System and Software Requirements Validation through Inspections: Constructive Reading and Mining Requirements from Natural Language Requirements Documents

Raymond J. Schneider
Bridgewater College, Department of Computer Science, Bridgewater, VA 22812, USA

**ABSTRACT:** Defects introduced early in the effort to engineer a system due to poorly identified requirements are generally seen as a major factor leading to high system and software costs, especially if the defective requirements are undetected until later development phases in the lifecycle of engineering a system. In software development, inspection methods have been particularly successful when applied to code inspections, but have often been substantially less effective when they are applied to natural language requirements specifications. Yet, the ultimate savings due to error detection, diagnosis, and correction before a trial system is produced are generally great. This paper addresses the problem of improving requirements inspections by exploring foundational issues, such as the ability of inspectors, the degree that skill is present as a stable and determining factor, and whether defect detection is influenced by intrinsic differences in difficulty of detection among defects. A new inspection technique, denoted Constructive Reading Inspection Process (CRIP), was developed and used to explore requirements inspection, which involves extracting the conceptual entities and their interrelationships as opposed to looking solely for defects. Inspections of phase products in software development is a best-practice in systems and software engineering, however reading inspection of natural language requirements is less productive than the same practices applied to code. Our study of reading inspections reveals issues largely overlooked in the past which suggest that low yields can be partly accounted for by cognitive factors, lack of uniform expectations, and the character of ordinary reading. The Constructive Reading Inspection Process focuses on revealing and illuminating the conceptual entities and inspecting those; thereby shifting the implied operational metaphor from fishing (looking for defects) to mining (exhuming and polishing entities). This paper describes the process, its use, and provides a preliminary evaluation of its effectiveness.

**INTRODUCTION**

Despite the many methods, formal and other, proposed for capturing system and software requirements (Davis, 1990; Sage and Palmer 1990; Robertson and Robertson, 1999), the use of natural language is pervasive in effective requirements practices (Young, 2001). The primary reason that formal methods have not made greater inroads into the requirements process is very likely that they are part of the context of software engineering rather than of the context of the many problem domains to which software engineering is expected to contribute solutions. If natural language requirements documents are a generally unavoidable feature of the early stages of projects, ultimately intended to lead to the engineering of a system that is trustworthy and which satisfies user needs, and if formal methods are generally not comprehended by the domain personnel who generate the requirements, one is necessarily faced with a conundrum. How can we transition from natural language to some representation that is
more formal and appropriate for use by system and software developers and still provide for the detection of as many defects in requirements as possible before the formality becomes an intractable obstacle to the domain experts?

An accepted “best practice”, which sets the context for this work, is that of software inspection (Wheeler, Bryczynski, and Meeson, 1996; Radice, 2002). Software inspection has been applied to code inspection with great success. The applicability of this form of inspection to natural language requirements inspection is much less direct and generally a more complex and time consuming undertaking, due in large part to the inherent ambiguity of natural language. We began our efforts by studying the ability of inspectors to locate defects in models of natural language specifications. There were a number of motivations, but an important one was to see if it would be possible to use fewer, better inspectors or a better inspection process to keep down the amount of time and effort they needed to be devoted to inspections to get a good result. The N-fold result of Martin and Tsai (1990) and Schneider, Martin, and Tsai (1992) clearly indicated that the use of more inspectors and more teams of inspectors led to more and more defects being discovered. Defects were found at an approximately uniform monotonically increasing rate in the first five inspections and at an only slightly diminished rate through ten independent inspections in these studies. Through use of approaches such as this, we can ultimately expect to find all of the requirements defects, but only at a very great expense in manpower and time. Perhaps if the inspections could be limited to the use of individuals instead of teams, and to using only the very best inspectors, or perhaps through adoption of a better inspection process; the results could ultimately be improved.

In our initial study (Schneider, 2001), inspection of natural language requirements were investigated using a set of four model Software Requirements Specifications (SRSs). The selected SRSs had been used by other researchers in current and past studies. The research was directed at gaining a greater understanding of the role that learning and skill play in the effectiveness of the traditional inspection process. Our findings determined that repeating the experiment with different SRSs did not appreciably change the relative effectiveness of the inspectors. This suggests that skill was more significant than learning, at least in the context of four SRS inspections done over a relatively short interval of time. Our experiment also confirmed that inspection had a relatively low probability of finding errors and that this probability varied substantially as a function of inspector skill and the number of inspectors (i.e. combining results). A small group of the most skilled inspectors, substantially outperforms larger groups of less skilled inspectors.

Two particularly interesting results observed in our initial studies were as follows.

1. A set of individuals approximated the results achieved by Martin and Tsai (1990) and Schneider, Martin, and Tsai (1992) which were obtained on a set of teams of about the same size. This suggests that teams may not be significantly more effective than individuals in performing inspections.

2. The incidence of solitaires, or defects found by only one individual or team, dominated the statistics, a feature that has been noted in passing in other earlier experiments (Tripp, Struck, and Pflug, 1991; Porter, Votta, and Basili, 1995). The question of why this is so, is an open issue at this time, but we believe it is due to the cognitive states of the inspectors which produce selective sensitivities, thereby biasing inspectors towards some kinds of defects and desensitizing them to others.

In light of these results, the focus of our research shifted to exploring the possibility that the inspection processes could be modified from a reading-based detection process, to a process which attempted to construct a model of the entities described in the SRSs and then by inspecting the resulting model. This approach was motivated in part by the success of the Software Cost Reduction (SCR) method developed at the Naval Research Laboratory (Heninger 1980). To these ends, we conceived a generic constructive reading method; from which two different embodiments emerged through an evolutionary sequence of
exploration, test, and enhancement activities.

This work confirmed the work of many other researchers in finding that the inspection of natural language documents yielded substantially lower yields than did code inspections and also resulted in a higher variability among inspectors. A further finding was that this ability was a stable skill of the inspectors and, within the limits of the current study, showed no tendency to change. Moreover, the presence of a strong solitaire effect could not be explained except by concluding that an inspector’s performance was strongly influenced by internal cognitive states over which they have little external control. These findings strongly suggested that better methods for natural language requirements inspection needed to be sought.

What we have termed “Constructive Reading” ultimately emerged as a method of extracting requirements entities from the matrix of natural language in which these requirements reside and then transforming them to a more formal setting without substantially changing their accessibility to the original authors and domain experts that have expressed the requirements.

We begin with an account of the original experiment. We then proceed to describe the detection characteristics of defects using the detection results of the original experiment, as well as an analysis of the properties of the defects using a defect detection classification model. Next, we present the Constructive Reading Methodology, including a process for its use, and show how it provides a conservative transformation of the original natural language requirements as a natural starting point for continued developments in requirements engineering. Finally, we discuss effectiveness and provide a preliminary evaluation of the process.

