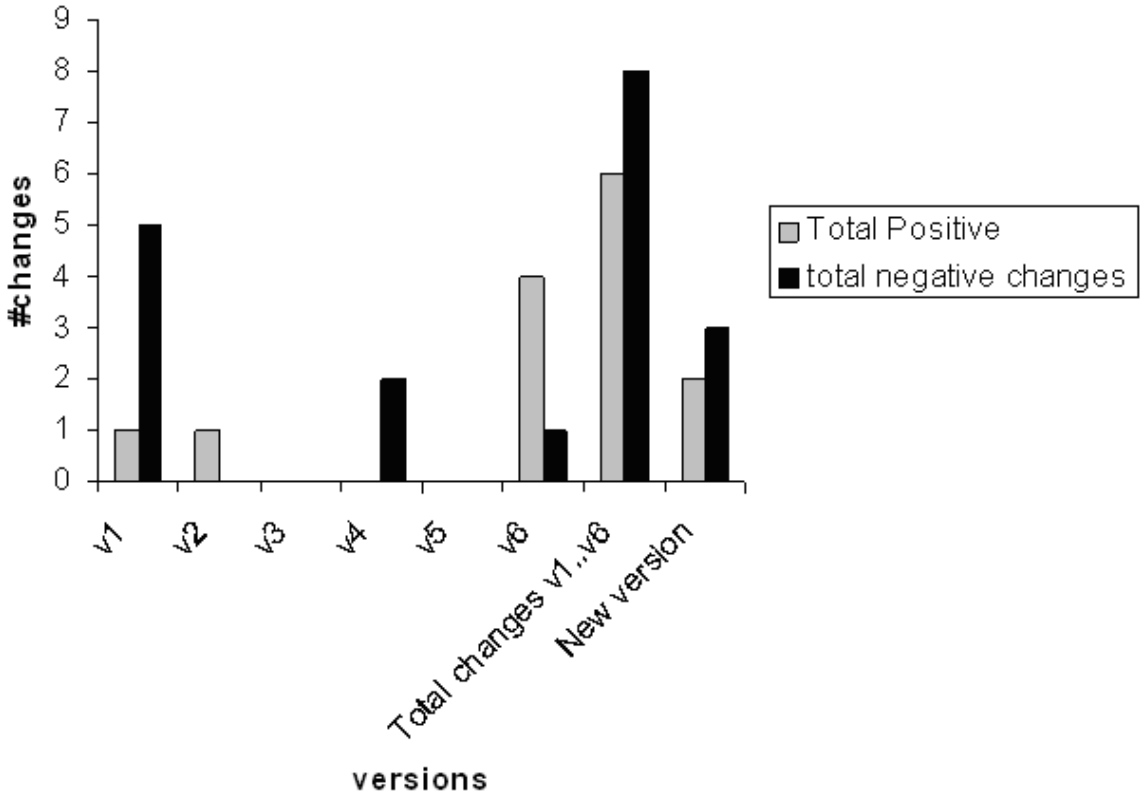
Figure 4. The distribution of modification types made at each algorithm version. The version numbers are as explained in the legend in Figure 2.

