

THE CLAIMS LIBRARY CAPABILITY MATURITY MODEL  
EVALUATING A CLAIMS LIBRARY

Christian Fehr Allgood

Thesis submitted to the Faculty of the  
Virginia Polytechnic Institute and State University  
in partial fulfillment of the requirements for the degree of

Master of Science  
in  
Computer Science and Applications

D. Scott McCrickard, Chair  
Osman Balci  
Manuel Pérez-Quñones

Blacksburg, VA

June 11th, 2004

Keywords: notification systems, software reuse, human-computer interaction, capability maturity  
model

# THE CLAIMS LIBRARY CAPABILITY MATURITY MODEL

## EVALUATING A CLAIMS LIBRARY

Christian Fehr Allgood

### ABSTRACT

One of the problem that plagues Human-Computer Interaction (HCI) software is its development cost. Many software companies forego the usability engineering aspect of their projects due to the time required to design and test user interfaces. Unfortunately, there is no “silver bullet” for user interface design and implementation because they are inherently difficult tasks. As computers are moving off the desktop, the greatest challenge for designers will be integrating these systems seamlessly into our everyday lives. The potential for reuse in user interfaces lies in reducing the time and effort required for this task, *without sacrificing design quality*.

In this work we begin with an iterative development cycle for a claims library based on prominent literature within the HCI and software engineering fields. We constructed the Claims Library to be a repository of potentially reusable notification system claims. We examine the library through theoretical and practical perspectives. The theoretical perspective reveals tradeoffs in the initial implementation that relate to Krueger’s taxonomy of reuse. The practical perspective stems from experience in designing and conducting usability testing for an in-vehicle input device using the Claims Library. While valuable, these examinations did not provide a distinct method of improving the library. Expecting to uncover a specific diagnosis for the problems in the library, it was unclear how they should be approached with further development efforts.

With this realization, we saw that a more important and immediate contribution would not be another iteration of the Claims Library design. Rather, a clarification of the underlying theory that would better inform future systems development seemed a more urgent and worthy use of our experience. This clarification would need to have several characteristics to include: composed of a staged or prioritized architecture, represents an ideal model grounded in literature, and possesses intermediate development objectives and assessment points.

As a solution, we propose the Claims Library Capability Maturity Model (CL-CMM), based on the theoretical deficiencies that should guide development of a claims library, as noted in the two evaluations. This thesis delivers a five-stage model to include process areas, goals, and practices that address larger threads of concern. Our capability maturity model is patterned after models in software engineering and human resource management. We include a full description of each stage, a gap analysis method of appraisal, and an example of its use. Several directions for future work are noted that are necessary to continue development and validation of the model.

*“Our knowledge is the amassed thought and experience of innumerable minds.”*

*– Ralph Waldo Emerson*

## ACKNOWLEDGMENTS

The work contained here would not have been possible without the guidance and support of Scott McCrickard and Christa Chewar. Thank you for your time, effort, and dedication. Chuck Holbrook and Maxim Moldenhauer were key contributors to the usability study that was conducted. "The Lab" - Jacob, Ali, Shatab, Dillon, Brandon - for their suggestions and encouragement. I would also like to thank Bill Yancey, III for his innovative thinking and his belief in my abilities. Finally, I thank my family and friends, who probably learned more about software reuse, HCI, notification systems, and capability maturity models than they ever wanted to know.

# Contents

<b>Abstract</b>	<b>ii</b>
<b>Acknowledgments</b>	<b>iv</b>
<b>List of Tables</b>	<b>x</b>
<b>List of Figures</b>	<b>xii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problems Facing Reuse . . . . .	2
1.2 The Claims Library . . . . .	3
1.3 Evaluating a Claims Library . . . . .	4
<b>2 Related and Preliminary Work</b>	<b>6</b>
2.1 Scenario-Based Development . . . . .	6
2.1.1 Motivation for Scenario-Based Development . . . . .	7
2.1.2 Claims Analysis . . . . .	7
2.1.3 Structuring Claims . . . . .	8
2.1.4 Task-Artifact Cycle . . . . .	8
2.1.5 Toward a Design Methodology . . . . .	9
2.1.5.1 Activity Design . . . . .	10
2.1.5.2 Information Design . . . . .	11
2.1.5.3 Interaction Design . . . . .	11
2.1.6 Claim Evolution and Relationships . . . . .	12
2.2 The Domain Theory . . . . .	12
2.2.1 Foundation of Cognitive Theory . . . . .	14
2.2.2 Generic and Generalized Tasks . . . . .	15

2.2.3	Claim Schema . . . . .	16
2.3	Notification Systems . . . . .	17
2.4	Creating a Claims Library . . . . .	19
2.5	Software Reuse . . . . .	20
2.5.1	Abstraction . . . . .	21
2.5.2	Selection . . . . .	22
2.5.3	Specialization . . . . .	23
2.5.4	Integration . . . . .	24
2.5.5	Cognitive Distance . . . . .	24
2.6	Capability Maturity Models . . . . .	25
	Maturity Levels . . . . .	26
	Process Areas . . . . .	26
	Goals . . . . .	26
	Practices . . . . .	27
2.7	Summary of Related Work . . . . .	27
<b>3</b>	<b>The Claims Library: A Theoretical Perspective</b>	<b>29</b>
3.1	Abstraction in the Claims Library . . . . .	30
3.2	Selection using the Claims Library . . . . .	33
3.3	Specialization using the Claims Library . . . . .	35
3.4	Integration using the Claims Library . . . . .	36
3.5	Summary . . . . .	37
<b>4</b>	<b>The Claims Library: A Practical Perspective</b>	<b>39</b>
4.1	Analysis . . . . .	40
	Voice Interface Scenario . . . . .	41
4.2	Design Phase . . . . .	43
4.2.1	Activity Design . . . . .	43
	Steering Wheel Input Device . . . . .	45
4.2.2	Information Design . . . . .	46
	Information Scenario . . . . .	48
4.2.3	Interaction Design . . . . .	50
4.3	Testing . . . . .	53
4.4	Summary . . . . .	57

<b>5</b>	<b>Claims Library Capability Maturity Model</b>	<b>60</b>
5.1	Creating the Claims Library Capability Maturity Model . . . . .	60
5.2	Maturity Level 1: Initial . . . . .	64
5.3	Maturity Level 2: Managed . . . . .	65
5.3.1	Claims Management . . . . .	65
	Purpose . . . . .	65
	Description . . . . .	65
	Summary of Goals and Practices . . . . .	67
5.3.2	Claim Verification . . . . .	67
	Purpose . . . . .	67
	Description . . . . .	67
	Summary of Goals and Suggested Practices . . . . .	68
5.3.3	Selection Mechanism . . . . .	68
	Purpose . . . . .	68
	Description . . . . .	68
	Summary of Goals and Practices . . . . .	69
5.3.4	Claim Harvesting for Breadth . . . . .	69
	Purpose . . . . .	69
	Description . . . . .	69
	Summary of Goals and Suggested Practices . . . . .	70
5.3.5	Repository Planning . . . . .	71
	Purpose . . . . .	71
	Description . . . . .	71
	Summary of Goals and Suggested Practices . . . . .	71
5.4	Maturity Level 3: Defined . . . . .	72
5.4.1	Claim Validation . . . . .	72
	Purpose . . . . .	72
	Description . . . . .	72
	Summary of Goals and Suggested Practices . . . . .	73
5.4.2	Claim Abstraction . . . . .	74
	Purpose . . . . .	74
	Description . . . . .	74
	Summary of Goals and Suggested Practices . . . . .	75
5.4.3	Claim Harvesting for Depth . . . . .	75

	Purpose . . . . .	75
	Description . . . . .	75
	Summary of Goals and Suggested Practices . . . . .	76
5.4.4	Organizational Training . . . . .	76
	Purpose . . . . .	76
	Description . . . . .	76
	Summary of Goals and Suggested Practices . . . . .	76
5.5	Maturity Level 4: Predictable . . . . .	77
5.5.1	Testing Frameworks . . . . .	77
	Purpose . . . . .	77
	Description . . . . .	77
	Summary of Goals and Suggested Practices . . . . .	78
5.5.2	Claim Integration Environment . . . . .	78
	Purpose . . . . .	78
	Description . . . . .	78
	Summary of Goals and Suggested Practices . . . . .	79
5.5.3	Design Records . . . . .	80
	Purpose . . . . .	80
	Description . . . . .	80
	Summary of Goals and Suggested Practices . . . . .	80
5.5.4	Quantitative Repository Management . . . . .	81
	Purpose . . . . .	81
	Description . . . . .	81
	Summary of Goals and Suggested Practices . . . . .	81
5.6	Maturity Level 5: Optimized . . . . .	82
5.6.1	Establishing New HCI Theory . . . . .	82
	Purpose . . . . .	82
	Description . . . . .	82
	Summary of Goals and Suggested Practices . . . . .	82
5.6.2	Suggest Design Solutions . . . . .	83
	Purpose . . . . .	83
	Description . . . . .	83
	Summary of Goals and Suggested Practices . . . . .	83
5.6.3	Inter-organizational Collaboration . . . . .	84

	Purpose . . . . .	84
	Description . . . . .	84
	Summary of Goals and Suggested Practices . . . . .	84
5.6.4	Organizational Innovation and Deployment . . . . .	84
	Purpose . . . . .	84
	Description . . . . .	85
	Summary of Goals and Suggested Practices . . . . .	85
5.7	Using the CL-CMM . . . . .	85
<b>6</b>	<b>Conclusions</b>	<b>88</b>
6.1	Future Work . . . . .	89
<b>A</b>	<b>CL-CMM Gap Analysis Charts</b>	<b>90</b>
	<b>Bibliography</b>	<b>100</b>

# List of Tables

2.1	Claim Relationship Types . . . . .	13
2.2	Sutcliffe’s Claim Schema . . . . .	16
3.1	Modified Claim Schema . . . . .	30
3.2	Summary of Strengths . . . . .	37
3.3	Summary of Weaknesses . . . . .	38
4.1	GPS System Input Methods . . . . .	40
4.2	ThumbType Root Concept . . . . .	43
4.3	Target Resource Usage . . . . .	45
4.4	Benchmark Driving Statistics . . . . .	55
4.5	Full Evaluation Driving Statistics . . . . .	56
4.6	Summary of Strengths . . . . .	58
4.7	Summary of Weaknesses . . . . .	58
5.1	Data Integrity Thread . . . . .	61
5.2	Facilitating Reuse Thread . . . . .	62
5.3	Data Acquisition Thread . . . . .	62
5.4	Repository Maintenance Thread . . . . .	62
5.5	Process Areas by Level and Thread . . . . .	63
5.6	CL-CMM Threads . . . . .	64
5.7	Example Gap Analysis Chart . . . . .	87
A.1	Gap Analysis Chart for Process Area 2.1: Claim Management . . . . .	90
A.2	Gap Analysis Chart for Process Area 2.2: Claim Verification . . . . .	91
A.3	Gap Analysis Chart for Process Area 2.3: Search Mechanism . . . . .	91
A.4	Gap Analysis Chart for Process Area 2.4: Claim Harvesting for Breadth . . . . .	92

A.5	Gap Analysis Chart for Process Area 2.5: Repository Planning . . . . .	93
A.6	Gap Analysis Chart for Process Area 3.1: Claim Validation . . . . .	94
A.7	Gap Analysis Chart for Process Area 3.2: Claim Abstraction . . . . .	94
A.8	Gap Analysis Chart for Process Area 3.3: Claim Harvesting for Depth . . . . .	95
A.9	Gap Analysis Chart for Process Area 3.4: Organizational Training . . . . .	95
A.10	Gap Analysis Chart for Process Area 4.1: Testing Frameworks . . . . .	96
A.11	Gap Analysis Chart for Process Area 4.2: Claim Integration Environment . . . . .	96
A.12	Gap Analysis Chart for Process Area 4.3: Design Records . . . . .	97
A.13	Gap Analysis Chart for Process Area 4.4: Quantitative Repository Management . . . . .	97
A.14	Gap Analysis Chart for Process Area 5.1: Establishing New HCI Theory . . . . .	98
A.15	Gap Analysis Chart for Process Area 5.2: Suggest Design Solutions . . . . .	98
A.16	Gap Analysis Chart for Process Area 5.3: Inter-organizational Collaboration . . . . .	99
A.17	Gap Analysis Chart for Process Area 5.4: Organizational Innovation and Deployment . . . . .	99

# List of Figures

- 2.1 An example of a claim indicating the upsides and downsides of using a color scale . . . 9
- 2.2 Task-Artifact Cycle . . . . . 10
- 2.3 IRC Cube . . . . . 18
- 2.4 Levels of Abstraction . . . . . 21
- 2.5 Parts of an Abstraction . . . . . 22
- 2.6 P-CMM Maturity Levels and Threads . . . . . 27
- 2.7 Maturity Model Components . . . . . 28
  
- 3.1 “Footprint” abstraction method. . . . . 32
- 3.2 Claims Library Search Interface . . . . . 34
- 3.3 Specialization . . . . . 35
  
- 4.1 Reused problem claim describing the tradeoffs in using a voice command interpreter 41
- 4.2 Reused activity claim describing the use of a remote control for interacting with an in-vehicle information system. . . . . 44
- 4.3 Reused information claim describing the use of auditory feedback when selecting an item . . . . . 46
- 4.4 Transformed design claim, reusing information from the claim in Figure 4.3 . . . . . 47
- 4.5 Driving Force Steering Wheel . . . . . 48
- 4.6 Initial Character Mapping . . . . . 49
- 4.7 Claim illustrating the benefits of a character mapping based on character frequency 50
- 4.8 Claim illustrating the benefits of an alphabetic character mapping . . . . . 50
- 4.9 Later Character Mappings . . . . . 51
- 4.10 Reused interaction claim describing the use of a graffiti interface . . . . . 52
- 4.11 Interaction claim about attaching the input device to a steering wheel . . . . . 52

# Chapter 1

## Introduction

Life is a collection of decisions that a person has made. Every moment of every day, we are forced to make choices ranging from the mundane to the monumental. The quality of one's life could be construed as a reflection of the quality of these choices. Often, we base our decisions on past experiences, recognizing similarities between the current and previous situations. Sometimes, we write out the pros and cons for making a decision, in an attempt to choose the best path to follow. Recognizing these tradeoffs allows us to choose between the lesser of two evils or the best from among equally promising options.

What if we could benefit not only from our own personal experiences, but from those of everyone who has arrived at the same fork in the road? Moreover, what if we could consider everyone else's unique rationale; their pros and cons? What if this knowledge was grounded in scientific fact instead of mere experience? How confident would you be if the decisions you made were built upon both personal experience and proven theory? Bearing in mind the connection between our decisions and our quality of life, the potential benefits of such knowledge are indisputable.

This hypothetical illustrates the reuse of knowledge. The goal of reuse is to increase **quality** and **efficiency** by leveraging experience that has already been gained. This concept has been applied to the realm of software engineering in an effort to reduce development cost:

Reuse is the systematic application of existing software artifacts during the process of building a new software system, or the physical incorporation of existing software artifacts in a new software system. An artifact: a piece of formalized knowledge that can contribute to the software engineering process. [14]

In [14], Dusink provides a massive compilation of reuse literature, demonstrating that software reuse can occur at any stage in the development cycle, targeting a wide range of components. For example,

Component-Based Development (CBD), which is gaining momentum among software developers, focuses on the reuse of source code within systems, and deals with the problem of finding, adapting, and combining existing code [18]. Techniques such as design patterns [17], ontology-based domain engineering [13] and problem frames [22] deal with the reuse of design structures and specifications.

One of the problems that plagues Human-Computer Interaction (HCI) software is its development cost. Many software companies forego the usability engineering aspect of their projects due to the time required to design and test user interfaces. Unfortunately, there is no “silver bullet” for user interface design and implementation because they are inherently difficult tasks [30]. As computers are moving off the desktop, the greatest challenge for designers will be integrating these systems seamlessly into our everyday lives [1]. The potential for reuse in user interfaces lies in reducing the time and effort required for this task, *without sacrificing design quality*.

## 1.1 Problems Facing Reuse

The major problem facing any reuse paradigm is the acquisition of knowledge in a reusable form. In [42], Wagner describes the knowledge acquisition problem facing domains which possess a “deep knowledge model,” such as usability engineering. He identifies the three following key factors: 1) knowledge structures embedded within the domain; 2) factors surrounding development; and 3) formalisms chosen to represent the knowledge. These factors are expanded into the following key questions that must be answered by all reuse techniques:

- What defines potentially reusable knowledge? Which information is worth reusing? What are its essential parts?
- Once a target body of knowledge is identified, how does one go about abstracting it? Typically knowledge that is currently in use has been tailored to exist in a specific context. How does one extrapolate reusable information from its current form without losing the essence of the artifact that makes it useful?
- What is the structure of the abstract form of this information? In order for future designers to be able to reuse a component he/she must understand it in its abstract form. The organization of the abstracted information and the terminology used to describe it must be well-defined from the perspective of potential designers.

These questions pertain more specifically to the process of design *for* reuse – creating generalized or abstract components that can be used later. Before design *by* reuse can occur, reusable elements must have been collected and made accessible. “Reuse can only survive if libraries of reusable

components have been built up” [39]. The benefits of reuse cannot be enjoyed until a foundation of knowledge has been established.

The next major issue is that of cognitive distance [23] – how difficult is it for someone to develop a new system using reusable components? Design *by* reuse involves two basic steps: 1) *identifying* reusable parts, and 2) *incorporating* them into the new system. *Identification* requires a high-level understanding of both the system being created and the part being reused. The designer must relate these two on this higher level to be able to determine the potential applicability of the component. *Incorporation* requires a similar understanding to recombine multiple reusable components into a new system. Some reuse techniques facilitate the identification process using sophisticated search techniques, often based on metadata related to the reusable information [37, 14, 23]. Other techniques automate the integration process, generating entire systems from reusable components [14, 23]. The degree to which the reuse technique shortens the gap between the vision of a final product and its realization can be measured in terms of cognitive distance, or mental exertion. Reuse paradigms should endeavor to minimize this cognitive distance.

The question that this thesis intends to explore is how these issues can be resolved through the systematic evaluation and improvement of a reuse technique. This involves the creation of a physical system based on the literature and its validation through design practice.

## 1.2 The Claims Library

Chapter 2 supplies the background needed to understand the work contained in this document. Based on the foundations of Scenario-Based Development, the Domain Theory, and notification systems research (our domain of interest) presented in this chapter, we recognized the potential of a claims library for supporting reuse in HCI. Krueger’s survey of reuse paradigms presents a taxonomy that can be used to investigate the technique and reveal both its qualities and its shortcomings. Building from this foundation, we describe the physical system, a claims library, that is investigated throughout this work.

Our Claims Library is the result of a seminar course that studied Alistair Sutcliffe’s Domain Theory. The Domain Theory is a reuse method that enhances Scenario-Based Development (SBD), functioning as a hybrid of expertise from cognitive and behavioral science applied to software requirements engineering [33]. It is rooted primarily in Gentner’s structure-matching theory of analogy, which states that people create loose connections across real-world problem spaces and reapply those solutions when they encounter similar situations [39]. The Domain Theory establishes a common

ground for all designers by 1) providing a finite set of abstract problem spaces or domain models and 2) defining the structure of reusable design information. Its goal is to facilitate the transference (reuse) of knowledge in the form of “designer-digestible packets” [38]. During SBD, explicit design knowledge is captured in the form of claims. Claims describe the psychological effects of an artifact within a specific context of use. The Domain Theory targets claims as the preferred medium for knowledge transfer (Sutcliffe’s “packets”).

The Claims Library is our interpretation of the Domain Theory applied to the reuse of knowledge within the realm of notification systems. Notification systems are interfaces specifically designed to support access to additional digital information from sources secondary to a user’s current activity. As a task-centric process, SBD lends itself to the development of these systems. Claims produced in the development of notification systems must simultaneously consider the primary and secondary tasks of the user. A given notification system can be categorized based on three critical parameters that describe the effect it has on a user: interruption, reaction, and comprehension (IRC) [27]. Notification system claims are analyzed based on how they produce these effects, resulting in an IRC rating.

We constructed the Claims Library to be a repository of potentially reusable notification system claims, as fully described in Chapter 2. Claims are classified based on the primary and secondary tasks that they support, abstract representations of their artifacts, and their IRC rating. The Domain Theory provides the task vocabulary used for this classification in the form of its generic and generalized task models. The claim structure used in the library originated from Sutcliffe’s 14-point claim format [39], altered slightly to take into consideration aspects specific to the design of notification systems, in particular its support of dual-tasks. This classification mechanism and well-defined structure provide the common ground for designers to access and reuse notification system design information.

### 1.3 Evaluating a Claims Library

Chapter 3 provides a theoretical analysis of the Claims Library for notification systems based on Krueger’s taxonomy in order to lend breadth to its theoretical basis. The library is discussed in terms of each facet of reuse, in particular how the its design contributes to the increase or decrease of cognitive distance. These observations lead to the postulation of strengths and weaknesses inherent in the library. As a method of reuse, a claims library faces the difficult problems of knowledge acquisition and cognitive distance. We looked to Sutcliffe’s and Krueger’s theories as a structured way to evaluate a claims library with respect to its ability to mitigate these issues. *How* does a claims

library overcome the knowledge acquisition problem? *How* does a claims library reduce cognitive distance?

Theoretical analysis on its own is insufficient as a approach to validation, so Chapter 4 extends the theoretical analysis through the development of an input device for a notification system. The chapter progresses through the stages of SBD, illustrating how the library contributed to the design and testing of the input device. This experience serves as a practical analysis, assessing the verity of the observations made in the previous chapter. The logical next step that follows these evaluation efforts would be continued iterative development of the Claims Library, improving it as a tool for design. However, we instead deemed it more important to address key deficiencies found in the guiding theory, which are codified in a capability maturity model.

Capability maturity models originate from software engineering, describing how an organization should improve its software development processes over time in order to improve the quality of the software that is produced. The power of the Software Engineering Capability Maturity Model (SW-CMM) (and the related maturity models incorporated into the Capability Maturity Model Integration framework [40]) is that it simultaneously describes a way of evaluating processes while providing a recommended direction for their development. The Claims Library Capability Maturity Model (CL-CMM) is a translation of this concept to the management and development of a claims library, similar to translations performed in [12], in which a People Capability Maturity Model (P-CMM) was created in order to manage an organization's human resources; in [6], in which a maturity model for software acquisition was developed (SA-CMM); and in [21], in which a documentation maturity model was proposed.

Chapter 5 describes the creation of the CL-CMM, based on the theoretical deficiencies that should guide development of a claims library, as noted in the two evaluations. First, the essence of the maturity model, as derived from these evaluations, is discussed in terms of its overarching concerns. This discussion is followed by the elaboration of the model's components. Finally, a brief example of how this model can be applied to the appraisal and development of a claims library ensues.

Finally, Chapter 6 summarizes the work contained in this document, and provides a recommendation for future directions that should be explored. Capability maturity models for software engineering are based on decades of research and practice. The cultivation of claims libraries and the subsequent refinement of the maturity model proposed here are the necessary and logical next steps in advancing the work contained in this thesis.

## Chapter 2

# Related and Preliminary Work

This chapter introduces the key influences of our research. In order to begin to assess our Claims Library, its underpinnings must be understood. It centers around the process of Scenario-Based Development (Section 2.1), with respect to its application to the design of notification systems (Section 2.3). The Domain Theory is an enhancement of this process with the goal of achieving the reuse of design knowledge (Section 2.2). The library employs key components of this theory in its design. Krueger then provides an excellent survey of the exceedingly broad domain of reuse (Section 2.5), supplying the basis for my discussion of its theoretical tenets. Finally, the structure of our Claims Library Capability Maturity Model was derived from the capability maturity models created by the Software Engineering Institute, particularly the Software Engineering Capability Maturity Model (SW-CMM) and the People Capability Maturity Model (P-CMM) (Section 2.6).

### 2.1 Scenario-Based Development

The Scenario-Based Development (SBD) perspective of HCI software design is the basis for the governing principles of the Claims Library. Understanding this process and its underlying components is integral to comprehending the work in this thesis. After a brief description of the motivation for SBD, Section 2.1.2 discusses the process of claims analysis. Section 2.1.3 formally presents the structure of a claim. Section 2.1.4 describes Carroll's task-artifact cycle, which furnished the basis for the SBD process (explained in Section 2.1.5). Finally, the notion of claim evolution, fundamental to understanding how claims are reused, is weighed upon in Section 2.1.6.

### 2.1.1 Motivation for Scenario-Based Development

In 1989, Carroll recognized a tension that existed between practice and theory within the field of Human-Computer Interaction. While some argued that psychological theory would provide the basis for user interface design, others insisted that only practical experience, “through a subjective process of discovery,” resulted in the true understanding of a design. Carroll determined that both of these techniques were problematic. Theory-based design was seldom conducted on a “non-trivial scale” resulting in limited applicability to real world problems. “Hermeneutics” contained “no systematic methodology, no conceptual framework, no explicit way to abstract from a particular experience” [7].

Carroll discovered a middle ground between these circles of thought – a connection between “science and design.” He observed that “HCI artifacts embody psychological claims in contexts of use: aspects of the interface engender psychological consequences and in this sense make claims about the user’s behavior and experience” [7]. He developed a method of evaluating an artifact to determine the claims that it contained. His intention was to bridge the gap between practice and theory by grounding real world design experience in something repeatable and empirically verifiable. The solution was his claims analysis method, described in the section that follows.