THE ORIGINAL EXPERIMENT

Our initial experiment was designed to be similar to that described in Porte, Votta, and Basili (1995) without the emphasis on differences in inspection methods and by using a set of natural language requirement specifications instead of the formal SCR-based requirements which they used in their pioneering work. Our emphasis was on individual performance and elimination of the Fagan (1976, 1986) collection meeting by simply doing aggregation of the individual defect reports along the lines of the deposition based method suggested by Votta (1993).

Four SRS’s were used in the phase one experiment. The first three are those used by Dr. Victor Basili (Basili, et al, 1996) and others in their investigation of perspective-based reading approaches. The fourth SRS is the Cold Springs Railroad specification used by Martin and Tsai (1990). The SRS’s are as follows.

1. Requirements Document for ABC Video System (ABC).
3. Requirements Document for a Parking Garage Control (PKG) System.

The motivation for using materials already used in previous work by others is the conviction that good science and engineering is built on the ability to compare results. The need for importance of experimental replication is stressed by many, such as (Brooks et al, 1996) and is the foundation of the scientific method. The need for comparability demands a common substratum. Each of these specifications has been used before or is being currently used: CSRR (Martin and Tsai, 1990; Schneider, Martin, and Tsai 1992); and ABC/ATM/PKG (Lanubili and Visago, 1996; Basili et al, 1996).

Three hypotheses were investigated by our experiments:

H1: Performance in inspections is a skill which can be selected for from among a group of individuals. On the basis of H1 we expect to find that individuals vary in skill level, but that once established, the skill level is an accurate predictor of performance from inspection to inspection.

H2: Learning takes place as a function of repeated inspections. We expect that a subject’s skill will
increase as a function of experience. The degree of this increase is important from a training point of view. For example, to what degree can a subject whose performance is initially low, be increased by training and experience?

**H3: Defects vary in difficulty and this has an impact on the performance of inspectors.** In principle we can measure the observed difficulty of a defect by observing its frequency of detection. A difficult defect is operationally defined as one that is only infrequently detected when a large group of inspectors inspect the same document.

**Description of Experiment**

Figure 1 depicts the order in which the four SRSs were presented to the inspectors. The experiment was designed to capture inspection data on individual performance in finding specification defects using the method described in (Porter 1995b) as Ad Hoc. The Ad Hoc method, to quote Porter and his associates, is one in which “… all reviewers use nonsystematic techniques and are assigned the same general responsibilities.” The Ad Hoc method was selected for this experiment because it is the most natural and involves the least intervention into the process. Since we are seeking to determine whether inspectors have a stable skill, denoted by hypothesis H1, it would be inappropriate to strongly bias the inspection process by imposing structure beyond familiarization with the nature of the items being sought, i.e. defects. The experiment was performed as a sequence of four SRS reviews and is now described.

**Preparation through an Orientation Lecture**

All inspectors were provided the same lecture on defects in natural language specifications. The lecture was not concerned with the provision of inspection training, only orientation to inspections. The inspectors were introduced to the idea of defects in specification and three taxonomies were described based on (Meyer 1985, Grady 1993, Schneider 1992, Porter 1995b). The lecture describes the types of things that constitute defects and how various writers have described defects in the literature. At the end of the lecture the work sheets for capturing the defects were described in relative detail to the inspectors.

---

*Fig. 1. Sequence of Inspections in Phase 1.*
The defect collection process consisted of marking a text that contains the defect using a highlighter on the inspector’s copy of the SRS, numbering it with the number in the defect list, and noting the reason why it was added to the defect list. The inspectors were also invited to classify each defect using one of the very simple defect taxonomies provided in the orientation lecture. The primary direction given on performing the inspection was that the inspection should take no more than two hours.

*Training SRS – The ABC Video System*

The ABC Video System SRS was used as the first SRS for all inspectors. It was selected for this use since the University of Maryland group also used it as the training SRS.

*Second Round — Split the Group into Two Groups of Approximately Equal Ability*

Using the raw data results from the training SRS, the group of participating inspectors was divided into two groups of approximately equal ability based solely on their defect count on the training SRS. One group was then given SRS-B (ATM) and the other group was given SRS-C (PKG) for the first review after the training review. The purpose of this was to provide a mechanism for detecting learning by varying the sequence of presentation so that sequence dependent effects could be examined. (Note: the author participated as Inspector 1. Inspector 1 is an engineering professional with over 30 years of experience in a range of software and hardware engineering projects. Inspector 1 was not included in the division of the graduate student population into two groups.)

No debriefing or review was provided between the inspections of subsequent SRS’s. This ensures that all observed improvement was due to self-learning and not a particular intervention activity.

*Third Round*

In the third round of the experiment, those inspectors that had received ATM in the second round now received PKG and vice versa.

*Final Round*

In the final round all inspectors inspected CSRR, the specification used in the N-fold inspection papers. (Martin 1990, Schneider 1992). This specification is slightly longer and somewhat different in format from the first three. It was put last to (1) maximize the reusability of the data. The first part of the experiment replicates the Ad Hoc portion of the PBR (Perspective Based Reading) experiments performed by the University of Maryland. The CSRR experiment replicates the work of Martin, Schneider and Tsai in an individual, rather than a team context. Placing the CSRR specification last ensured that the inspectors would have the maximum experience possible within the context of the experiment. One objective was to see if there was a significant difference between N-fold inspection practiced by individuals versus N-fold inspection practiced by teams.

*Results of the Experiment*

Table 1 illustrates the number of defects that each inspector listed on their work sheets. The division in the two specifications in the middle of the table denotes the fact that inspectors 1 through 4 inspected SRS-B, the ATM specification second, while inspectors 5 through 7 inspected SRS-C, the PKG specification second.

*Experiment Conclusions*

In the context of our initial three hypotheses we concluded the following:

**H1:** Performance in inspections is a skill which can be selected for from among a group of individuals.
Table 1
Inspection Data. (Note: * Inspector 2 did not complete CSRR)

<table>
<thead>
<tr>
<th>Inspector</th>
<th>SRS-A (ABC)</th>
<th>SRS-B (ATM)</th>
<th>SRS-C (PKG)</th>
<th>SRS-D (CSRR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32</td>
<td>81</td>
<td>48</td>
<td>26</td>
</tr>
<tr>
<td>2</td>
<td>23</td>
<td>27</td>
<td>20</td>
<td>— *</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>18</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>12</td>
<td>16</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>22</td>
<td>26</td>
<td>23</td>
<td>26</td>
</tr>
<tr>
<td>6</td>
<td>17</td>
<td>55</td>
<td>28</td>
<td>33</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>17</td>
<td>9</td>
<td>11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Specification</th>
<th>SRS-A (ABC)</th>
<th>SRS-B (ATM)</th>
<th>SRS-C (PKG)</th>
<th>SRS-D (CSRR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Defects</td>
<td>83</td>
<td>119</td>
<td>134</td>
<td>95</td>
</tr>
</tbody>
</table>

**TRUE**

We found that inspector skill is measurable and significant. This is supported by the data in Figure 2. Although it varies somewhat from inspection to inspection, skill ranking measured by percent of errors detected is stable from inspection to inspection. Here, the inspectors were rank correlated pair-wise between each pair of inspections. Rankings were significant using the Spearman Rank Correlation Coefficient, a standard statistical measure in hypothesis testing, at a level less than or equal to 0.025. This means that any one of the pairwise findings could happen by chance with a probability of less than or equal to one chance in 40. The probability that they would occur jointly is less than one chance in 1600 since the first and last pairs are independent.

