A Trust Framework for Autonomic Computing Systems

By

Priyanka Agrawal
B.I.S., G.G.S. Indraprastha University, 2004

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTERS OF SCIENCE

In the Department of Computer Science

SUPERVISORY COMMITTEE

A Trust Framework for Autonomic Computing Systems

By

Priyanka Agrawal
B.I.S., G.G.S. Indraprastha University, 2004

Supervisory Committee

Dr. Hausi A. Müller, Supervisor (Department of Computer Science)

Dr. Daniela Damian, Department Member (Department of Computer Science)

Dr. Margaret-Anne Storey, Department Member (Department of Computer Science)

Dr. Ahmed E. Hassan, External Examiner (Department of Electrical and Computer Engineering)

Dr. Hausi A. Müller, Supervisor (Department of Computer Science)

Dr. Daniela Damian, Department Member (Department of Computer Science)

Dr. Margaret-Anne Storey, Department Member (Department of Computer Science)

Dr. Ahmed E. Hassan, External Examiner (Department of Electrical and Computer Engineering)

ABSTRACT

Present-day IT environments are complex, heterogeneous tangles of hardware, middleware and software from multiple venders that are becoming increasingly difficult to integrate, install, configure, tune and maintain. In order to combat this increasing level of complexity, automating many of the functions associated with computing today seems to be a reasonable solution. IBM, inspired by the autonomic nervous system of the human body which regulates without any conscious intervention, chose to call this paradigm Autonomic Computing—computing using adaptive and self-managing systems with minimal human intervention. Autonomic computing poses several research challenges. In an endeavor to hide complexity, autonomic systems give up, to a certain extent, accountability to the user. Consequently, autonomic systems exhibit fewer cause and effect relationships and therefore engender trust and adoption issues. In other words, the system itself takes over control whereby it may or may not operate as per user expectations during its operation. Our goal is to develop a framework of trust that will be useful for developers of autonomic computing applications or self-managed systems dealing with trust issues. Our approach gathers key trust topics, issues, nomenclature, taxonomies, and user task models from the literature which are then distilled and pruned to form our own trust framework which is intended to aid developers in the design of self-managed systems. We then use IBM's Tivoli provisioning system, which is one of the most successful autonomic systems, to consolidate our framework. Finally, we evaluate our framework by trying to identify the strengths and weaknesses with respect to trust of self-managed systems by performing a case study on a non-deployed autonomic e-Commerce prototype.

Table of Contents

SUPERVISORY COMMITTEE ii

ABSTRACT iii

Table of Contents iv

List of Tables viii

List of Figures ix

Acknowledgements x

Dedication xi

Chapter 1: Introduction 1

1.1 Introduction to Autonomic Computing 1

1.2 Problem Statement 4

1.3 Motivation 5

1.4 Approach 7

1.5 Thesis Overview 9

Chapter 2: Related Work 10

2.1 Introduction 10

2.2 Communication and Interaction to Build Trust 10

2.3 Understanding Task Models in Adaptive Systems 13

2.4 Autonomic Computing and Trust 16

2.4.1 The Correlation between the Process of Development of Trust in Autonomic Computing and Autonomic Computing itself 17

2.4.2 Measuring trust in Autonomic Computing 18

2.4.3 Mapping Autonomic Computing maturity levels to different levels of trust 19

2.4.4 Trust in Automation 21

2.4.5 Existing Autonomic Applications 23

2.5 Autonomic applications 23

2.5.1 Tivoli—A Deployed Autonomic System 23

2.5.2 The E-Marketplace Application—A Non-deployed Prototype of an Autonomic System 24
2.5.2.1 Architecture 24
2.5.2.2 Autonomous E-shopping 26

2.6 Concerns Affecting Trust in e-Commerce 27

2.7 Chapter Summary 29

Chapter 3: Trust Framework 30

3.1 Defining and Categorizing Trust 30

3.1.1 Classification on the basis of Type 31
3.1.1.1 Simple Trust 31
3.1.1.2 Blind Trust 32
3.1.1.3 Authentic Trust 32

3.1.2 Classification on the Basis for Development 32
3.1.2.1 History based source of trust 33
3.1.2.2 Personal disposition 33
3.1.2.3 Shared category membership 34
3.1.2.4 Role behaviour 34

3.1.3 Classification on the basis of definition 35
3.1.3.1 Building on the existing categories from Lee 37
3.1.3.2 Introducing New Trust Categories 39

3.2 Chapter Summary 41

Chapter 4: Type and Nature of Information Exchange to Build Trust 42

4.1 Type of Information 44

4.1.1 Self-Configuration 44

4.1.2 Self-Optimization 45

4.1.3 Self-Healing 45

4.1.4 Self-Protection 45

4.2 Nature of Information Exchange 46

4.2.1 User expectations 50
4.2.1.1 Functionality and goal-oriented performance assessment 50
4.2.1.2 Modus operandi 51
4.2.1.3 User-perceived state 53
4.2.1.4 Quality of Service 55

4.2.2 Intention and willingness to act on tasks and services 58
4.2.2.1 Service awareness – tracking tasks or services 58
4.2.2.2 Context awareness – event detection 59
4.2.2.3 Resource awareness – support and maintenance of services 61

4.2.3 Users' belief and confidence 63

4.2.3.1 User environment 63
4.2.3.2 Context awareness 64
4.2.3.3 Configuration details 64
4.2.3.5 Status indicators 65
4.2.3.6 Data review 67
4.2.3.7 Visibility of information and tracking of processes 69
4.2.3.8 Recourse and undo 69
4.2.3.9 Trial runs 70

