Understanding Design Patterns with Design Rationale Graphs

Elisa Baniassad and Gail C. Murphy
Department of Computer Science
University of British Columbia
2366 Main Mall Vancouver BC Canada V6T 1Z4
{bani, murphy}@cs.ubc.ca

Christa Schwanninger
Siemens AG, CTSE 2
Otto-Hahn-Ring 6, 81739
Munich Germany
christa.schwanninger@mchp.siemens.de

ABSTRACT

A Design Pattern presents a proven solution to a common design problem using a combination of informal text, diagrams, and examples. Often, to suitably describe an issue, the author of a Design Pattern must spread and repeat information throughout the Pattern description. Unfortunately, spreading the information can make it difficult for a reader to grasp subtleties in the design, leading to possible misuses of the Pattern.

In this paper, we introduce the Design Rationale Graph (DRG) representation that connects and visualizes related concepts described in a Design Pattern. The localization of concept information is intended to help improve a reader's understanding of a Design Pattern. Improved comprehension of a Pattern could aid the use of a Pattern during implementation, and the reading of code built upon the Pattern. In addition to describing the DRG representation, we present a tool we have built to support the semi-automatic creation of a DRG from Design Pattern text, and we report on a small study conducted to explore the utility of DRGs. The study showed that readers with access to a DRG were able to answer questions about the Pattern more completely and with more confidence than those given the Design Pattern alone.

Keywords

Design patterns, Design rationale, Design comprehension, Program understanding

1. INTRODUCTION

What is so exciting about [design] patterns? It is probably the fact that they constitute a 'grass roots' effort to build on the collective experience of skilled designers and software engineers. Such experts already have solutions to many recurring design problems. Patterns capture these proven solutions in an easily-available and, hopefully, well-written form.[3]

Authors of Design Patterns face a difficult task: they must try to convey complex problems and solutions in a comprehensible way to readers of the Pattern. To achieve this task, Design Pattern authors use a mixture of informal text, diagrams and examples. Text is used to describe the context of the design problem, the solution, and the forces impinging upon a solution, amongst other issues. Diagrams are used primarily to convey the structure and dynamics of the presented solution. Typically, a Design Pattern will be reviewed and revised several times to help ensure the desired information is being conveyed.

Readers of a Design Pattern, who are most often software developers, also face a difficult task. To use the Pattern appropriately, or to understand the use of the Pattern, a developer needs a thorough understanding of when the design applies, how it works, and why it works. In a Pattern, much of this information is typically presented through the informal text. Text is used because it provides the needed expressiveness. However, the use of text comes at a price: readers can find it difficult to grasp subtleties of the design because information pertinent to an issue is spread throughout the Pattern text. Even for small Patterns, this spreading of information can be problematic. For example, in an exploratory study we conducted, (Section 4), we found that software developers who believed they understood the 14-page Visitor Pattern [7] were unable to correctly answer a question about the basic operation of the Pattern.

To help Pattern readers, we have created the Design Rationale Graph (DRG) representation, which supplements the original Design Pattern. A DRG aggregates, organizes, and visualizes information in the Design Pattern according to design elements. Design elements are any concept or entity described in the Design Pattern text. A reader can query a DRG to find all design elements related to a particular concept and can view those elements in context. The identification and viewing of detailed design information in context facilitates a developer's understanding of how the design described in the Pattern works. In particular, a DRG can help a reader understand the rationale—the why—behind choices made in the design presented by the Pattern.

We begin with a brief example of a DRG created from the Visitor Pattern and a description of how the DRG has helped developers improve their understanding of the Pattern (Section 2). Then, we provide a more detailed description of the DRG representation, the operations available to help a developer navigate around a DRG, and the semi-automatic tool we have developed to create DRGs from Pattern text (Section 3). Next, we describe an exploratory study we conducted to determine if