### 2.1.2 Claims Analysis

Claims analysis of an artifact centers around user activity. This analysis is driven by the pursuit of *task coverage* and *task depth*. Carroll condenses Norman’s stages of action into three categories: “identifying and clarifying a goal, planning and acting toward its achievement, and evaluating the results of action” [7]. *Task coverage* refers to the degree to which a particular artifact possesses claims across each of these categories. Similarly, *task depth* refers to the degree to which an artifact contains multiple claims for each category. Typically, artifacts exhibit considerable task coverage and depth, embodying a significant number of claims in each category. Claims analysis involves explicitly identifying a sufficient number of claims for an artifact along those lines.

Carroll’s analysis method is an instantiation of Scriven’s mediated evaluation [35], “a compromise, or melding, of intrinsic and pay-off evaluation” [9]. An intrinsic evaluation attempts to determine the inherent characteristics of a design in terms of its *proposed* context of use. Intrinsic evaluations are “mired in goals,” both implicit in the design and intended by the designer; goals which may or may not be empirically testable. Pay-off evaluations, on the other hand, examine an artifact in use, yet lack the basis for attributing success or failure to specific aspects of a design.

Mediated evaluation uses the explicit goal formation that occurs during intrinsic evaluation as

the guiding influence for later pay-off evaluations of a design. Claims analysis employs the “notion of psychological rationale as the intrinsic evaluation and conventional usability testing as the pay-off evaluation” [9]. An artifact’s usability is determined by the coherence of “the myriad claims and their interrelations” that exist within that artifact [7]. The explicit capture of design goals in the form of claims later guides usability testing of that design.

### 2.1.3 Structuring Claims

Usability evaluations are expensive in terms of time and effort, and often the results of these evaluations do not extend beyond individual design iterations. Claims offer a medium that can be used to capture knowledge, “to save and generalize the results” of evaluation work, allowing “further develop[ment of] the theories” rooted in the designs being evaluated [7]. A well-defined claim structure is essential to facilitating knowledge transfer. Carroll provides a schema for claims that “incorporat[es] both conjectured goals in the design, and the downside risks that might” occur [9]:

*IN situation, artifact feature CAUSES desirable psychological consequence(s), BUT MAY ALSO CAUSE undesirable psychological consequence(s)*

For the purposes of this thesis, “desirable psychological consequences” will be referred to as “upsides” and will be denoted by a plus (+) sign. Similarly, “undesirable psychological consequences” will be referred to as “downsides” and will be denoted by a minus (−) sign.

Consider the example claim in Figure 2.1, noting its structure. This claim embodies the tradeoffs in using a color scale within a certain context, in this case to represent a user’s interest level. The claim has two distinct positive effects. The first is that by incorporating something familiar, connections can be made and the scale will be easily learned. The second upside highlights the use of color as a highly recognizable, but less distracting alternative to animation. The designer would also have to consider the potential downside effect of using this artifact: a color scale may not expand well to a broad range of levels.

### 2.1.4 Task-Artifact Cycle

Carroll extended the ideas in his evaluation framework into the task-artifact cycle (see Figure 2.2). The task-artifact cycle describes the iterative evolution of HCI artifacts. First, user requirements are generated from an analysis of the tasks they need to perform, and artifacts are developed to support those tasks. These artifacts then present new possibilities to the user, which in turn create the desire to accomplish new and different tasks. The task-artifact cycle grounds claims in the “scope of a

### Using Color Scale to Represent Interest

Using a color scale to represent a user's interest level in a category...

- + can leverage several existing metaphors, making the scale intuitive to the user
- + is easily noticeable (high reaction), yet not as distracting as an animation (low interruption)
- is not scalable to a broad spectrum of interest levels

Figure 2.1: An example of a claim indicating the upsides and downsides of using a color scale

basic task usage situation,” or a “scenario.” Scenarios provide a means of better understanding user activity. They elaborate the *situation* feature of Carroll's original claim schema.

Notice from Figure 2.2 that claims and scenarios exist in parallel with each other in two distinct states. These states represent the theory-practice tension identified by Carroll. One begins from narrative and causal theory of user tasks (left side of the figure). By developing scenarios and synthesizing claims, a scenario representation of an artifact is produced, along with a complimentary claims analysis (right side of the figure). The transition that occurs from left to right is the intrinsic evaluation of an artifact. The claims analysis and scenario representation of an artifact constitute living, breathing HCI knowledge found in a current design. Once observed and evaluated, these become the narrative and causal theory that are built upon for the next generation of artifacts. This second transition results from the pay-off evaluation of the artifact (depicted as arrows going right to left).

#### 2.1.5 Toward a Design Methodology

Carroll synthesized these parts (claims, scenarios, task analysis) into the SBD process. SBD begins with a task analysis of the system to be built, achieved through various field studies. The analysis should consider the problem space from the perspective of each of the stakeholders (individuals or organizations with an interest in the system's development), and should achieve good task coverage and depth. After the analysis is complete, problem scenarios are written to describe the problem space in detail. “Problem scenarios should illustrate and put into context the tasks and themes discovered in the field studies” [34]. A claims analysis of each problem scenario is performed, describing the various aspects of the current solution. This initial analysis results in a root concept, or overall vision, which provides the general high-level goals and assumptions for the system. The root concept, problem scenarios and claims analysis drives the three subsequent phases of development:

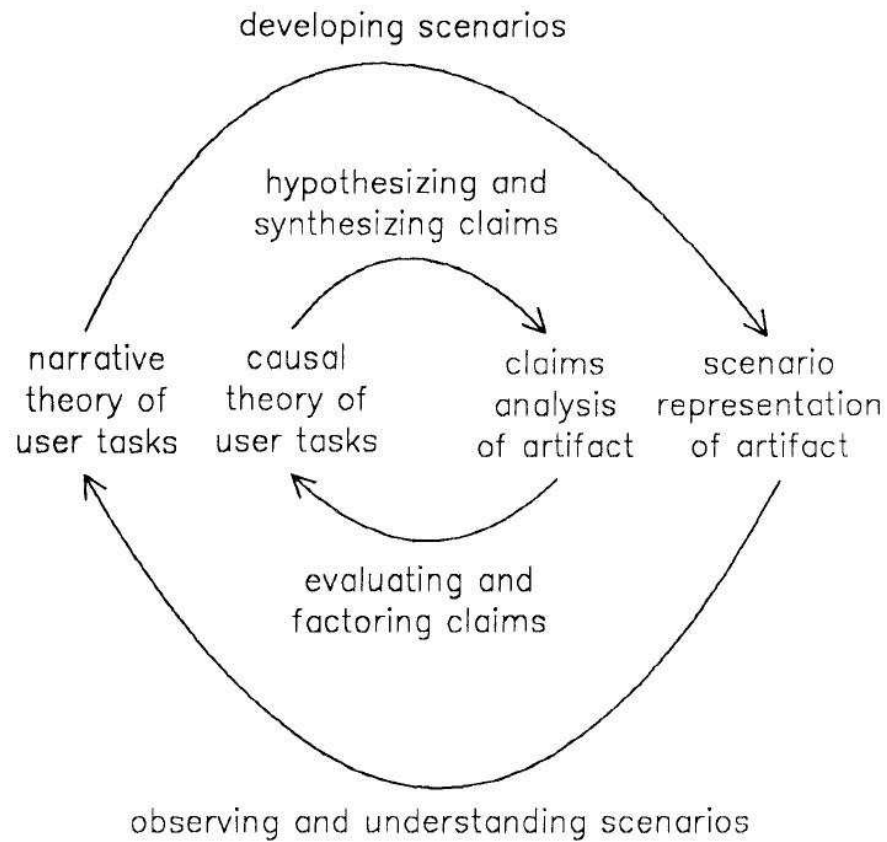


Figure 2.2: The task-artifact cycle illustrating knowledge transfer from [8]

*activity design, information design, and interaction design.*

### 2.1.5.1 Activity Design

Activity design describes the overall functionality of the system being designed. At this stage, the designer considers the problem scenarios and claims from the analysis phase. Problem scenarios describe the current practice, and their associated claims highlight both the positive and negative aspects of the current solution. Activity design should describe a new solution that preserves as much of the positive aspects of the current solution, while addressing the negative aspects using HCI theory and innovation. The result of this process should be the creation of activity scenarios which describe “typical or critical services” of the system [34], focusing purely on functionality.

As an example, consider the design of a website. The activity design phase would not be concerned with the layout or color scheme that is used, or deciding whether clicking something or simply mousing over it is the better way to perform some action. Activity design is only concerned with what the user can do. Can the user navigate to other pages contained in the site from the

current page? Can the user submit information to the system? Can the user search for information contained in the site? This design phase deals specifically with the various tasks or actions that a viewer of the site could perform.

#### 2.1.5.2 Information Design

The next two design phases are distinguished by where they fall in relation to Norman's Stages of Action [31]. Information design addresses the stages that make up the Gulf of Evaluation: perception, interpretation, and making sense. "The goal of information design is to specify representations of a task's objects and actions that will help users perceive, interpret, and make sense of what is happening" [34]. Information scenarios should elaborate activity scenarios, incorporating information design choices that allow a user to understand which state the system is in and what functionality is available to them.

Continuing the website example from above, information design would be concerned with the layout and color scheme of the site. Information design is concerned with how aspects of the system are presented to the user. What color are hyperlinks and should they be underlined? Should graphic buttons be used instead? Should the navigation bar be located at the top of the page horizontally or on either side of the page vertically? Information design describes how everything that a user would see on the webpage should look.

#### 2.1.5.3 Interaction Design

Interaction design addresses Norman's Gulf of Execution: forming system goals, planning action, and execution. "The goal of interaction design is to specify the mechanisms for accessing and manipulating task information" [34]. Interaction design describes *how* a user accomplishes a task using the system. It must take into account aspects of the previous design phases. Interaction scenarios provide a complete picture of the system's interface, extending the tasks described in activity scenarios to explicitly describe how a user would perform these actions, while taking into consideration how parts of the system are represented for the user (from the information design).

Completing the website example, interaction design would be concerned with the difference between clicking and mousing over. Interaction design describes *how* a user accomplishes a goal using the website. How does a user navigate fields within a form on the webpage? How does a user access some piece of information contained in a database front-ended by the site? How does a user navigate from page to page? Interaction design describes how users of the website would accomplish the goals defined in the activity design, keeping in mind the site's visual objects and lay out detailed in the information design.

### 2.1.6 Claim Evolution and Relationships

At each design phase of SBD, an iterative process of claims analysis occurs. Claims are created in parallel with scenarios. They highlight and document the key design decisions within each scenario that should be examined during future usability studies (intrinsic evaluation). Prototypes are then created, and claims are empirically tested (pay-off evaluation). The information gathered from these studies either provides support for a claim or indicates the need for further scenario and claim refinement. This, in essence, is the task-artifact cycle. Claims evolve over time as the design of a single system progresses or as claims are reused from system to system. Capturing the knowledge embodied in the evolution of claims is a valuable design resource, containing within it the record of a system's improvement over time. While it is unclear how this should be done, it is the purpose of this thesis to explore the possibilities and prescribe a solution.

Wahid recognized the diversity of claim evolution, and identified six claim relationship types that can occur [43]. Table 2.1 briefly describes this extensible set relationships. The first two relationship types (Predicating/Postulating, Executing/Evaluating) are geared toward understanding relationships across SBD design phases. The remaining relationships allow insight into how a claim evolves within the development of a single system over design iterations or how a claim was reused in the creation of a new system. As stated previously (Section 2.1.2), the usability of a design is determined by the claims about that design and the relationships among those claims. The relationship framework described here “permits one to understand the process that was used to derive a new claim or reuse a claim in another domain, . . . provok[ing] reflection and creative thought processes” [43] which may aid in the transfer of design knowledge.

Sutcliffe and Carroll also describe a factoring process by which abstract claims may be generated from original design claims [7, 37]. A walk-through method based on Norman's Stages of Action, this process is intended to aid in the reuse of claims by removing the specific context in which the original claim existed, so that it may then be applied to a new one. Their factoring process is equivalent to the creation of generalizing claims from specifying claims, described in the relationships above. Sutcliffe more fully develops how reuse might occur using claims to encapsulate design knowledge in his Domain Theory, described in the next section.

## 2.2 The Domain Theory

Sutcliffe devised a reuse technique founded in cognitive science and applied to the realm of HCI software development called the *Domain Theory*. His motivation was “to capture and formalize reusable knowledge at the requirements specification level, essentially translating the memory that

<b>Relationship</b>	<b>Description</b>
Predicating / Postulating	The predicating-postulating relationship occurs between claims within the problem space and claims that describe the solution space. Predicating claims present aspects of a problem domain based on a task analysis. Postulating claims propose solutions to the problems identified by these claims. Typically, this is the relationship that exists between claims derived from a problem scenario and claims that exist within activity scenarios.
Evaluating / Executing	The evaluating-executing relationship occurs between claims produced during the information design of a system and corresponding interaction design claims. While these two aspects of design addresses a separate part of user activity, they are often closely related, and it is important to acknowledge when this occurs.
Generalizing / Specifying	Generalizing claims are the result of the broadening of the scope or context of a claim. Similarly, specifying claims result from the specialization of more general claims. This is a very important relationship in terms of claim reuse which will be discussed later.
Fusing / Diffusing	Fusing claims are the result of the combination of two or more claims into a superclaim. Similarly, diffusing claims are the result of the breakdown of a superclaim into smaller, atomic claims. Typically this relationship occurs over the course of design iterations when claims are refined with usability testing.
Translating	Translating claims are created when a claim is transformed so that it applies in a new context. This is the equivalent of generalizing a claim and then specializing it for a new situation. The translating relationship is often the result of claim reuse.
Mitigating	Mitigating claims occur when design claims are improved, usually as a result of usability testing. Problems made evident through evaluation are alleviated and a new mitigating claim is created.

Table 2.1: Descriptions of the claim relationship types summarized from [43].

expert consultants retrieve when they recognize” that they have arrived at a problem similar to one that they have encountered before [39]. He proposed that this information be recorded in the form of claims, and that claims should be stored in a designer-accessible repository - a *claims library*.

While the spirit of the Domain Theory is reinforced by our Claims Library, specifically in its reliance upon structure-matching as a means of information retrieval, only certain components of Sutcliffe’s work are utilized in its design. For brevity, only those components are discussed here. Section 2.2.1 contributes a short description of the underlying cognitive theory behind the Domain Theory. Section 2.2.2 describes the task models that were borrowed, supplying the language used in the Claims Library’s retrieval mechanism. Section 2.2.3 describes the claim schema developed by Sutcliffe and Carroll [37, 39] for use in the Domain Theory.

### 2.2.1 Foundation of Cognitive Theory

The Domain Theory can be viewed from two perspectives. “First it is a *theory of abstraction* in that it defines a schema for generic domain knowledge, and second it is a *theory of naturally occurring expertise*” [39]. Its existence as a *theory of abstraction* is based primarily on Gentner’s structure-matching theory of analogy, which describes how people form abstract connections between information in different domains. Gentner’s theory “proposes that abstract schema are learned when people recognize and solve problems . . . built from the concrete source domain [with] mappings from the source to the analogue schema and to a new target domain” [39].

An example of this is how people make connections between the flow of water and the flow of electricity. Despite being very different in composition, connections can be made through the abstract concepts of particles (water molecules and electrons), connectors (insulated wire versus pipes), and flow regulation. Sutcliffe cites the importance of this phenomena: “Establishing common abstractions that maximize shared knowledge and minimize erroneous interdomain transfer is a key concern of the Domain Theory” [39].

As a *theory of naturally occurring expertise*, the Domain Theory stems from Schank’s theory of ecological memory. Schank’s work suggests that “[m]emory is developed by generalization from many specific instances of similar episodes, so we apply a generic script to situations by recognizing their entry conditions” [39]. Sutcliffe attempts to capture the knowledge contained by experts in a set of generic problem domain models. His models consist of objects and the operations that can be performed on them. Part of this framework are generic and generalized tasks, described in the next section.

### 2.2.2 Generic and Generalized Tasks

As part of the Domain Theory, Sutcliffe defines a generic task structure. Though similar to generic task models that have already been described in literature [3, 5, 45, 46], the Domain Theory task structure focuses on “human mental activity,” including “the cognitive precursors of action” [39]. In other words, this task structure places emphasis on the functional requirements associated with user interaction. One example is the generic task of monitoring (adapted from [39]):

#### *Monitor*

*Monitoring observes the environment for state changes of interest. Monitoring task variants depend on the nature of the event to be detected, which may be discrete (a single change such as a gunshot) or continuous (gradual temperature change). Events may be located in space and time, involve the movement of objects, or record changes to the attributes of objects and properties of the environment.*

Within the Domain Theory, generic tasks describe the basic set of actions that can be performed on objects. They are “self-contained procedures that run from initiation to completion without halting, to achieve a goal” [39]. They are complete units of user activity.

Generic tasks have an atomic relationship with another Domain Theory task model: generalized tasks. Generalized tasks are compositions of generic tasks, providing a molecular view of user activity. The following is an example of a generalized task (adapted from [39]):

#### *Planning-Scheduling*

*This is the act of organizing some resource or event so that it should happen at a certain time or occur in a specific place. The essence of planning is ordering the future. Planning is interpreted in the cognitive sense of creating a memory structure of intentions, i.e. a task goal structure that describes a high-level approach to solving a problem or carrying out an activity. Human planning has variations such as spatial planning layouts of objects, temporal planning of schedules, and planning that involves allocation. Scheduling involves planning with sequential or temporal constraints. Temporal planning involves the additional complexity of sequencing and synchronizing activity. The major problem in Planning is uncertainty. In deterministic domains in which the behavior and properties of objects are known, plans can be formed in detail. In non-deterministic domains no planning may be possible because of the high level of entropy. In between are most domains in which change is a fact of life but some regularity exists. Plans, therefore, usually have to start with a default or idealized version and then expect to be revised in light of experience. Planning can be fully automated if detailed knowledge can be captured; however, in domains with heuristic and informal knowledge, humans carry out the planning with computer support in the form of lists of goals, operations, constraints and plan templates, with sorting and ranking functions. As Planning has to respond to change, this generalized task is associated with*

<b>1. Claim ID:</b>	unique identifier for the claim
<b>2. Title:</b>	given by the claim's author
<b>3. Author:</b>	researcher(s) or practitioner(s) who developed the original claim
<b>4. Artifact:</b>	brief description of the product or application in which the claim originated
<b>5. Description:</b>	natural-language description of the claim
<b>6. Upside(s):</b>	positive effect of using the claim on a system goal or usability
<b>7. Downside(s):</b>	negative effect of using the claim on a system goal or usability
<b>8. Scenario:</b>	scenario in which the claim was derived or is now used
<b>9. Effect:</b>	desired, and preferably measurable, system goal-usability effect that the implemented claim should achieve
<b>10. Dependencies:</b>	other design problems that have to be solved to achieve the claim's effect
<b>11. Issues:</b>	design issues influenced by the claim
<b>12. Theory:</b>	underlying theory explicitly referenced by the author
<b>13. Relationships:</b>	interclaim links that describe how claims evolved during a history of investigation
<b>14. Scope:</b>	scopes the reuse context by linking the claim to a Domain Theory model or component thereof

Table 2.2: Sutcliffe's 14-point claim schema from [39]

*Analysis-Modeling (another generalized task) and the Monitor, Interpret, and Evaluate generic tasks.*

Generalized tasks are different from generic tasks in that they describe a much broader range of activity. Generalized tasks may go through several intermediate stages of action, as its independent generic task components are completed. This distinction in granularity lends itself to the description of dual-task user activity found in notification systems, described in Section 2.3.

### 2.2.3 Claim Schema

The Domain Theory encapsulates reusable knowledge about a system's design in the form of claims, borrowed directly from Carroll's task-artifact framework and SBD. Table 2.2 depicts the 14-point claim schema that Sutcliffe defines to support knowledge transfer. This new schema extends Carroll's original structure, explicitly including links to theoretical references, design dependencies and issues, interclaim relationships, and metadata associated with storing the claim for reuse, such as a claim ID, title, the claim's author, and scope. The scope field relates the claim directly to the Domain Theory. Its purpose is to convey the context of reuse for the claim, allowing the claim to be "classified and organized in a library" [39]. The Domain Theory is intended to provide the "generalized context of use to situate design knowledge encapsulated in claims" [39].

One interesting aspect of this claim schema is the inclusion of a scenario field. Sutcliffe appears to abate the importance of scenarios by making them components of a claim as opposed to equal

entities. Carroll's task-artifact cycle appears to place comparable emphasis on each, and perhaps suggests the greater importance of scenarios with his Scenario-Based Development framework. One might expect Sutcliffe to perhaps incorporate scenarios as separate parts of his Domain Theory, as opposed to giving them a subordinate role.

The problem that arises with the reuse of claims, however, is that they are grounded in a specific context (the scenario), which hinders reuse in another situation. Once claims have been recorded in this format, Sutcliffe recommends generalizing claims using a process similar to the walk-through method described in [37]. The original scenario of use helps in generalizing the claim, yet is intended to have little involvement when the claim is reused, perhaps only providing insight into previous specific instantiations. This dichotomy of the tight relationship between claims and scenarios versus the attempt to reuse claims in new situations is one that will be reiterated throughout this thesis.

The Claims Library investigated here does not employ a method of generalizing claims. Rather, it uses the generic and generalized task models of the Domain Theory as part of an abstract representation of a claim. This structure fits nicely with the dual-task nature of the type of system for which the library was designed. These *notification systems* are described in the next section.

## 2.3 Notification Systems

What is a notification system? *Notification systems* are interfaces specifically designed to support user access to additional digital information from sources secondary to a user's current activity. Status programs, stock tickers, and e-mail alert programs are commonly found examples of notification systems, enabling access to a variety of information instantaneously. The secondary nature of the demanded information is a key characteristic of notification systems: "some primary task typically dominates the attention of the user, with attention diverted as minimally as possible [by the notification system] to allow the user to maintain awareness at a desired level" [27].

Notification systems are not relegated to the desktop. They are major components of mobile computing devices (PDAs, cell phones), in-vehicle information systems (IVISs), and real world interfaces. While desktop notification systems typically manifest themselves in the form of secondary visual displays [27], these new domains pose interesting challenges in presenting information. Traditional, graphical displays such as computer monitors are not readily available, so other mediums have been employed to convey information.

McCrickard [27] defines three critical characteristics of notification systems: interruption, reaction, and comprehension. The degree to which each of these characteristics are present merits in the assignment of a numeric value between 0 and 1. The collective rating for all three characteristics

is referred to as an IRC rating. IRC ratings are valuable in understanding and expressing a user's secondary task goals, as "notification systems research should focus on exploring the balance between interruption, reaction, and comprehension design objective parameters" [27]. A notification system that unnecessarily interrupts a user performing a primary task hinders that user, rather than providing support. A notification system that inadequately notifies the user of the existence of new information negates the potential of the supporting system. A notification system that provides insufficient levels of information defeats the purpose of that system. These characteristics are described in more detail below. Figure 2.3 illustrates this three-dimensional space describing eight general types of notification systems based on their varying levels of interruption, reaction, and comprehension.

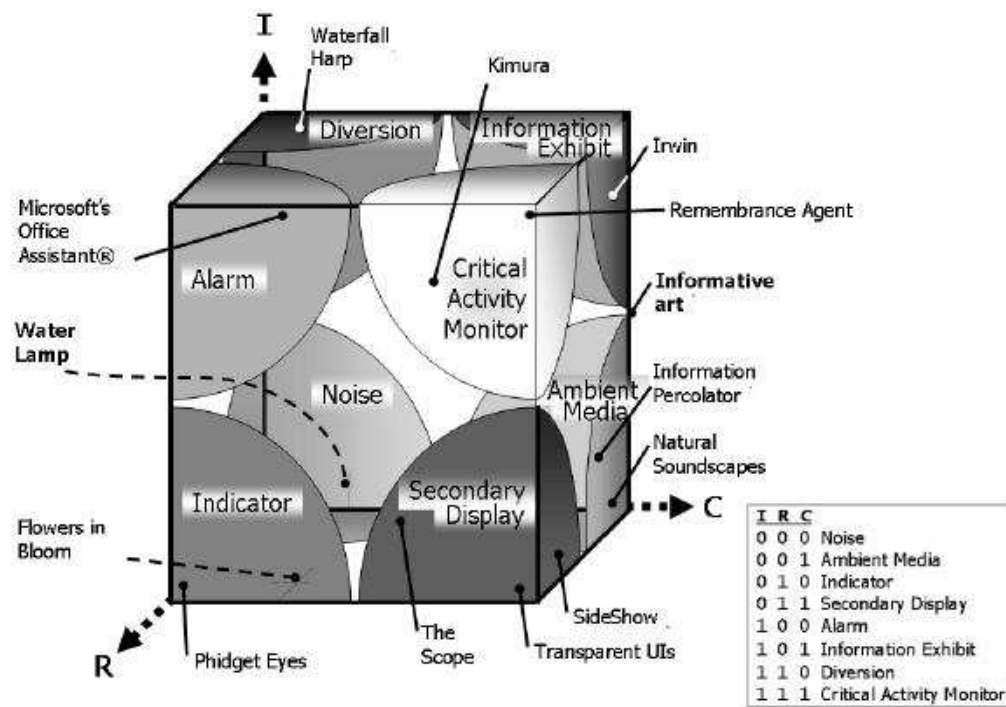


Figure 2.3: IRC cube [28] illustrating eight types of notification systems as they pertain to regions of varying levels of interruption, reaction, and comprehension.