4.2.4 User and task behaviour *71*
4.2.4.1 User-attributes 71
4.2.4.2 Task-specific preferences 72
4.2.4.3 Supplier information 74
4.2.4.4 Preferences relative to QoS 75
4.2.4.5 Specification 76

4.2.5 Reducing uncertainty in the mind of the user *77*
4.2.5.1 Alternative suppliers to support a service 78
4.2.5.3 User task-specific preferences 79
4.2.5.4 Changes in environment 79
4.2.5.5 Error messages 80
4.2.5.6 Unforeseen circumstances with recommendations 81

4.3 Development of the trust framework **82**

4.4 Evaluation of Tivoli **84**

4.5 Chapter Summary **86**

Chapter 5: Evaluation **87**

5.1 Trust Framework with respect to Autonomic Systems **87**

5.1.1 User expectations and autonomic systems *87*

5.1.2 Willingness to act and autonomic systems *88*

5.1.3 User belief and autonomic systems *88*

5.1.4 User behaviour and autonomic systems *88*

5.1.5 Reducing uncertainty and autonomic systems *89*

5.2 Nature and Type of Information Exchange with respect to the E-Marketplace Application **89**

5.2.1 User expectations *89*
5.2.1.1 Functionality and goal-orientated performance assessment 89
5.2.1.2 Modus operandi 90
5.2.1.3 User perceived state 90
5.2.1.4 Quality of Service 91

5.2.2 Intention and willingness to act on tasks and services *91*
5.2.2.1 Service awareness 91
5.2.2.2 Context awareness 91
5.2.2.3 Resource awareness 91

5.2.3 Users' belief and confidence *92*
5.2.3.1 User environment 92
5.2.3.2 Context awareness 93
5.2.3.3 Configuration details 93

5.2.3.4 Status Indicators 93
5.2.3.5 Data review 94
5.2.3.6 Visibility of information and tracking of processes 94
5.2.3.7 Recourse or undo 95
5.2.3.8 Trial runs 96

5.2.4 User behaviour *96*
5.2.4.1 User-attributes 96
5.2.4.2 Task-specific preferences 97
5.2.4.3 Supplier information 97
5.2.4.4 Preference relative to QoS 97
5.2.4.5 Specification 98

5.2.5 Reducing uncertainty in the mind of the user *99*
5.2.5.1 Alternative suppliers to support a service 99
5.2.5.2 QoS 99
5.2.5.3 User task-specific preferences 99
5.2.5.4 Changes in environment 99
5.2.5.5 Error messages 100
5.2.5.6 Unforeseen circumstances with recommendations 100

5.3 Evaluation of the E-Marketplace application **100**

5.4 Chapter Summary **102**

Chapter 6: Conclusions **103**

6.1 Summary **103**

6.2 Contributions **105**

6.3 Limitations **106**

6.4 Future Work **107**

Bibliography **108**

List of Tables

Table 2.1 List of design modules and descriptions..... 25

Table 4.1 Trust framework in autonomic systems.....46

Table 4.2a Level of trust and nature and type of information exchange to build trust with autonomic systems for the attitude or expectation category 47

Table 4.2b Level of trust and nature and type of information exchange to build trust with autonomic systems for the intention or willingness to act category 48

Table 4.2c Level of trust and nature and type of information exchange to build trust with autonomic systems for the belief and confidence category 48

Table 4.2d Level of trust and nature and type of information exchange to build trust with autonomic systems for the user behaviour category.....49

Table 4.2e Level of trust and nature and type of information exchange to build trust with autonomic systems for the uncertainty reduction category 49

Table 4.3 Development of the trust framework by studying Tivoli.....83

Table 4.4 Evaluation of Tivoli with respect to our trust framework 85

Table 5.1 Evaluation of E-Marketplace application with respect to our trust framework.....101

List of Figures

Figure 1.1 The autonomic manager and elements in an autonomic system 3
Figure 1.2 Autonomic computing properties 4
Figure 1.3 The exploratory approach to research.....8

Figure 2.1 An autonomic trust element.....17
Figure 2.2 The relationship among calibration, resolution, and automation capability in defining appropriate trust in automation.....22

Figure 3.1: Framework of trust.....31
Figure 3.2 Pictorial representation of trust framework for autonomic systems based on definitions.....36

Figure 4.1 Tivoli Store Manager: health monitor.....51
Figure 4.2 Tivoli Storage Manager: the restore feature with Hints and Tips..... 53
Figure 4.3 Tivoli Storage Manager: setting preferences and filters to suit user perceived state.....53
Figure 4.4 Tivoli Storage Manager: the system status feature depicts expected refresh rate..... 55
Figure 4.5 Tivoli Storage Manager: the system status feature depicts the state of the system 59
Figure 4.6 Tivoli Storage Manager: the restore feature depicts file hierarchy.....61
Figure 4.7 Tivoli Storage Manager: the restore feature communicates resource availability.....63
Figure 4.8 Tivoli Storage Manager: the system status 66
Figure 4. 9 Tivoli Storage Manager: with message boxes communicating current state 67
Figure 4.10 Tivoli Storage Manager: displaying data already entered.....68
Figure 4.11 Tivoli Storage Manager: feature that is communicates user attributes 72
Figure 4.12 Tivoli Storage Manager: task specific preferences.....74
Figure 4.13 Tivoli Storage Manager: preferences with respect to QoS.....76

Figure 5. 1 User interface for the Market Manager Agent in the E-Marketplace application 92
Figure 5.2 User interface for Seller Agent in the E-Marketplace application.....94
Figure 5.3 User interface for Seller Agent in the E-Marketplace application.....95
Figure 5.4 User interface for Buyer Agent in the E-Marketplace application 96
Figure 5.5 User interface for Buyer Agent in the E-Marketplace application..... 98

Acknowledgements

A journey is easier when you travel together. Interdependence is certainly more valuable than independence. This thesis is the result of work whereby I have been accompanied and supported by many people. It is satisfying to now have the opportunity to express my gratitude to them.