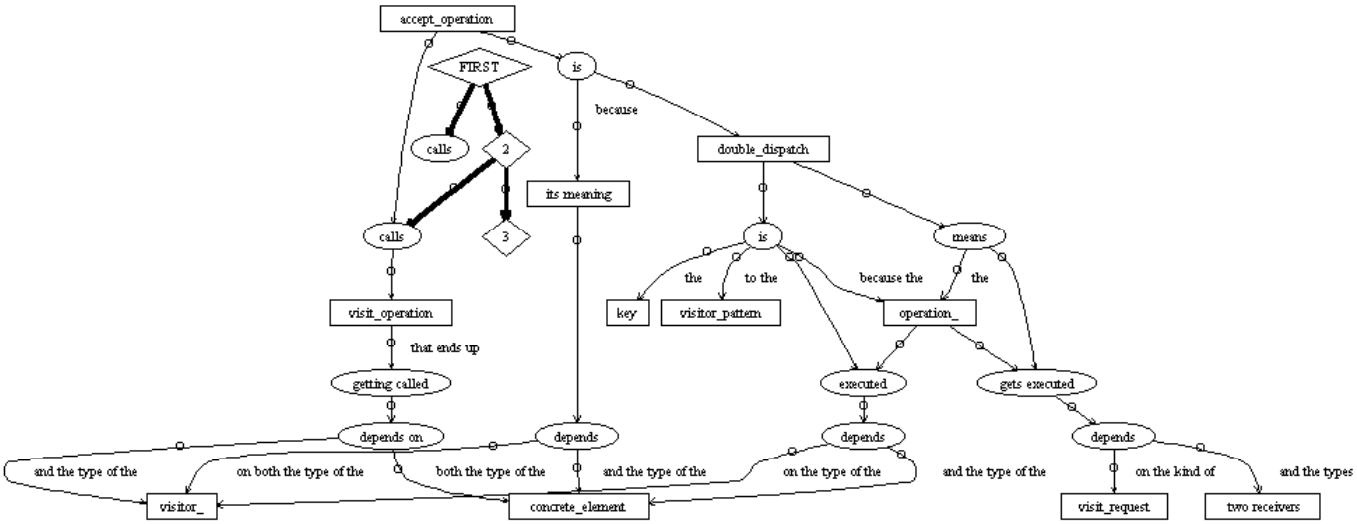

Figure 1: A DRG about double dispatching in the Visitor Pattern

the DRG representation and tool shows promise in helping increase the design detail and context reported by developers when investigating questions about a Pattern: this study involved software developers from both academia and industry and considered two patterns, Visitor and Reactor [14] (Section 4). We conclude with a discussion of outstanding issues (Section 5), related work (Section 6) and a summary of the paper (Section 7).

2. EXAMPLE: A DRG OF VISITOR

To illustrate a DRG in action, we show how the tool can be applied to help understand the Visitor Pattern.

The Visitor Pattern supports the selection of a method to execute based on both the type of the initial recipient of a message and on the type of the sender of that message—the caller. In other words, the Visitor Pattern provides support for double dispatching. A user of the Visitor Pattern must understand that double dispatching is at the core of the Pattern to apply the Pattern correctly and to reap its benefits. Surprisingly, when we asked two pattern-aware developers to read Visitor and answer a few questions, they were unable to answer completely a question about how the method to execute is chosen. One developer responded that the method was chosen based only on the type of the caller; the other responded that the method was chosen based only on the type of the message recipient.

In contrast, developers who had access to the DRG representation and tool were able to answer the question correctly. Figure 1 shows the portion of the DRG used by the developers. This DRG localizes information about the method determination that was spread throughout the Pattern text. The rectangles in the DRG represent design elements and nouns found in the pattern, the oval nodes represent verbs, and the edges are labelled with phrases linking the nouns and verbs. The subject of a verb points into the verb; the object of the verb is pointed to by the verb. Diamond-shaped nodes indicate a sequence of events.

Reading this DRG, a developer can determine several pertinent facts: the `accept` operation calls the `visit` operation as the second in a sequence of three calls; the `accept` operation is a double dispatch operation because its meaning depends on both the type of the `visitor` and the type of the `concrete element`; the `visit` operation that is called depends on the type of the `visitor` and the type of the `concrete element`; and double dispatch is the key to the Visitor Pattern.

The full DRG representation of the 14-page Visitor Pattern is large, comprising over 250 nodes and 400 edges. Examining the entire DRG to help understand the Pattern is thus impossible. Instead, developers use a set of operations to select relevant portions of the DRG. For example, the DRG in Figure 1 was produced by including the paths that contained the word "depend", and then expanding on the relationship between the `accept` operation and the `visit` operation. A developer might choose to search for the word "depend" because they recall from, or see in, the pattern text that the execution *depend*ed on a number of factors. After viewing the graph produced by this first query, the developer may chose to expand the graph to include invocation information between the `accept` and `visit` operations. This pair of queries collects information about double dispatch and puts the information in the context of the calling structure explained in the pattern.

3. DESIGN RATIONALE GRAPHS

Design Rationale Graphs are intended to help developers understand informal design text, such as that found in Design Patterns, by semi-automatically structuring, amalgamating and graphically displaying the text. In this section, we describe how to create, read and manipulate a DRG.

3.1 Creating a DRG

Creating a DRG for a Design Pattern requires two inputs: the text comprising the Pattern, and a dictionary of design elements specific to the Pattern. We define a design element as an entity, participant, or concept. Some examples of design elements are names used in the implementation, such as method names or class names; other examples are concepts described in the Pattern, such as double dispatch.

Providing the Pattern text is easy as it can be extracted from a digital representation of the Pattern. This extracted text requires one step of pre-processing by the DRG user before it can be input to the DRG tool: the user must annotate the text to include sequential information. The annotation involves adding the word "First" to the beginning of the first sentence in a set of steps, and the word "then" to the beginning of each subsequent sentence. Although this might appear onerous, it took less than 10 minutes to annotate the text of the Visitor Pattern.