*Interruption* refers to the way in which a notification system elicits the full concentration of the user, shifting his/her focus from the primary task to the secondary one. Depending upon the scenario of use, this transition of focus may or may not be desired. In some situations, the interruption of the primary task is preferred due, perhaps, to the importance of the information being delivered by the

notification systems. In other situations, performance of the primary task is of paramount concern, and an interruption would be undesirable.

Within certain contexts, undesired interruption which hinders the performance of a primary task could have grave consequences. Interaction with an in-vehicle notification system could potentially affect a driver's ability to navigate his/her vehicle. In this instance, human safety is at stake. A poorly designed notification system could result in serious injury or death. Regardless, distraction caused by high levels of interruption are of detriment to the primary task, but this distraction may be desired by the user if the notification is of higher importance.

*Reaction* refers to the "rapid and accurate response to important information provided by notification systems" [27]. While closely related to interruption, the goal of reaction is to sufficiently convey the existence of notifications, without unnecessarily interrupting the user. In this case, timely and exact responses to a notification are more important than the complete allocation of the user's attention. Colors, shapes, sounds, and motion are all used to convey the current state of the system, and provide the ability to highlight changes in this system [27].

When using a notification system, it is important in many situations for a user to remember the information that is conveyed over time. This is referred to as *comprehension*. The degree to which a notification system facilitates the understanding of information is reflected in both what a user can retain from current notifications, but also in how the user begins to anticipate future notifications. Systems with high levels of comprehension allow the user to recognize patterns in the information being presented, which may lead to the perception of underlying trends.

Scenario-Based Development lends itself to the design of notification systems because of its definitive attention to user tasks. Claims analysis of usage scenarios is a clear, concise way of evaluating interruption, reaction, and comprehension without forgetting the dual-task nature of the system. While the development of non-traditional notification systems requires innovative cogitation, there is potential in reusing aspects of traditional notification system design. Perhaps the reuse of claims will lead to the rapid advancement of next-generation notification systems.

## 2.4 Creating a Claims Library

Sutcliffe's Domain Theory calls for the creation of a claims library: "To enable effective reuse, claims have to be classified and organized in a library so that designers can locate the appropriate claims for their current application problem" [39]. We constructed a claims library for notification systems based on the Domain Theory components described above, as well as our knowledge of SBD and, more importantly, our expertise with this class of system. This section discusses the development

of the entire project, while the theoretical aspects of the library are left for discussion in Chapter 3.

Creation of the library consisted of a backend implementation, data collection process, and user interface design. Our Claims Library was built with open source software, using a MySQL database (<http://www.mysql.com>), Java 2 Platform, Enterprise Edition (J2EE) architecture (<http://java.sun.com>), and Hibernate (<http://www.hibernate.org>) for the object/relational mapping and data persistence. The data collection process spanned more than fifty systems, including those designed within the project group, through research collaboration with other designers, and identified in literature. We intentionally selected systems that included a variety of implementation platforms and supported a range of user tasks. Our user interface design was bolstered by an internal heuristic evaluation.

Initial testing was performed to assess the potential for our library. Nine participants attempted to apply knowledge contained in the library to a series of design tasks. While these tasks tested many aspects of the entire library, we focused on how useful the participants felt the claims were in their design process and how intuitive and meaningful the retrieval process was. We presented our system to Sutcliffe in the Spring of 2003, and received positive feedback. Finally, our work [33] was published later that year in the 2003 IEEE International Conference on Information Reuse and Integration (<http://parks.slu.edu/IRI2003/>), describing our design experience and future work for the library. This work is continued here, using our Claims Library as the focus of the theoretical and practical evaluations in the following chapters.

## 2.5 Software Reuse

Software reuse is a broad topic that has been thoroughly explored within computer science, yet still feels as if it is a new frontier. As exhibited in Dusink's compilation of research on the subject, software reuse can take many forms and elicits a number of interpretations. Krueger's survey paper is an attempt at identifying the essential pieces and using them to evaluate a broad range of reuse techniques [23]. He defines four axes by which reuse techniques should be analyzed: abstraction, selection, specialization, and integration. The collective implementation of these features results in an overall level of mental effort required of the user of the technique, referred to as cognitive distance. Krueger examines eight approaches in an effort to assess the level of cognitive distance that they impose.

Section 2.5.1 describes abstraction, perhaps the most important part of any reuse technique. Abstraction plays a role in selection (Section 2.5.2), specialization (Section 2.5.3), and integration (Section 2.5.4). Section 2.5.5 explores the idea of cognitive distance and how it is used to evaluate a reuse technique. Understanding our Claims Library in terms of the cognitive distance it produces

allows us to identify and mitigate limitations in its design.

### 2.5.1 Abstraction

“Abstraction plays a central role in software reuse. Concise and expressive abstractions are essential if artifacts are to be effectively reused” [23]. *Abstraction* is perhaps the most important feature of any reuse paradigm. It is the means by which knowledge is transferred from previous experience. “An abstraction for a software artifact is a succinct description that suppresses the details that are unimportant to a software developer and emphasizes the information that is important” [23]. In other words, an abstraction captures the essence of an artifact.

An artifact can exist in one of two forms - its specification and its realization. The specification of an artifact is its higher-level, abstracted form. Similarly, the realization of an artifact is its concrete existence, often referred to as its implementation. (Since the term “implementation” often implies the source code of a program, and reuse extends well beyond this type of artifact, Krueger uses “realization” instead.) The process of abstraction is simply the transformation of an artifact from realization to specification. Additionally, abstraction can occur in layers, where the specification of an artifact at one level becomes the realization of the next, higher level (see Figure 2.4).

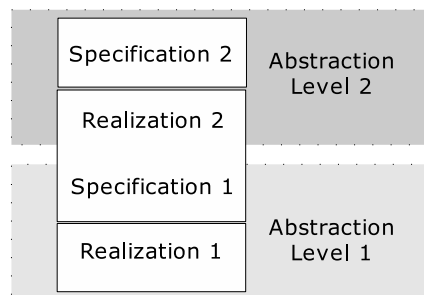


Figure 2.4: Levels of abstraction (adapted from [23]). A specification supplies the realization for the next higher level of abstraction.

Continuing Krueger’s description of abstraction, he identifies three parts of an abstraction. An abstraction specification contains a “variable” part and a “fixed” part. The fixed part is best described as the inherent, unchangeable characteristics of the artifact. The variable part is still an essential aspect of the artifact, but changes from realization to realization (see Figure 2.5). Each abstraction realization also contains a “hidden” part that consists of the details inconsequential to the user of the abstraction. Krueger uses the example of a stack abstraction. The last-in-first-out (LIFO) semantics of a stack is an “invariant” trait, so that would be considered the fixed part of the abstraction. The type of element stored in the stack can change from case to case, making it the variable part. Finally, the depth of the stack might be inconsequential to the concept of a stack, so

it makes up the hidden part [23]. (In another case, stack depth may be an important characteristic, and could be deemed a fixed or variable part; it is all up to the designer of the abstraction.)

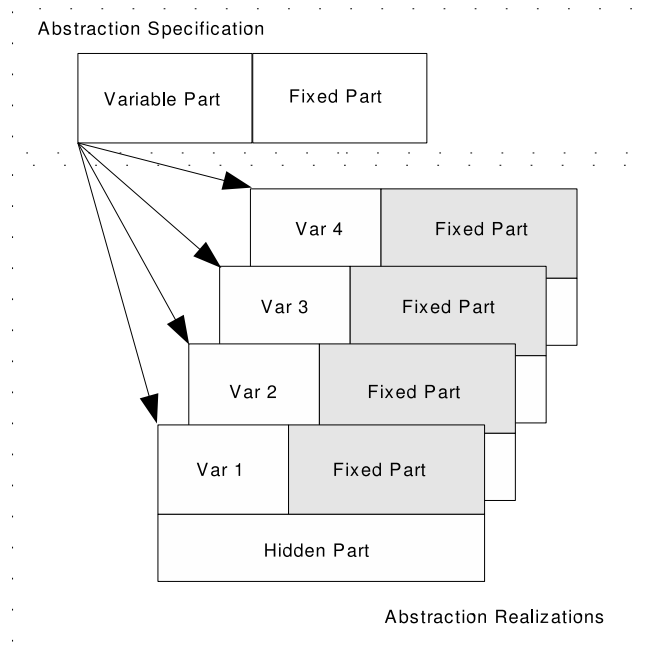


Figure 2.5: Parts of an abstraction (adapted from [23]). An abstraction specification contains fixed and variable parts, while an abstraction realization has an extra hidden part. The variable part of an abstraction is different from realization to realization.

Reuse hinges on the abstraction of information. Knowledge is captured in the form of an abstraction specification. Reuse occurs when someone transforms an abstraction specification to an abstraction realization by defining its variable parts. The potential for reuse of an artifact depends on its level of abstraction. While a higher level of abstraction results in a broader range of applicability, it lacks in detail which may reduce its utility. Lower levels of abstraction may benefit from greater utility, but will be limited in its scope of reuse. Sutcliffe explains this using the analogy of a “reuse spotlight” [39]. Picturing a flashlight pointing at the ground, as it moves upwards (higher level of abstraction), the light covers a wider area, but is not as intense. Likewise, when it moves downwards (lower level of abstraction), the light is more intense, but covers a smaller area.

## 2.5.2 Selection

The *selection* of reusable information is the first step of the design *by* reuse process. Developers must “locate, understand, compare, and select the appropriate artifacts from a collection” [23]. There are two qualities to the process of selection that should be achieved: *facility* and *accuracy*. *Facility* refers to the ease with which developers are able to retrieve information. Often, reuse techniques employ

some sort of classification scheme and search mechanism that guides the location of components, instead of forcing users to sift through massive amounts of data. A search mechanism could be as simple as a keyword field or as complex as a diverse set of fields for which the developer provides values [37].

*Accuracy* refers to how relevant located information is to the design problem. Typically, the classification scheme is based on some abstract idea of the problem space. Developers supply criteria that describes the current problem, and solutions to similar problems are returned. For example, a developer requires a sorting algorithm that can order a uniquely structured dataset in a minimum amount of time. Some library search mechanisms might only let the developer search for sorting algorithms in general. Others might allow the specification of detailed criteria like the speed or the makeup of the dataset. Selection accuracy would be how close the search results were to the desired solution.

Abstraction plays an interesting role in selection. When supplying search criteria, the developer must create an abstract view of the system being created in order to relate it to the information in a reuse library. This often involves decomposing the problem (e.g. a sorting algorithm would only be a small part of an entire system) and searching for solutions to the various parts, requiring a significant amount of mental effort on the part of the user. However, a well-designed reuse technique can alleviate the cognitive load by employing “concise abstractions” [23] that may be used as a common ground from which developers may begin the selection process.

### 2.5.3 Specialization

*Specialization* is the transformation of reusable knowledge so that it applies in a new domain. The difficulty in specialization is inversely proportional to the information’s level of abstraction. The higher the level of abstraction, the easier it is to specialize to a more narrow domain [23] (keeping in mind the Sutcliffe’s reuse spotlight tradeoff). If the information has a low level of abstraction, specializing it for another domain involves abstracting that information to a higher level and then narrowing the scope again, this time to the new domain. Needless to say, the cognitive load of having to work from specific instances of information rather than abstracted versions is significantly higher.

Ultimately, specialization boils down to refining the variable portion of an abstraction, described above. Again, the importance of well-defined abstractions comes into play. Having a distinct delineation between the fixed and variable parts of an abstraction facilitates specialization, and reduces the mental effort for the developer.

### 2.5.4 Integration

Once a conglomerate of reusable information is selected and specialized, it must be combined in such a way that the various pieces will work coherently. This is the process of *integration*. The developer must think about the pieces abstractly in order to determine how they will all fit together. Krueger makes a general distinction between an abstraction specification and an abstraction realization. The specification refers to *what* the artifact does, while the realization of the artifact describes *how* the artifact performs its function [23]. During integration, *what* each of the reusable components do must be considered in order to put them all together. Clearly defining the specification and realization parts of an abstraction will aid in integration. Again, abstraction plays a key role in how a subsequent aspect of reuse operates.

“Reuse technologies typically have an integration framework” that developers use “to combine a collection of selected and specialized artifacts into a complete software system” [23]. A high-level language like Java, for example, could be considered a reuse technique in that it provides a collection of reusable source code in the form of classes and packages. The program itself becomes the integration framework, allowing the inclusion of various packages and requiring strict syntax for incorporating the various classes and methods into a new program.

### 2.5.5 Cognitive Distance

Reuse techniques can ultimately be evaluated in terms of cognitive distance – “an intuitive gauge of the intellectual effort required to use the technique” [23]. In other words, cognitive distance is the amount of mental effort required for a developer to go from the vision of a system to its implementation using a reuse technique. Reduction of cognitive distance can be accomplished in two ways [23]:

1. Reduce the amount of intellectual effort required to go from the initial conceptualization of a system to a specification of the system in abstractions of the reuse technique.
2. Reduce the amount of intellectual effort required to produce an executable system once the software developer has produced a specification in terms of abstractions of the reuse technique.

The first method involves designing abstractions so they coincide with how people already reason about the information. For example, Java works with commonly-used object-oriented concepts like classes and methods. Developers familiar with object-oriented programming can easily envision their system in these terms. The second method requires some sort of automation of the specialization

process whereby an abstraction specification is transformed into an abstraction realization. These general methods suggest potential paths toward improvement that can be made in the development of a claims library.

## 2.6 Capability Maturity Models

In this work we put forth a Claims Library Capability Maturity Model (CL-CMM), derived from the capability maturity models created by the Software Engineering Institute (SEI). Although discussion of the CL-CMM occurs later, this section elaborates the notion of capability maturity models and their various applications. This concept was originally developed in an effort to assess and improve upon an organization's software development processes.

*A capability maturity model delineates the characteristics of a mature, capable process. It identifies the practices that are basic to implementing effective processes as well as advanced practices. It also assigns to those practices associated maturity levels ranging from unrepeatable to a mature, well-managed process. Typically a path is recommended through the various practices for achieving higher levels of maturity and, therefore, improve an organization's processes [12].*

Since 1991, when the SEI released the Capability Maturity Model for Software Engineering (SW-CMM), the concept has both evolved and been applied to many different areas. The original SW-CMM splintered into similar maturity models for systems engineering (SE-CMM) and integrated product and process development (IPPD-CMM). Recognizing a number of similarities among these models and the co-existence of these processes within a single organization, the SEI developed the Capability Maturity Model Integration (CMMI) framework. This framework supports the various representations that evolved from the original model, and provides a means of organizing them within one maturity model.

In 1995, the notion of a capability maturity model was extended into the realm of human resources with the creation of the People Capability Maturity Model (P-CMM). The P-CMM deals with “the key elements of managing and developing the work force of an organization” [12]. The P-CMM borrowed the classic staged model from the SW-CMM, adapting it to apply to the processes involved with people management. Similar translations can be found in literature, like the maturity model for software acquisition (SA-CMM) [6] and the documentation maturity model [21].

Along the same lines as these translations, our CL-CMM applies the framework of capability maturity models to the management and development of a claims library. The CL-CMM employs

a modified version of the *staged representation* used in the SW-CMM and P-CMM. A staged representation focuses on measuring improvement using *maturity levels*. Maturity levels are composed of *process areas*. To achieve a maturity level, an organization must meet the specific *goals* of each process area on that level. The process areas of future maturity levels build upon the those of previous ones. Thus, earlier levels establish a foundation for later stages and collectively describe a clear and precise path of development. These components are discussed in more detail below.

**Maturity Levels** “A maturity level is a defined evolutionary plateau” [40]. Each maturity level contains process areas geared toward improving in a particular category, or focus. The achievement of a maturity level serves two purposes. The first is that it acts as an evaluation or assessment. In terms of the SW-CMM, attainment of a maturity level speaks to an organization’s process capability, providing “a way to predict the future performance of an organization within a given discipline or set of disciplines” [40]. With respect to the CL-CMM, maturity levels describe the maturity of the repository in terms of its content, facilitation of reuse, and underlying maintenance processes. The second purpose is that it provides a foundation for future maturity levels. In addition to acting as a comparative rating system, maturity levels are also milestones in achieving a larger development goal.

**Process Areas** “A process area is a cluster of related practices . . . that, when performed collectively, satisfy a set of goals considered important for making significant improvement in that area” [40, 12]. They describe the processes both integral to growth within a maturity level and important to overall improvement. For example, process areas found in the second maturity level of the P-CMM include: staffing, work environment, communication and coordination, training and development, and compensation. These areas support the overall focus of the maturity level: “[m]anagers taking responsibility for managing and developing their people” [12]. Additionally, the P-CMM groups process areas *across* maturity levels by “common areas of concern” referred to as “threads” (see Figure 2.6). Threads are addressed throughout the model, and represent larger goals as opposed to the “organizational transformations associated with the maturity levels” [12].

**Goals** Goals are unique to process areas and describe what must be done to satisfy the purpose of the process area. “The goals of a process area summarize the states that must exist for that process area to have been implemented and institutionalized” [40, 12]. The degree to which these goals have been achieved provide the basis for assessing capability in a process area. The terms “implemented and institutionalized” imply that these goals are effectively and consistently achieved.

Maturity levels	Process Area Threads			
	Developing individual capability	Building workgroups & culture	Motivating & managing performance	Shaping the workforce
<b>5</b> Optimizing	Continuous Capability Improvement		Organizational Performance Alignment	Continuous Workforce Innovation
<b>4</b> Predictable	Competency Based Assets Mentoring	Competency Integration Empowered Workgroups	Quantitative Performance Management	Organizational Capability Management
<b>3</b> Defined	Competency Development Competency Analysis	Workgroup Development Participatory Culture	Competency Based Practices Career Development	Workforce Planning
<b>2</b> Managed	Training and Development	Communication & Coordination	Compensation Performance Management Work Environment	Staffing

Figure 2.6: “Matrix” of maturity levels and process area threads in the P-CMM [12].

**Practices** A practice is “a subprocess within a process area that contributes to achieving a goal” [40, 12]. Each goal has associated with it a set of suggested practices. Ideally, the effectiveness of these practices has been proven over time, but a list of practices is by no means exhaustive or all-encompassing. The evaluation of an organization’s practices ultimately determines maturity.

The relationship among these components is illustrated in Figure 2.7. Maturity models are made up of process areas which act coherently toward a particular stage of improvement. Process areas contain certain goals which are addressed by corresponding practices.

## 2.7 Summary of Related Work

The work contained in this thesis spans a vast range of research throughout computer science, drawing together areas of knowledge and software reuse, software engineering, and HCI. This section was not meant to be a thorough exploration of each of these topics, but rather an examination of how they apply to the Claims Library and the evaluation thereof. SBD and the Domain Theory laid the groundwork for the development of the library with Carroll’s task-artifact cycle and Sutcliffe’s task models and explicit claim schema. The Claims Library was built upon knowledge of notification

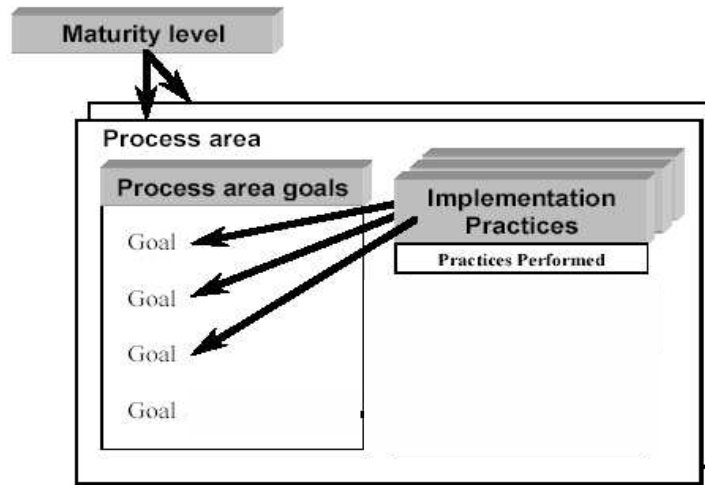


Figure 2.7: Relationships among capability maturity model components; maturity models are made up of process areas which contain goals; goals are satisfied by the implementation of various supporting practices (adapted from [12]).

systems, a specific type of user interface that supports access to information in a dual-task situation, and it was through the development of such a system that the library was evaluated from a practical perspective. The theoretical analysis of the repository conducted in this document is based on the work of Krueger. The ultimate contribution of this thesis, a Claims Library Capability Maturity Model, was derived from capability maturity models originating in the discipline of software engineering.

## Chapter 3

# The Claims Library: A Theoretical Perspective

Krueger's survey paper on software reuse provides the basis for the theoretical evaluation of our Claims Library. This chapter serves as a detailed critique of the underlying principles of the library, in terms of the four axes of reuse identified by Krueger: abstraction, selection, specialization, and integration. He provides a set of fundamental questions to explore his taxonomy [23]:

**Abstraction** What types of artifacts are reused and what abstractions are used to describe the artifacts?

**Selection** How are reusable artifacts selected for reuse?

**Specialization** How are generalized artifacts specialized for reuse?

**Integration** How are reusable artifacts integrated to create a complete software system?

From this exploration will come an understanding of our Claims Library as a reuse technique. Every reuse technique imposes some degree of mental effort on the user. Krueger refers to the measurement of this effort as cognitive distance. Effective reuse techniques are designed to minimize cognitive distance. Throughout this evaluation, aspects of our Claims Library that act to reduce cognitive distance will be highlighted as **strengths**. Similarly, those that increase cognitive distance will be noted as **weaknesses**. Section 3.5 summarizes these strengths and weaknesses and draws conclusions about how the library can be improved.

### 3.1 Abstraction in the Claims Library

*What types of artifacts are reused?* Our Claims Library stores notification system design claims in a format similar to Sutcliffe’s 14-point schema (see Table 2.2). His schema was altered slightly to better support claims about notification systems (see Table 3.1). A major concern in the design of a notification system is the dual-task situation in which it will be used. The Scope field was split into two separate fields, the claim’s primary and secondary tasks, reflecting the importance of this concern. The Effect field was enhanced to include an IRC rating. Finally, additional metadata (Editor, Parent Component, and Usage Log fields) was incorporated for maintenance purposes.

<b>1. Claim ID:</b>	unique identifier for the claim
<b>2. Title:</b>	given by the claim’s author
<b>3a. Author:</b>	researcher(s) or practitioner(s) who developed the original claim
<b>3b. Editor:</b>	person who created the claim record for the library and established the IRC rating
<b>4. Artifact (Design Abstraction):</b>	brief description of the product or application in which the claim originated
<b>5. Description:</b>	natural-language description of the claim
<b>6. Upside(s):</b>	positive effect of using the claim on a system goal or usability
<b>7. Downside(s):</b>	negative effect of using the claim on a system goal or usability
<b>8. Scenario:</b>	scenario in which the claim was derived or is now used
<b>9. Effect (IRC Rating):</b>	desired, and preferably measurable, system goal-usability effect that the implemented claim should achieve
<b>10. Dependencies:</b>	other design problems that have to be solved to achieve the claim’s effect
<b>11. Issues:</b>	design issues influenced by the claim
<b>12. Theory:</b>	underlying theory explicitly referenced by the author
<b>13. Relationships:</b>	interclaim links that describe how claims evolved during a history of investigation
<b>14a. Scope (Primary Task):</b>	generalized task description
<b>14b. Scope (Secondary Task):</b>	generic task description
<b>15. Parent Component:</b>	system name and information
<b>16. Usage Log:</b>	running commentary on experience in using the claim

Table 3.1: Format modifying Sutcliffe’s original 14-point claim schema from [39]

*What abstractions are used to describe the artifacts?* Krueger maintains that abstraction is at the heart of any reuse technique. In fact, it exists in some form in each of the other three aspects of reuse. As a result, the effectiveness of the abstraction process is directly related to the overall cognitive

distance produced by the reuse technique. Every software abstraction has two levels. The higher of these two levels is known as the abstraction *specification*. The lower, more detailed level is known as the abstraction *realization*. The difference between two realizations of the same specification is known as the *variable* part of the abstraction. The aspects of an abstraction that do not change from realization to realization are known as the *fixed* part. (For a more detailed explanation, see Section 2.5.) These terms will be used below to describe abstraction in the Claims Library.

Abstraction of claims never occurs in the traditional sense that an explicit abstract version of a claim is created and stored in the library. This is due to the nature of claims as part of the SBD process. Claims about a design artifact are inherently linked to a scenario of use because their effects (upsides and downsides) are determined by the context in which they are used. Specifically in the case of notification systems, consideration of the primary and secondary tasks is critical in the system design. These are usually contained in a claim’s scenario of use. In order to understand the effects that a design artifact has on these tasks, the scenario must be preserved.