**Inspector Performance**

![Inspector Performance Graph]

Figure 2. Summary of Inspector Performance.
H2: Learning takes place as a function of repeated inspections.

**FALSE OR VERY WEAK**

Average performance was not significantly affected by repeated performance of inspections. If there was any effect it was a slight lowering of performance since the last two inspections had an average detection rate of 20.8% versus the first two which had an average detection rate of 22.9%. Individual inspector performance varied somewhat more than the average, but in both directions. Inspector 2 degraded in rank in each of the first three inspections and did not complete the fourth. This suggests a growing lack of interest more than a lack of skill. Inspector 6 scored 4th on the first inspection and then scored 2, 2 and 1 which suggests a jump discontinuity learning event, possibly in motivation. The other five inspectors never changed rank by more than one rank either up or down. Since there was no discussion of results between inspections, the only opportunity for learning was the process of inspection itself. The data indicates that learning by doing was not a significant effect in this experiment.

H3: Defects vary in difficulty and this has an impact on the performance of inspectors.

**INCONCLUSIVE**

It was originally expected that there would be enough inspectors that the data could be partitioned and that defect difficulty could be estimated by the number of inspectors detecting and then statistical re-sampling of the data could verify a difficulty effect by showing that the less skilled inspectors didn’t detect the more difficult defects. While it can’t be ruled out that defects differ in difficulty, they almost certainly do to some extent, the number of inspectors available was too small to demonstrate any direct effect on a defect by defect basis.

Subsequent analysis of the data, aggregated by defect types, suggests that an effect related to what an inspector “sees” as a defect was at work. Some categories of defects proved to be harder to see than others.

**Experimental Surprises**

The experiment revealed several surprises. Table 2 illustrates the distributions of error detections. The low effectiveness of the individual inspection of nominally 20% was not a surprise. It corresponded

<table>
<thead>
<tr>
<th>#Inspectors Detecting</th>
<th>ABC</th>
<th>ATM</th>
<th>PKG</th>
<th>CSRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>57</td>
<td>64</td>
<td>94</td>
<td>74</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>27</td>
<td>20</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>14</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>83</strong></td>
<td><strong>119</strong></td>
<td><strong>134</strong></td>
<td><strong>95</strong></td>
</tr>
</tbody>
</table>
approximately to the N-Fold result for single teams. Particularly surprising, however, were the following results.

1. There is a nearly total lack of evidence for learning from inspection to inspection.
2. There is an extremely high incidence of solitaires, i.e. defects detected by only a single inspector.
3. There is a considerable lack of evidence supporting difficulty as a factor in performance.

Two factors strongly suggest that inspections, as currently practiced for SRS inspection using Fagan-like methods, may not be the right thing to do. The first factor is the low detection rates scored by individual inspectors, nominally 20%. This indicates that the Ad Hoc process is a weak process. The second factor is the high solitaire rate which suggests that what defects are found depends largely on uncontrolled cognitive factors that differ considerably among and across inspectors.

FACTORS INFLUENCING DIFFICULTY

The idea that difficulty played a role in defect detection was impossible to establish on a simple interpretation of the data from the experiment. However, a related issue, that of classifying defects by kind, was explored and led via an unexpected path to a demonstration that difficulty, or at any rate selective sensitivity to defects by kind, played a role. We conducted a survey of the literature to discover how defects had been classified in the past and turned up relatively few papers on classifying defects in the context of the inspection of requirements documents.

Three defect taxonomies were examined: Bertrand Meyer’s *Seven Sins of theSpecifier* (Meyer, 1985), *Omissions and Commissions* by the N-Fold researchers and extended by Porter, et al. (Porter 1995b) and *The Hewlett Packard Taxonomy* described by Grady (Grady 1993). We decided to use a taxonomy that in spirit, if not in substantial detail, was derived from Grady. We called this taxonomy *The Constructive Defect Taxonomy*.

**Constructive Defect Taxonomy (Entity-Attribute-Value)**

The Constructive Defect Taxonomy relates defects to the kinds of things that are being described, hence the term constructive as relating to the parts that go into the construction of something. This taxonomy has a distinct reality model view of the SRS based on the notion that reality is composed of physical things and non-physical things and that SRS documents are about these things. In order to classify a defect we form a descriptive three-tuple of words which jointly provide the classification. The form of the tuple is <entity> <attribute> <value> and the terms corresponding to the components of the tuple are listed in Table 3 below.

The system is inherently extensible by adding terms to the various components of the tuple. However, to keep the classification system maximally usable, it is important to avoid multiplying categories without good reason. This system was developed in a heuristic manner by classifying the defects found in the inspections. Each defect in the SRS defect lists was classified without constraints on the terms used. Then the lists were studied to weed out redundancy and synonyms. The results are presented in Table 3. No claims are made for this taxonomy other than it was found to be heuristically useful. There are 350 possible distinct tuples, each of which can be thought of as a defect category. In the development data set there were 442 defects which were classified into 83 categories.

Application of the Constructive Defect Taxonomy involves:

1. Deciding what kind of entity a defect involves (i.e. What is defective?)
2. Deciding what aspect of the entity is defective? and
3. Deciding in what way that aspect is defective?
Table 3
Constructive (Entity-Attribute-Value) Defect Taxonomy Categories

<table>
<thead>
<tr>
<th>&lt;entity&gt;</th>
<th>&lt;attribute&gt;</th>
<th>&lt;value&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meta* = M [41(8,33)]</td>
<td>Reference = R [33(8,25)]</td>
<td>Missing = M [161(44,117)]</td>
</tr>
<tr>
<td>Data = D [142 (47,95)]</td>
<td>Definition = D [304(96,208)]</td>
<td>Wrong = W [48(17,31)]</td>
</tr>
<tr>
<td>Physical = P [77(15,62)]</td>
<td>Content = C [65(15,50)]</td>
<td>Ambiguous = A [47(7,40)]</td>
</tr>
<tr>
<td>Process = Pr [166(53,113)]</td>
<td>Value = V [2(1,1)]</td>
<td>Confusing = C [16(5,11)]</td>
</tr>
<tr>
<td>Constraint = C [16(3,13)]</td>
<td>Usage = U [19(1,18)]</td>
<td>Redundant = R [5(1,4)]</td>
</tr>
<tr>
<td></td>
<td>Initialization = I [13(4,9)]</td>
<td>Incomplete = I [120(37,83)]</td>
</tr>
<tr>
<td></td>
<td>Enforcement = E [6(1,5)]</td>
<td>Meaningless = Mn [8(5,3)]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unnecessary = U [10(4,6)]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contradictory = Cn [12(2,10)]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Out of Order = O [15(4,11)]</td>
</tr>
</tbody>
</table>