First and foremost, I would like to extend my deepest sense of gratitude and reverences to my supervisor, Dr. Hausi Müller, for his guidance, encouragement, moral and financial support throughout this research. I cherish and am grateful for the opportunity that he provided me with to pursue graduate studies under his supervision. Thank you for fostering such a healthy research environment and for extending your concern towards me not only as a student but also as a person. This had made me feel so much more comfortable in a foreign country.

I am also grateful to all the Rigi research group members who would almost endlessly lend an ear to me when I would have doubts or issues with my research. I would like to acknowledge their help for providing me with doses of reality, constructive criticism and up-front remarks, suggestions on how to improve my thesis, narrow my research domain and expose areas that I may need to explore to defend myself.

I would be failing in my duty if I do not express my gratitude towards my sister Bhawna and brother-in-law Shishir in Victoria who steadily stood by me during every phase of life during my entire stay in Victoria. I would not be where I stand today, had they not been there for me. Also, words cannot say how deeply I am indebted to Dr. Ashok Agarwal, Physics Department, University of Victoria and his family for their generosity and help.

Last but not the least, I take the privilege of thanking my affectionate parents, with whose blessings and unending love, the completion of this thesis has been possible. Thank you for your understanding, support, and stable influence. Thank you to my sister Muskan in India, for always believing in me and making me feel special and to my niece Aashi for the joy and love she has given to our family.

I am also thankful to my friends for always remaining in contact and for expressing their concern for me across the miles through phone calls, chat, and email while I was working. Thank you to everyone else, who kept asking me right from the start: "Is your Masters Degree over yet? How much longer will you take?" which motivated me in many visible and invisible ways.

Dedication

For my parents

Chapter 1: Introduction

"There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult." — C.A.R. Hoare

1.1 Introduction to Autonomic Computing

Present-day IT environments are complex, heterogeneous tangles of hardware, middleware and software from multiple venders that are becoming increasingly difficult to integrate, install, configure, tune and maintain [Keph2005b]. Computing systems' complexity appears to be approaching the limits of human capability, yet the march towards increased interconnectivity and interconnection rushes ahead unabated [KeCh2003].

Even if there are enough skilled people, as computing evolves, the complicated web of interacting applications in a typical IT environment overwhelms the decision making capabilities of human administrators. Thus, the very benefits information technology aims to provide seem to be threatened. As introduced by IBM's Senior Vice-President of Research, Paul Horn, at Harvard University in a keynote address in March 2001, such a scenario inevitably points towards the need to automate many of the functions associated with computing today. IBM chose to call this notion of automated and self-managing software systems requiring minimal human intervention *Autonomic Computing*. The word autonomic stems from the nervous system that acts involuntarily as reflexes to self-configure, self-optimize, self-heal, self-protect the biological system in human beings. Analogously, autonomic software contributes to its own management by taking appropriate actions when sensing problems and adjusting to various circumstances just as human reflexes do for the body.

Autonomic systems operate on a set of policies or rules and are thus effective with respect to mundane and repetitive tasks [THLM2003]. Also computer systems are

good at record keeping and maintaining a knowledge base. Hence there are many benefits to automating tasks and having systems being capable of self-management.

While the term autonomic computing was coined by IBM in 2001, the concepts of adaptive and self-managing systems have been discussed in the literature for many years [Sous2005]. The autonomic computing literature has grown at an amazing rate since the seminal papers by IBM researchers appeared in January 2003 [KeCh2003, AHPR2003, GaCo2003, HIPD2003, RMDN2003].

IBM has developed an architecture blueprint for autonomic computing [KeCh2003, IBM2006]. According to this blueprint, an autonomic element consists of an autonomic manager and a managed element with an underlying control loop which is signified by the grey ellipse as depicted in Figure 1.1 below. A collection of interacting autonomic elements or even a single autonomic element constitutes an autonomic system.

An autonomic manager in turn consists of a monitor, an analyzer, a planner, and an execution engine operating on a knowledge base (Figure 1.1). The interface between the autonomic manager and the managed element comprises sensors and effectors. The data exchanged across this interface is event-based and standardized so that a single autonomic manager can control multiple resources or a single resource can be controlled by different mangers. Policies are either hard-wired into the autonomic manager or injected into an autonomic element through its own effectors [Bowe2005]. The monitoring process is responsible for collecting data from the sensors as well as filtering and organizing data for storage in the knowledge base. The analysis engine detects symptoms by analyzing the accumulated data and the planner devises corrective actions which are executed through the effectors (i.e., tuning parameters or executing scripts).