Strict separation between specification and realization is never achieved within the Claims Library framework. Instead, the library incorporates a “footprint” method of abstraction [33]. Each specific claim exists in an abstract, four-dimensional space defined by the primary and secondary tasks it supports, a universal description of its effects on a user as a notification system (IRC rating), and a general description of its originating artifact, referred to as a “design abstraction.” The language used to describe the primary and secondary tasks was borrowed directly from Sutcliffe’s generic and generalized task models (see Section 2.2.2). Generalized tasks are molecular in relation to generic tasks, and were thus chosen to define primary tasks, which tend to be broader in the range of activity they encompass. The more granular generic task model was suitable for defining the independent, atomic notification tasks.

Krueger notes that one of the ways in which cognitive distance can be reduced is through the use of “succinct and expressive” abstractions [23]. The Domain Theory task models provide a finite set of well-defined, abstract representation of the context inherent in a notification system claim. IRC ratings clearly describe a claim’s effect using unambiguous, numerical values to indicate degrees of interruption, reaction, and comprehension. Additionally, the 16-point schema used to represent the realization of a claim is a concise structure that is easily recognized and understood. All of these contribute to the reduction of cognitive distance.

**Strength** The 16-point claim schema, generic and generalized task models, design abstractions, and IRC ratings used in the “footprint” representation of a claim are “succinct and expressive,” reducing the mental effort required to understand a claim and its abstract representation.

At best, the “footprint” representation is only a pseudo-specification (see Figure 3.1). True specifications contain the fixed and variable parts of an abstraction. In this instance, the context of a claim would constitute the variable part, as it changes from reused claim to new claim. In terms of the “footprint,” the context is made up of the primary and secondary task descriptions and the design abstraction. However, the scenario field of a claim also elaborates a claim’s context, yet are not included as part of the specification, making it incomplete. Including the scenario as part of the specification poses a problem because at this level of abstraction, the scenario loses its context, thus alters the claim’s effect.

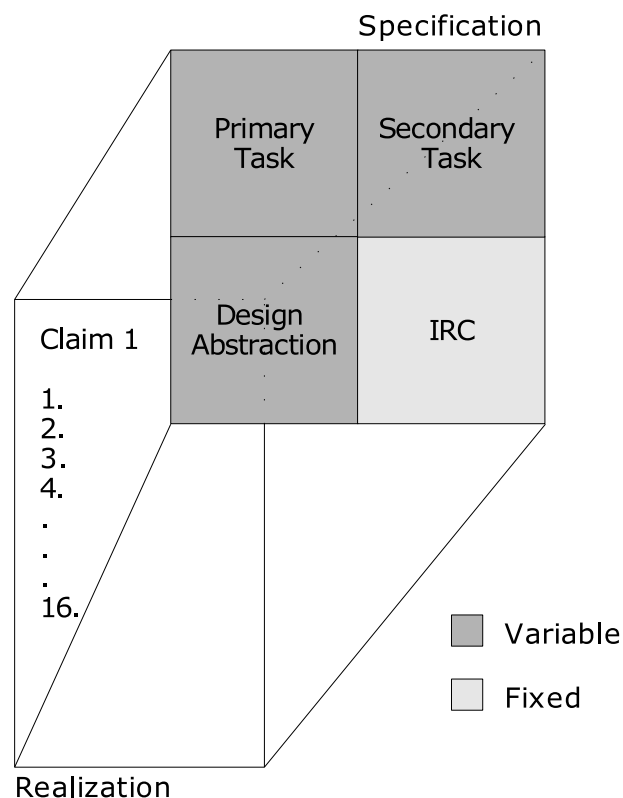


Figure 3.1: “Footprint” representation of a claim. The “footprint” is not a true abstraction because it does not fully capture the variable and fixed parts of a claim in the four fields shown.

**Weakness** The abstraction method used in the Claims Library is incomplete, lacking the explicit creation of an abstraction specification, which could lead to problems during selection, specialization, and/or integration.

The fixed part of the “footprint” would be the IRC rating. When a designer is reusing a claim, the effect of that claim is not intended to change from realization to realization. The previous statement is qualified by the word “intended” because the effect of an artifact is determined by its context of use.

When the context of use is changed, the potential exists for the effect to change as well. However, it is the *intention* of the designer to reproduce the same or similar effect described by the original claim. (Future testing is required to affirm or refute the effect.) Again, the “footprint” parameter is inadequate in describing part of the abstraction specification. The upsides and downsides of a claim are also important in understanding a claim’s effect, and should be considered in this fixed part. Thus, the abstraction method is incomplete in its representation of a claim, which could lead to problems in later aspects of reuse.

## 3.2 Selection using the Claims Library

*How are reusable claims selected for reuse?* There are three basic parts to selection. *Classification* occurs during the design *for* reuse process, when information is being abstracted and stored. *Location* of information is the first step in the design *by* reuse process, and involves narrowing down a large set of data to the potentially reusable. Finally, *selection* itself is the recognition that a piece of information is both applicable and useful, and is thus chosen to be reused.

*Classification* of claims is based on the “footprint” abstraction method described above. Claims are indexed based on their primary and secondary tasks, design abstraction, and IRC rating. A web interface (see Figure 3.2) was built, employing a search mechanism based on this classification. Designers specify one or all of the above criteria in an attempt to *locate* claims that might relate to the current design problem. Carroll and Sutcliffe note that a “classification scheme is limited by the consensus on terminology that can be achieved among the indexers and users” [37]. The solution to this problem is the use of abstractions that are close to the way in which people reason about artifacts informally [23]. The Domain Theory task models, design abstraction terminology, and IRC rating provide a common ground that is easily-accessible by designers.

This classification method also can be used to guide the knowledge acquisition process. All reuse paradigms face a bottleneck in collecting reusable information. During this stage in development, analysis of the four-dimensional space created by the “footprint” representation is able to reveal areas where the content is thin. Concentration on these areas increases the efficiency in attacking this problem.

**Strength** The “footprint” abstraction method used as the basis for claim classification establishes a common ground between how the claims were indexed and how designers reason about claims, reducing cognitive distance.

**Notification Systems Claims Search**

Get Parameters >> Interruption:  Reaction:  Comprehension:

*Item definitions can be accessed by mousing over checkboxes. If no selection is made, that criteria is not considered in the search.* **SEARCH**

Primary Task(s):	Notification Task(s):	Design Concern(s):
<input type="checkbox"/> Analysis-Modeling	<input type="checkbox"/> Assemble	<input type="checkbox"/> Affordances
<input type="checkbox"/> Diagnosis	<input type="checkbox"/> Associate	<input type="checkbox"/> Animation
<input type="checkbox"/> Explanation-Advising	<input type="checkbox"/> Classify	<input type="checkbox"/> Audio
<input type="checkbox"/> Forecasting	<input type="checkbox"/> Communicate	<input type="checkbox"/> Color
<input type="checkbox"/> Information Acquisition	<input type="checkbox"/> Compare	<input type="checkbox"/> Configurability
<input type="checkbox"/> Information Retrieval	<input type="checkbox"/> Decide	<input type="checkbox"/> Error Recovery
<input type="checkbox"/> Judgement-Decision Making	<input type="checkbox"/> Disassemble	<input type="checkbox"/> Feedback
<input type="checkbox"/> Matching	<input type="checkbox"/> Evaluate	<input type="checkbox"/> Fonts
<input type="checkbox"/> Navigation	<input type="checkbox"/> Explain	<input type="checkbox"/> Grouping
<input type="checkbox"/> Other	<input type="checkbox"/> Identify	<input type="checkbox"/> Input Method
<input type="checkbox"/> Planning-Scheduling	<input type="checkbox"/> Interpret	<input type="checkbox"/> Interface Control
<input type="checkbox"/> Progress Tracking	<input type="checkbox"/> Locate	<input type="checkbox"/> Metaphor
<input type="checkbox"/> Validation-Testing	<input type="checkbox"/> Model	<input type="checkbox"/> Screen Space
	<input type="checkbox"/> Monitor	<input type="checkbox"/> Transition
	<input type="checkbox"/> Orient	<input type="checkbox"/> Video
	<input type="checkbox"/> Plan	
	<input type="checkbox"/> Record	
	<input type="checkbox"/> Search	
	<input type="checkbox"/> Select	
	<input type="checkbox"/> Sort	
	<input type="checkbox"/> Test	
	<input type="checkbox"/> Transform	

Figure 3.2: The Claims Library search interface illustrating how the “footprint” method of abstraction can be used to search for claims by specifying a primary task, a secondary task, a design abstraction, or IRC rating.

**Strength** The classification method increases the efficiency in attacking the knowledge acquisition problem by allowing the identification of thin content areas.

*Selection* of a claim for reuse is determined by its perceived relation to the current domain and the effects that it produces. The key weakness identified in the abstraction method was the lack of a true abstraction specification. Assessing a claim’s potential for reusability would be made easier by the existence of a higher-level representation of that claim. Specifications suppress the inconsequential parts of the artifact, so the designer only has to consider those aspects which are pertinent. In this case, nothing is hidden, and the designer is exposed to additional detail which must be sorted through. While the “footprint” generalizes about the context of use and effect, it supplies too little to be the sole factor in selection. The lack of a generalized version of the claim

increases the mental work required to select it for reuse.

**Weakness** When selecting a claim, the lack of a generalized version of the claim in the library forces the designer to sort through additional detail that may not be pertinent to its reuse, increasing cognitive distance.

### 3.3 Specialization using the Claims Library

*How are generalized artifacts specialized for reuse?* Specialization is the process by which a reusable piece of information is applied to a new domain. As described in Section 2.5.3, the difficulty of this process is determined by the level of abstraction in which the information exists. The “footprint” method of abstraction falls short again, requiring developers to work from claim realizations. A full translation from one realization to another must occur, as opposed to merely altering the variable parts of a specification (see Figure 3.3).

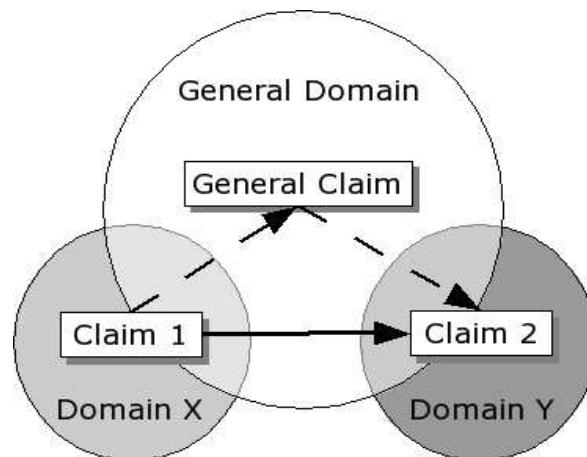


Figure 3.3: Specialization

Requiring a full translation has two negative impacts. Obviously, there is a significant increase in cognitive distance, illustrated in the figure. The designer must conceptualize an abstract form of the claim before translating it to a new domain. The solid arrow represents the net transfer of Claim 1 from Domain X to Claim 2 in Domain Y, however the dotted arrows show the actual path of thought. The general domain that relates the two claims was perhaps identified during the selection process, but the formation of the general claim occurs during specialization.

The second negative impact is less obvious. While an expert developer may understand how claims will apply in a new domain, novices could struggle with accurately performing such translations. The result is misuse instead of reuse, which may be seriously detrimental to the final

design product, requiring additional iterations of development to iron out the problems. Working from a generalized claim increases accuracy by reducing the margin of error involved in the overall translation.

**Weakness** Specialization is hindered by the lack of true abstraction, requiring designers to transfer knowledge from one specific domain to another. This both increases cognitive distance and the potential for misinterpretation by novice designers.

### 3.4 Integration using the Claims Library

*How are reusable artifacts integrated to create a complete software system?* Integration refers to the process by which reusable information is recombined to form new solutions. There are two ways in which claims can be integrated into the development of a system. First, claim may be used in creating a scenario. Aspects of this claim (design artifact, effects) become a part of the scenario. New claims extracted from this scenario should be able to trace their parts back to the reused claim. This is defined here as a *scenario-first approach*, whereby the reused claim contributes to directly to the creation of a scenario, which leads to the generation of a new claim. The second possibility is that a claim is transformed into a new claim before an entire scenario is developed (*claim-first approach*). This new claim is then incorporated into one or multiple scenarios of use. Again, its roots should be traceable back to the original claim.

In either case, scenarios supply the integration framework for reusing claims. However, the Claims Library treats scenarios as secondary aspects of design, attaching them to claims, as opposed to recognizing them as an equal (if not greater) part of the design process. In fact, the library is deficient of any integration framework. Claims cannot be saved, compared, or viewed collectively. A claim's original scenario of use provides some insight into how these claims can be incorporated in a new context, but this task becomes daunting with a large number of reusable claims from separate domains are recombined. Ultimately, the integration of claims into a new system is left completely up to the designer.

**Weakness** The Claims Library does not contain any integration framework, nor does it aid the process. Instead, the task of integrating claims is left up to the designer.

### 3.5 Summary

Effectively reducing cognitive distance hinges upon the abstraction method employed by the reuse technique [23]. However, abstraction of a claim poses difficulties because of its fixture within a scenario of use. Generalization of claims risks the loss of its effect because the claim becomes removed from its native context. Rather than creating abstract versions of claims, the library indexes them in a four-dimensional, abstract space that describes the context and effect of the claim, and stores them in their original state. The evaluation above reveals both the positive (Table 3.2) and the negative (Table 3.3) consequences of this decision.

Shortening the gap between a designer’s conceptualization of a system and the information in the library is one approach to reducing cognitive distance [23]. The Claims Library is effective in this endeavor by providing an easily accessible classification method. User activity described by primary and secondary tasks is bounded by the Domain Theory task models. The artifact description and claim effect are likewise restricted by a finite set of design abstractions and the IRC rating system, respectively. The 16-point claim schema clearly defines a claim’s constitution, removing any structural ambiguity. While a designer’s notion of a system and a claim can stretch as wide as the human imagination, these descriptions clearly mark the boundaries of the information in the library in an approachable way. This method of classification also supplies a metric for measuring and progressing toward resolving the knowledge acquisition problem. Inadequate data coverage can be easily recognized and more quickly resolved.

<b>Strengths</b>
The 16-point claim schema, generic and generalized task models from the Domain Theory, and IRC ratings used in the abstract representation of a claim are “succinct and expressive,” reducing the mental effort required to understand a claim and its abstract representation.
The “footprint” abstraction method used as the basis for claim classification establishes a common ground between how the claims were indexed and how designers reason about claims, reducing cognitive distance.
The classification method increases the efficiency in attacking the knowledge acquisition problem by allowing the identification of thin content areas.

Table 3.2: Summary of strengths identified by theoretical evaluation of the Claims Library.

Unfortunately, the “footprint” representation of a claim is insufficient in supporting selection, specialization, and integration. Each of these processes relies either entirely or in part on the existence of a true abstraction. The beauty of abstraction is its suppression of unnecessary detail, yet the library burdens designers with instances of complete claims. Determining which claims are appropriate for reuse (selection) and exactly how to use them (specialization and integration) become more difficult tasks. In addition to the increase in effort, there is a greater margin error,

particularly with non-expert users.

Weaknesses
The abstraction method used in the Claims Library is incomplete, lacking the explicit creation of an abstraction specification, which could lead to problems during selection, specialization, and/or integration.
When selecting a claim, the lack of a generalized version of the claim in the library forces the designer to sort through additional detail that may not be pertinent to its reuse, increasing cognitive distance.
Specialization is hindered by the lack of true abstraction, requiring designers to transfer knowledge from one specific domain to another. This both increases cognitive distance and the potential for misinterpretation by novice designers.
The Claims Library does not contain any integration framework, nor does it aid the process. Instead, the task of integrating claims is left up to the designer.

Table 3.3: Summary of weaknesses identified by theoretical evaluation of the Claims Library.

Improvements to the fundamental parts of the Claims Library could resolve these issues. True abstractions should be stored in the library, and this could be accomplished through the incorporation of a claim factoring process like the one described by Carroll and Sutcliffe. They recommend, however that “claims analysis [be] allowed to progress through several generations before factoring is attempted” to avoid the generalization of “immature knowledge” [37]. A greater emphasis should be placed on scenarios as well. Scenarios appear to be the solution to the absence of an integration framework within the library. They are the means by which claims are combined, depicting how each aspect of a design works as a coherent unit.

While the theoretical evaluation identified potential areas in which our Claims Library could be improved, it was unclear at this juncture exactly how or to what degree a designer would be affected. Instead of using this information as the basis for a redesign of the library, we chose to perform a practical analysis in order to gain a deeper understanding of these issues. Perhaps the use of a “footprint” method of abstraction *was* the solution to the dilemma posed by attempting to reuse claims grounded in specific context within a different domain. Perhaps Sutcliffe’s task models would *not* prove to be easily-accessible by designers. The valuable insight provided from the theoretical analysis supplied the groundwork for evaluating the design experience described in the next chapter.

## Chapter 4

# The Claims Library: A Practical Perspective

In order to provide a balanced evaluation of the Claims Library, a practical evaluation was conducted from the perspective of a designer developing an input method for an In-Vehicle Information System (IVIS). A good portion of the data within the library pertains to the work of Chuck Holbrook [20]. Holbrook formulated a design analysis technique that focused on input for dual-task situations. His testing involved the comparison of three IVIS input mechanisms, and resulted in the creation of related claims and their addition to the library.

Perhaps the most interesting aspect of this design experience was how using the Claims Library affected the SBD process. In a non-reuse situation, designers begin with their own knowledge of the design space, collected from the requirements analysis and previous personal experience, which may be limited within a new domain. The subsequent design phases of SBD would consist of the generation of scenarios, followed by the extraction of specific claims from that scenario. Claims are then validated or refuted by testing. Scenarios are modified based on the test results, and a new design iteration begins.

In a reuse situation, the generation of scenarios is enhanced by existing claims in the library. Instead of beginning from personal knowledge of the domain, designers have access to a repository of design knowledge that helps them in creating scenarios. The design described here will follow this procedure, listing the key claims retrieved from the library first, followed by a scenario generated from certain aspects of these claims, and then extracting the new claims that exist within this scenario which should incorporate some aspects of the retrieved claims. These aspects will be highlighted in each design phase.

This chapter describes a single iteration of the SBD process. Section 4.1 describes the analysis phase of SBD, an examination of the current problem space. Section 4.2 discusses the subsequent design phases. Section 4.3 describes the initial testing that was performed on the input device. These processes lead to the identification of **strengths** and **weaknesses** in the Claims Library. Section 4.4 summarizes overall performance of the library and provides suggestions for improvement.

## 4.1 Analysis

The development of an IVIS requires much more thought and analysis than the development of most other notification systems due to the simple fact that design mistakes could cost lives. The primary task of the user of in-vehicle systems is driving, and it is imperative that the safety of the user is never put into jeopardy. Many states have already enacted laws that prohibit the use of distracting devices, cell phones in particular, while driving.

However, more and more automobiles are being manufactured with systems that are designed to support the primary task of navigation. Global Positioning Satellite (GPS) systems aid the driver in arriving at his or her destination efficiently, minimizing the risk of getting lost. In-vehicle entertainment systems have been incorporated to increase the pleasure of a long road trip. The intention of these systems has never been to interfere with a person's ability to drive, yet with the current design of these systems, the driver's safety is called into question.

Occasionally, these systems require some sort of textual input from the user. Table 4.1 lists the most popular methods of receiving textual input from a user utilized by common in-vehicle GPS systems. The table also indicates the positive and negative aspects of each input device within this context (entering text) identified in [20, 41].

Input Methods	Positive	Negative
Touchscreen	High familiarity Can be directly manipulated by the user	Highly visually-intensive Requires at least one hand
Remote control	Potentially requires less visual attention as devices have a feel to which users can become accustomed	Requires at least one hand for input If the user is not familiar with character mapping, could cause higher levels of distraction
Voice recognition	Hands-free Requires little to no visual attention from the driver	Poor voice recognition software Problems caused by environmental noise

Table 4.1: Positive and negative aspects of in-vehicle GPS system input methods discussed in [20, 41].

Tijerna et. al. conclude that visual-manual text input methods such as the touchscreen and

remote control are dangerous to use while driving [41]. Users of the voice input method performed remarkably well. One of the metrics used to measure how detrimental a device was to one's driving performance was lane exceedences. Subjects using voice input did not have one lane exceedence during the entire trial. Holbrook's work contributed the claim in Figure 4.1 to the library that helped summarize voice input.

### Decide While Navigating

A voice command interpreter allows a user to perform a variety of operations using a very logical interface...

- + Minimizes unwanted interruption by providing a method for novice users to input a variety of complex commands with only knowledge of the command language with logical correlations to human-human communication
- + Minimizes unwanted interruptions by not requiring physical interaction with a device
- + Spoken commands that correspond to the names of system capabilities can increase comprehension of the system affordances
- + Literally thousands of commands can be supported at once
- The performance of the system depends on quality of the recognition engine
- Noisy environments can greatly effect system performance
- Performance may differ greatly between users
- May cause unwanted interruption for users unfamiliar with the command language
- May cause slower reaction times for simple tasks that only have a small number of commands that could all map to buttons on a remote or other interface that does not require remembering a command language

Figure 4.1: Reused problem claim describing the tradeoffs in using a voice command interpreter

The “Decide While Navigating” claim and others found in the library describing different in-vehicle input devices helped in the creation of scenarios used to highlight important features of the problem space. The following scenario further explores voice input:

**Voice Interface Scenario** *Mark and his family have been travelling across the United States for the last few weeks. He and his wife wanted their kids to finally see all there is to see in the country during their summer vacation. On a whim, Mark decides that he would like to take his family to an early colonial settlement on their way up the East Coast, a destination that was not on their original itinerary. Mark begins to spell the letters of their destination at the audio prompting from*

*the van's GPS. While he is speaking, the noise from the kids in the backseat interferes with the voice recognition by the system. After quieting them down, he is forced to begin again. He speaks the characters one-by-one, until the city was recognized as a unique location by the system. Mark is a little frustrated, both by the misbehavior of his children and by the amount of time it took for him to enter his destination.*

Voice recognition would seem to be ideal for in-vehicle systems, however the problem exhibited in this scenario seems to be the major drawback: environmental noise. In [32], Oviatt points out that environmental noise actually introduces two restrictions on the performance of a voice interface. Obviously, the first is that the environmental noise itself “contaminates the speech signal” [32]. The second is something referred to as the Lombard effect [24], which states that people speak differently when there is background noise. While this appears to increase understandability for other human listeners, voice recognition systems have a harder time processing the effected speech.

The Claims Library proved to be very useful in the generation of problem scenarios. While claims were important in reinforcing design issues identified through research, perhaps the greater value was found in their concise articulation of these issues. As described in the previous chapter, claims are stored in their original state in the library. Applying a claim in a new domain requires significant mental effort, but at this stage in development, claims in the library only need to be examined in their current form. No translation is required to understand how a claim pertains to its original problem space.

**Strength** Storing claims in their original state provides valuable insight into the analysis phase of design with little effort required on the part of the developer.

The culmination of this analysis was the creation of a root concept, described in the next section.

A driver's safety is of the utmost importance in the development of this type of interface. How does one achieve the greatest level of safety? Based on the analysis above, an ideal input device will have two key requirements. First, the driver's eyes should never be required to leave the road. Allowing the user to enter text without requiring visual feedback is ideal. Secondly, the driver's hands should never be required to leave the steering wheel. Quick response to driving conditions should not be compromised when providing textual input to a system.

Secondary qualities of an input device are desired in addition to safety. Ward et. al. describe various tradeoffs that should be considered when designing an input device [44]. A tradeoff exists between a device's size (number of buttons) and the size of the set of characters to which the buttons map. Device design should maximize this tradeoff by allowing the largest set of characters for the fewest number of buttons.

There is also a tradeoff between “efficiency and ease of learning” – the design “must favor the selection of common characters, while still being learnable” [44]. Often, this requires leveraging designs to which a user is already accustomed. A good example of this is the addition of miniature QWERTY keyboards to personal digital assistants (PDAs). People are used to typing in text on a standard keyboard, so it was natural to incorporate this method on a new platform in spite of its inefficiency.

The root concept of the input device is summarized in Table 4.2. This table indicates both the primary and secondary goals that design of this device will hope to achieve. In addition to these more general goals, a tool developed by Chewar was used to determine a target IRC value for this notification system [11]. Responses to various questions about the intended use of the system produces the recommended levels of interruption, reaction, and comprehension that should be achieved. This value is included in the table.

<b>Primary Goals</b>
Require little visual attention from the driver
Allow the driver to keep his/her hands on the wheel at all times
<b>Secondary Goals</b>
Maximize ratio between character map and device size (number of buttons)
Do not sacrifice learnability for efficiency
<b>Target IRC:</b> 0.33 / 0.82 / 0.88

Table 4.2: Root concept highlighting primary and secondary goals and the target IRC of the input device design.

## 4.2 Design Phase

The following sections describe each of the design phases. The entire design process is not documented here. Rather, each section exposes key design decisions, providing an overall picture of the input device, and highlights how the Claims Library either contributed to or hindered the process.

### 4.2.1 Activity Design

The activity design of a notification system is unique in that the developer is designing an interface for a secondary set of tasks. In this particular instance, the primary task of the user is driving an automobile, a task that requires much of the user’s concentration. The input device for an IVIS needs to be somewhat transparent in its support of text entry, so that it does not interfere with the driving task, yet still allows the efficient use of the input device.