One of the interesting aspects of the results was that the ratio of multiple detections to solitaires among defect types appeared to distribute the defect kinds in an order which could be plausibly interpreted as difficulty. The classification is recorded by using the tuple of letters which characterize it. For example PrRA is Process-Reference-Ambiguous. The notation, ex. [33(8,25)], in the table gives the number of defects incorporating that category and the split between multiple detections and nulls and solitaires. For the total of 442 defects classified, 126 were multiple detections and 316 were solitaires and nulls. The ratio of multiple detections to solitaires and nulls is almost exactly 40%. Categories greater than 40% on a difficulty interpretation would be easier than average and categories that are less than 40% would be more difficult than average. The fractional difficulty values from this interpretation are as follows:

- Meaningless 1.666 (easiest to find)
- Unnecessary 0.666
- Wrong 0.5483
- Confusing 0.4545
- Incomplete 0.4457
- Missing 0.376
- Out of Order 0.3636
- Redundant 0.25
- Contradictory 0.20
- Ambiguous 0.175

When one examines this ordering it is significant that the order is plausible. Meaningless things should stand out. It is plausible that Unnecessary and Wrong things would be easier to see than Confusing or Incomplete things. All of the categories Missing, Out of Order, Redundant, Contradictory and Ambiguous require knowing about more than one thing in order to define them. Something missing is missing in some context. Redundant means that it has been said more than once, while contradictory means that there is something repeated that is in conflict with other statements. Ambiguous is less direct than contradictory. It is interesting and important to note that the ordering is both plausible in the large, in that stark singular things like unnecessary and wrong are easiest to see, whereas things that involve relations among other things are harder to see. These relational “things” are also in a plausible order. The ordering obtained among the entities is as follows.

- Data 0.4947
- Process 0.469
- Meta- 0.2424
- Physical 0.2419
Constraint 0.2307
Data and Process are both very common and relatively easy to detect than Meta-Physical and Constraint entities, which are harder. Finally, looking at the Attribute category results in the ordering:
Value 0.5 (since there were only 2 this may not be very significant)
Definition 0.4615
Initialization 0.4444
Reference 0.32
Content 0.3
Enforcement 0.2
Usage 0.0555

This discussion illustrates that the notion of difficulty measured as the ratio of multiple detections versus nulls and solitaires produces a plausible ordering of the categories. This suggests that the notion of “difficulty”, in particular difficulty interpreted as degree of visibility, as opposed to difficulty interpreted as hard to understand, plays a role in the detection of SRS defects. Later work we performed on tracing defects using CRIP-2 products and the phase one detection results further illuminates this distinction in a simple Defect Recognition model.

The Constructive Reading Inspection Process: CRIP-1 and CRIP-2

The development of the Constructive Defect Taxonomy focused on the notion of defects as associated with the conceptual entities that were described in the document and not defects in the document itself (the Meta- category). This led to the idea that perhaps defects would be more easily seen if the SRS was analyzed from the point of view of the entities which were buried in the SRS. This led to two efforts to develop such a methodology and the insight that what we were doing was moving from one metaphorical representation of inspection to another, from fishing to mining.

Fishing
The fish (or defects) are hidden in a sea of surrounding text that gives some indications of the presence of the fish. The inspector is a fisherman using various lures, bait, nets (techniques) and careful attention to what is being done to catch the fish. But fishing is a chancy and uncertain process. Sometimes the fisherman’s attention lapses and the fish get away. In any case, there are always many more fish left behind and uncaught than are caught. The inspection process might be summarized as follows in the context of the fishing metaphor.

• You catch some fish (defects)
• Probably more little ones than big ones
• Some people are better at it than others
• More fisherman catch more fish
• The big fish sometimes get away
• You never catch all the fish

Mining
The mining metaphor focuses not on finding defects, but on finding the ideas that form the SRS. The defects become a byproduct of inspecting the ideas. A mining process requires a mining methodology. The notion of mining calls to mind ideas of digging for treasure, panning for gold as well as related ideas of refining, polishing or as is the case with jewels and diamonds, setting in an attractive way. The mining methodology for SRS inspection consists of four sub-processes:

1. Extraction — getting to the ideas that are in the SRS and separating them from the multitude of things that are not relevant, in Meyer’s terms the noise.
2. Filtering — organizing the ideas into a standard form so they can be critically examined.
3. Assaying — checking the ideas for completeness, correctness, and clarity.
4. Refining and Proofing — correcting the final product to produce the desired result.

The mining metaphor can be used as a check-list for preparing a variety of inspection processes which take a different perspective from traditional fishing inspections which concentrate on finding defects and not on clarifying and proofing ideas.

Since it follows the mining metaphor, CRIP is a process of inspection that seeks to discover the SRS ideas by decoding them from the document and then through inspecting these ideas and not the natural language requirements document itself. Through this, the inspection of the document is seen as a two part effort:

1. A transcription from natural language to some other more precise form which is still accessible to ordinary readers, and
2. An effort to translate the resulting more precise embodiment of the ideas in the requirements statement into a canonical or standard form which aids subsequent inspection.

The degree of precision possible using these kinds of methods is a function of the degree of formality and rigor applied. However, as formality and rigor is increased, accessibility to those untrained in the methods naturally decreases. This suggests that a better approach might be a middle-way, one that conforms to the relative comfort levels of both users and developers who are the primary stakeholders in the requirements phase of engineering the system.

**CRIP Variants**

CRIP begins with a transcription process which transforms the natural language SRS into a more formal representation using column oriented spreadsheet representations to allow easy processing. Two variants were developed. The first used a statement oriented approach. This required the transcriber to accomplish a mental mapping of natural language statements onto an English-like artificial language mapping to an entity model space which was termed the Entity-Basis-Model (EMB). The initial EBM which was defined and explored used statements in an extendable Slot-Template format based on an underlying object oriented entity set. A variety of statement types sketched in Figure 3 in an abbreviated fashion capture general processes describing classification, composition, activities and relational dependencies in explicit declarative sentences that are sufficiently English-like to be easily understood and sufficiently formal to form a bridge from the natural language specification to design. Classification and composition templates should serve to illustrate the idea.