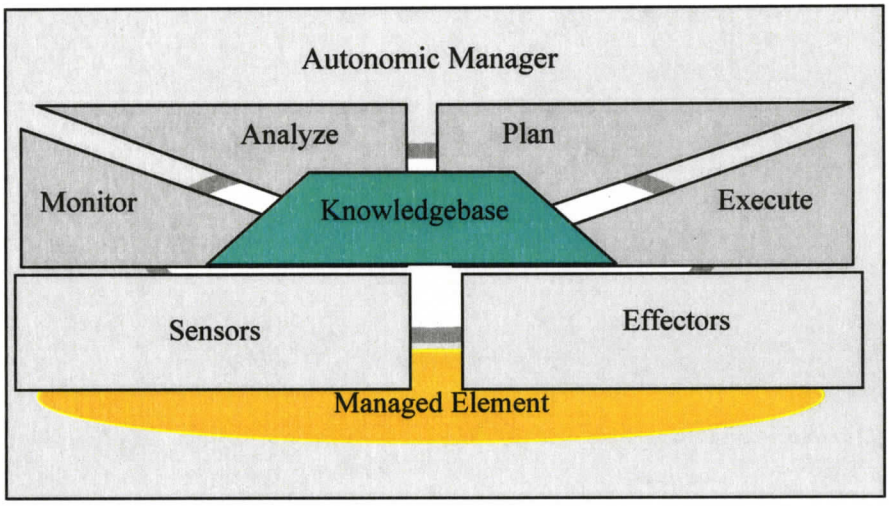

Figure 1.1: Autonomic element

To be autonomic, a system must know itself as well as its boundaries and environment, manage itself (e.g., configure and reconfigure itself, continually optimize itself, recover or heal from malfunction, or protect itself), and function in a heterogeneous world—while keeping its complexity hidden from the user [MuMy2004]. Note that an autonomic application need not satisfy all of the characteristics simultaneously. For example, IBM concentrates on the following four self-management functions: self-configuration, self-healing, self-optimization and self-protection. Self-configuration provides increased responsiveness by adapting dynamically to changing environments. Self-healing systems deliver resiliency by discovering, diagnosing and preventing disruptions. Operational efficiency is achieved by tuning resources and balancing workloads with self-optimizing systems. Information and resources are secured with self-protecting systems by anticipating and protecting against attacks.

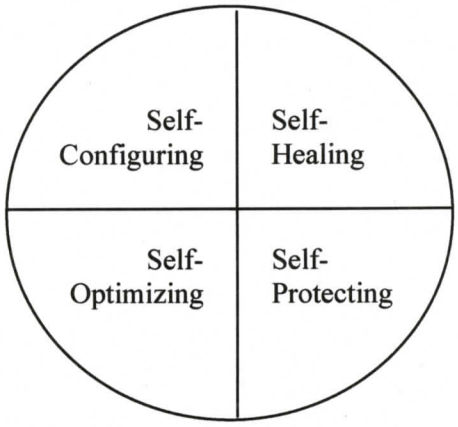

Figure 1.2: Autonomic computing properties

1.2 Problem Statement

Autonomic computing has gained widespread attention over the last three years for its vision of developing applications with autonomic or self-managing behaviours [Keph2005b]. Various approaches for the design and implementation of autonomic systems have emerged, including the use of goal policies [KeWa2004], utility functions [KeWa2004], intelligent monitoring, data mining, reinforcement learning, and planning. Important engineering concerns arise related to security, privacy and trust [CSAW2005, Keph2005, Keph2005b]. In autonomic computing, the system takes full or partial control over itself and thereby potentially alienating the user during its operation. Consequently, autonomic systems lead to trust and adoption issues since the system no longer displays cause and effect relationships [Keph2005b, RMDN2003]. In other words, autonomic systems in an endeavour to hide complexity, give up accountability to a certain extent. Non-autonomic systems have a relatively direct connection between the users' commands and the actions taken by the software [KAMK2005]. Russel *et al.* rightly argue that autonomic systems, in an endeavour to hide complexity, appear like ghosts in that action and decisions are taken mysteriously without the knowledge of the user [RMDN2003]. Chan *et al.* state that the autonomic approach to solve complexity increases the administrators' skepticism towards automation—how is an administrator to believe that an autonomic system will help his or her systems perform better

[CSAW2005]? Hence, engendering trust in an autonomic system is a hard problem. Much attention has been devoted to the importance but lack of transparency in an autonomic system's decision process [LMRW2005, GaCo2003] and the need for an autonomic system to know itself [HIPD2003]. An autonomic software system lives in a dynamic environment where autonomic elements interact and collaborate with each other and their environment. Hence, in order to reap the benefits of autonomic computing systems, building trust with autonomic systems is a grand challenge and an important issue.

1.3 Motivation

Trust is one of the most desirable qualities in any close relationship [ReHZ1985]. Significant benefits can accrue from trust in business relationships [LeWe1985] and software systems today play a vital role in businesses, particularly e-Commerce applications. As a matter of fact, modern systems depend on trust [Gidd1991] and autonomic systems are rightly an integral part of modern software systems. Researchers have stressed the fact that it is important to build trust in order to automate computing functionalities in particular and increasing trust will in turn allow a further increase in autonomicity in existing autonomic systems [LoLi2002]. Newly deployed autonomic applications on the other hand will be more readily accepted and adopted if trust issues are addressed. In fact, the issue of trust is in many ways at the core of self-managing systems, and the issue is quite large and complex [LMRW2005]. There is not only a need, but also an absence of mechanisms, to develop a rapport with autonomic systems [BMKB2005]. In this thesis, we propose a framework for building 'trustworthy' autonomic applications that will hopefully help developers with their design choices. The concern is over building autonomic systems that are usable [BMKB2005]. Norman clearly states how feedback is an important element of usability [Norm1998]. Therefore, having systems that are usable refers back to the need to have interaction between the user and the system, thus directly pointing at accountability. Trust directly relates to accountability. To understand this, consider the following scenarios:

Scenario 1:

"You are requested to provide your bank details to person X."