Providing the dictionary of design elements is somewhat more involved. To help the user with this step, our tool presents the user with a list of nouns found in the Pattern text. The user then peruses the list and selects the design elements. The choice of design elements dictates the structure of the DRG: a noun designated as a design element is represented by one node in the graph whereas separate nodes are used to represent occurrences of non-design element nouns. Among the noun phrases selected as design elements for the Visitor DRG were "double dispatch", "visitor", "accept operation", and "concrete element. Nouns not chosen as design elements included "key", "meaning", "class", and "call". For the Visitor Pattern, it took about five minutes to choose the 17 design elements for the dictionary from the 93 nouns in the Pattern text. Since the process of creating a DRG is not onerous, if a user reading a DRG finds a concept not captured as a design element, the user may iteratively update the dictionary and recreate the DRG.

To create a DRG from the Pattern text and the dictionary, our tool uses a parts-of-speech tagger, called LTCHUNK [11] to identify the noun and verb phrases in Pattern text sentences. Running LTCHUNK on a sentence from the Visitor Pattern results in the following mark-up: noun phrases are enclosed in double square brackets and verbs are enclosed in double parentheses.[1]

> *Paragraph 2, page 339:* [[double_dispatch]] ((is)) the [[key]] to the [[visitor_pattern]] because [[the operation_]] ((executed)) ((depends)) on the type of the [[visitor_]] and the type of the [[concrete_element]].

Our tool processes sentences one at a time. For each sentence, the tool must determine the nodes and edges to be introduced into the DRG. Noun phrases are mapped to nodes as described above. Each occurrence of a verb phrase introduces a new node into the graph. Edges are determined as follows. The first graph node identified in a sentence is considered a source node, regardless of whether it is a node based on a verb or a noun phrase. Each subsequent node encountered in the same sentence is considered as a destination node and an edge is introduced into the graph between the source and destination node. When a node based on a verb phrase is encountered, the source node is reset to the verb node. The edges identified in this way are labelled by the phrase, if any, linking the noun and verb phrases.

Thus, in the sentence from Paragraph 2, page 339 stated above, "double_dispatch" is created as a node in the graph and is considered a source node. Since the next phrase, "is" is a verb phrase, a new node is introduced into the graph, an edge is created between "double_dispatch" and "is", and the source node is reset to be the "is" node. When the next phrase is encountered, "key", a node is introduced for the phrase, and an edge is created from "is" to "key" with label "the". The fragment of the DRG created from this sentence can be seen in Figure 1. The remaining sentences used to create Figure 1 are shown below.

1. *End of page 338:* Double dispatch means the operation that gets executed depends on the kind of visit request and the types of two receivers.
2. *End of page 338:* The accept operation is double dispatch because its meaning depends on both the type of the visitor and the type of the concrete element.
3. *Middle of page 337:* The visit operation that ends up getting called depends on both the type of the concrete element and the type of the visitor.

Our tool outputs the DRG in the AT&T graphviz format [1]. The graphviz (dotty) package can then be used to view the DRG.

3.2 Reading a DRG

A DRG preserves all the text from the original Design Pattern. Sentences from the Pattern are shown as chains of verb phrases with the subjects and objects attached to the chain. The first verb in a sentence has no incoming edges from verb nodes, and the final verb node in a sentence will have no outgoing edges to verb nodes.

Regardless of their position in the chain, verb nodes will have an incoming edge from their subject, and an outgoing edge to their object. Hence, to find the subject of a verb, a user follows the edge incoming to the verb node backwards to a noun or design element. To find objects of a verb, a user follows the outgoing edges. For example, in Figure 1, the `accept` operation points to a `calls` verb, hence it is the subject of the verb. The same `calls` node points to the `visit` operation, which is the object of that verb.

For the sake of maintaining contextual information, it may be necessary for a user to understand the ordering of sentences from the Pattern text. For instance, adjacent sentences may describe a sequence of calls between methods. Ordering is shown through sequences, diamond-shaped nodes, in the DRG. In a sequence, the first verb is pointed to by the `FIRST` node, the second is pointed to by the node labelled `2`, and so on. For example, in Figure 1, the `FIRST` node points to an unexpanded `calls` verb. The `2` node is read next, and points to another `calls` verb that links the `accept` operation to the `visit` operation. A third `calls` verb follows, which is also unexpanded.

For increased clarity, the DRGs are shown in colour when presented on screen. Noun nodes are shown as blue, verb nodes as purple, and nodes related to adjectives are shown as pale yellow. Each new sequence has its own colour, and all sequence nodes in a particular sequence share that colour. This helps readers identify verbs that are parts of the same sequence even when the entire sequence is not shown.

3.3 Manipulating a DRG

Because even small Patterns produce large graphs, a user needs support in manipulating a DRG to produce a useful view. To help the user generate views pertinent to a concept of interest, our tool has operations to expand or subtract portions of the

[1] Underscores were introduced during dictionary pre-processing.

graph related to nodes specified by one or more regular expressions. The expansion or subtraction can be with relation to the entire graph or with relation to a sub-graph.