![Fig. 3. A Generic Meta-Model (Entity Basis Model).](image-url)
Classification


Composition


and the inverse form:

[A|AN|THE] <part of item> IS PART OF [A|AN|THE] <composite item>. //IPO can be used as a contraction for IS PART OF.

When these ideas were manually applied to an analysis of the four example SRSs it was found that the process was far too tedious and time consuming to be practical as a manual process. Moreover, while using spreadsheet processing to sort the entities into clusters of common objects prior to creating canonical forms worked, the EBM had to be force-fit into the column oriented spreadsheet format and this added additional complexity to the process. Despite these inadequacies, CRIP-1 worked well. However, it was difficult to compare it to the Fagan type inspection process since it did not support the objective of tabulating defects as entities. Nevertheless, the process showed sufficient promise that we searched for an expression of the process that would be easier to transcribe and better matched to spreadsheet analysis methods. This yielded a different EBM, the SPOR (Subject, Process, Object, Relation) model which was instantiated as CRIP-2.

Martha Kolln, in her book “Understanding English Grammar” (Kolln, 1990) points out that “Ten sentence patterns account for the underlying skeletal structure of almost all the possible grammatical sentences in English.” She goes on to say “… Probably the easiest way to think about those sentence skeletons is to recognize them as a series of slots, or functions, each one of which is filled by a particular structure, or form.” This observation fitted well into the <slot> + <template> view that CRIP-1 had adopted, with the added advantage that all the ten patterns fitted into a four slot (hence four spreadsheet column) format. The template text disappears from the EBM and in its place are a set of generic column headings. All ten patterns fit into this form and while the patterns, as Kolln says, “… are much simpler than most of the sentences we use in speech and writing …” we must first understand the simple skeletons. Thus CRIP-2’s SPOR EBM were initially adopted because of this fact about English, and the fact that the resulting transcription method was conformal to English and slot oriented. It is still necessary, however, for a transcriber to simplify through paraphrase, complex sentences that do not map immediately onto this form. We will describe the SPOR EBM later. This directly accomplishes the two objectives of making the transcription process substantially easier, since the EBM is much closer to English syntax, and the resulting transcription directly and naturally maps onto spreadsheet columns for easy manipulation.

The Effect of Distance on Defect Recognition

Throughout the study of defect detection represented by this work, there was an ongoing tension associated with two questions:

- What is a defect?
- How is a defect recognized?

For operational reasons we chose to accept as defects those things that inspectors considered defective after being exposed to various examples of defects in the context of the defect classification systems introduced earlier. The next effort is that of searching for a model of defect detection.
ANALYSIS OF DEFECT DETECTION USING A SIMPLE MODEL

Model of Defect Recognition

A model of defect recognition must be provided to define the recognition process. We will express the defect recognition process as one of the inspector seeing something that is perceived to be wrong or require clarification. The inspector sees the defect either directly or indirectly. To see it directly is to see it without needing to see any other thing, hence the standard for a direct perception is within the control of the inspector. The directly seen defect either

1. Fails to match the inspector’s expectations (i.e., fails to conform to an internal standard which the inspector maintains and produces a question of interpretation.), or

2. matches something for which the inspector has bad expectations.

The first we will classify as DMx (Direct Match Failure) and the second we will classify as DMb (Direct Match Bad).

The indirectly seen defect is a defect which requires two or more statements to be apprehended by the inspector to recognize the defect. These are also classified in two kinds:

1. two statements that directly conflict in some way in the inspector’s mind, and

2. a statement which produces an expectation that there will be one or more other statements that will resolve an expectation and these are not found.

The first we will classify as IC (Indirect Conflict) and the second as IM (Indirect Missing information).

In each kind of defect recognition (direct and indirect), the standard for recognition is the expectations of the inspector since the recognition of a defect is a human cognitive act. The difference is primarily one of apprehending one thing (the statement recognized to be defective directly in the first case) or apprehending more than one thing (a problem which to be seen requires more than one statement).

The Analysis

To explore the recognition model we examined the 83 defects from the ABC SRS. The raw data consisted of logs and classifications of the defects and the CRIP-2 data from the ABC SRS consisting of the SPOR transcription and various analysis products. To allow references between the two data sets, the data was prepared by

1. locating all the defects in the original SRS and labeling them from the log, and

2. finding the statement number in the CRIP-2 transcription of the ABC SRS that corresponded to the location of each defect.

This effort was not entirely successful since the transcription did not include the 8 meta-defects. This is not, however, seen as a defect of the process since meta-defects are not defects of the ideas but of the requirements document itself. This left 75 defects to analyze.

The Distance Metric

The cognitive distance is defined as the difference between statement line numbers separating the two statements required in the case of IC defects that are required to recognize the defect. Where more than one statement was found in conflict, and there were only ten of these in this effort, only one was used.

Processing the Defects

A spreadsheet was developed with the seventy-five defects that remained. The defects were classified according to the four recognition categories. There were 10 DMb, 10 DMx, 13 IM and 42 IC. The observed probability of detecting these defects was calculated by dividing the number of detections by the number of detection opportunities. The eight meta-defects were detected with a probability of 0.304. DMb defects were detected with a probability of 0.286, DMx with a probability of 0.143, IM with a probability of 0.176 and IC with a probability of 0.204. Two defects were eliminated as redundant.
Of the various categories, only the DM categories can be directly apprehended since they can be “seen” at a single site in the SRS without reference to any standard but an internal one. The difference between DMb and DMx probabilities is large and suggests that the inspectors have large differences in their internal standards. Note that DMb is the closest to the Meta value of 0.304 which is the category likely to have the greatest intrinsic agreement. So we attribute the low value of DMx to the fact that the internal expectations of the inspectors differ. The low value of IM is likely to be due to the open ended character of that kind of defect. A statement is seen which raises an expectation in the inspector’s mind. However, the defect is only recorded if the inspector remembers that he had an expectation that was never satisfied.

As noted, two of the IC defects were eliminated due to redundancy. The remainder were analyzed based on the distance between the statements which jointly defined them. One would expect one of four kinds of matches a subject match \([S1,S2]\), an object match \([O1,O2]\) or one of two kinds of cross-match, \([S1,O2]\) or \([O1,S2]\). These four kinds of matching will result in the statements being close together in the SPOR spreadsheet processed form termed an ASSEM, which presents statements with the same subject or object together.

To study the effect of inter-statement distance on recognition probability the 26 IC statements which shared an entity category and the 14 IC statements that crossed categories were examined separately. The distance between the two statements in the SPOR transcription jointly defining defects was termed the Transcription Delta (T-Delta). The defects were sorted by T-Delta and the observed probability of detection was calculated for the top half and for the bottom half separately. This data is presented in Table 4.