Scenario 2:

"You are requested to provide your bank details to the account manager for verification purposes. The information provided by you shall be kept confidential and any losses incurred due to leakage of information will be made good."

In these scenarios who would you trust when providing your bank details?

A logical answer to this would be: Scenario 2 because some degree of accountability has been provided to you. It seems to be a natural human tendency to trust any person or system that is 'accountable' or in other words provides us with the relevant information in the domain of concern. Hence, we can say that accountability and trust are complementary and interwoven and thus go hand in hand [AHPR2003].

Conversely, building trust should also facilitate reliance upon information received from the system about uncertain environmental states and their accompanying outcomes in risky situations [ScHT1973]. Established trust leads to positive interpretation of others' behaviour, commitment, open communication, greater flexibility, learning and knowledge sharing, and satisfaction. Hence, established trust is influenced by trust itself [Falc2004]. Telford *et al.* states that earning trust of users such as Data Base Administrators in an autonomic system like DB2 will allow the application to be 'more' autonomic [THLM2003]. Users who trust an autonomic system will more readily give out information to the system for execution and will be more responsive towards the system input. This in turn will allow autonomic systems to react more appropriately and accordingly in unforeseen situations and therefore add to the knowledgebase and policies to use in similar situations in the future.

Trust fosters the willingness of one person to increase his or her vulnerability to the actions of another person whose behaviour could not be controlled [Zand1972]. Ideally, in autonomic systems developed trust should encourage the user to 'let go' and rely on the system's decisions and actions during the course of its operation. Trust in autonomic systems shall facilitate their ready acceptance by users and foster confidence in their competence, thereby rightly achieving the goal that IBM has set forth—automation with minimal human intervention. Also, since autonomic systems aim at reducing complexity, as stated by Egger *et al.,* trust may help reduce complexity when there is incomplete information about other parties, promoting risk taking among consumers [Egge2001].

Building tools and techniques to instill trust should be an integral part of the development of an autonomic system. Although a lot of work has been done on the issues of trust, little has been done in the field of trust that specifically applies to autonomic computing and autonomic systems; and hence the motivation for this work.

1.4 Approach

We advocate information exchange and communication as a tool to build trust with autonomic systems. Using our trust framework as a canvas, we discuss how the theory of communication helps build trust with software systems. In our thesis, we advocate an exchange of information between the user and the autonomic application in order to build trust incrementally. Improved interaction will allow autonomic systems to be more autonomous, exhibiting increased initiative without losing the users' trust. Higher levels of trust and usability should in turn lead to improved adoptability. This thesis addresses the trust problem by developing an in-depth understanding of trust and applying the understanding to autonomic systems thereafter.

We follow an exploratory approach for our research. In an exploratory approach, usually little is known of the subject under study at the outset of the research [Mode2004]. The system under study is investigated from different viewpoints, and the researcher, in the process, gains 'insights' of the subject under study as depicted in Figure 1.3. Hence the researcher deepens his or her understanding. This method may therefore reveal valuable new aspects about the subject. In our case, the insight led to a trust framework for autonomic computing systems. We look at trust from different viewpoints—from the way trust is defined to how trust is formed. We also review the literature on trust to gather information on key trust topics, issues, nomenclatures and taxonomies as a basis for our research.

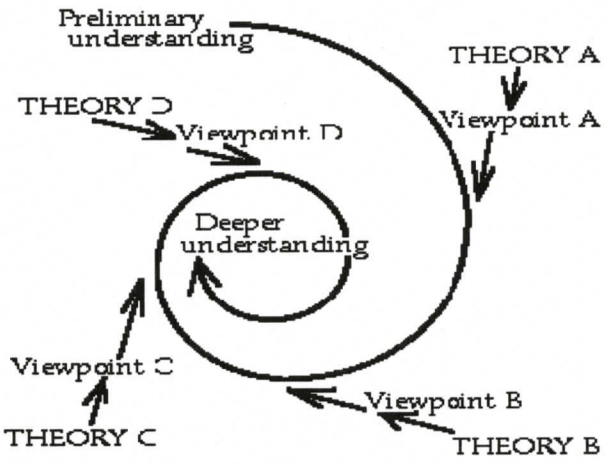

Figure 1.3: Exploratory research approach
Courtesy: http://www2.uiah.fi/projects/metodi/177.htm

A framework is a simplified description of a complex entity or process. In our case, the complex entity was trust itself. Our framework provided a simplified description of trust by:

Presenting a nomenclature (naming conventions) of trust;
Presenting a taxonomy (classification) of our nomenclature;
Presenting a Venn diagram of the definition of trust;
Identifying the different levels of trust; and
Reviewing a method to measure trust [CSAW2005].

In this thesis we distill and prune the key topics of our trust framework. We investigate Tivoli provisioning system, an IBM application that has significant autonomic capabilities, to determine the nature and pieces of information exchange to build trust. To evaluate the strengths and shortcomings with respect to trust in Tivoli, we look at the nature and type of information exchanged in an adaptive system [Sous2005].

Upon identification of the nature and type of information to be exchanged, our approach involves meticulous planning and deciding which pieces of information among these should be inherently built-in or provided to the system and which others should be learned or communicated with time. We propose a three-stage trust model to describe the varying levels of trust in autonomic applications. We describe the existing

methods to measure trust in autonomic systems and outline which pieces of information need to be exchanged with each level of trust.