For example, the user who expanded the DRG for the Visitor Pattern based on the accept and visit operations would have asked for an expansion based on the regular expressions `visit.*`, `accept.*` and `impl.*` to get all related implementation nodes to the operations of interest.

The DRG manipulation operations are currently supported by a set of command-line tools. A graphical interface to make these operations easier to apply is planned, but has not yet been implemented.

4. EXPLORATORY STUDY

To determine whether the DRG representation can help readers who are trying to understand a Design Pattern, we conducted a small, exploratory study. The hypothesis of this study was that the use of a DRG would increase the amount of design detail and context reported by Design Pattern readers.

4.1 Study Format

We broke the study into two blocks, each of which consisted of four, single participant trials. In each block, half of the participants, the test group, had access to a DRG of the Design Pattern; the other half, the control group, worked only from the Pattern. All participants were asked questions about the Design Pattern. We compared the responses of the control group to the test group within each block. We then compared the results between the blocks.

The participants in the first block were software developers from Siemens AG. These participants worked with the 22-page Reactor Pattern [14].[2] The second block involved four graduate students from the University of British Columbia (UBC). These participants worked with the Visitor Pattern.

4.1.1 Patterns Used

We used two different Patterns to help reduce the likelihood that a problem in understanding the Pattern was related to the way in which the Pattern was written, or to the questions we asked. The two Patterns we used have different authors and are of differing size: the Visitor Pattern is short but subtle; the Reactor Pattern is longer and more detailed.

4.1.2 Participants

We kept the skill set of participants within each block as similar as possible. All of the participants in the Reactor experiment were non-native English speakers, with similar experience in reading and writing English. Each participant possessed the equivalent of a Bachelor's degree in Computer Science, and had at least one year of experience working with Java in an industrial setting. Each participant was screened to have a certain level of exposure to Design Patterns, but no exposure to the Reactor Pattern.

[2] The Reactor architectural pattern allows event-driven applications to handle service requests sent to applications by one or more clients.

The participants in the Visitor Pattern experiment were all PhD students at the University of British Columbia. None of the participants had previous exposure to Design Patterns. Each participant was a native English speaker.

4.1.3 Experimental Set-up

In each trial, a participant was given the same set amount of time to read a hard-copy of the assigned Design Pattern. At Siemens, the participants were given one hour to read the Reactor pattern: At UBC the participants were given 20 minutes to read the Visitor pattern.

After reading the pattern, participants in DRG trials were asked to put away their copy of the Pattern. They were then given a 20-minute tutorial on reading a DRG. After this tutorial, they were asked a predetermined list of questions about the Pattern. They were not allowed to refer to the hard-copy of the Pattern while answering the questions. They were allowed to iteratively ask the experimenter to perform an operation on the presented DRG of the Pattern and were able to view the DRG resulting from the operation. We chose the approach of the experimenter performing the DRG query because of usability concerns about the current DRG tool interface.

Participants in the control group were asked the same predetermined questions about the Pattern. In contrast to the DRG trials, these participants were allowed to refer to their copy of the Pattern, and any notes they had taken while reading the pattern.

Participants in trials involving the Visitor Pattern were asked to answer, as fully as possible, three questions:

1. What allows the Visitor to directly access the concrete element?
2. How is it determined which operation is executed?
3. What is the sequence of events that occur in the Visitor Pattern?

For the Reactor study at Siemens, the experimenter was not on-site. Instead, the experimenter interacted with the participants over the phone and over the web. These participants were asked a different set of three questions:

1. What does the logging handler register with, and what does it register for?
2. About what does the synchronous event demultiplexer notify the initiation dispatcher?
3. What happens after a connection request arrives?

It was reasonable to expect that both the control trial and the DRG trial participants could answer these questions for two reasons. First, the questions we asked of the participants about the Pattern could be answered based solely on the information in the text of the Pattern. Second, both the control and the DRG trial participants were given ample time to read the Pattern, and participants in the control group were allowed to re-read the Pattern as much as they wanted within the allotted time. The control trial participants were thus not at a disadvantage compared to the DRG trial participants.

4.1.4 *Evaluation Questions*

After the participants had responded to the Pattern-specific questions, they were asked follow-up questions.

Participants in the control trials were asked about their confidence in their answers to the Pattern questions, how they used the Pattern text to reach their answers, and from where they drew their answers in the Pattern text.

Participants in DRG trials were asked four questions.

1. Did the graphs help you visualize design entities?
2. Did the graphs help you visualize relationships between entities?
3. Did the graphs help you feel more confident about your answers?
4. Would you choose to use this tool again?

4.1.5 *Study Limitations*

The format of the study has several drawbacks, including the small number of participants, the small number of Patterns, and the lack of a group who had both the Pattern text and a DRG available.