From the analysis above, it was clear that a traditional visual-manual device would be unaccept-

able, and that the problems associated with voice input should be avoided. The claim depicted in Figure 4.2 inspired the idea of attaching the input device to the steering wheel of the car. Moreover, it is an excellent example of how the language used in a claim contributes to how it is understood. The phrase “at the user’s fingertips” was all that was needed to jumpstart thought about device placement. Both of the downsides of this claim would be mitigated by combining the tool used for the primary task with the device used for the secondary task. Location of the device would not be an issue, nor would the need for the user to choose the wheel over the input device.

#### **Select while Navigating**

The portability of a remote control allows a user to remain in contact with the interface while performing other tasks...

- + Enables quick reaction to notification events since the remote will be at the users fingertips.
- + May lessen interruption if the user has to locate the interface
- May cause unwanted interruption if the remote is not in the user’s hand and has to be located
- May cause unwanted interruption if the user needs to use the hand holding the remote for the primary task.

Figure 4.2: Reused activity claim describing the use of a remote control for interacting with an in-vehicle information system.

On the other hand, some of the language used in the claim is confusing. The title itself was not particularly descriptive: “Select While Navigating.” Little if anything can be gleaned about the claim from that title. Also, the second upside of the claim says that interruption may be decreased “if the user has to locate the device.” Perhaps the claim author intended the phrase to read “if the user has already located the device,” which would make more sense. Interruption would be less of an issue if the device was readily available to the user. Other claims that were examined also exhibited the same inconsistencies. Some had not even been assigned an IRC rating, one of the key components of claim classification. None of the claims appeared to have any theoretical backing; nor did they exhibit any interclaim relationships, describing how the claim may have evolved over the course of design. These discrepancies all lead to both confusion on the part of the designer and hesitancy to reuse the information.

**Weakness** Inconsistencies in claim data (poorly chosen titles, lack of IRC ratings, missing theoretical backing and interclaim relationships) lead to confusion and insecurity about the information in the library.

Holbrook identifies four critical resources that should be considered when designing any input method: tactile, audible, visual, and cognitive [20]. *Tactile* refers to the amount of physical interaction a user has with the device. For example, a mouse would have a high tactile rating, while gaze input would be low. *Audible* resources are the sounds produced both by the user as input and by the system as feedback. *Visual* resources are typically used to process feedback from the system, but simply refers to the amount of attention required on the part of the user. *Cognitive* refers to the amount of mental effort required to interact with the input device. Input methods are classified based on the degree to which they require these resources. The following scenario was used to reason about how each of them would be utilized in the proposed solution. (The results are summarized in Table 4.3.)

**Steering Wheel Input Device** *Mike is running late for a meeting with a potential client. He leaves his office in a rush without directions to the meeting location. He remembers the address, and intends to enter it as he is driving. Using the input device attached to his steering wheel, Mike enters each character of the address to his in-vehicle GPS. Midway through entering the address, Mike determines from the audio feedback that he had just entered the wrong character. He deletes the most recent character and replaces it with the correct one. When he finishes typing, he submits the text to the system. Instantly, the system recognizes the address and begins guiding him in the right direction. Mike arrives at his meeting on time and is able to secure the new client.*

Resource	Usage
Tactile	High
Audible	Medium
Cognitive	Medium
Visual	Low

Table 4.3: Target resource usage for steering wheel input device in conjunction with audio feedback.

The proposed device is highly tactile, as it will likely require a number of buttons or knobs in order to support a sufficient character set. Both its use of audio feedback and placement on the steering wheel reduce the level of required visual resources. As the device does not require any additional voice input, the required audible resources remain at a medium level. Recalling which characters have been entered is supported by the audio confirmation, but using the device will be far from “instinctive” [20], resulting in a medium requirement of cognitive services.

### 4.2.2 Information Design

Working from this new idea for a wheel-attached input device, information design is concerned with supporting the functionality outlined in the activity design through the meaningful representation of objects and possible actions [34]. Two major issues of this phase are discussed in this section: audio feedback and character mapping. Audio feedback when a user performs some action is a non-visual representation of the results of that action. In terms of Norman's Stages of Action, the playback would be the user's *perception* of information. The design of the character mapping contributes to a user's overall comprehension, allowing the user to make sense of the system and formulate system goals.

In an effort to significantly reduce visual attention requirements, the input device proposes the use of audio feedback as actions are performed. These actions can be enumerated from the activity scenario above: entering a character, deleting a character, and submitting input to the system. As each character is entered by the user, a recording of that letter is played back. As each action is performed (e.g. submitting input, deleting a character), some audio feedback is returned. The library produced the claim in Figure 4.3 about auditory feedback.

#### Identify while navigating

When a scroll key is pressed on a tactile interface, auditory input feedback speaks the name of the highlighted list element...

- + Enables the user to react to the new selection
- + Enables comprehension of the other elements in the list seen while browsing
- + Will not rely on the visual resources of the driver
- May cause unwanted interruption to driving by placing a cognitive burden on the user to understand spoken words especially if the text to speech system is poor
- If the list is large, it may take a longer time to scroll through than a visual list.

Figure 4.3: Reused information claim describing the use of auditory feedback when selecting an item

While this claim deals with supporting the selection of items from a list, many of the same principles apply to this secondary task. Some aspects do not apply at all (e.g. impeding scroll time), while others fit perfectly (e.g. not relying on visual resources). Other aspects of the claim will need to be modified slightly. For example, the cognitive burden on the user will not be as great because the output will be predictable and a text-to-speech system will not be required. Single characters, sounds, and perhaps short phrases will be used instead. Similarly, a user's reaction will

not be to a potential selection in this case. Rather, users react to the entry of a character or the performance of some action. The resulting claim is shown in Figure 4.4.

#### **Auditory feedback for text input**

Auditory feedback is given as users provide input to the system. . .

- + Enables the user to react to text entry
- + Enables comprehension of the entered text
- + Will not rely on the visual resources of the user
- May cause some interruption by placing a cognitive burden on the user to understand spoken characters and words

Figure 4.4: Transformed design claim, reusing information from the claim in Figure 4.3

Character mapping has an impact on one of the secondary goals found in the root concept: learnability. The character mapping is also influenced by the interaction design described in the next section. For the purpose of coherence, the mapping is discussed first, as it is a part of the information design process, but this is an excellent example of how information and interaction design often occur in parallel. The claims discussed below have an evaluating-executing relationship with claims in Section 4.2.3.

From the interaction design, the system places a directional pad at the 10 o'clock position of the steering wheel, and four buttons arranged in the shape of a diamond at the 2 o'clock position (see Figure 4.5). Considering the nine distinct positions of the directional pad (N, NW, E, SE, S, SW, W, NW, and none selected), there are 36 combinations of direction and button that can be mapped to individual characters. A character is entered by first selecting a direction and then pressing a button, or by not selecting any direction and pressing a button by itself.

The learnability-efficiency tradeoff became apparent. Arranging characters in a familiar way will increase the chance that the user will be able to quickly learn the mapping. For example, most people are highly familiar with the alphabet, and an alphabetic character mapping would be easily learnable. However, due to the nature of the device, the secondary directions (NE, SE, SW, NW) are more difficult to select than the standard cardinal directions. Characters could instead be placed by frequency of use, aligning the less frequent characters with the harder combinations. For example, “E” would be entered by selecting north on the directional pad and button 1 (see Figure 4.6). This tradeoff is summarized in the claims found in Figure 4.7 and Figure 4.8.

After much thought, the decision was made to use the alphabetic mapping over the character



Figure 4.5: Steering wheel with directional pad and buttons at 10 o'clock and 2 o'clock, respectively.

frequency mapping. Learnability was one of the goals identified in the root concept because it contributes to lowering the overall interruption level of the input device. If a user can easily learn the character mapping, fewer cognitive resources will be required for determining how to type a letter. Those resources can then be used for primary task performance. Figure 4.9 shows two later depictions of the character mapping that was used. The image on the right is the final product, and was used during usability testing. Character mapping begins at the north direction and begins clockwise around the pad. Likewise, buttons are numbered clockwise beginning at the top of the diamond formation, and characters are arranged alphabetically.

The information design process revealed another deficiency of the Claims Library. The current interface provides no method of saving, comparing, or combining claims. When searching through the repository for relevant information, multiple browser windows had to be opened in order to look at more than one claim simultaneously. These claims exist in a purely textual format that makes manipulation of the information difficult as well. This is perhaps an elaboration of the weakness found in the theoretical evaluation, identifying a lack of integration environment.

**Weakness** Interface provided no method of saving, comparing, or combining claims, making claim manipulation difficult.

The following information scenario summarizes the claims found in this section:

**Information Scenario** *Mark begins entering characters of a new location using the input device attached to the steering wheel. His destination begins with the letter 'S'. He remembers that the letter*

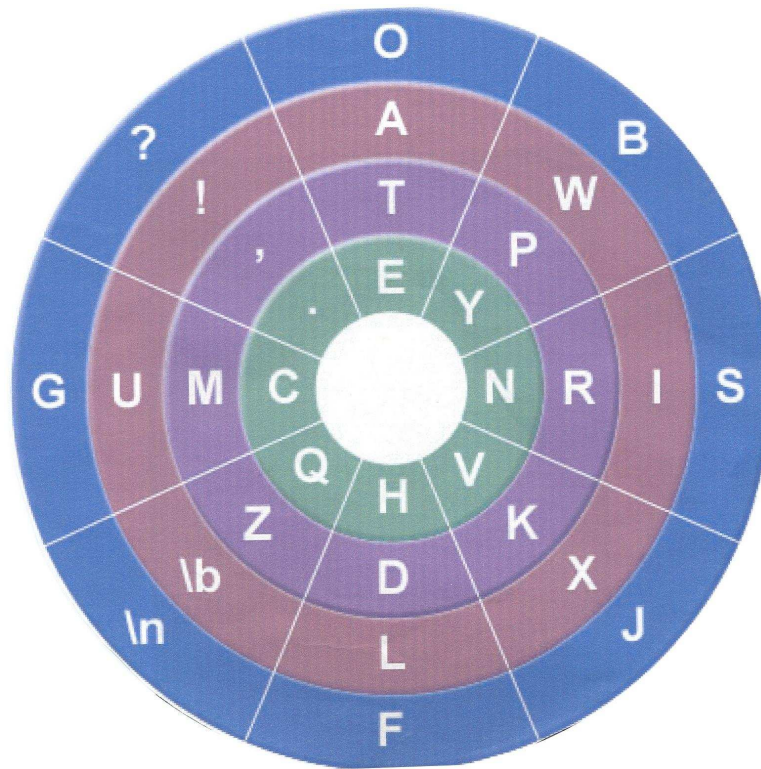


Figure 4.6: Initial character mapping by frequency of use.

'Q' is located at the combination of pressing down on the directional pad and Button 1. Because the characters are arranged alphabetically, he knows that 'S' must be located at down and Button 3. His next letter is 'O.' He has a general idea of where 'O' might be located, so he selects a combination in that general area. From the voice feedback he receives, he realizes that he made a mistake, selecting 'M' instead. He deletes 'M', and can now deduce where 'O' is located. He enters the correct character, and continues. When he finishes, he glances briefly at the GPS display to check his input and then enters the text.

In conclusion, it is important to note that the audio feedback is not intended to completely replace a visual interface, as witnessed at the end of the scenario. However, the amount of required visual attention is reduced significantly. Furthermore, buttons are numbered clockwise, beginning at the top of the formation (as described above). The mention of Button 3, for example, refers to the button at the bottom of the diamond (see Figure 4.5).

The next section describes the closely-related interaction design phase.

### Mapping Characters by Frequency of Use and Ease of Selection

Mapping characters so that the most commonly used are the easiest to select...

- + reduces the physical requirements of the user to a minimum
- + improves typing speed once mapping is learned
- may be harder for the user to learn the mapping, as the characters would have no recognizable order

Figure 4.7: Claim illustrating the benefits of a character mapping based on character frequency

### Mapping Characters Alphabetically

Mapping characters alphabetically...

- + increases learnability by applying a well-known, easy-to-remember order to the character mapping
- may hinder typing speed, as frequently used characters may be more difficult to type

Figure 4.8: Claim illustrating the benefits of an alphabetic character mapping

## 4.2.3 Interaction Design

Interaction design describes *how* users accomplish their desired goals. The physical actions that someone performs to enter text using an input device fall into this category of design. Though the work contained in this section follows information design, this was not the case during development. Particularly with input devices, but also in general, interaction and information design have significant influences on each other.

As an example, consider the design of the character mapping described above. Characters are mapped first by direction and then by button instead of vice versa. This was a result of the decision to require users to first physically select a direction on the directional pad and then select a button. The physical action of pressing a button is very similar to the sensation of typing on a keyboard, as opposed to selecting a direction on a directional pad. Consequently, the graphical representation and conceptual design of the character mapping changed dramatically.

When driving, even with both hands on the steering wheel, a person's thumbs are typically less involved in controlling the car. When hands are placed in the recommended driving position (10-and-2), the thumbs are pointed toward the inner portion of the steering wheel. The buttons and

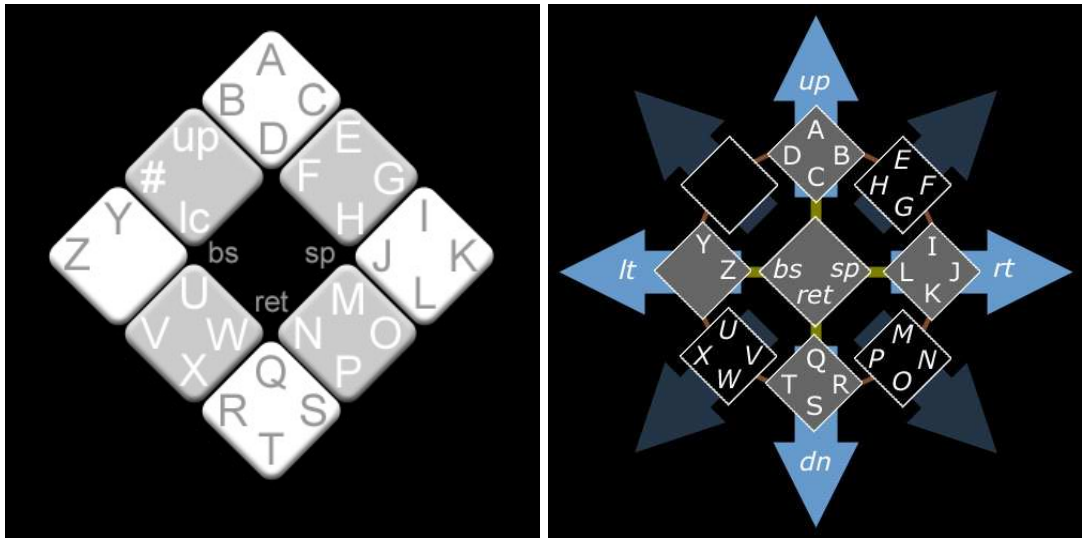


Figure 4.9: Graphical depictions of later character mappings. The image on the left illustrates an alphabetical mapping, clockwise around the directional pad and top-to-bottom, left-to-right for the buttons. The final design is on the right, mapping characters clockwise around the directional pad *and* around the buttons.

directional pad were chosen to be placed at this location, exploiting this fact. (This design choice was made *before* the Logitech ® Driving Force wheel was found. Fortunately, no one had to build the hardware from scratch.) As a result, the input device became known as ThumbType.

The Claims Library proved useful in supplying claims dealing with in-vehicle input devices. Finding claims from the exact domain, such as the one described in Figure 4.10, made it easy to adapt the claims via small changes in the design artifact. While many of the claims in the library are related to the secondary task of selecting from a list (a result of Holbrook’s work in [20]), the reused claim provided here was used to “perform a variety of operations.” As a more general claim, altering it for text entry was easier. Figure 4.11 shows the resulting interaction claim.

Moreover, the library appeared to return an accurate set of claims from the search criteria that was entered. This reveals two qualities. First, it indicates the understandability of the classification mechanism that is used to organize information. The way in which the designer reasoned about the input device was easily mapped to a primary task, secondary task, design abstraction, and to a lesser degree, the IRC rating. (The IRC portion of the search mechanism was not fully functional during development. As mentioned above, not all claims had IRC ratings, either. The rating was useful in discriminating against results that were returned when it was present, but was not consistently used in searching for claims.)

**Decide for Navigation**

A graffiti command interpreter allows a user to perform a variety of operations using a single physical interface...

- + Minimizes unwanted interruption by providing a method for expert users to input a variety of complex commands with a single interface
- + Minimizes unwanted interruptions by providing a hardware device location that never changes
- May cause unwanted interruption for users unfamiliar with the command language
- May cause slower reaction times for simple tasks that only have a small number of commands that could all map to buttons on a remote or other interface

Figure 4.10: Reused interaction claim describing the use of a graffiti interface

**Strength** Search mechanism produced accurate claims in terms of relevance to the design problem, facilitating their reuse, and indicating the definition and utilization strong classification method.

Second, it implies that claims were accurately classified when they were first entered into the repository. Just as important as having a good classification method is carefully indexing where information should go. Misplaced claims, however appropriate, might never be found by a designer. By the same token, misplaced claims could also cause confusion when they do not appear to have any relevance to the search criteria.

**Input Device Attached to Steering Wheel**

Using an input device located on the steering wheel, consisting of a directional pad and buttons...

- + Minimizes unwanted interruption by providing a method for expert users to input a variety of characters with a relatively small device
- + Minimizes unwanted interruptions by providing a hardware device location that never changes
- + Increases reaction time by mapping characters to a directional pad-button combination
- May cause unwanted interruption for users unfamiliar with the input method

Figure 4.11: Interaction claim about attaching the input device to a steering wheel

One disappointment, however, was the lack of claims from other domains. A variety of different keyboards have been developed, such as the chorded keyboard [25], FingeRing [16], and half-QWERTY keyboard [26]. With the rise in popularity of mobile phones and PDAs, innovative research has resulted in the creation of new text input methods for these devices [4, 19, 25, 36]. The particular attraction to small, mobile computing devices is that text input is almost exclusively performed by the thumbs. Surely parallels could have been drawn from these areas.

**Weakness** Sparsity of relevant claims outside the realm of in-vehicle information systems indicates a lack of coverage in terms of tasks and design abstractions.

Had a different design project been chosen for this thesis, would the library have been as successful in fostering claim reuse? Did this design fall short of its potential because it was limited by the knowledge in the Claims Library (and the knowledge of the developer)? Still early in development, the library has apparent holes in its content. This is an illustration of the knowledge acquisition problem facing all reuse paradigms. The silver lining is that even from the relatively small amount of available information, the reuse of claims aided in the creation of the ThumbType interface.

### 4.3 Testing

A usability study was conducted in order to further examine the design of the text input device. The study was conducted around a similar scenario to those described above. Two sets of ten users performed a primary activity (driving a car simulator) while entering text using one of two input methods: ThumbType and a simulated voice input device. A true voice recognizer was not used for this study. Rather, a “wizard of oz” approach was used, similar to the one used in [2]. Voice recognition takes time to configure, and must be calibrated for each subject. As in [2, 41], users entered words by spelling them, character by character. These were entered via keyboard by the test administrator.

The testing platform consisted of three parts: a driving simulator, a text input display, and a survey tool. The simulator was developed by Maxim Moldenhauer using the DIVERSE toolkit and OpenGL’s Performer software [29]. The survey tool was written by Holbrook, and modified for the dynamic creation of entrance and exit survey pages as well as testing instructions pages. The tool also recorded the responses to these surveys in comma-separated value (CSV) format, which is easily imported into spreadsheets or statistical software. Both of these tools were used in some form for Holbrook’s work. The simulator used the default “town” world included with Performer, employing a physics engine written by Moldenhauer for navigation.

The text input display was very basic, displaying a small, static map image and a text entry box. The simulator and input display both ran on a SuSE Linux 9.0 machine with dual-monitor display. The monitors were arranged such that there was a main display directly in front of the user, and a secondary display slightly lower and to the right of the user. The main display was connected to a KVM switch which allowed the transition between a laptop running Windows2000 and the Linux machine. The survey software was written for Windows, necessitating this configuration. A wireless keyboard and two wireless mice were used so that the test administrator could set up various parts of the evaluation without needing to be directly in front of the system.

Subjects sat directly behind a Logitech ® Driving Force steering wheel, originally designed for use with the Playstation2 entertainment system. A dynamic shared object (DSO) was created to read input from this USB device. The DSO was used by the driving simulator for reading changes in heading and acceleration. The text input display used the Simple DirectMedia Library (SDL) to read input from the user (either from the keyboard or directly from the wheel in the case of the ThumbType input method). SDL is a powerful library used in many open source games because of its compatibility with OpenGL's 3-D engine and event processing capabilities.

The input method was the only part of the test that was varied among users. Subjects were given a short entrance survey upon beginning the evaluation. In addition to typical demographic information, this survey asked questions related to the subject's experience 1) driving, 2) playing videogames, and 3) with dual-task, in-vehicle situations (using a cell phone or GPS system). The purpose of these questions was to determine if there were any underlying trends between this experience and one's performance during the evaluation.

After the entrance survey, users were given a brief introduction to the primary and secondary tasks which they were going to be performing (driving and entering text, respectively). Then, each subject was given an opportunity to become more familiar with the driving simulator. They were required to complete a short course consisting of seventeen checkpoints scattered around the "town." The course was designed such that users would become familiar with the various types of turns that they could possibly encounter during the course of the evaluation, in addition to becoming familiar with how the "car" handled and how navigation from checkpoint to checkpoint would proceed. Data was collected during these initial trials, and were used as a baseline (benchmark) for all subjects.

Next, users were given a description of the input method that was going to be used in the evaluation. They had an opportunity to practice with each method. The text input display was brought up, and users were able to enter text until they were satisfied with their performance. Understandably, this practice did not last long with voice input subjects. They were asked to enter a city and state abbreviation, exactly as would be asked of them during the real evaluation.

Typically, one or two locations were dictated before the user was ready to continue. These were entered on the wireless keyboard by the administrator, as each character was spoken. The only restriction was that the participants speak clearly enough for the administrator to understand the character.

None of the ThumbType users had ever seen the input device before the evaluation, so more practice was required. In addition to the text input display, the character mapping (right-hand side of Figure 4.9) was displayed as a reference for the user. Basic tasks were suggested for the user to complete, such as entering the entire alphabet forwards and backwards, entering the user's first and last name, and entering their home town and state abbreviation. Upon completion, the subject was then asked to perform the latter two tasks with the mapping out of view (minimized by the administrator), using the voice feedback to guide their input. Once these tasks were completed, the user proceeded to the next portion of the evaluation.

In the full evaluation, users were told that they were beginning a spontaneous road trip with their friends, and as the subject drove around town, the people in the backseat suggested destinations for their trip. The driver had to enter these suggestions into the GPS system so their trip could be planned. Each subject navigated a course that was twice as long as the practice, consisting of different locations. Periodically, the test administrator would verbally suggest a city and state. Subjects had to enter the name of the city and the state abbreviation, with appropriate spacing, while driving. Before the evaluation started, participants were reminded that their driving was the key concern, and not to sacrifice their performance in order to enter text. Destination prompts were given roughly ten seconds after the previous destination had been entered into the system.

The input display program was also built to record both typing metrics and driving metrics that were supplied by the simulator via a message passing protocol. Table 4.4 and Table 4.5 contain the driving statistics from the benchmark test and full evaluation, respectively. With the exception of the "Number of Times Speeding per Trial" field, t-tests performed on the benchmark data did not reveal any significant differences between the two data sets. This indicates that the drivers all had about the same driving ability, and that no one group fortunately had better drivers than the other.

Metric (mean, standard deviation)	ThumbType		Voice	
<b>Total Time (seconds)</b>	298.600	45.5	312.500	127.18
<b>Collisions per Trial</b>	2.900	1.79	3.500	1.72
<b>Number Times Speeding per Trial</b>	0.900	0.99	2.800	2.49
<b>Number Times Too Slow Per Trial</b>	7.300	2.45	7.300	4.45
<b>Spinouts per Trial</b>	0.500	0.97	1.200	3.46
<b>Number of Large Accelerations</b>	0.800	1.14	1.300	1.42
<b>Speed</b>	16.929	2.27	18.294	2.68

Table 4.4: Benchmark Driving Statistics

Metric (mean, standard deviation)	ThumbType		Voice	
<b>Total Time (seconds)</b>	607.400	109.4	473.900	52.7
<b>Collisions per Trial</b>	1.100	2.18	4.600	3.98
<b>Number Times Speeding per Trial</b>	8.100	4.31	6.100	3.00
<b>Number Times Too Slow Per Trial</b>	2.700	2.26	2.800	2.74
<b>Spinouts per Trial</b>	0.000	0.00	0.100	0.32
<b>Number of Large Accelerations</b>	0.400	0.70	1.300	2.26
<b>Speed</b>	17.920	1.60	20.701	2.06
<b>Speed (Typing)</b>	17.838	1.72	19.685	2.52
<b>Speed (Driving Only)</b>	18.040	2.00	21.205	2.06

Table 4.5: Full Evaluation Driving Statistics

As expected, the interesting statistical results occurred during the actual evaluation, when users were using the input methods to enter text. The voice group performed better in almost every category, having significantly faster completion times and average speeds. Speeding occurrences, driving too slow, spinouts and large accelerations were statistically insignificant between the groups based on t-tests. Perhaps the most surprising observation was that drivers using ThumbType had recognizably fewer collisions during the trial. They averaged less than a quarter of the number of collisions as the voice input drivers (1.1 to 4.6 per trial). A possible explanation for this result is that while entering text, ThumbType users are required to have both hands on the wheel. This potentially leads to better control of the automobile while driving and typing. More testing would be needed to validate this hypothesis, however.