In order to evaluate our trust framework, we apply our findings to an autonomic e-Commerce prototype called the E-Marketplace application as a case study. The Electronic Marketplace Application was designed at the University of Victoria by Jing Zhou [Zhou2006]. The system is based on a Mobile Multi-Agent autonomic architecture to achieve e-Commerce functionalities. The system exhibits autonomic properties such as self-protecting and self-healing from failures.

Based on these case studies, we feel that we are able to guide the development of trustworthy autonomic applications to some extent. In particular, designers can use our trust framework to gain understanding for designing incremental trust for autonomic applications. This is our main contribution. The trust framework presented in this thesis is also a starting point for other researchers to follow and build up on.

1.5 Thesis Overview

This thesis is organized as follows. Chapter 2 provides insight into the work related to our domain of research. Chapter 2 also presents background on autonomic systems and a brief overview of trust related concepts and approaches. Chapter 2 discusses how trust can be measured. Chapter 3 introduces our proposed framework of trust in detail. Chapter 4 categorizes the nature and type of information to be exchanged to build trust and shows how this fits in with the framework of trust as discussed in Chapter 3. Chapter 5 is an attempt to evaluate our trust framework. Chapter 6 summarizes the contributions of this thesis and proposes areas for future research.

Chapter 2: Related Work

This chapter reviews previous studies and analyses trust factors in non-autonomic systems, autonomic e-Commerce systems and autonomic systems in general. There are many tools, infrastructures, concepts and methods to build trust; here we simply outline the ones that have influenced us the most. We also study related areas such as task models and other related work which may have directly or indirectly affected this research.

2.1 Introduction

Trust is a central concept in relationships, both because it is a pre-requisite for any kind of collaborative activity, and because it is one of the primary outcomes of and reasons for engaging in relationship development [Bick2003]. The literature on trust spans the disciplines of sociology, social psychology and philosophy. Several models of trust have been developed in the areas of software agents and artificial intelligence.

2.2 Communication and Interaction to Build Trust

Research on human-computer interfaces has produced several interesting results with respect to trust. It has been shown that trust in intelligent systems is higher for systems that can explain and justify their decisions [MiLa1992]. Also, it has been pointed out how the inclusion of comprehensive product information can influence a user's perception to trust an interface of a system [LeKM2000]. Moon demonstrated that a computer, which uses a reciprocal strategy—a deepening self disclosure in its text based conversation with the agent—will cause the user to rate it as more attractive, divulge more intimate information, and become more likely to buy a product

from the computer [Moon1998]. Analysis has been done and experiments have been performed to justify how applications having a human-computer relationship are advantageous to the human participant [Bick2003]. For example, when a user who is using an autonomic system that does not display a direct cause and effect relationship to the user, building user trust in the system will allow him or her to 'let go'. Bickmore *et al.* describe how software agents can prove to be useful in building relationships with computer systems. They design and advocate the use of Embodied Conversational Agents (ECA) that are anthropomorphic interface agents which engage a user in real time dialogue, using speech, and other verbal channels to emulate the experiences of human face to face interaction. They argue that such relational agents, which are computational artifacts that involve the user in small talk, can be designed to build and maintain long-term, social-emotional relationships with their users and help people negotiate and maintain better relationships. They present a case study with a real time, multimodal, life sized ECA, called Real Estate Agent (REA), which processes the conversation and its content. They perform an empirical evaluation of REA that determines and confirms the effectiveness and success of relational agents and such an approach to building relationships. Hence, all this and more demonstrates that building relationships via information exchange plays an important role in the effective use of intelligent systems.

Autonomy, pro-activeness, and goal-directed interactivity with their environment are distinguishing characteristics of software agents. Viewing autonomic elements as agents and autonomic systems as multi-agent systems make it clear that agent-oriented architectural concepts will be critically important [Keph2005b]. Researchers have described agents as an implementation method for autonomic software systems. Therefore, we assume that tools and techniques to build trust with an agent based system should be the same for and also apply to autonomic applications. Therefore we envision that building trust in autonomic systems using the concept of information exchange between the users and the autonomic systems is appropriate and worthwhile.

An integral part of a good relationship is trust [ReHZ1985]. Based on the success of relational agents in building relationships, the goal of this thesis is to recommend guidelines for the design of trustworthy autonomic applications by using the tool of communication. The success of relational agents underlines and provides reassurance to the fact that our approach of looking at information exchange for solving issues of trust in autonomic software systems is appropriate and worthwhile.

Researchers have proposed to combat the problem of trust in autonomic computing via communication and interaction [KAMK2005, CSAW2005]. In our thesis as well, we suggest communication and information exchange as a tool to help build user trust in autonomic applications. We advocate humans as modal managed elements to help build trust in autonomic systems [KAMK2005]. We also acknowledge the fact that humans are an integral part of any autonomic application. Hence, the autonomic system must account to the user.

The dual principle of initiative and interaction [KAMK2005] brings up the fact that while using communication as a tool to build trust with users of an autonomic application; we also need to consider the level of cognitive overload. Hence, we suggest exchanging different sets of information at different instances of time to limit the information overload in the minds of the user (cf. Tables 4.2a – 4.2e). We advocate exchanging more information at levels of no trust. At levels of partial trust, the information exchange is expected to decrease. At levels of full trust, we believe that information exchange can be filtered and hence information exchange is expected to decrease with increasing levels of trust.