The small number of users and Design Patterns in our study affects the generalizability of our results. We chose to limit the number of users and Patterns because we were focusing on an initial determination of whether the DRG representation showed promise. We believe our results can answer this question because we varied the background of the participants, including both experienced software engineers and graduate students, and because we selected Patterns that differed in authorship, style, and length.

In our study, we chose to have the test group use only the DRG rather than both the text of the Pattern and the DRG because we wanted to isolate the use and effectiveness of the DRG. At this exploratory stage, we did not want to give the participants a choice about the degree to which they relied on the DRG to answer their questions.

4.2 Study Results

The results of our study supported our hypothesis: DRG participants gave more complete and detailed responses than control participants. Our analysis of the results also provided two

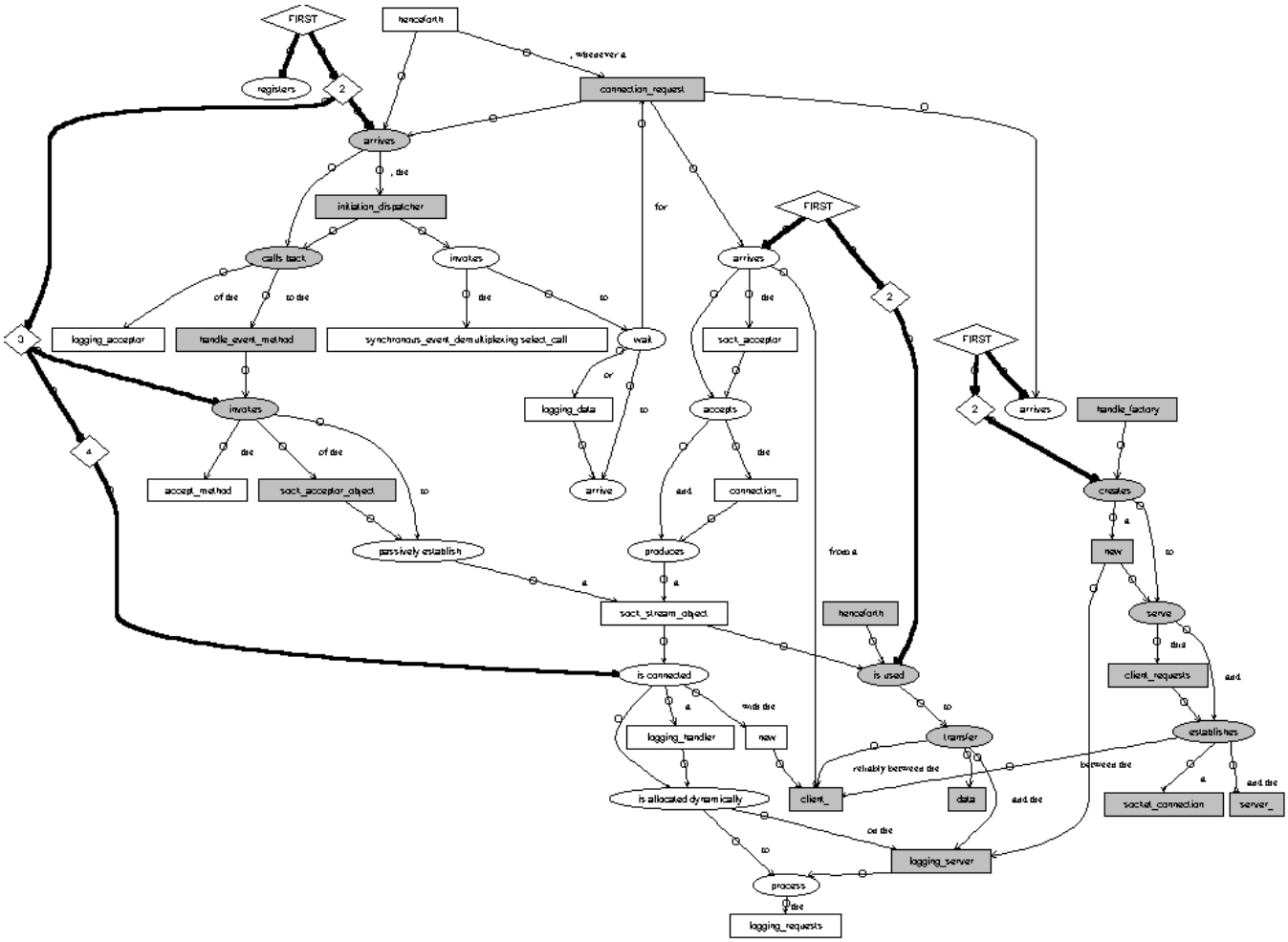

Figure 2: DRG for Reactor Experiment Question 3.

Gray nodes show the pattern portions considered by both control and DRG participants.

unexpected results: DRG participants tended to take more time answering the questions, and they reported more confidence about their responses.