<table>
<thead>
<tr>
<th>Defect Recognition Category</th>
<th>#</th>
<th>Observed Probability of Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>META</td>
<td>8</td>
<td>0.304 Highest</td>
</tr>
<tr>
<td>DMb</td>
<td>10</td>
<td>0.286 Highest non-Meta</td>
</tr>
<tr>
<td>DMx</td>
<td>10</td>
<td>0.143 Lowest</td>
</tr>
<tr>
<td><strong>Total Direct</strong></td>
<td>20</td>
<td>0.214</td>
</tr>
<tr>
<td>IC-Entity High TDelta</td>
<td>13</td>
<td>0.208</td>
</tr>
<tr>
<td>IC-Entity Low TDelta</td>
<td>13</td>
<td>0.242</td>
</tr>
<tr>
<td>IC-XCAT High TDelta</td>
<td>7</td>
<td>0.163</td>
</tr>
<tr>
<td>IC-XCAT Low TDelta</td>
<td>7</td>
<td>0.204</td>
</tr>
<tr>
<td>IM</td>
<td>13</td>
<td>0.176</td>
</tr>
<tr>
<td><strong>Total Indirect</strong> *</td>
<td>53</td>
<td>0.202</td>
</tr>
<tr>
<td>ALL DEFECTS</td>
<td>83*</td>
<td>0.213</td>
</tr>
<tr>
<td>* 2 IC defects were redundant but were included in Total</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Summary of Trace Findings**

The following summarizes our observations concerning the experimental trace findings.

1. Defects Vary in Observed Probability of Detection by Defect Recognition Category in ways that suggest that different Defect Recognition Categories relate to real differences in “visibility” of the defect.

2. IC Defects which are grouped by high and low distance measures in the transcriptions have relatively lower observed detection probability than defects that are closer together.
3. Statements which together share a common Subject or Object term are detected more easily than those that don’t share a common term (Entity versus XCAT at high and low TΔ respectively).

4. Defects which have multiple associated statements that share common terms typically sort to be very close to each other, which would make their recognition in the sorted transcription easier.

A Total of 35 defects (26 IC and 9 others that were IM and DMb or DMx), involving multiple statements, were sorted from originally far from one another to very close. Figure 4 shows the distance between related statements before sorting (nominally 100 statements) and Figure 5 shows the distance between the same statements after sorting (worst case 16 and most one or two). Table 4 shows that distance reduces detectability. Thus these 35 defects would be substantially more detectable after sorting than before sorting.

Fig. 4. Defect Recognition Distance Before CRIP.

Fig. 5. Defect Recognition Distance After CRIP.
CONSTRUCTIVE READING

This section briefly discusses the Constructive Reading process termed CRIP-2 SPOR. A more complete treatment can be found in (Schneider 2001). We have previously mentioned that the initial version of CRIP proved to be tedious when done manually. Moreover, since there was no intrinsic fit to the spreadsheet paradigm, processing was a somewhat hit or miss affair. In the course of this initial exploratory phase however, some of the inspectors had explored alternative representations and came close to the SPOR representation which was further motivated by studying the normal grammatical structures of English. The Subject-Process-Object-Relation (SPOR) representation was explored in detail through the expedient of using it on the model specifications. CRIP-2 followed the basic idea of the mining metaphor’s four steps of 1) Extraction, 2) Filtering, 3) Assaying, and 4) Refining and Proofing.

Extraction → Transcription Structured for Spreadsheet Processing

Extraction is implemented as a transcription process which converts the SRS from its Natural Language source to a spreadsheet form that facilitates processing using sorting. This requires a columnar format in which the columns are meaningful and consistent. The method adopted in CRIP-2 combines ideas suggested by student inspectors in the experiment and were extended by the author. Specifically the CRIP Transcription process uses a minimalist English structure that sees all sentences as being of the form

```
(line #) <SRS page #> <entity/agent> <process/activity> <entity/object> <relation> <related entity>
```

(each bracketed term is a separate column)

This is also referred to as Subject/Process/Object/Relation or Agent/Activity/Object/Relation order. The primary distinctions are those of transitive sentences in English. The Subject or Agent acts performing a Process or Activity that affects the Object which is the receiver of the action, often with some condition or mediation Relation which modifies the activity. For example, line 148 of the ATM transcription reads:

```
<S> <P> <O> <R------------------------- >
ATM dispenses Money IF transaction is successful. — the ATM is the Agent, dispenses the Process, and Money the Object. The conditional Relation is IF and the condition is a successful transaction.
```

Operations on Transcribed Statements

The four slots labeled S-P-O-R in the CRIP together form a statement and the entire set of statements \( \{S,P,O,R\} \) comprise the requirements document’s transcription, i.e. \( T=\{S,P,O,R\} \). Because it is close to English patterns it is readily accomplished and conservative. The operations used to manipulate the transcription are just regroupings of the statements to produce focused views of the transcription. Most of the time the R column can be neglected as a sort entity since it is a component of only three of the patterns and when present is reflexive on the object.

Subject Sorts

We consider first the sorts that begin with S as the primary key. There are two possible secondary keys, neglecting R and these are P and O. We are more interested in grouping operations together that all affect the same Object than to group by the same processes that affect different Objects, so the most effective secondary key is likely to be O. The result can be designated: \( T_{SOP} =\{S_1,P_2,O_3,R\} \), where the subscripts now refer to the sort order. When the transcription is sorted in this way, all the Subject interactions with the same Objects will be grouped together. This immediately identifies the connectivity among Subject and Object entities. Thus, we see that SOP sort produces information of the kind:
\{S \text{ (particular Subject)} \mid \text{Process}_1 \ldots \text{Process}_n \mid O \text{ (particular Object)}\}

This form is a listing of the ways the Subject influences the Object and can be mapped as functions of the Subject (functional mapping) or functions of the Object initiated by the Subject (object oriented mapping).

One of the key ideas is that the Process is a link between the Subject and Object and must be “sited” or owned by one or the other. There are certain processes that are used by many Subjects and/or applied to many Objects and these should be reified. But this is usually done following a convention that can be summarized as: \textit{A Sawmill is a Verb}. The particular function: \textit{sawing wood} becomes a Subject: \textit{Sawmill}, which is a reification, or perhaps subjectification (if the function is self-triggering), of a process. So we pass wood to the \textit{Sawmill} to be sawed because: \textit{A Sawmill saws Wood}.