Although interaction has been recommended in several papers, what exactly needs to be conveyed by the system to the user has not been detailed. This thesis explicitly defines the nature and type of information exchange and hence provides details on the interactivity that is required between humans and autonomic systems to build trust. It is important to apply and extend the approach of interaction and initiative with autonomic systems to an existing autonomic prototype as a case study in order to evaluate the framework. In fact, an in-depth study to build trust in autonomic systems has never been performed. Therefore, the goal of this research is to understand trust itself with respect to autonomic systems and then lay out specifically how and what type of information exchange between the user and the autonomic system helps the users feel as being a part of the loop for building trust. Application of the concepts presented in [KAMK2005] to a domain specific area like e-Commerce makes things more believable, understandable, and helps test its feasibility.

2.3 Understanding Task Models in Adaptive Systems

Information exchange and communication is a popular approach to building trust. However, the type of information to be communicated depends a lot on the task at hand. Different applications tend to have different domains of operation and hence the nature and type of information exchange with each system is contextual. As such, it is imperative to understand user task models so that we may identify the tasks for an e-Commerce application in particular, which is part of our research. We feel trust is relative to properties such as task urgency, choice, or personal factors. Understanding the details of scalability of task management across heterogeneous environment will help identify the general user expectations or requirements which can then be applied to our research. Therefore, in our research we borrowed ideas and concepts from Sousa's Ph.D. thesis where he presents an architectural framework for adaptive systems based on high-level models of what users need from the computing environment for each of their tasks [Sous2005]. An adaptive system is a device or mechanism, which is changed or changes so as to become suitable to a new or special application or situation. Adaptive systems are basically self-configuring systems and therefore a special case of autonomic systems. Thus, with respect to developing incremental trust over time with a user community, there is little difference between the specific notion of an adaptive system and the more general notion of a self-managed system.

The main contributions of Sousa's thesis are (1) an architectural framework that defines a new software layer dedicated to gathering knowledge about user tasks to promote system-wide awareness of user tasks and preferences; (2) a component dedicated to managing the environment based on abstract models of user tasks and a global view of the environment to promote coherent system-wide configuration and adaptation decisions; (3) a strategy for matching user needs and preferences to environment capabilities using a utility framework; (4) a three-level dynamic adaptation method based on (a) fine-grained adaptation policies within resource-aware applications, (b) adaptation to changes in the capabilities of an environment, and (c) adaptation to changes in the requirements of user tasks. The new software layer and the component managing the environment proposed by Sousa roughly correspond to the notion of an autonomic element and an autonomic manager, respectively. The strategy for matching

user needs and dynamic adaptation method can be used as a blueprint for meeting user trust expectations and adapting the information exchange while developing trust incrementally.

In his Ph.D. thesis, Sousa addresses heterogeneity and resource variations in the environment, by presenting task models which represent the user's preferences with respect to alternative ways to carry out the task and preferred Quality of Service (QoS) tradeoffs. The task models adopted in his work address three fundamental properties of scalable task management. First, task models provide a handle for the coordinated use of a set of services in the environment. Second, to address scalability in space, and in particular heterogeneity and resource variations in the environment, task models represent the user's preferences relative to features and QoS. Third, to address scalability in time, task models establish an enduring identity that enables users to find tasks long after they are gone from the list of currently active tasks [Sous2005].

Sousa's ideas on how to address scalability in space, we feel, are most interesting and relevant to our research. The following aspects of the scalability in space category have been used and applied in our thesis: QoS, supplier preferences, configuration details, context awareness and specification. The following is a brief overview of some of Sousa's ideas that we found intriguing, relevant and applied to our research. Chapter 4 describes the applicability of Sousa's work to our research in more detail.

- **QoS**: QoS preferences for a user may change with time and tasks. Users may prefer to have different QoS tradeoffs for a given service in different tasks [Sous2005]. Sousa in his Ph.D. thesis gives an instance wherein Fred is watching a video over a network link and the bandwidth suddenly drops. Now, reducing the frame rate or the image quality will depend on Fred's preferences for the current task. If Fred is watching a sports event, he may prefer frame rate to be preserved at the expense of image quality. On the other hand, for watching a documentary on painting, the reverse may be preferable. Furthermore, the preferred QoS tradeoffs may change during the task. Sousa cites an instance where there is a task that involves automatic translation of natural language. The users in this case may prefer faster responses over accuracy of translation during the introductory part of the conversation, but they may prefer the opposite once the conversation gets more involved. According to Sousa, while the simplest form

of expressing a tradeoff is to indicate which dimension is preferred, this form has very limited expressive power. For instance, a user might indicate that response time is preferred over accuracy of translation. However, how short of a response time will satiate the user? And even if accuracy is less important, what if it degrades so much that the translations become unusable? A clearly more powerful form is to express the thresholds that characterize a tradeoff. For example, if Fred requires highly accurate translations, he may be willing to wait up to 30 seconds for an answer. In Sousa's work, QoS tradeoffs are set by expressing user happiness with the level of quality provided along each QoS dimension. For example, when faster responses are preferred over accuracy of translation, the QoS preferences set stricter happiness thresholds for response time and looser thresholds for accuracy. The tradeoff can be reversed by relaxing the thresholds on response time and tightening the thresholds on accuracy.