4.2.1 Detail and Completeness of responses

For both Patterns, we observed that the DRG participants provided highly detailed and precise answers. In contrast, control participants using the original form of the Pattern typically responded with higher-level conceptual information. This observation particularly held for information that could be considered "obscure".

The first question in the Reactor trials, for instance, dealt with a specific example from the Pattern. The control trial participants tended to answer about the general case, rather than about the situation described specifically in the example. Although their answers demonstrated that they understood the relevant concept, they missed stating the precise type of events registered for by the logging handler used in the example, the type of event handler it registered, and details about the entity with which it registered. The DRG participants did not miss any of these details.

In the third question of the Reactor trials, the participants were asked to explain what happens after a connection request arrives. To help them answer the question, the DRG trial participants asked to see a graph relating specifically to the arrival of connection requests. The graph shown in Figure 2 was created by querying for arrival in the context of connection requests, and then by expanding the sequences in which the arrival nodes appeared.

The answers given by the DRG trial participants were more complete and more detailed than those given by the control trial participants. Figure 2 depicts the difference in the answers of the two groups. The graph shows the details expressed by the DRG participants. The details given by either of the control trial participants are coloured gray. The colouring of nodes shows that the control participants missed many design details. Among them, the passive establishment of a `sock_stream` object, and the invocation of the synchronously demultiplexing `select` call.

4.2.2 Willingness to Explore

The DRG participants spent more time answering the questions than the control participants, who all took less than five minutes to answer each of the questions.

When we asked the control participants why they did not take more time to re-read the Pattern to provide detail, two of the control participants said they did not know it was required, implying that they would not voluntarily do so, and the other two explicitly said they did not feel they would get anything more out of "flipping through" the text.

For example, participant A, in the Visitor control trials, responded incorrectly to Question 1. When asked why he did not look in more detail for the relevant parts of the Pattern, one Visitor participant said:

> *[it] seemed familiar, but I didn't think I could flip back and find it. I did kind of hesitate with the text going 'do I remember at all where that was, or am I going to have to re-read the whole thing?', and then decided I had a pretty good idea where [it] appeared.*

In contrast, the DRG participants spent approximately half an hour answering each question. At some point, each of these participants noted that they believed they had collected all necessary information, but wished to continue exploring "just to be sure". When asked why, one participant responded:

> *Well ... it's that [with a DRG] I can start by looking in at a place where I believe is a starting point where I want to begin, and then I can go a little bit out from there, and I can go a little bit down a path and kind of go "no that's not working out" and quite quickly go another way.*
>
> *Whereas maybe in the text, it's more like, maybe I'll start a paragraph and I won't know where its going, and I'll think "I can cut that", but I feel like maybe I've wasted a lot of time, and I SHOULD have read that paragraph. And I feel like [a DRG] helps me very quickly zoom in on the relationships.*

Another indication of the willingness to explore was the inclination of the participants to modify their original answers based on new information gathered either from the text or from the DRG. All control participants were asked to look through the Pattern and report the source of their answers. They all used this perusal to support their original answers, even when those answers were incorrect as happened with two Visitor participants.

Both during their exploration of the DRG and upon later reflection, the DRG participants all noted that they had incompletely, or incorrectly answered the questions before they began exploration, and that they were able to improve their answers through the use of the DRGs.

4.2.3 Level of confidence in answers

Although all control trial participants reported that they could not be fully confident about the completeness of their answers, they were confident about the correctness of their answers. Only one Reactor control participant admitted little confidence about the correctness of all his answers. The two Visitor control participants expressed complete confidence in their answers to questions two and three, but less confidence about question one, although they both strongly believed that they were partially correct. The other Reactor control participant felt confident about all his answers. Too see if they were correct in their levels of confidence, we examined their answers.

As mentioned before, the Reactor control participants both answered all the questions mostly correctly, while missing details of the answer.

The control participants in the Visitor experiment both rightly lacked confidence on question one, which they both answered incorrectly. For Question two, about which they were both highly confident, they answered incompletely, forgetting that it is both the type of the visitor and the type of the concrete element that determine which visit operation is eventually called. One said that it was only the type of the visitor, the other said only the

type of the concrete element. In question three, only one participant was able to give details about the invocation of the visit and accept operations; neither recalled how the accept operation was called (even though one was working from the interaction diagram shown in the text, page 335, [7]).

The DRG participants all stated that the graphs helped them collect the relevant information together, so they could answer the questions more completely and with more detail. They all noted that they felt more confident answering the questions using the graphs, than they did answering from memory before using the graphs. Three of the four DRG participants noted that if they were able to refer both to the text and to the graph, they would feel most confident about their answers.

4.3 Summary of Results

We found the results of the study encouraging from three perspectives: DRG readability, support for detailed understanding of design concepts, and support for linking design context to design elements.

After a 20-minute tutorial, all participants in the DRG trials were able to read the graphs with relative ease, and were able to collect the information displayed in the graphs to fully answer the questions posed.