All the slots of a transcription statement may not be occupied. Perhaps the most common are imperative and intransitive sentence forms. The first has only the implied subject “you” or “it” — things like \textit{Read the Bar Code}. There is an implied Subject which would be obtained by inverting the sentence as \textit{Bar-Code Reader Reads the Bar Code}. But it is transcribed as <\text{blank}>,<\text{Read}>, <\text{Bar Code}/the, <\text{blank}>. (note that articles and modifiers are generally translated to the end, ex. Bar Code/the, House/green, so that sorting will operate on the right words. Another intransitive sentence would look like \text{Menu initialized} and would be transcribed directly as <\text{Menu}>, <\text{initialized}>,<\text{blank}>, <\text{blank}>. The second form would be captured by the SOP sort, but the first would only appear in the sort in the set of all imperative statements with a subject that is <\text{blank}>.

\textbf{Object Sorts}

Similar to the Subject sort, there are two possible secondary keys for the Object sort. There is less rationale for choosing between them since some Objects will be used by several Subjects and in that case one might find that a OPS sort order would best cluster the data. The alternative is the OSP order which like the SOP order emphasizes the Subjects and Objects over the Processes. The Subjects and Objects are generally the entities which exhibit persistence in the environment of any application. With the exception of existential verbs (IS/ARE) and compositional verbs (HAS), most processes are transient, and may be thought of as having short lifetimes relative to the Subjects and Objects.

\textit{Filtering} \rightarrow \textit{Sorting the Transcription into Entity Sorts}

Filtering is a process of grouping the data into concentrated form for inspection. The primary columns in the transcription are Subject/Process/Object similar to Class/Responsibility/Collaborator columns in CRC cards but intended by design to be closer to English. The filtering process, which was developed through experience, begins with sorting the data by columns. Then each primary sort key is treated as if it were a particular view of the system, that view is analyzed by grouping the entities. This results in a set of derivative views of the transcription, which conserve the transcription.

\textit{Sorting by Primary Columns – Subject/Process/Object}

There are six possible permutations of sort order for three things taken two at a time:

SPO, SOP, PSO, POS, OSP, OPS

Of these six sort orders, the most useful appear to be SOP, OSP, and PSO. The first two restructure the transcription by persistent entities, i.e. subjects and objects, and the last structures the transcription statements by process and subject, i.e. initiating agent. The first two facilitate an object-oriented view of the SRS, while the last facilitates a transformational view. SPO has also been useful. Different inspectors may find different orderings or even different sorts helpful. For example, with experience, inspectors might sort on the relation column and create a SR sort order to associate Subject and Relations.
Fig. 6. Diagram of Baseline and Intermediate CRIP Products.

The sorting process is quickly accomplished using Excel by first copying the transcription to a new worksheet, and then selecting all the lines of the transcription except for the headings. The specific sort is defined by clicking on the menu item Data, selecting Sort and then structuring the sorting order. The result is a sorted transcription. The original transcription can be recovered by selecting all the lines of the transcription and resorting on the pre-pended transcription line number.

Figure 6 illustrates the procedures which jointly define CRIP-2. The process is divided into a Baseline process, and an Intermediate process. These terms are chosen to reflect the probability that a future CRIP process which extends these methods would be termed “Advanced.” An Advanced process would incorporate the existing process elements and increase the level of automation to allow products to be generated in a semi- or even fully automated manner.

Figure 6 is a diagram which captures both the order of processing and the nature of the processing of an SRS. The first step in this is to transcribe the SRS in an SPOR format which yields a spreadsheet with semi-formalizes the SRS but with minimum violence to its readability. In addition to the SPOR columns there are also page number and line number columns appended to the SPOR columns. Reading from the top down, this is step 1. Step 2 takes and makes copies of the transcription into tabs in the spreadsheet for as many or as few of the sorts as are desired, nominally SPO, PSO, OSP, and OPS although others can be added such as RSP. These are nothing but versions of the original transcription sorted by the primary keys and secondary keys in key order, so SPO for example means we sort by Subject first, then by Process and then by Object. This leaves a spreadsheet with exactly the same lines as the original transcript, but sorted in SPO line order.

Each sort takes only moments, so the actual time once the transcription is complete to get to the third tier is small. The third tier is the ASM sorts. This involves going through the SORTS and inserting heading lines for groups with the same subject in an SPO sort, or the same object in an OSP sort. It groups together statements with the same primary key value. Thus if you are interested in all the places where the Customer is referenced as a subject you look in the SPO sort under Customer.

The MISC spreadsheet tab is one in which singleton statements are all grouped together from the
various ASM sorts and then checked to make sure that they are not singletons in all three primary keys. If they were then cutting them would involve loss of information. Below the third tier are more specialized products which are used to study the SRS or to build interpretations of it. The ASSEM is an elementary example. It is a spreadsheet that combines the SPO and OSP sorts into a single tab such that the same entities (Subject value = Object value) are grouped together. This groups all uses of an entity together in a single group in a transcript. Up to this point all the various tabs are conservative – they are just the original transcription at the top of the chart sorted into different orders and grouped. The ASSEM is conservative also, but doubled since the SPO and OSP sorts are two different views of the entire transcription. Each of the elements in Figure 6 will now be described.

Canonical Forms (Variations on a theme)
Canonical forms can range from stopping at the assembly stage, the SRS-ASMs and the ASSEM combined Subject/Object assembly. Then the CRIP can be used as an aid to more conventional reading inspection, a super index into the SRS. The independent view of the SRS facilitates and enhances single inspector inspection. The most ambitious use of CRIP is as a full constructive method for building a model of the ideas in the SRS. We will refer to the first form as Baseline CRIP and the second as Intermediate CRIP. In order to achieve Baseline CRIP, one must create a transcription, a set of entity sorts, and a set of assembly versions through the list sort and ASSEM. The tabs of a typical SPOR Baseline CRIP spreadsheet are as follows.

Baseline CRIP Tabs

RawENG — This represents the transcript in the selected style. The SPOR style spreadsheet has the headings: <lines> <page> <Subject> <Process> <Object> <Relation><Related Entity><Comment><Extended Comment>

Entity Sorts — This is comprised of SPO,PSO,OSP View sorts, additional sorts include OPS and RSP.

Assembly Sorts — These represent SPO-ASM, PSO-ASM, OSP-ASM, which are just Entity Sorts with inserted column labels for groups of common entities and singletons relegated to a MISC category. The ASSEM assembly listed below in Intermediate CRIP-2 is a combination of a Subject assembly sort and an Object assembly sort.

LIST — This represents the collection of S-P-O group headings, relations can also be listed if desired.

MISC — This is a spreadsheet that collects all the MISC statements from the assembly views together and then sorts them on line number. This allows a check that no lines were triple singletons and hence escaped consideration by falling exclusively into the MISC category. This never actually happened in the experiment. If it happens it signals a defect in either the SRS or the transcription.

Intermediate CRIP
The Baseline CRIP is essentially the product of the first two CRIP steps of Extraction and Filtering. Intermediate CRIP goes beyond the baseline process by introducing additional products, specifically additional tabs and constructions using the products of the baseline as inputs.