From the statistical results and the exit survey conducted, however, it was determined that the low interruption objective was not achieved by this design. While no participant deemed ThumbType extremely dangerous to use, none felt as if it did not have any effect on their driving. Five of the ten participants indicated that use of the input device was moderately costly in terms of their performance; two indicated that it was very costly. Moreover, six of those eight claimed the device reduced their driving ability by about half. Time to complete the course and average speeds were lower than those of the voice input subjects as well.

On the other hand, three participants did not think text input with ThumbType was very costly, and this was supported by the lack of collisions for this group of participants. Overall, the interruption rating for this device is moderate-to-high (0.65). Reaction times for entering text were also moderate, averaging about 4.5 seconds between each entered character, compared to about 1 second for voice users. Comprehension was extremely high for the users of ThumbType. Questions were asked in the exit survey about the content of the text entry tasks. On average, subjects using this input method answered nine out of ten questions correctly, compared to a 65-percent accuracy for those using voice input. Table 4.3 summarizes the resulting IRC ratings in relation to the desired

ratings determined during the analysis phase.

Parameter	Desired	Result
Interruption	0.33	<b>0.65</b>
Reaction	0.82	<b>0.50</b>
Comprehension	0.85	1.0

One of the major hindrances in completing the usability study was finding a suitable test environment. The framework used in [20] would have been ideal, but had not been preserved for future use. In working with Holbrook and Moldenhauer<sup>1</sup>, a similar system was reconstructed, but the time and effort in doing so was substantial. Holbrook's survey tool is highly customizable, and could be a staple in any usability testing suite. Moldenhauer's driving simulator is more complex and is restricted to supporting specific types of evaluations. In this instance, however, making the software available would have expedited the overall process and allowed the developer to focus on system design rather than locating test materials.

**Weakness** Testing framework used to validate claims was not provided or described in the Library, but would be helpful to designers in their evaluation efforts.

The next section summarizes the strengths and weaknesses, such as the one above, that surfaced during the design experience.

## 4.4 Summary

A couple of observations became clear over the course of the design experience described above. The Claims Library has immense promise in both the reinforcement of ideas and, more importantly, in the transfer of knowledge. However, there is a lot of work that needs to be done in order for it to reach its potential.

The strengths summarized in Table 4.6 demonstrate that, in spite of its relatively short life span, the library can contribute in meaningful ways to system design. The lack of generalization cited in the previous chapter as a shortcoming from a theoretical point of view proved helpful in the initial analysis of the domain. Problems identified through research were concisely expressed in the form of design claims, reinforcing the developer's understanding of the domain and aiding in the visualization of potential solutions.

The theoretical strengths of the library were supported by the accuracy of claims returned during the design phases. The well-defined classification mechanism facilitated both the location of relevant

<sup>1</sup>Many thanks to Chuck and Max for their efforts in helping locate, configure, and run their software.

<b>Strengths</b>
Storing claims in their original state provides valuable insight into the analysis phase of design with little effort required on the part of the developer.
Search mechanism produced accurate claims in terms of relevance to the design problem, facilitating their reuse, and indicating the definition and utilization strong classification method.

Table 4.6: Strengths identified through design experience

claims, but also appears to have helped when claims were originally indexed. Developers are able to logically reason about a design in terms of the search criteria. Similarly, maintainers of the repository can easily recognize where data belongs that it can be later accessed and used.

All of the weaknesses (see Table 4.7) can be attributed in some way to the youth of the system. Every reuse technique must cross the mountain of knowledge acquisition, and the Claims Library has only reached the foothills. It is very tempting to populate the library with information without taking the time to first examine its caliber. This is the real challenge in acquiring knowledge – not achieving sheer numbers, but doing so without sacrificing quality for time. Sparsity of data is a temporary problem; inconsistency diminishes utility and prevents the repository from reaching its potential. Over time, as the system is used, superficial faults such as the inability to manipulate claims will be recognized and resolved. Just as the content inside the library is still growing, so is the interface.

<b>Weaknesses</b>
Inconsistencies in claim data (poorly chosen titles, lack of IRC ratings, missing theoretical backing and interclaim relationships) lead to confusion and insecurity about the information in the library.
Sparsity of relevant claims outside the realm of in-vehicle information systems indicates a lack of coverage in terms of tasks and design abstractions.
Interface provided no method of saving, comparing, or combining claims, making claim manipulation difficult.
Testing framework used to validate claims was not provided or described in the Library, but would be helpful to designers in their evaluation efforts.

Table 4.7: Weaknesses identified through design experience

Echoes of a lack of integration framework (from theoretical analysis) can be heard in the interface deficiencies, however. A conscious effort to support claim integration might have resulted in the inclusion of comparative tools or a development environment for reusing claims. Testing utilities might also have been a part of the overall integration framework, allowing designers to validate groups of claims through usability evaluations. The major anticipated weakness was the absence of truly abstract claims, however the closeness of the information in the library to the design problem made knowledge transference trivial.

Still, the analysis did not provide a distinct method of improving the library. Expecting to uncover a specific diagnosis for the problems in the library, it was unclear how they should be approached with further development efforts. For example, should resolving the sparsity of data precede the inclusion of an integration framework? Different observations in the theoretical analysis and practical experience provide contradicting evidence for prioritizing the important next steps. Additionally, it was unclear if problems observed were symptoms of larger problems or the causes thereof. In reflecting on the magnitude of the reuse literature that guided library development, the task of creating a sequential development plan proved daunting.

With this realization, we saw that a more important and immediate contribution would not be another iteration of the Claims Library design. Rather, a clarification of the underlying theory that would better inform future systems development seemed a more urgent and worthy use of our experience. This clarification would need to have several characteristics to include: composed of a staged or prioritized architecture, represents an ideal model grounded in literature, and possesses intermediate development objectives and assessment points. With this, we turn to the notion of a capability maturity model, which we are familiar with from our exposure to software development practice.

## Chapter 5

# Claims Library Capability

## Maturity Model

This chapter describes the creation of the Claims Library Capability Maturity Model (CL-CMM). In the tradition of other maturity models [40, 12, 6, 21], the CL-CMM is both a means of appraisal and a blueprint for improvement. The first section discusses the overall design of the model in terms of the organization of its subsequent parts. These parts are described in more detail in the sections following. The final section briefly illustrates the application of the CL-CMM, using the Claims Library as an example.

### 5.1 Creating the Claims Library Capability Maturity Model

In a sense, the previous two chapters constitute a *mediated* evaluation of the Claims Library. *Intrinsic* evaluation was performed by inspecting the library from a purely theoretical point of view. The *payoff* evaluation examined the library in practice, as a designer attempting to reuse the information it contains. Results from the intrinsic evaluation helped identify the strengths and weaknesses in the library’s design. Moreover, these results were influential in understanding observations made during the payoff evaluation; observations which also supplied valuable insight that could not be seen within a theoretical scope. Collectively, the ascertained strengths and weaknesses can be generalized to describe areas of concern that a claims library should address.

These areas of concern are referred to as “threads,” borrowing from the P-CMM [12]. Four threads were defined for the CL-CMM, each having some basis in both the theoretical and practical evaluations: data integrity, facilitating, data acquisition, and repository maintenance. *Data Integrity*

involves the verification and validation of claims; in other words, making sure that claims are classified correctly and that they are justified by some theoretical or empirical evidence. Ultimately, the value of reusable information lies in trusting that it has been tried and tested, so that it can be built upon with confidence. The theoretical analysis indicated a strong claim schema and classification structure, which both contribute to establishing data integrity. The practical evaluation, however, illustrated inconsistencies in the library’s data that caused confusion and hesitation on the part of the designer. Table 5.1 summarizes the purpose and foundation of this thread.

Theoretical Basis	Practical Basis
The theoretical analysis described the need for a well-defined claim structure and classification method. These provide a “succinct and expressive” [23] representation of a claim, making its information accessible to the designer, and thus more easily reused.	The practical analysis reinforced the importance of a good classification method, as accurate claims were returned using the search mechanism. However, inconsistencies in the data lead to confusion and insecurity about the claim. Missing IRC ratings, poorly-worded titles and descriptions, and absent theoretical backing and claim history, negatively impact the reuse process.
<b>Data Integrity</b> is concerned with clearly defining the information structure, language standards, and classification method, as well as validating claims via testing or theory.	

Table 5.1: Basis for the Data Integrity Thread of the CL-CMM.

A claims library should strive to facilitate the reuse of its information. *Facilitating Reuse* is synonymous with reducing cognitive distance. Regardless of the data quality, the inability to find and use this information negates its purpose. Proper implementation of Krueger’s taxonomy enables the identification and incorporation of reusable information. The theoretical analysis noted the lack of a true abstraction, a problem that caused weaknesses in each of the other facets of reuse, particularly specialization. With the current implementation of the library, designers are forced to work from specific instances of claims, requiring significant mental effort, where an abstract representation of a claim would reduce this effort. While not all of these weaknesses were apparent during the practical analysis (possibly because of the chosen design project), the lack of an integration framework stood out in this phase. Table 5.2 summarizes the purpose and foundation of this thread.

The cost of resolving the knowledge acquisition bottleneck is great for any reuse technique. It is important to efficiently solve this problem without sacrificing the quality of data for collection speed; thus the creation of the *Data Acquisition* thread. The theoretical analysis revealed that the “footprint” classification method supplies an excellent guide for knowledge acquisition. Along the same vein as Carroll’s concept of *task coverage*, where Norman’s Stages of Action are used to guide claim analysis, the “footprint” serves as a coverage matrix that can be quickly analyzed to determine areas shallow in content. The practical analysis noted the scarcity of claims outside

Theoretical Basis	Practical Basis
The theoretical analysis pinpointed the lack of a true abstraction as the potential cause for problems in later aspects of reuse. Abstract representations of information are important in reducing the cognitive distance imposed on the designer, a key problem facing any reuse paradigm. Not having any integration framework also contributes to an increase in cognitive distance, impacting how information is combined to form a new system design.	The practical analysis was quick to point out the need for a means of integrating and manipulating claims. Truly abstract claims did not seem to pose a problem, however, but this may have been because the chosen design problem was closely related to much of the information already in the repository.
<b>Facilitating Reuse</b> is accomplished by implementing the facets of Krueger’s taxonomy, focusing on the reduction of cognitive distance. Particular respect should be paid to abstraction, the dominant aspect of this taxonomy.	

Table 5.2: Basis for the Facilitating Reuse Thread of the CL-CMM

the domain of in-vehicle information systems, though it did not pose a problem for development. Table 5.3 summarizes the purpose and foundation of this thread.

Theoretical Basis	Practical Basis
All reuse techniques face a knowledge acquisition bottleneck early in development. However, the theoretical analysis revealed that the “footprint” classification method can be used to guide the collection of content to efficiently resolve this issue.	The practical analysis notes the scarcity of claims currently in the library (outside the domain of in-vehicle information systems).
<b>Data Acquisition</b> focuses on strengthening the knowledgebase effectively and efficiently.	

Table 5.3: Basis for the Data Acquisition Thread of the CL-CMM

Less obvious from the theoretical analysis is the need for some sort of *Repository Maintenance*. Krueger implies the need for a managing entity in describing his taxonomy, referring often to the “creator” or “designer” that implements each facet of reuse [23]. Sutcliffe is less subtle, going so far as to enumerate individual roles within the management of a claims library [39]. Inconsistencies in the data found during the practical analysis reinforce this notion. Table 5.4 summarizes the purpose and foundation of this thread.

Theoretical Basis	Practical Basis
Krueger implies and Sutcliffe explicitly states the need for some managing entity to control the definition and development of a reuse technique [23, 38].	Inconsistencies in the data contained in the Claims Library strongly echo the need for management processes.
<b>Repository Maintenance</b> focuses on the management and development of the library	

Table 5.4: Basis for the Repository Maintenance Thread of the CL-CMM

The Claims Library Capability Maturity Model employs a structure closely related to that found in the P-CMM, which in turn was originally derived from the SW-CMM (both discussed in Section 2.6). In terms of the CMMI framework for software-related processes, the CL-CMM will use a *staged representation*, grouping process areas within maturity levels. A staged representation was chosen in favor of a *continuous representation* due to the nature of the library. Within a staged representation, process areas grouped in a maturity level are aimed at a singular underlying goal, or focus. Maturity levels are ordered such that the attainment of one maturity level implies a basis for attaining subsequent levels. As a reuse mechanism, the Claims Library faces two major problems, described in Section 1.1: knowledge acquisition and cognitive distance. Lower levels of the CL-CMM concentrate on the knowledge acquisition bottleneck, while the higher levels attempt to enhance the data found in the library and improve the reuse process.

The five maturity levels of the CL-CMM are: Initial, Managed, Defined, Predictable, and Optimized. The terminology used here is borrowed directly from the P-CMM. Table 5.5 describes the focus and lists the process areas of each maturity level.

Maturity Level	Focus	Process Areas
Level 1: Initial	Inconsistent repository management and content	
Level 2: Managed	Structure and organize reusable information and define repository management practices	Claim Management Claim Verification Search Mechanism Claim Harvesting for Breadth Repository Planning
Level 3: Defined	Ensure data integrity and develop claim depth	Claim Validation Claim Generalization Claim Harvesting for Depth Organizational Training
Level 4: Predictable	Support development process and manage repository quantitatively	Testing Frameworks Claim Integration Environment Design Records Quantitative Repository Management
Level 5: Optimized	Foster innovation and collaboration	Establishing New HCI Theory Suggest Design Solutions Inter-organizational Collaboration Organizational Innovation and Deployment

Table 5.5: Each maturity level contains process areas across each of the threads described above.

While process areas are grouped within maturity levels, the P-CMM also uses threads to group process areas across maturity levels (see Figure 2.6), indicating how these concerns are continually addressed throughout development. Progress within each thread builds incrementally from lower maturity level to higher maturity level. Table 5.6 shows the CL-CMM process areas grouped by

both maturity level and thread.

	Data Integrity	Facilitating Reuse	Data Acquisition	Repository Maintenance
2	Claim Management Claim Verification	Search Mechanism	Claim Harvesting for Breadth	Repository Planning
3	Claim Validation	Claim Abstraction	Claim Harvesting for Depth	Organizational Training
4	Testing Frameworks	Claim Integration Environment	Design Records	Quantitative Repository Management
5	Establishing New HCI Theory	Suggest Design Solutions	Inter-organizational Collaboration	Organizational Innovation and Deployment

Table 5.6: CL-CMM Threads

The rest of this chapter provides a description of each maturity level and their associated process areas, goals, and practices. Goals are required components of the CL-CMM, meaning they are “essential contributors” [12] to the maturity of a claims library. Practices are expected components, meaning they only describe typical parts of a maturity model, intended to support process area goals. In each section, a maturity level will be described in terms of its focus. Then, each of the process areas within that maturity level will be discussed. (The Initial maturity level has no process areas, as all libraries begin at this level.) This discussion will consist of a statement of purpose, a description, and a summary of the goals and practices that were mentioned in the description. The goals from each process area can be used to assess whether or not a process area has been achieved. The practices that are provided are **suggested** practices based on the mediated evaluation of the library and the related research. For brevity, the practices are not described individually, but are rather explicated in the description of the process area.

## 5.2 Maturity Level 1: Initial

All claims libraries begin with an Initial maturity level. At this stage of development, the content of the library is very loosely organized. Data is inconsistent, and there are no complex classification mechanisms. Search tools for claim retrieval are limited to keyword input, if any exist at all. There are no explicit processes for maintaining the library, resulting in “ad hoc or chaotic” management. The library is probably only useful to experienced designers who are able to separate flawed data from sound data.

## 5.3 Maturity Level 2: Managed

The focus of the Managed maturity level is to begin laying the foundation for the content and maintenance of a claims library. The process areas within this maturity level are geared toward eliminating the knowledge acquisition bottleneck, while establishing certain standards for the data that will be stored in the repository. Additionally, defining the roles of the people needed to maintain the library and plan its development are essential activities at this level. When the process areas of this maturity level have been implemented, a claims library will have: 1) a claim schema and classification method, 2) a broad coverage of data, 3) a search mechanism allowing the access of claims based on the classification method, and 4) clearly defined roles and a plan for managing and cultivating the library.

The key process areas for this maturity level are: Claims Management, Claim Verification, Selection Mechanism, Claim Harvesting for Breadth, Repository Planning. They are described in more detail in the sections below.

### 5.3.1 Claims Management

**Purpose** The purpose of Claims Management is to define the structure of data stored in the repository, including a classification method and a standard vocabulary to be used within a claim, in order to establish consistency of data.

**Description** The establishment of a claim structure should be the initial focus in the development of a claims library. This structure will influence the overall development of the library, as it dictates how many of the process areas will operate. For example, the classification aspect of the claim structure will have a major impact on the Library Search Interface that designers use to locate reusable information. This classification method will also have some effect on the Data Measurement and Analysis process area, which uses that method to define data coverage.

Other process areas are even more closely tied to how a claim is structured. The Claim Verification process area relies on the standards established by this process area in order to determine how a claim should be verified. Process areas in subsequent maturity levels have similar dependencies. Claim Evolution Management will depend on a flexible structure that allows the recognition of relationships among claims within the repository. Claim Validation requires that the original claim schema allow for the linking of theory or testing results to a particular claim.

While the Claims Library for notification systems was used in creating this maturity model, it is intended to be general enough that other claims libraries can be built from it. The first aspect of this process area is defining the structure of a claim. The Claims Library used a modified version of

Sutcliffe's 14-point format described in Section 2.2. This structure was altered to capture information that pertains specifically to notification systems [33]. A developer for another claims library might design their own claim format, focusing on the aspects of a claim that pertain to a different type of system.

However, there are a few minimal requirements for this claim structure. A claim schema must contain the following fields: title, description, upsides, downsides, scenario of use, and theoretical basis. It is recommended that some form of Sutcliffe's format be used for the sake of completeness, but these pieces of information form the bare minimum for a claim within a claims library. The essence of a claim is captured in the description, upsides, downsides, and scenario of use. The title provides a short summary of what is described by the claim. The need for the theoretical basis of a claim will be discussed later, but this field is required so that data integrity can be ensured.

Additionally, the language used within these fields, particularly the description and title, should adhere to some standard. One of the major problems identified in the practical evaluation of the claims library was the vagueness of many claim titles ("Select while navigating"). Claim titles and descriptions were returned as the results of a search, and many designers would be confused about the content of a claim with such a title. Standards for claim wording should be defined in this process area.

Due to the focus on notification systems, the Claims Library uses a classification method that distinguishes between the primary and secondary tasks of the user. In another claims library, this distinction may not be necessary. Similarly, an IRC rating is used to describe the effect of a claim. Using this rating would not make much sense when developing a non-notification system. It is within this process area that the classification method is defined to meet the needs of the organization maintaining the library.

While the classification method is completely up to the designers of the library, there are a few guidelines that should be followed. The classification method should describe at least two aspects of a claim: its context and its effect. Moreover, the language used to describe the context and effect should be finite and well-defined. Using the Claims Library as an example, the Domain Theory provided a complete, concise vocabulary to describe primary and secondary tasks in the form of generic and generalized task models. These tasks represent context abstractly as part of the classification method. A typical claims library where there might not be a distinction between primary and secondary tasks, a single task field could be used, defined by a different task model, perhaps one from [3, 5, 45, 46].

The following is a list of goals and suggested practices for this process area.

### Summary of Goals and Practices

**Goal 1:** Define a Claim Standard

**Practice 1:** Obtain an understanding of the definition of a claim

**Practice 2:** Identify the minimum requirements for a claim

**Practice 3:** Define a method of categorization

**Practice 4:** Define commonly used terminology and a language standard

**Practice 5:** Provide a checklist to determine if a claim meets minimum requirements and standards

**Goal 2:** Monitor Submitted Claims

**Practice 6:** Provide a single means of receiving new claims

**Practice 7:** Use standard claim checklist to determine if claim should be approved for inclusion in the library

**Goal 3:** Provide Author Feedback

**Practice 8:** Track and provide claim evaluations to the author, whether or not claim was approved for the library

### 5.3.2 Claim Verification

**Purpose** The purpose of Claim Verification is to ensure that a claim meets the standards defined by the Claim Management process area and to verify that a claim is classified properly within the library.

**Description** Once the structure and classification methods have been established, as claims are authored, they should be verified before they are made available for reuse. This is the first step in achieving data integrity, one of the main threads of concern within the CL-CMM. Claim Verification ensures that claims are understandable to the designer and that they are classified correctly within the library. Reuse of claims depends on the ability for a designer to locate and identify potentially reusable information. If a claim was indexed improperly within the library, a designer might never find it, even though it may very well be the solution to all of his/her problems. Similarly, if a claim title or description contains vague or unclear language, a designer could possibly ignore the claim without ever seeing its full contents.

Verification of a claim's proper location within the repository requires an understanding of the classification method defined in the Claim Management process area, but may also require the use of special tools. For example, one of the fields of classification within the Claims Library is an IRC rating. In [11], Chewar describes a method for calculating IRC rating based on both empirical and analytical assessments. This method can be applied to an individual claim and will determine the appropriate rating. Not all claims within the Claims Library have been verified with this method, however, and as a result searching based on IRC is difficult, often yielding poor returns (from Section 4.4). Similarly, determining if a claim was assigned the appropriate primary and secondary tasks will require some form of assessment by people familiar with the Domain Theory task models. In summary, a claims library must define how one can clearly determine where a claim fits within the library.

### Summary of Goals and Suggested Practices

#### Goal 1: Prepare for Verification

**Practice 1:** Examine claim structure and classification mechanism

**Practice 2:** Establish verification criteria and procedures

#### Goal 2: Verify Claims

**Practice 3:** Perform verification

**Practice 4:** Analyze verification results and possibly identify corrective action

**Practice 5:** Perform corrective action, if needed

### 5.3.3 Selection Mechanism

**Purpose** The purpose of the Selection Mechanism process area is to define a method by which users can access information in the claims library.

**Description** The need for a selection mechanism is emphasized in [23] (see Section 2.5.2). This is crucial in supporting reuse, and it requires both an understanding of the data within the library *and* the means by which that data is organized. It involves creating the interface that designers will use to locate and compare information within the repository. The classification method described in Claim Management should provide the basis for creating a selection mechanism. In [37], Sutcliffe and Carroll point out the differences between richly-linked selection mechanisms, such as a hypertext system “link[ing] claims across multiple access paths,” and more efficient selection mechanisms based

on facets of the claim schema. The first has the potential to lead to an increasing amount of design knowledge, while the second narrows search results, reducing complexity and allowing designers to reach reusable data faster.

The Claims Library currently employs a mechanism similar to one also described in [37], whereby indexing is based on “generic models” thus “increas[ing] the scope of [a] claim’s potential reuse.” In this case, indexing is based on the classification method described above and in Section 2.5.1. This is an area of the Claims Library that should be examined in more depth, possibly focusing on the importance of a particular facet of the classification (is the primary task more important than the other search criteria?) or a comparison of various selection methods (hypertext search versus schema search).

### Summary of Goals and Practices

**Goal 1:** Define a selection mechanism

**Practice 1:** Understand claim classification method

**Practice 2:** Identify potential selection interface designs

**Practice 3:** Define retrieval process

**Goal 2:** Deploy selection mechanism

**Practice 4:** Implement selection mechanism

**Practice 5:** Provide interface for use by designers

### 5.3.4 Claim Harvesting for Breadth

**Purpose** The purpose of this process area is to aid in the population of data within a claims library so that coverage of the abstraction space is achieved.

**Description** The need for claim harvesting is described in [39], and it is the first step in addressing the knowledge acquisition problem. “Harvesting” refers to the active extraction of claims from existing system in order to populate a claims library. Ideally, claims will be submitted to a claims library by designers over the course of a system’s development. However, not only does this process take time, but the claims produced may not satisfy holes in library that need to be filled. In order to expedite the process, experts (known as “knowledge harvesters” in [39]) should actively pursue the creation of claim content recognized in existing systems that fill these holes.

What is meant by a “hole” in a claims library? At this maturity level, the collection of data should focus on something similar to what Carroll describes as “task coverage” [8], whereby claims are generated such that the range of all possible user tasks have an associated claim. The classification method described above should provide an  $n$ -dimensional space within which claims exist. Harvesting for Breadth implies that data exists in all possible locations within this space.

Using the Claims Library as an example, a four-dimensional space is used to classify information within the library. This space is made up of context descriptions (primary and secondary tasks and a design abstraction) and an IRC rating, which summarizes a claim’s effect. Harvesting for Breadth would result in locating claims with the expressed purpose of filling gaps within this space. If there are no claims related to a particular design abstraction, then claims would be gathered to fill this void.

Continuing this example, the range of IRC ratings proves a little more difficult. Should each possible value be accounted for in the library (e.g. claims with an I-value of 0.10 versus claims with an I-value of 0.11)? Or should there be a range of each category that is covered (e.g. a certain number of claims with I-values between 0-0.25 and the same number of claims for I-values between 0.26-0.5)? These are questions that should be addressed within this process area.