The DRG participants answered the questions in a more detailed way than those using the Pattern because the DRG participants examined portions of a relevant DRG sub-graph containing the details before answering. The control participants, in contrast, referred specifically to only one portion of the text per question, and even then, they did not delve deeply enough in the text to draw out all relevant details.

Finally, the DRG participants noted design concepts that provided context for the design elements involved in answering the questions. For instance, in the Reactor trials, only the DRG participants noted information about how a process blocks while awaiting arrival of events. This information helps ensure the concept of the responsiveness of servers to clients. In the case of Visitor, only the participants in the DRG trials connected the double dispatch concept to how the method to be executed is determined. In each of the DRG trials, the participant noted the relevant concept information only after seeing it connected to

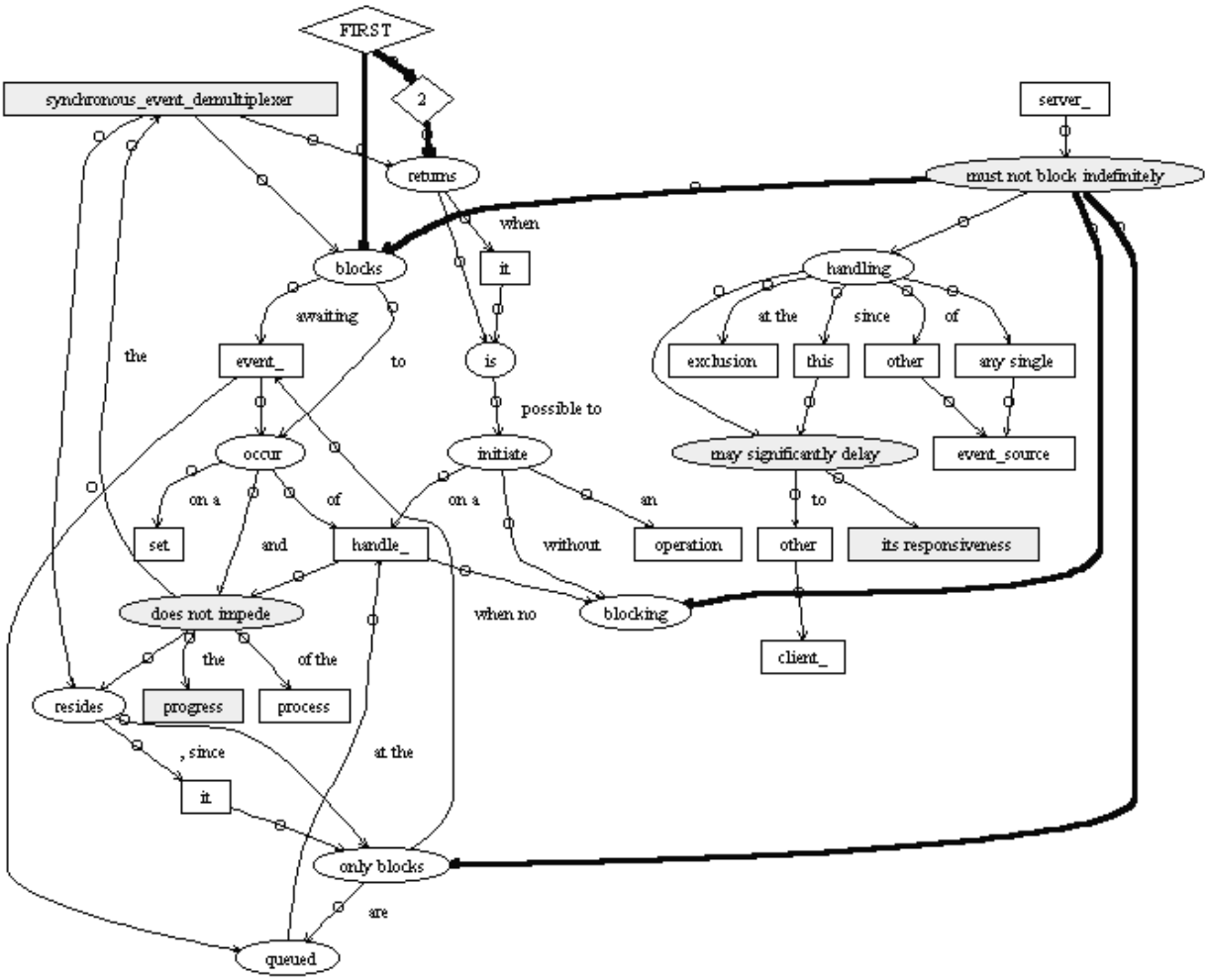

Figure 3: How the synchronous event demultiplexer contributes to server responsiveness

parts of the graph being viewed.