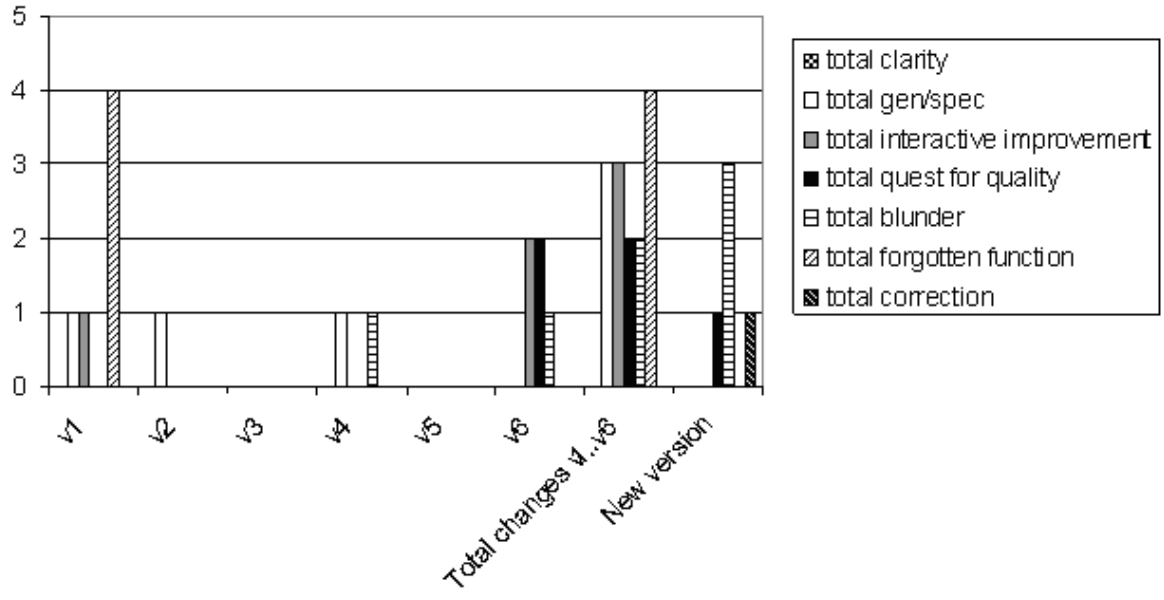
Figure 5. Distribution of types of statements in the algorithm according to their location in the narrative guideline. Categories I-IV are explained in Section 2.6. The numbers by the pie slices indicate the number of statements belonging to a category and their relative percentages

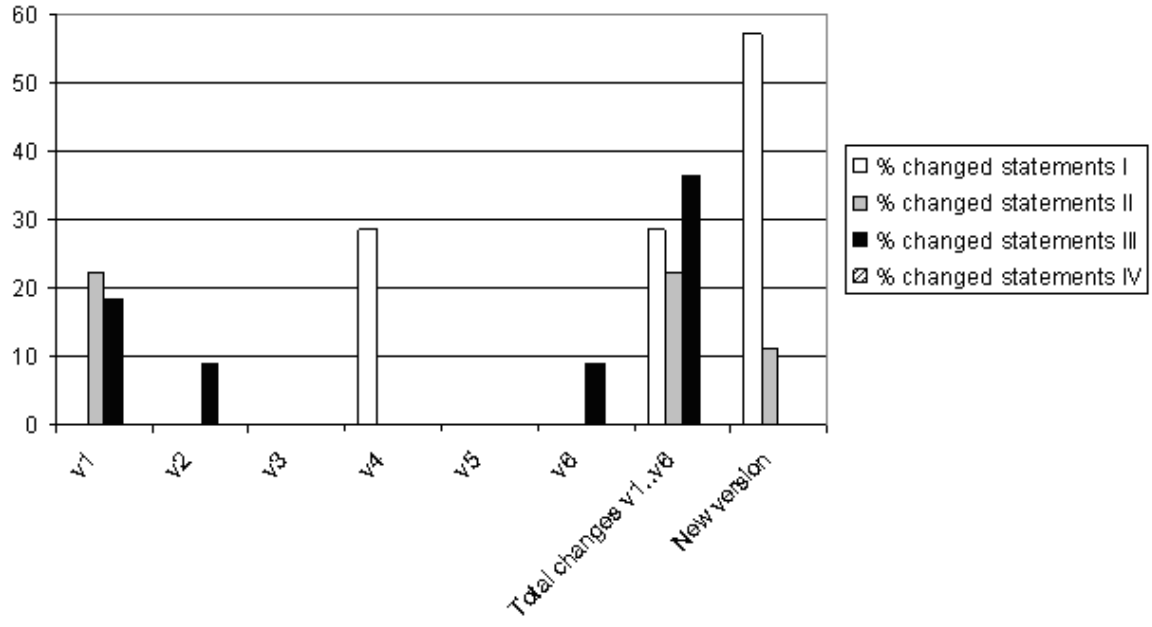
Figure 6. The percentage of modified statements, arranged by their location in the guideline, in the different algorithm versions.