### Summary of Goals and Suggested Practices

#### Goal 1: Define Claim Content Analysis Technique

**Practice 1:** Establish objectives for the analysis

**Practice 2:** Specify the measures that will be taken

**Practice 3:** Specify analysis procedures

#### Goal 2: Provide Analysis Results

**Practice 4:** Collect content measurement data

**Practice 5:** Analyze content measurement data

**Practice 6:** Identify “holes” in claim content

#### Goal 3: Generate data to fill “holes”

**Practice 7:** Identify existing systems with potential filling material

**Practice 8:** Extract claims about the system

**Practice 9:** Verify that claims remedy lacking content areas

**Practice 10:** Add information to the repository

### 5.3.5 Repository Planning

**Purpose** The purpose of Repository Planning is to establish and define the processes involved in maintaining a claims library, including the definition of management roles and procedures.

**Description** This process area was derived from the Project Planning process area from the CMMI framework [40]. Within the CMMI, Project Planning refers to activities involved in managing a software development project. Similar activities are involved in the maintenance and development of a claims library, a complex entity that extends well beyond what its users are exposed to (claim content and search interface). This process area involves the backend of the repository, planning its overall development and describing the roles of the people involved in such an endeavor. For example, the Claim Harvesting for Depth process area mentions the need for a “knowledge harvester.” This is one of many roles that will need to be defined. It is important to delegate the various responsibilities involved in maintaining a claims library. It is recommended that HCI experts fulfill many of the management positions due to their knowledge of the domain.

Repository Planning also entails making decisions regarding the technology used to implement the library. Should the interface be web-based? Which backend database should be used for data storage? What resources are available for maintaining the library? These are important questions that the administrators of the system will need to address. Repository Planning should be used to formally describe the details of a library’s design from an implementation perspective. In a sense, it is a lot like managing a software project, therefore, many of the goals and practices from the CMMI’s Project Planning process area were borrowed for use here.

#### Summary of Goals and Suggested Practices

**Goal 1:** Establish estimates

**Practice 1:** Estimate the scope of the project

**Practice 2:** Establish estimates required tasks

**Practice 3:** Define estimates of effort and cost

**Goal 2:** Develop plan for the repository

**Practice 4:** Define repository goals

**Practice 5:** Establish budget and schedule

**Practice 6:** Plan for data management

**Practice 7:** Plan for project resources

**Practice 8:** Plan for needed knowledge and skills

**Practice 9:** Plan for stakeholder involvement

**Practice 10:** Establish repository plan

**Goal 3:** Institutionalize repository management process

**Practice 11:** Establish an organizational policy

**Practice 12:** Define roles in managing the repository

**Practice 13:** Assign responsibility

**Practice 14:** Identify and involve relevant stakeholders

**Practice 15:** Monitor and control the repository

## 5.4 Maturity Level 3: Defined

The solid foundation of structure and content provided by the Managed maturity level allow the Defined maturity level to improve the library in other areas. With an established claim schema and broad range of information, focus shifts toward ensuring the quality of data and deepening the content. Moreover, claims within the library are generalized, making the reuse of information easier on the designer. Management roles and practices are improved upon with the creation of tools and training materials. When the process areas of this maturity level have been implemented, a claims library will have: 1) well-structured claims that have been fully tested and validated, 2) broad and deep coverage of data, 3) a search mechanism that produces abstract claims that are easily reused in different contexts, and 4) a set of management practices codified in training material and supported by various toolsets.

The key process areas for this maturity level are: Claim Validation, Claim Abstraction, Claim Harvesting for Depth, Organizational Training. These process areas are discussed in more detail below.

### 5.4.1 Claim Validation

**Purpose** The purpose of Claim Validation is to increase the integrity of data in the library by supporting claims with proven theory or empirical testing.

**Description** This is the next level of the Data Integrity thread within the CL-CMM, building upon the Claim Management and Claim Verification process areas. These areas established a claim

structure and classification method, as well as a means of verifying that a claim was classified correctly. Claim Validation goes a step further and ensures that claims within the library are rooted in theory or empirical testing. This is the difference between an “identified” claim and an “attested” claim. To this point, claims in the library are merely “identified,” meaning that they have been observed in an artifact or exist as part of a scenario. The process of validating a claim involves transforming them into “attested” claims, meaning that they have been “empirically and theoretically substantiated and refined” [37].

This process area involves identifying the theoretical basis for each claim in the library, and making this distinction between “identified” and “attested.” Attaining Maturity Level 3 means that all of the claims within the library have been tried and tested according to one of the six categories described in [37]: theory prediction, grounded theory, experimentally derived, empirically grounded, proof of concept, or operationally grounded. These basically fall into two more general categories: theory-based and experiment-based. Claims should be shown to have some grounding in either proven theory or repeatable test results.

Claim Validation should be added as a minimal requirement for accepting a claim into the library. Claim authors must go through the extra step of testing their claims before they can be added to the collection, *and* some proof of these results should be included with the claim. Additionally, maintainers of the library should retroactively validate claims within the repository. This can be accomplished in several ways. The original claim author can be contacted to see if the claim had been validated in any way. Maintainers of the repository could run their own tests to validate a claim or set of claims. (If this approach is used, it is recommended that a group of claims is tested because these efforts are expensive.) Claim validators could research underlying theory that could be attributed to a claim in the library.

Claims that cannot be validated by testing or theory, or claims that are refuted by either, should not be used by designers. These claims should either be removed from the library or simply not included in search results. Authors should be allowed to perform testing in support of their claims, so it makes sense to save the claims rather than removing them completely.

### **Summary of Goals and Suggested Practices**

**Goal 1:** Define methods for validating claims

**Practice 1:** Support inclusion of empirical testing results

**Practice 2:** Support references to underlying theory

**Practice 3:** Define claim validation methods

**Practice 4:** Define invalidation procedure

**Goal 2:** Retroactively validate claims within the library

**Practice 5:** Identify groups of similar claims for validation

**Practice 6:** Identify groups of claims from the same author

**Practice 7:** Perform theory research, conduct testing, or contact author

**Practice 8:** Include validation results with claim or invalidate claim, if needed

**Goal 3:** Incorporate validation criteria into claim standard

**Practice 9:** Require testing or theoretical validation before including claim in repository

**Practice 10:** Store proof of validation with claim

### 5.4.2 Claim Abstraction

**Purpose** The purpose of this process area is to recognize the levels of abstraction that naturally exist within the library so that general forms of a claim can be recognized and more easily specialized.

**Description** The theoretical analysis of the Claims Library lead to the identification of two problems leading to the increase of cognitive distance. This process area exists within the thread of Facilitating Reuse, and focuses on the specialization aspect of Krueger’s taxonomy of reuse. The problem with specializing claims from the library is that claims are found in their specific state (or abstraction realization), and designers must make an extra mental leap when translating claims to a new domain. This extra leap involves, either implicitly or explicitly, creating a general form of the claim within the library and specializing it within the new context.

Wahid points out that one of the relationships that can exist between two claims is a generalizing-specifying, whereby one claim is a more specific version of another. Occurrences of this natural relationship should exist within the library at this stage of development. This process area involves identifying this relationship in order to aid developers in translating claims. If a designer selects a claim for reuse, the library should provide all claims which exist as more general forms of this original claim. Designers will be able to work from these general forms instead of having to formulate these themselves.

In addition to identifying this relationship, a process of claim factoring should also occur. Claim factoring is the process of assessing “the generalized contribution of a claim” [37], using a walk-through method based on Norman’s Stages of Action. Carroll and Sutcliffe recommend that HCI experts perform claim factoring, and that “claims analysis [be] allowed to progress through several

generations before factoring is attempted,” so that “immature knowledge” is not generalized before it is validated. Relying on the Claim Validation process area, validated claims are prime targets for factorization. The result of this process is the creation of a new, abstract claim (or claims) to be stored in the repository.

### Summary of Goals and Suggested Practices

**Goal 1:** Assign generalizing-specifying relationships to claims

**Practice 1:** Allow for inclusion of claim relationships

**Practice 2:** Identify pairs of claims with generalizing-specifying relationship in the library

**Practice 3:** Link claims via this relationship

**Goal 2:** Factor claims

**Practice 4:** Identify validated claim or claims for factorization

**Practice 5:** Perform walkthrough

**Practice 6:** Generate new abstract claim

**Practice 7:** Verify new claim’s location in repository

**Practice 8:** Link claim to original with generalizing-specifying relationship

### 5.4.3 Claim Harvesting for Depth

**Purpose** The purpose of this process area is to aid in the population of data within a claims library so that adequate depth of the abstraction space is achieved.

**Description** This process area builds on Claim Harvesting for Breadth from Maturity Level 2. At this stage of development, a claims library should have at least one claim in each position of the  $n$ -dimensional space defined by the classification method. Now the idea is to reach an acceptable depth at each of these positions in terms of the number of claims in each “bucket.” As more and more claims are reused within a claims library, it is to be expected that certain areas of the abstract space will grow deeper in number faster than others. It is to be expected that as claims are reused, the newly created claims will exist in the same or similar  $n$ -dimensional space, thus providing depth. Claims should be harvested for those areas lacking in depth.

**Summary of Goals and Suggested Practices****Goal 1:** Define Claim Content Analysis Technique**Practice 1:** Establish objectives for the analysis**Practice 2:** Specify the measures that will be taken**Practice 3:** Specify analysis procedures**Goal 2:** Provide Analysis Results**Practice 4:** Collect content measurement data**Practice 5:** Analyze content measurement data**Practice 6:** Identify shallow areas in claim content**Goal 3:** Generate data to fill shallow areas**Practice 7:** Identify existing systems with potential depth material**Practice 8:** Extract claims about the system**Practice 9:** Verify that claims remedy lacking content areas**Practice 10:** Add information to the repository**5.4.4 Organizational Training**

**Purpose** The purpose of Organizational Training is to define the procedures and techniques used in training people involved in maintaining a claims library.

**Description** This is another process area derived almost directly from the CMMI framework. This is the second process area devoted to the Repository Maintenance thread. It extends the ideas in Repository Planning by providing the appropriate training in the processes involved in maintaining a claims library. Roles were defined in Maturity Level 2, and at this point, it is likely that best-practices have evolved and tools have been developed for each of those roles. This process area is designated to formally training people involved in the maintenance of the library in their respective roles and to use the tools that have been created to facilitate their work.

**Summary of Goals and Suggested Practices****Goal 1:** Define best-practices and develop tools**Practice 1:** Enumerate and codify set of best-practices

**Practice 2:** Identify areas to be improved with tools

**Practice 3:** Design and implement tools

**Practice 4:** Document intended use of tools

**Goal 2:** Provide training

**Practice 5:** Identify strategic training needs

**Practice 6:** Establish training capability

**Practice 7:** Administer training

**Practice 8:** Assess training effectiveness

## 5.5 Maturity Level 4: Predictable

The trend in the first two maturity levels has been resolving the knowledge acquisition problem. The Predictable maturity level shifts gears, focusing on the improvement of design by reusing claims in the library. The entire design process is addressed, from the initial creation of claims to their testing to their incorporation into an overall design specification over multiple design iterations. Management processes quantitatively measure how the library affects the reuse process and steers library growth toward enhancing these aspects. When the process areas of this maturity level have been implemented, a claims library will have: 1) reusable testing frameworks associated with well-structured, fully-validated claims, 2) data that is not only broad and deep in terms of the classification method, but also span several design iterations, 3) an integration framework for manipulating abstract claims and incorporating them into new contexts, and 4) the extension of management practices to identify potential areas where the library can enhance the reuse process.

The key process areas for this maturity level are: Testing Frameworks, Claim Integration Environment, Design Records, Quantitative Repository Management. These process areas are discussed in more detail in the sections below.

### 5.5.1 Testing Frameworks

**Purpose** The purpose of the Testing Frameworks process area is to store and make available tools and processes for testing the validity of claims.

**Description** At this point in the maturation of a claims library, all of the claims within the library have been both verified and validated. It is now the responsibility of the maintainers of the library

to provide related testing frameworks to designers for the purposes of their own testing. This will serve two purposes. The first is that it will increase productivity on the part of the designer. Having a ready-made testing frameworks for claims that are reused within the library will expedite SBD iterations. Frameworks should be flexible enough that they can be applied in numerous contexts.

One example of this type of framework is the survey tool created by Holbrook, used in the usability study of ThumbType. Usability studies often consist of the following format: an entrance survey, test guidelines and instructions, and an exit survey. Holbrook's tool provides a configurable, wizard-style environment where the designer can specify all survey questions and test instructions, and the tool records them in comma-separated value (CSV) format.

This process area involves both linking claims to testing frameworks that were used to validate them and allowing designers to search for testing frameworks which fit their design situation.

### Summary of Goals and Suggested Practices

**Goal 1:** Store testing frameworks within library

**Practice 1:** Allow for the inclusion of testing frameworks

**Practice 2:** Define classification method for testing frameworks

**Practice 3:** Index frameworks based on classification method

**Practice 4:** Link testing frameworks to claims, when applicable

**Goal 2:** Provide search interface for locating testing frameworks

**Practice 5:** Create additional search interface for browsing frameworks

**Practice 6:** Include links to testing frameworks from claims

### 5.5.2 Claim Integration Environment

**Purpose** The purpose of the Claim Integration Environment process area is to facilitate reuse by providing a system whereby claims are easily incorporated into a new design.

**Description** One of the major problems identified by the theoretical analysis of the Claims Library was that there was no explicit means for integrating claims into a new design. Integration is the final part of Krueger's taxonomy, and is a key part of design *by* reuse. This process area builds upon the progress made within the thread of Facilitating Reuse.

From both the theoretical and practical analysis, the importance of scenarios during the reuse process is emphasized. When claims are reused, they either provide the basis for the creation of a

new scenario, and the claims gleaned from this new scenario link back to the original, reused claims in some way, or they are directly translated into individual claims that are then incorporated into a new scenario. Either way, scenarios appear to represent the integration framework for reusing claims, and a claims library needs to account for this.

This process area involves organizing claims based on the scenarios with which they are associated (typically multiple claims exist within a scenario). Designers should have the option to view results by scenario in addition to being able to recognize individual claims. Scenarios should contain links to or a list of the claims that they contain. Providing this type of information to the designer will aid in development of his/her own scenarios because he has an example to work from, thus reducing cognitive distance.

This also involves the ability of designers to store reusable claims in some way. During the practical evaluation of the Claims Library, one of the drawbacks was that claims could not be marked or saved in any way. Searches had to be repeated in order to find useful claims again. Claims did not exist in an easily accessible format, only textually within a webpage, so their information had to be copied directly from the page. Providing a claim in an easily manipulated format would help ease the integration process as well.

### **Summary of Goals and Suggested Practices**

#### **Goal 1:** Store scenarios individually

- Practice 1:** Define schema for recording scenarios
- Practice 2:** Define classification method for scenarios
- Practice 3:** Link scenarios to the multiple claims it contains

#### **Goal 2:** Allow searching and selection of scenarios

- Practice 4:** Create search mechanism for claims based on classification method
- Practice 5:** Provide links to related claims from scenarios
- Practice 6:** Provide links to related scenarios from claims

#### **Goal 3:** Provide tool for claim manipulation

- Practice 7:** Identify requirements for tool
- Practice 8:** Design tool for claim manipulation
- Practice 9:** Implement tool for claim manipulation
- Practice 10:** Integrate tool with claims library

### 5.5.3 Design Records

**Purpose** The purpose of the Design Records process area is to enhance the information stored in the Claims Library by storing entire system design records for viewing by designers.

**Description** This is the next process area within the Data Acquisition thread of concern for the CL-CMM. To this point, the library has been populated both in breadth and depth, and all of the data has been verified and validated along the Data Integrity thread. This process area involves storing entire design records for systems within the library. While claims encapsulate reusable design knowledge in its atomic form, design records may provide insight to designers that cannot be gleaned from a single claim or even a scenario of use. Design visualization tools such as CERVi [43], could be built directly into a claims library.

Additionally, design records should illustrate the evolution of claims throughout the design process. Claim relationships (described in Section 2.1.6) should be recognized across design iterations, and these relationships should be stored within the repository. Designers should also be able to search for the claims located at the deepest levels of these design iterations, adding a third dimension to the claims in the library (breadth, depth, and now evolutionary depth). Over the course of a design, claims about an artifact improve through testing and redefinition (task-artifact cycle). One could argue that the quality of claims increases with development, thus a designer should be able to specify a minimum evolutionary depth in his search results.

#### Summary of Goals and Suggested Practices

**Goal 1:** Provide means for designers to store entire system designs

**Practice 1:** Establish the concept of a design project

**Practice 2:** Define the schema for storing system designs

**Practice 3:** Link design records to claims in the repository

**Practice 4:** Design and implement visualization tool

**Goal 2:** Record evolutionary relationships among claims

**Practice 5:** Include means to capture evolutionary relationships

**Practice 6:** Link claims using these relationships, within and across projects

**Goal 3:** Allow search of claims based on evolutionary depth

**Practice 7:** Define classification of evolutionary depth

**Practice 8:** Include evolutionary depth in search criteria based on classification

### 5.5.4 Quantitative Repository Management

**Purpose** The purpose of this process area is to manage a claims library based solely on quantitative aspects.

**Description** The purpose of reuse is to save time and effort by not duplicating work that has already been done. Inherently, the benefits of reuse are hard to measure because they are a comparison to a process that was not performed (i.e. designing *without* reuse). However, as entire design projects are being stored within the library, it becomes easier to track when and how claims are being reused. This process area involves analyzing the library in order to understand how it is contributing to improving the cost of HCI software development. Repository management should be geared to maximize the positive trends identified by this analysis. For example, if studies indicate that the Claim Integration Environment process area expedites the development process, then the repository should focus its own growth on enhancing or maintaining this aspect.

#### Summary of Goals and Suggested Practices

**Goal 1:** Define analysis techniques and metrics

**Practice 1:** Establish objectives for the analysis

**Practice 2:** Specify the measures that will be taken

**Practice 3:** Specify analysis procedures

**Goal 2:** Provide Analysis Results

**Practice 4:** Collect measurement data

**Practice 5:** Analyze measurement data

**Practice 6:** Identify contributions to reuse process

**Goal 3:** Manage repository based on analysis

**Practice 7:** Define method of enhancing or maintaining aspect based on analysis

**Practice 8:** Implement method

**Practice 9:** Assess effectiveness of method

## 5.6 Maturity Level 5: Optimized

The Optimized maturity level represents the culmination of the development of a claims library. The focus of this level is the advancement of HCI design on a grand scale through collaboration and innovation. The library now contains a wealth of knowledge that should be explored and shared as a whole. The maintenance of the library is now able to focus on unique approaches to strengthening the entire system based on insight gathered over the course of the library's growth. When the process areas of this maturity level have been implemented, a claims library will have: 1) extrapolated HCI theory from the validated claims in the library, proven using various testing frameworks, 2) been interwoven with other repositories to create massive collections of data, 3) the ability to suggest entire design solutions based on designer criteria, and 4) experience-driven improvements to the underlying maintenance processes.

The key process areas for this maturity level are: Establishing New HCI Theory, Suggest Design Solutions, Inter-organizational Collaboration, Organizational Innovation and Deployment. These process areas are discussed in more detail in the sections below.

### 5.6.1 Establishing New HCI Theory

**Purpose** The purpose of this process area is to recognize the existence of new HCI theory encapsulated by the claims within a claims library.

**Description** Over the course of ensuring Data Integrity, claims have been validated and tested, potentially many times over. It is possible to recognize underlying trends among the claims within a library, thus leading to the identification and documentation of new HCI theory. This is a complex task involving expert analysis of claims with respect to theories of psychology, information visualization, cognitive science, computer science and other related fields. Once potential theory has been identified within the library, it should be validated, possibly through the use of testing frameworks linked to the involved claims.

#### Summary of Goals and Suggested Practices

**Goal 1:** Propose new HCI theory

**Practice 1:** Identify trends among claims in the library

**Practice 2:** Develop hypothesis centering around a group of claims

**Practice 3:** Research current theories which may imply or support the hypothesis

**Practice 4:** Propose new HCI theory based on research and trends

**Goal 2:** Validate theory through testing

**Practice 5:** Design battery of tests to prove new theory

**Practice 6:** Incorporate appropriate testing frameworks from library, if needed

**Practice 7:** Conduct testing

**Practice 8:** Analyze results

### 5.6.2 Suggest Design Solutions

**Purpose** The purpose of this process area is to reduce cognitive distance by suggesting entire design solutions based on criteria specified by designers.

**Description** Suggest Design Solutions is the final process area along the Facilitating Reuse thread. Krueger notes one of the ways in which cognitive distance can be reduced is through the automated mapping between abstraction specification and abstraction realization [23]. In other words, once a potential system is defined in the abstract terms of the reuse technique, the degree to which that technique helps in creating the final product affects cognitive distance. To this point, the process areas within the CL-CMM have improved the individual aspects of reuse; providing a search mechanism (selection), abstracting claims (specialization), and supplying a claim integration framework (integration). The purpose of this process area is to take designer criteria and, using sophisticated analysis techniques, suggest a set of claims as an entire design solution. The Design Record process area from Maturity Level 4 also helps in this process. Algorithms for determining design solutions may be based on the complete design records that are stored in the repository.

#### Summary of Goals and Suggested Practices

**Goal 1:** Develop algorithm for suggesting solutions

**Practice 1:** Examine design records contained in the repository

**Practice 2:** Establish criteria for defining a system based on examination of other systems

**Practice 3:** Propose algorithm for suggesting solutions

**Practice 4:** Test algorithm

**Goal 2:** Suggest solutions based on designer criteria

**Practice 5:** Provide interface for collecting criteria

**Practice 6:** Process criteria based on algorithm

**Practice 7:** Return results

### 5.6.3 Inter-organizational Collaboration

**Purpose** The purpose of this process area is to leverage the knowledge contained in other claims libraries.

**Description** The claims library evaluated in this thesis was specific to the realm of notification systems. Ideally, others will use this maturity model in creating their own claims libraries, perhaps focusing on a different domain or even encapsulating the whole of HCI design. At this stage of maturity, the maintainers of a claims library should seek out other mature libraries in an effort to combine their knowledge. While not calling for the full integration of two libraries, involving the merging of claim schemas and classification methods, providing a common search interface for tapping into multiple repositories would be sufficient in achieving Inter-organizational Collaboration. Organizations should work together to define and implement a collaboration strategy.

#### Summary of Goals and Suggested Practices

**Goal 1:** Define collaboration strategy

**Practice 1:** Identify potential collaboration opportunities

**Practice 2:** Compare library structure and design

**Practice 3:** Define strategy for collaboration based on comparison

**Goal 2:** Implement collaboration strategy

**Practice 4:** Define requirements for collaborative system based on the strategy

**Practice 5:** Design and implement the system

### 5.6.4 Organizational Innovation and Deployment

**Purpose** The purpose of Organizational Innovation and Deployment is to define and implement incremental and innovative improvements to the repository.

**Description** Another process area adapted directly from the CMMI framework, Organizational Innovation and Deployment focuses on the overall enhancement of a claims library through innovative approaches to achieving the goals defined in the Repository Planning process area. The infrastructure that has been established throughout the Repository Maintenance thread has resulted in the definition and understanding of the underlying management processes on qualitative and quantitative levels. This understanding allows for the identification of improvements that can be made to the repository, and this process area involves the proposal and implementation of such solutions.

### Summary of Goals and Suggested Practices

**Goal 1:** Select improvements

**Practice 1:** Collect and analyze improvement proposals

**Practice 2:** Identify and analyze innovations

**Practice 3:** Pilot improvements

**Practice 4:** Select improvements for deployment

**Goal 2:** Deploy improvements

**Practice 5:** Plan the deployment

**Practice 6:** Manage the deployment

**Practice 7:** Measure improvement effects

This concludes the description of the CL-CMM. The next section provides an example of how the CL-CMM might be applied to a claims library.

## 5.7 Using the CL-CMM

There are many different appraisal methods based on capability maturity models. The Standard CMMI Appraisal Method for Process Improvement (SCAMPI) is the “most rigorous” method applied to the CMMI framework [40]. The document describing the P-CMM mentions four different appraisal methods, including one that was developed over the course three years, aptly named the People CMM-Based Assessment Method [12].

The CL-CMM needs more refinement before a full-fledged appraisal method can be based on it. Such refinement takes more time and thorough investigation of claims libraries, both of which are outside the scope of this thesis. However, to illustrate how an appraisal method could be based

on the CL-CMM, one of the P-CMM methods, known as Gap Analysis, has been implemented and included here. “A gap analysis is an organizational analysis that examines the organization’s workforce activities against a benchmark standard, and identifies the gaps or shortcomings” [12]. In this case, the CL-CMM supplies the “benchmark standard” against which a claims library is measured.

This type of appraisal is intended to be a “self-assessment” led by someone familiar with the CL-CMM, but conducted by someone familiar with the claims library in question. Gap analysis charts are created, like the one shown in Table 5.7, for each process area. One column describes how each suggested practice is currently implemented, if at all. The second column is used to annotate recommended methods of improving or implementing a practice, based on the current state of affairs.

The example table (part of Appendix A) shows the gap analysis chart for the Claim Management process area, filled out for the Claims Library. In many cases, a missing practice results in the absence of subsequent practices. For example, providing author feedback cannot occur if claims are not monitored as they are added to the library. As the Claim Management process area falls within the second maturity level, and it has not been satisfied, the Claim Library is still only at maturity level 1. For the P-CMM, a gap analysis is not thorough enough to officially determine maturity levels. A People CMM-Based assessment or joint assessment are required, and often take months of preparation and analysis.