Intermediate CRIP Tabs and Constructions
TRIM and CUT Scripts — These are the first sheets which are individually non-conservative, but are jointly conservative. The CUT script ends up with redundant and questionable statements leaving
the TRIM script which is usually a variant of the SPO-ASM script as a condensed version of the SRS sorted by Subject and Process.

IS(ARE) and HAS — These are clip-scripts since they are just the IS(ARE) and HAS script clipped from the PSO-ASM. They are handy to look at statements which made classification and composition statements. They can be usefully sub-sorted into SPO or other orders to collect all the IS and HAS statements together by Subject or Object.

Extended Assembly Script — This is a constructed script which cuts SPO or SOP and OSP scripts together to form an ASSEM script which groups Subject References and Object References. This can optionally be used as the basis of the TRIM and CUT script activity. It combines the Subject and Object view which is advantageous, however it doubles the number of lines to be inspected intercutting the full transcription with itself.

RSP Sorts — These are sorts that extend beyond the SPO entities can be helpful if the transcriber uses orderly conventions in making the transcription. Sorting on the Relation field will group statements with common functions such as conditional statements that use S-does P- to O- IF - condition. The Relation column and the columns beyond can also be used explicitly by the transcriber to include markers for special kinds of statements, ex. ASSUMPTION, CONDITION, or to preserve various identifiers or aspects of the requirements document which otherwise would be lost. Examples: (1) identifiers such as FR#n used in ATM to allow functional requirements statements to be regrouped; (2) sequence information that is multi-statement in extent, a transcriber could use a step code as FR#1 STEP 1 in the comment field. A definite transcription style will tend to emerge on repeated use of the process as the CRIP implementors gain experience with the method.

Context Diagram — The context diagram is a visual composition which can be done synergistically at the same time that the Container View and the Temporal Structure are being prepared. The context diagrams done in this work generally use a container metaphor to represent parts and a road or connection metaphor to represent sequence. In the context diagram entities are exhibited inside one another or optionally with a line adorned with a bullet to capture PART-OF relationships. Lines generally show flow of messages, control, or activation.

Container View — The text based container view is an indented representation of the SRS entities which uses indentation, emphasis (bold, etc.) and prefix symbology used to create an abbreviated listing of the major entities with behaviors and rules associated with the entities that perform them or initiate them. A diagrammatic Container View is a graphical restructuring of the entities (Subjects and Objects) showing in a visual way, through overlapping, how the entities relate to one another (see Schneider, 2001 for many examples).

Temporal Structure — The time dimension is represented by the lifelines of entities which exhibit strong persistence in the application. Behaviors of these entities are either on-going, hence looping, behaviors, or triggered behaviors, hence initiated by some kind of stimulus usually provided by a different entity from the one executing the behavior.

EVALUATION OF CRIP

Use of the CRIP methodology and implementation process using the SPOR model, demonstrates that
single individuals could readily create a comprehensive analysis of a requirements document in timeframes that were much less, measured in person hours, than those using tradition Fagan-like reading inspection methods. The primary difference in the two efforts is the creation of the SPOR transcription, which in the cases studied took an average of 12 minutes per page and covered 64 document pages. Once the transcription is accomplished, the document is analyzed using spreadsheet methods and then inspected entity by entity. It is difficult to compare these two kinds of methods because they share so little in common. The reading model depends on the inspector “seeing” the defect as they read. This is intrinsically related to the expectations of the inspector. By contrast the CRIP model collects together all the statements about the various entities in the SRS, putting them in close proximity. When this is accomplished the probability of seeing deficiencies is enhanced as demonstrated in Table 4 where cross category detection, for example, was increased by 25%. However, to look at CRIP in this way is to put on the glasses of the current metaphor, the search for defects by reading and what CRIP accomplishes is to focus on the entities themselves and inspect them for consistency and integrity. The CRIP methodology coalesces the entities into clusters of functionality that can be inspected constructively by putting the pieces together. The method looks more for integrity than for defects. By clustering all the statements about the customer together, a model of the customer can be extracted from the SRS and looked at individually. This can be done for each entity. The result is the construction of LISTS of entities (the set of Subjects, Processes, and Objects clustering in the sorts), and the raw material to build comprehensive views, spatially and temporally using diagrammatic or textual methods using the CRIP tabs as raw material. This yields much more insight than reading alone. Space considerations do not allow us to present all of the CRIP products; however, it cannot be less effective than Fagan inspection as it contains an intense Fagan reading in the transcription process and the subsequent analysis increases the likelihood of detecting significant classes of the hardest to detect defects. Thus on inspections of any cardinality, CRIP can be expected to be superior. This suggests its effectiveness and future studies and independent experiments to provide confirmation of this effectiveness.

CONCLUSIONS

Our primary conclusions to this work are as follows.
1. Fagan like reading inspections of natural language SRSs is a low yield process whose popularity is mainly due to its success in code inspections.
2. A Constructive Reading Inspection Process (CRIP), particularly one using the SPOR form provides a better alternative because it is semi-formal while retaining contact with the domain experts, it is objective and not dependent on internal cognitive states in the inspectors. Instead of providing just a marked up SRS as a product, it provides entity lists, Subject/Object utilization patterns and a variety of other objective products which provide a natural bridge to subsequent activities.

This paper presents an approach which appears to have direct and relatively immediate application in industrial development settings, especially those constrained by small team size or time pressures, where traditional methods such as extended Fagan inspections by multiple teams are not feasible.

REFERENCES

Note: ISERN (International Software Engineering Research Network) Reports may be found in postscript or Adobe PDF format at http://www.iese.fhg.de/network/ISERN/pub/isern_biblio_tech.html


Schneider, R. (2001). *Improving Requirements Inspection Through The Use of a Constructive Reading Inspection Process*. Dissertation Submitted to the Graduate Faculty of George Mason University, Spring UMI # 3000908.


Ray Schneider received a B.S. in Physics from St. Joseph’s college, an M.S. in Engineering Science from Pennsylvania State University, and a Ph.D. in Information Technology from George Mason University, the latter in 2001. Dr. Schneider worked as a research scientist for the United States Navy at the Naval Air Development Center from 1966 until 1977 developing magnetic sensors and signal processing technology. From 1977 until 1989 he worked for Sperry Defense Systems Division, later Unisys, where he rose to Director of the Superconducting Electronics Technology Center and later Chief Scientist of the Washington Systems Engineering Center. In 1989 he joined ComSonics Inc., a developer and manufacturer of cable television test equipment, as Director of Engineering. In 2002 he was a visiting Assistant Professor of Computer Science at James Madison University. He is currently an Assistant Professor of Mathematics and Computer Science at Bridgewater College in Bridgewater, Virginia. His interests include software and systems engineering, signal processing, knowledge representation and programming languages.