5. DISCUSSION

To date, our focus has been on the utility of the DRG concept and the feasibility of creating DRGs from Design Pattern text. Given the early stage of this research, there are a number of outstanding issues related to using, creating, and representing DRGs. We discuss several issues in each of these categories.

5.1 Tracing Rationale

A primary motivating factor behind the development of DRGs is the desire to help software developers understand the *why* behind the source code in a system. All too often, a software developer who is faced with making a change to an existing system must guess why the source code has been implemented in a particular way. An incorrect guess about the design rationale behind a piece of code can lead to the violation of properties of the system, such as the particular structuring chosen to ease future changes, the performance of the system, or even the intended behaviour.

The use of Design Patterns to implement systems can help make the rationale of a design decision more apparent. A developer who recognizes the use of a Pattern can read the Pattern to more fully understand the design problem and solution. However, as we have noted in this paper, it can be hard for a reader of a Pattern to link together all relevant rationale information with particular pieces of the solution.

Although we have focused on examples in this paper primarily dealing with how a solution described in a Pattern works, we believe DRGs can also be helpful in linking the information contained in the Pattern with pieces of the solution. For example, Figure 3 shows the result of two queries that combine to describe how, through its process blocking conventions, the Synchronous Event Demultiplexer design entity contributes to server responsiveness. The nodes shown in gray in Figure 3 are highlighted to show nodes of special interest. The dark edges show how low-level implementation details correspond to server responsiveness.

5.2 Establishing Relationships

There are two limitations to our current DRG creation algorithm: pronouns and synonyms. Currently, pronouns and synonyms in the input text must be massaged to enable the formation of an appropriate DRG.

The problem with pronouns is that they are interpreted as new noun nodes, and are thus inserted into the graph, without adequate linkage to the concept or entity to which they refer. This problem can be overcome by replacing the pronouns with appropriate concrete nouns. Typically, this massaging is necessary only for pronouns that appear at the beginning of sentences. If a pronoun appears in the middle of the sentence, it typically refers to a noun within the sentence. Since most queries do not break sentences up when returning portions of interest in the graph, these pronouns will be attached to the concept or entity to which they refer.

Along the same lines, there is no automatic support for synonyms. Currently, to assure that all synonyms of a noun or concept are linked it is necessary to replace all mentions with one common label. This ensures that all the related paths will converge into one node.

More sophisticated text analysis support and the input of a synonym dictionary could help address these problems.

5.3 Complementary Design Documentation

A DRG does not add or infer any information that was not present in the design documentation from which it was created. A DRG simply provides a different view of the information. A DRG is thus meant to complement, not replace, the existing documentation.

In particular, a DRG does not retain the ordering of sentences in the design documentation. As a result, it is difficult to read large bodies of text from a DRG. A DRG allows specific concepts and entities to be explored while the documentation explains the "story" of the design.

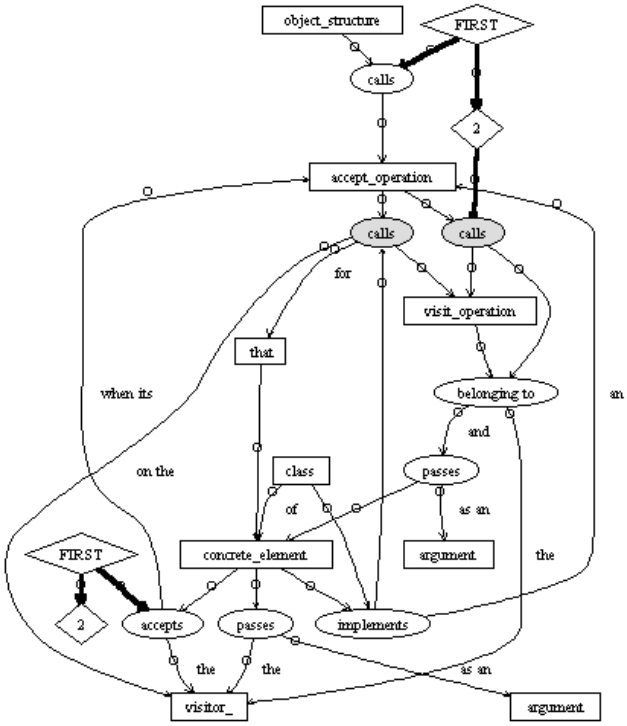

Figure 4: Gray nodes show duplication

5.4 Improving the DRG Representation

Observing the participants in the DRG trials, we learned of three areas needing improvement in the DRG representation: sequences, examples, and repeated concepts.

5.4.1 Sequence interpretation

Three of the DRG participants had trouble correctly interpreting the sequence information. The main problem was that sequence nodes point to the first verb in a sentence, but not all the subsequent verb nodes in that same sentence. The reader is supposed to read the entire sentence pointed to by the `FIRST` node before reading the sentence attached to the `2` node. For instance, in Figure 4, the second step in the upper-most sequence is not merely that the `accept` operation calls the `visit` operation, but that it calls the `visit` operation that belongs to