It is not certain whether a gap analysis will be sufficient for appraising a claims library. Considering the current scale of the Claims Library, a gap analysis is perhaps more than adequate, however, as the library grows in content and functionality, more thorough investigation may be needed. This is representative of the entire Claims Library Capability Maturity Model; this document provides a substantial beginning, but time is needed for it to more fully develop. Along these lines, the next chapter summarizes the work contained in this document and identifies areas of future work.

Claim Management	Current State	Proposed Improvements
<b>P1</b> Obtain an understanding of the definition of a claim	Understand the definition of a claim is exemplified in the creation of a claim schema based on Sutcliffe's 14-point format	None
<b>P2</b> Identify the minimum requirements for a claim	The minimum requirements for a claim are implied in the required fields in the database, but are never explicitly indicated	Define the minimum requirements for a claim; Are a title, a description, upsides, and downsides enough? Is a scenario also required?
<b>P3</b> Define a method of categorization	Categorization is accomplished using the "footprint" representation of a claim	None
<b>P4</b> Define commonly used terminology and a language standard	Commonly used terminology is exhibited in the task models and set of design abstractions that are employed. Language standards do not exist, as witnessed in inconsistencies in claim titles, descriptions, and other textual fields	Define and employ language standards for claim fields
<b>P5</b> Provide a checklist to determine if a claim meets minimum requirements and standards	No checklist exists due to lack of explicit definition of minimum requirements and standards	Once requirements and standards are clearly defined, generate checklist
<b>P6</b> Provide a single means of receiving new claims	System provides a common interface for an author to enter claims, but claims are entered directly to the database, without being checked	Add intermediate step of checking claims after they are submitted
<b>P7</b> Use standard claim checklist to determine if claim should be approved for inclusion in the library	Claims are not checked when they are submitted to the repository	Once received claims are checked and claim checklist is created, use to stamp claims for approval
<b>P8</b> Track and provide claim evaluations to the author, whether or not claim was approved for the library	Claims are not evaluated, so author feedback cannot be provided	Provide author feedback after evaluating claim using checklist Make checklist available to authors so they can perform self-checking before they submit

Table 5.7: Example use of the Gap Analysis chart for the Claim Management process area, filled out for the Claims Library for notification systems.

## Chapter 6

# Conclusions

“Our knowledge is the amassed thought and experience of innumerable minds” [15]. It is interesting how Emerson’s words are able to summarize the work contained in this document so well. Thought is the purely intellectual mode of understanding the world, while experience is the result of living and interacting with it. Intrinsic and pay-off evaluations? The dichotomy of theory and practice? Thought influences how the world is experienced, while experience changes how the world is thought about. The task-artifact cycle? Claims? And what is *knowledge*? For Emerson, it is the collection of the thoughts and experiences of “innumerable minds” – a claims library?

The reuse of HCI design *knowledge* may be the key to reducing the cost of usability testing. Building from “amassed thought and experience” avoids redundancy and inspires innovation; design and testing are given a head start and can progress faster. However, the challenge lies in cultivating a rich repository of *knowledge*. How should such an entity grow and develop? How should information be located and reused? This thesis has started to answer these important questions through the following contributions:

- Instantiation of a claims library as an interpretation of the literature
- A theoretical analysis of the Claims Library based on Krueger’s taxonomy of reuse
- A practical analysis of the Claims Library from the point of view of a designer, resulting in the creation of an in-vehicle input device for a notification system
- A Claims Library Capability Maturity Model based on the two evaluations and in support of the assessment and development of claims libraries

The generality of the CL-CMM extends its utility far beyond the Claims Library for notification systems to other such repositories, making it a greater contribution to the advancement of HCI.

## 6.1 Future Work

While this thesis provides a strong beginning, there is more work that needs to be done. The CL-CMM was created as a proof-of-concept from the perspective of one person based on the evaluation of one library. In contrast, the capability maturity model for software engineering was founded on decades of research and the experiences of thousands of software developers. Further refinement of this model is needed, but will have to come over time as the Claims Library evolves, adapts, and is used.

The following is a summary of impending future work:

- Applying the model to multiple libraries to provide insight into how the model could be refined
- Verifying the CL-CMM through design process in industry
- Identifying value-adding features and ensuring they are emphasized by the CL-CMM
- Creating administrative tools in library to assist process area attainment
- Looking at the application of the CL-CMM to systems beyond claims libraries

Additionally, the Claims Library has already begun to morph into the LINK-UP system [10], with changes that extend the functionality beyond merely locating claims. It would be interesting to see how closely related these new developments are to the process areas within the CL-CMM, showing whether the natural needs that were identified to improve the system were also accounted for in this model.

All of the repositories examined in continued investigation will need to be used in real design projects. In some cases, these projects should push the envelope of cross-domain reuse in a sort of stress test of the library. The purpose of a claims library is to improve the efficiency and quality of HCI software. Demonstrations are needed to show how their use can positively impact the design process. At the same time, these experiences will reveal strengths and weaknesses in the libraries and in the CL-CMM.

On a broader scale, there is potential for the creation of a capability maturity model for usability engineering (UE). Capability maturity models typically apply to much broader areas, encompassing entire disciplines. The current CMMI framework integrates process improvement for software engineering, systems engineering, and integrated product and process design. Developing a usability engineering capability maturity model and including it in the CMMI framework would help establish interface design as a key aspect in the development process.

## Appendix A

# CL-CMM Gap Analysis Charts

Claim Management	Current State	Proposed Improvements
<b>P1</b> Obtain an understanding of the definition of a claim		
<b>P2</b> Identify the minimum requirements for a claim		
<b>P3</b> Define a method of categorization		
<b>P4</b> Define commonly used terminology and a language standard		
<b>P5</b> Provide a checklist to determine if a claim meets minimum requirements and standards		
<b>P6</b> Provide a single means of receiving new claims		
<b>P7</b> Use standard claim checklist to determine if claim should be approved for inclusion in the library		
<b>P8</b> Track and provide claim evaluations to the author, whether or not claim was approved for the library		

Table A.1: Gap Analysis Chart for Process Area 2.1: Claim Management

<b>Claim Verification</b>	<b>Current State</b>	<b>Proposed Improvements</b>
<b>P1</b> Examine claim structure and classification mechanism		
<b>P2</b> Establish verification criteria and procedures		
<b>P3</b> Perform verification		
<b>P4</b> Analyze verification results and possibly identify corrective action		
<b>P5</b> Perform corrective action, if needed		

Table A.2: Gap Analysis Chart for Process Area 2.2: Claim Verification

<b>Search Mechanism</b>	<b>Current State</b>	<b>Proposed Improvements</b>
<b>P1</b> Understand claim classification method		
<b>P2</b> Identify potential selection interface designs		
<b>P3</b> Define retrieval process		
<b>P4</b> Implement selection mechanism		
<b>P5</b> Provide interface for use by designers		

Table A.3: Gap Analysis Chart for Process Area 2.3: Search Mechanism

Claim Harvesting for Current State		Proposed Improvements
Breadth		
<b>P1</b>	Establish objectives for the analysis	
<b>P2</b>	Specify the measures that will be taken	
<b>P3</b>	Specify analysis procedures	
<b>P4</b>	Collect content measurement data	
<b>P5</b>	Analyze content measurement data	
<b>P6</b>	Identify “holes” in claim content	
<b>P7</b>	Identify existing systems with potential filling material	
<b>P8</b>	Extract claims about the system	
<b>P9</b>	Verify that claims remedy lacking content areas	
<b>P10</b>	Add information to the repository	

Table A.4: Gap Analysis Chart for Process Area 2.4: Claim Harvesting for Breadth

<b>Repository Planning</b>	<b>Current State</b>	<b>Proposed Improvements</b>
<b>P1</b> Estimate the scope of the project		
<b>P2</b> Establish estimates required tasks		
<b>P3</b> Define estimates of effort and cost		
<b>P4</b> Define repository goals		
<b>P5</b> Establish budget and schedule		
<b>P6</b> Plan for data management		
<b>P7</b> Plan for project resources		
<b>P8</b> Plan for needed knowledge and skills		
<b>P9</b> Plan for stakeholder involvement		
<b>P10</b> Establish repository plan		
<b>P11</b> Establish an organizational policy		
<b>P12</b> Define roles in managing the repository		
<b>P13</b> Assign responsibility		
<b>P14</b> Identify and involve relevant stakeholders		
<b>P15</b> Monitor and control the repository		

Table A.5: Gap Analysis Chart for Process Area 2.5: Repository Planning

Claim Validation	Current State	Proposed Improvements
<b>P1</b> Support inclusion of empirical testing results		
<b>P2</b> Support references to underlying theory		
<b>P3</b> Define claim validation methods		
<b>P4</b> Define invalidation procedure		
<b>P5</b> Identify groups of similar claims for validation		
<b>P6</b> Identify groups of claims from the same author		
<b>P7</b> Perform theory research, conduct testing, or contact author		
<b>P8</b> Include validation results with claim or invalidate claim, if needed		
<b>P9</b> Require testing or theoretical validation before including claim in repository		
<b>P10</b> Store proof of validation with claim		

Table A.6: Gap Analysis Chart for Process Area 3.1: Claim Validation

Claim Abstraction	Current State	Proposed Improvements
<b>P1</b> Allow for inclusion of claim relationships		
<b>P2</b> Identify pairs of claims with generalizing-specifying relationship in the library		
<b>P3</b> Link claims via this relationship		
<b>P4</b> Identify validated claim or claims for factorization		
<b>P5</b> Perform walkthrough		
<b>P6</b> Generate new abstract claim		
<b>P7</b> Verify new claim's location in repository		
<b>P8</b> Link claim to original with generalizing-specifying relationship		

Table A.7: Gap Analysis Chart for Process Area 3.2: Claim Abstraction

<b>Claim Harvesting for Depth</b>	<b>Current State</b>	<b>Proposed Improvements</b>
<b>P1</b> Establish objectives for the analysis		
<b>P2</b> Specify the measures that will be taken		
<b>P3</b> Specify analysis procedures		
<b>P4</b> Collect content measurement data		
<b>P5</b> Analyze content measurement data		
<b>P6</b> Identify shallow areas in claim content		
<b>P7</b> Identify existing systems with potential depth material		
<b>P8</b> Extract claims about the system		
<b>P9</b> Verify that claims remedy lacking content areas		
<b>P10</b> Add information to the repository		

Table A.8: Gap Analysis Chart for Process Area 3.3: Claim Harvesting for Depth

<b>Organizational Training</b>	<b>Current State</b>	<b>Proposed Improvements</b>
<b>P1</b> Enumerate and codify set of best-practices		
<b>P2</b> Identify areas to be improved with tools		
<b>P3</b> Design and implement tools		
<b>P4</b> Document intended use of tools		
<b>P5</b> Identify strategic training needs		
<b>P6</b> Establish training capability		
<b>P7</b> Administer training		
<b>P8</b> Assess training effectiveness		

Table A.9: Gap Analysis Chart for Process Area 3.4: Organizational Training

Testing Frameworks	Current State	Proposed Improvements
<b>P1</b> Allow for the inclusion of testing frameworks		
<b>P2</b> Define classification method for testing frameworks		
<b>P3</b> Index frameworks based on classification method		
<b>P4</b> Link testing frameworks to claims, when applicable		
<b>P5</b> Create additional search interface for browsing frameworks		
<b>P6</b> Include links to testing frameworks from claims		

Table A.10: Gap Analysis Chart for Process Area 4.1: Testing Frameworks

Claim Integration Environment	Current State	Proposed Improvements
<b>P1</b> Define schema for recording scenarios		
<b>P2</b> Define classification method for scenarios		
<b>P3</b> Link scenarios to the multiple claims it contains		
<b>P4</b> Create search mechanism for claims based on classification method		
<b>P5</b> Provide links to related claims from scenarios		
<b>P6</b> Provide links to related scenarios from claims		
<b>P7</b> Identify requirements for tool		
<b>P8</b> Design tool for claim manipulation		
<b>P9</b> Implement tool for claim manipulation		
<b>P10</b> Integrate tool with claims library		

Table A.11: Gap Analysis Chart for Process Area 4.2: Claim Integration Environment

Design Records		Current State	Proposed Improvements
<b>P1</b>	Establish the concept of a design project		
<b>P2</b>	Define the schema for storing system designs		
<b>P3</b>	Link design records to claims in the repository		
<b>P4</b>	Design and implement visualization tool		
<b>P5</b>	Include means to capture evolutionary relationships		
<b>P6</b>	Link claims using these relationships, within and across projects		
<b>P7</b>	Define classification of evolutionary depth		
<b>P8</b>	Include evolutionary depth in search criteria based on classification		

Table A.12: Gap Analysis Chart for Process Area 4.3: Design Records

Quantitative Repository Management		Current State	Proposed Improvements
<b>P1</b>	Establish objectives for the analysis		
<b>P2</b>	Specify the measures that will be taken		
<b>P3</b>	Specify analysis procedures		
<b>P4</b>	Collect measurement data		
<b>P5</b>	Analyze measurement data		
<b>P6</b>	Identify contributions to reuse process		
<b>P7</b>	Define method of enhancing or maintaining aspect based on analysis		
<b>P8</b>	Implement method		
<b>P9</b>	Assess effectiveness of method		

Table A.13: Gap Analysis Chart for Process Area 4.4: Quantitative Repository Management

	Establishing New HCI Theory	Current State	Proposed Improvements
<b>P1</b>	Identify trends among claims in the library		
<b>P2</b>	Develop hypothesis centering around a group of claims		
<b>P3</b>	Research current theories which may imply or support the hypothesis		
<b>P4</b>	Propose new HCI theory based on research and trends		
<b>P5</b>	Design battery of tests to prove new theory		
<b>P6</b>	Incorporate appropriate testing frameworks from library, if needed		
<b>P7</b>	Conduct testing		
<b>P8</b>	Analyze results		

Table A.14: Gap Analysis Chart for Process Area 5.1: Establishing New HCI Theory

	Suggest Design Solutions	Current State	Proposed Improvements
<b>P1</b>	Examine design records contained in the repository		
<b>P2</b>	Establish criteria for defining a system based on examination of other systems		
<b>P3</b>	Propose algorithm for suggesting solutions		
<b>P4</b>	Test algorithm		
<b>P5</b>	Provide interface for collecting criteria		
<b>P6</b>	Process criteria based on algorithm		
<b>P7</b>	Return results		

Table A.15: Gap Analysis Chart for Process Area 5.2: Suggest Design Solutions

<b>Inter-organizational Collaboration</b>		<b>Current State</b>	<b>Proposed Improvements</b>
<b>P1</b>	Identify potential collaboration opportunities		
<b>P2</b>	Compare library structure and design		
<b>P3</b>	Define strategy for collaboration based on comparison		
<b>P4</b>	Define requirements for collaborative system based on the strategy		
<b>P5</b>	Design and implement the system		

Table A.16: Gap Analysis Chart for Process Area 5.3: Inter-organizational Collaboration

<b>Organizational Innovation and Deployment</b>		<b>Current State</b>	<b>Proposed Improvements</b>
<b>P1</b>	Collect and analyze improvement proposals		
<b>P2</b>	Identify and analyze innovations		
<b>P3</b>	Pilot improvements		
<b>P4</b>	Select improvements for deployment		
<b>P5</b>	Plan the deployment		
<b>P6</b>	Manage the deployment		
<b>P7</b>	Measure improvement effects		

Table A.17: Gap Analysis Chart for Process Area 5.4: Organizational Innovation and Deployment

# Bibliography

- [1] GREGORY D. ABOWD AND ELIZABETH D. MYNATT. Charting past, present, and future research in ubiquitous computing. *ACM Trans. Comput.-Hum. Interact.*, 7(1):29–58, 2000.
- [2] DOUG A. BOWMAN, CHRISTOPHER J. RHOTON, AND MARCIO S. PINHO. Text input techniques for immersive virtual environments: An empirical comparison. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 2002.
- [3] J. BREUKER AND W. VAN DER VELDE. *CommonKADS Library for expertise modelling*. IOS Press, 1994.
- [4] LEE BUTTS AND ANDY COCKBURN. An evaluation of mobile phone text input methods. In *Third Australasian conference on User interfaces*, pages 55–59. Australian Computer Society, Inc., 2002.
- [5] S. K. CARD, T. P. MORAN, AND A. NEWELL. *The psychology of human computer interaction*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1983.
- [6] SOFTWARE ENGINEERING INSTITUTE CARNEGIE MELLON UNIVERSITY. The software acquisition capability maturity model (sa-cmm). Online at <http://www.sei.cmu.edu/arm/SA-CMM.html>, March 2003.
- [7] J. M. CARROLL AND W. A. KELLOGG. Artifact as theory-nexus: hermeneutics meets theory-based design. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 7–14. ACM Press, 1989.
- [8] JOHN M. CARROLL AND MARY BETH ROSSON. Getting around the task-artifact cycle: how to make claims and design by scenario. *ACM Transactions on Information Systems*, 10(2):181–212, 1992.
- [9] JOHN M. CARROLL, MARK K. SINGLEY, AND MARY BETH ROSSON. Integrating theory development with design evaluation. *Behavior and Information Technology*, 11:247–255, 1992.
- [10] C. M. CHEWAR, EDWIN BACHETTI, D. SCOTT MCCRICKARD, AND JOHN E. BOOKER. Automating a design reuse facility with critical parameters: Lessons learned in developing the LINK-UP systems. In *Proceedings of the Conference on Computer-Aided Design of User Interfaces (CADUI)*, Funchal, Island of Madeira, Portugal, 2004.
- [11] C. M. CHEWAR, D. SCOTT MCCRICKARD, AND ALISTAIR G. SUTCLIFFE. Unpacking critical parameters for interface design: Evaluating notification systems with the IRC framework. In *Accepted in Proceedings of the ACM Conference on Designing Interactive Systems (DIS '04)*, Boston, MA, 2004.
- [12] BILL CURTIS, WILLIAM E. HEFLEY, AND SALLY A. MILLER. People Capability Maturity Model: Version 2.0. Technical report, Software Engineering Institute, July 2001.

- [13] RICARDO DE ALMEIDA FALBO, GIANCARLO GUIZZARDI, AND KATIA CRISTINA DUARTE. An ontological approach to domain engineering. In *Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pages 351–358. ACM Press, 2002.
- [14] LIESBETH DUSINK AND JAN VAN KATWIJK. Reuse dimensions. In *Proceedings of the 1995 Symposium on Software reusability*, pages 137–149. ACM Press, 1995.
- [15] RALPH WALDO EMERSON. *Letters and Social Aims*. J. R. Osgood, 1876.
- [16] MASAOKI FUKUMOTO AND YASUHITO SUENAGA. “fingering?": a full-time wearable interface. In *Conference companion on Human factors in computing systems*, pages 81–82. ACM Press, 1994.
- [17] ERICH GAMMA, RICHARD HELM, RALPH JOHNSON, AND JOHN VISSIDES. *Design Patterns*. Addison-Wesley, 1995.
- [18] NASIB S. GILL. Reusability issues in component-based development. *SIGSOFT Software Engineering Notes*, 28(4):4–4, 2003.
- [19] NAMBU HIROTAKA. Reassessing current cell phone designs: using thumb input effectively. In *CHI '03 extended abstracts on Human factors in computing systems*, pages 938–939. ACM Press, 2003.
- [20] CHUCK HOLBROOK. Input methods for notification systems: A design analysis technique with a focus on input for dual-task situations. Master’s thesis, Virginia Polytechnic Institute and State University, 2003.
- [21] SHIHONG HUANG AND SCOTT TILLEY. Towards a documentation maturity model. In *Proceedings of the 21st annual international conference on Documentation*, pages 93–99. ACM Press, 2003.
- [22] MICHAEL JACKSON. Problem analysis using small problem frames. In *Proceedings of WOFACS '98*, volume 22, pages 47–60, March 1999.
- [23] CHARLES W. KRUEGER. Software reuse. *ACM Computing Surveys*, 24(2):131–183, 1992.
- [24] E. LOMBARD. Le signe de l’elevation de la voix. *Annals Maladiers Oreille, Larynx, Nez, Pharynx*, 37:101–119, 1911.
- [25] KENT LYONS, THAD STARNER, DANIEL PLAISTED, JAMES FUSIA, AMANDA LYONS, AARON DREW, AND E. W. LOONEY. Twiddler typing: one-handed chording text entry for mobile phones. In *Proceedings of the 2004 conference on Human factors in computing systems*, pages 671–678. ACM Press, 2004.
- [26] EDGAR MATIAS, I. SCOTT MACKENZIE, AND WILLIAM BUXTON. Half-qwerty: a one-handed keyboard facilitating skill transfer from qwerty. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 88–94. ACM Press, 1993.
- [27] D. SCOTT MCCRICKARD, RICHARD CATRAMBONE, C. M. CHEWAR, AND JOHN T. STASKO. Establishing tradeoffs that leverage attention for utility: empirically evaluating information display in notification systems. *International Journal of Human-Computer Studies*, 58:547–582, 2003.
- [28] D. SCOTT MCCRICKARD, C. M. CHEWAR, JACOB P. SOMERVELL, AND ALI NDIWALANA. A model for notification systems evaluation—Assessing user goals for multitasking activity. *Transactions on Computer-Human Interaction*, 10(4):312–338, December 2003.

- [29] MAXIM MOLDENHAUER AND D. SCOTT MCCRICKARD. Effect of information modality on geographic cognition in car navigation systems. In *Proceedings of the IFIP TC.13 Conference on Human-Computer Interaction (INTERACT)*, Zurich, Switzerland, 2003.
- [30] BRAD MYERS. Challenges of hci design and implementation. *interactions*, 1(1):73–83, 1994.
- [31] DONALD A. NORMAN. Cognitive engineering. In *User Centered System Design: New Perspectives on Human Computer Interaction*, Donald A. Norman and Stephen W. Draper, editors, pages 31–62. Lawrence Erlbaum Associates, 1986.
- [32] SHARON OVIATT. Multimodal system processing in mobile environments. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 21–30. ACM Press, 2000.
- [33] CATHERINE PAYNE, C. F. ALLGOOD, C. M. CHEWAR, CHUCK HOLBROOK, AND D. SCOTT MCCRICKARD. Generalizing interface design knowledge: Lessons learned from developing a claims library. In *Proceedings of the 2003 IEEE International Conference on Information Reuse and Integration (IRI)*, 2003.
- [34] MARY BETH ROSSON AND JOHN M. CARROLL. *Usability Engineering: Scenario-Based Development of Human-Computer Interaction*. Morgan Kaufman Publishers, 2002.
- [35] M. SCRIVEN. *Perspectives of Curriculum Evaluation*, pages 39 – 83. Rand McNally, 1967.
- [36] MIIKA SILFVERBERG, I. SCOTT MACKENZIE, AND PANU KORHONEN. Predicting text entry speed on mobile phones. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 9–16. ACM Press, 2000.
- [37] A. G. SUTCLIFFE AND J. M. CARROLL. Designing claims for reuse in interactive systems design. *Int. J. Hum.-Comput. Stud.*, 50(3):213–241, 1999.
- [38] ALISTAIR SUTCLIFFE. On the effective use and reuse of hci knowledge. *ACM Trans. Comput.-Hum. Interact.*, 7(2):197–221, 2000.
- [39] ALISTAIR SUTCLIFFE. *The Domain Theory: Patterns for Knowledge and Software Reuse*. Lawrence Erlbaum Associates, 2002.
- [40] CMMI PRODUCT TEAM. Cmmi for systems engineering, software engineering, integrated product and process development, and supplier sourcing: Staged representation. Technical report, Software Engineering Institute, March 2002.
- [41] LOUIS TIJERNA, EDWIN PARMER, AND MICHAEL J. GOODMAN. Driver workload assessment of route guidance system destination entry while driving: A test track study. In *Proceedings of the 5th ITS World Congress, Seoul, Korea, October 12-16, 1998*, Seoul, Korea, October 1998.
- [42] WILLIAM P. WAGNER. Issues in knowledge acquisition. In *Proceedings of the 1990 ACM SIGBDP conference on Trends and directions in expert systems*, pages 247–261. ACM Press, 1990.
- [43] SHATAB WAHID, C. F. ALLGOOD, C. M. CHEWAR, AND D. SCOTT MCCRICKARD. Entering the heart of design: Relationships for tracing claim evolution. In *Proceedings of the Sixteenth International Conference on Software Engineering and Knowledge Engineering*, June 2004.
- [44] DAVID J. WARD, ALAN F. BLACKWELL, AND DAVID J. C. MACKAY. Dasher – a data entry interface using continuous gestures and language models. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 129–137. ACM Press, 2000.

- [45] R. WEHREND AND C. LEWIS. A problem-oriented classification of visualization techniques. In *First IEEE Conference on Visualization*, pages 139–143, Los Alamitos, CA, 1990.
- [46] M. X. ZHOU AND S. K. FEINER. Visual task characterization for automated visual discourse synthesis. In *Human Factors in Computing Systems: CHI 98*, C.-M. Karat, A. Lund, J. Coutaz, and J. Karat, editors, pages 392–399, New York, 1998. ACM, ACM Press.