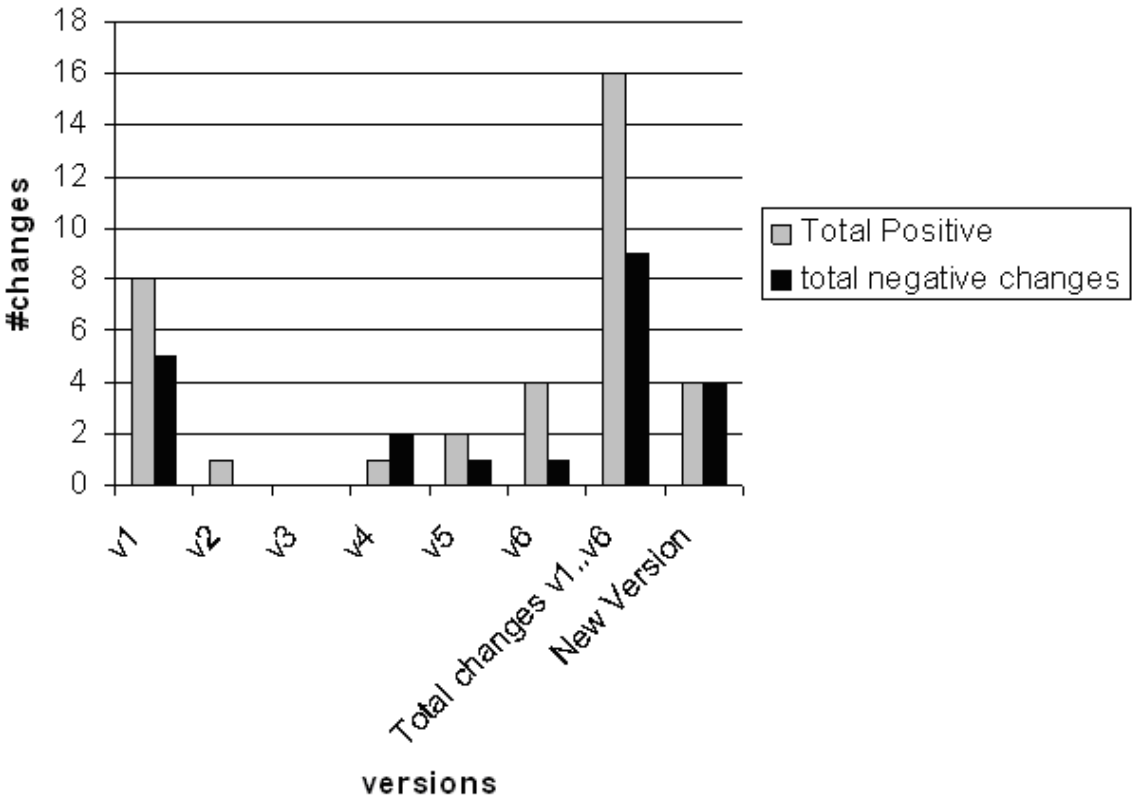

6, 18%
7, 21%
11, 34%
9, 27%
total number statements I
total statements II
total statements III
total statements IV


16
14
12
10
8
6
4
2
0
v1
v2
v3
v4
v5
v6
Total changes v1..v6
New version
Total Changes


9
8
7
6
5
4
3
2
1
0
#changes
Total Positive
total negative changes
v1
v2
v3
v4
v5
v6
Total changes v1..v6
New version
versions


5
4
3
2
1
0
v1
v2
v3
v4
v5
v6
Total changes v1..v6
New version
total clarity
total gen/spec
total interactive improvement
total quest for quality
total blunder
total forgotten function
total correction


60
50
40
30
20
10
0
v1
v2
v3
v4
v5
v6
Total changes v1..v6
New version
% changed statements I
% changed statements II
% changed statements III
% changed statements IV


18
16
14
12
10
8
6
4
2
0
#changes
Total Positive
total negative changes
v1
v2
v3
v4
v5
v6
Total changes v1..v6
New Version
versions