- **Supplier preferences**: Sousa notes that users may be willing to use different service suppliers in different circumstances. He quotes an instance where Fred starts reviewing the video at home, using MS Word as a supplier for editing his notes, and decides to resume that task at the office, where he has a desktop running Linux. According to Sousa, if only Linux native text editors are available, say Emacs and Vim, Fred may prefer using Emacs to Vim (or vice-versa). Sousa mentions that during automatic configuration, supplier preferences play a key role in choosing among alternative components to provide a given service. A task model includes supplier preferences for every service in the task [Sous2005].
- **Configuration details**: The configuration management of a multimedia adaptive application called Aura as suggested in his Ph.D. thesis is addressed as follows. By finding the available components that best fit the user's needs for each task, the infrastructure enables users to take full advantage of diverse environments. Furthermore, by keeping track of user preferences with respect to QoS, of resources demanded by alternative computing modalities, and resource availability, the infrastructure can select the optimal configuration and carry out dynamic adaptation to changes in the environment. By automatically configuring environments, on demand, and by continuously adapting to changes in the environment, the infrastructure reduces the burden of routine configuration and reconfiguration tasks for the user.

- **Context awareness:** Sousa in his Ph.D. thesis outlines how context awareness is important. He presents an architectural framework describing ways to sense changes in context and adapt to them [Sous2005].
- **Specification:** Sousa cites an instance: During a task that involves automatic translation of natural language, a user may prefer faster responses over accuracy of translation during the introductory part of the conversation, but may prefer the opposite once the conversation becomes more involved [Sous2005]. We feel that allowing the user to specify task based and other preferences and demonstrating to the user that those preferences are applied by the system, should help in gaining users confidence.

Hence, we use some of Sousa's terminologies and concepts to develop our trust management framework for autonomic systems. Knowing user requirements and expectations with an understanding of users' task models in general, helps us filter and drill down into the nature and type of information exchange necessary for building incremental trust when we look at Tivoli. While Sousa's work is based on an interactive multimedia application called Aura [Sous2005], our research focuses on an e-Commerce autonomic prototype called E-Marketplace Application [Zhou2006].

2.4 Autonomic Computing and Trust

In order to accomplish our goals we need to study and understand existing autonomic applications. We need to understand how we may map the process of building trust to an autonomic element itself.

2.4.1 The Correlation between the Process of Development of Trust in Autonomic Computing and Autonomic Computing itself

As discussed in Chapter 1, an Autonomic Element constitutes an architectural component of autonomic systems. In this thesis, we propose an analogous element called 'Autonomic Trust Element' as depicted in Figure 2.1 below in an attempt to map the architectural components of an autonomic element (cf. Figure 1.1) to the process of building trust (cf. Figure 2.1).

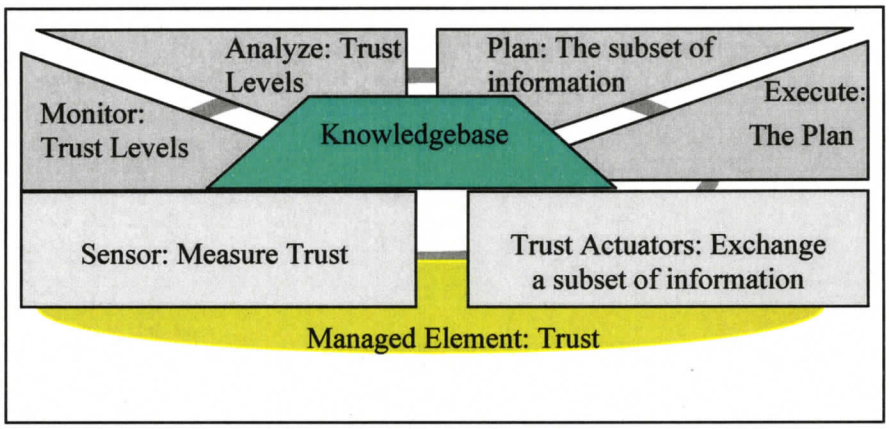

Figure 2.1: An autonomic trust element

We argue that the process of building incremental trust with autonomic applications is simply an autonomic or self-managing process itself. We perceive user trust in an autonomic application as a managed resource. In order to have a trustworthy autonomic application, we need to measure trust at different instances to make adjustments accordingly. In order to accomplish this, any decrease or increase in the level of trust needs to be monitored and analyzed based on user reactions or actions [CSAW2005]. When the level of trust increases or decreases, the system needs to plan and execute so as to adjust to the varying levels of trust. The feedback loop plays a vital role in the policy-based management of trust. Hence building trust in autonomic applications in itself conforms to the architecture of an autonomic element. Once the trust level has been determined, the planner needs to decide which subset of information needs to be presented to the user (i.e., how to increase or decrease the amount of information exchanged with the user). The executer and the trust actuators

may then in turn, display the specified pieces of information to the user. In Chapter 4, we present a layout of the nature and type of information exchange in autonomic systems. We group them into sets depending on the levels of trust which may be picked up by the planner and be presented to the user while implementing such an autonomic trust element.

2.4.2 Measuring trust in Autonomic Computing

We now review techniques to measure trust. Chan *et al.* proposed the concept of *Overall Trust Index (OTI)* and *Instantaneous Trust Index (ITI)* for autonomic systems to measure trust and thus allow the system to adapt to the varying levels of trust [CSAW2005]. The OTI for an autonomic manager reflects the level of trust of a system administrator in that autonomic manager [CSAW2005]. The ITI can be computed, for example, by examining what fraction of suggested actions from an autonomic manager the user accepts unchanged, or by examining how extensive the changes that the user makes to the suggested actions are [CSAW2005].

Let ITI = $f(m_1, m_2, \ldots m_n)$, where m_1, m_2 ... m_n are weights assigned to each user modification, and 0 <= ITI <= 1. The OTI is then computed as follows: OTI = $f(ITI_1, ITI_2, \ldots ITI_N)$ where ITI_1, ITI_2 ... are the ITIs for each execution of the Autonomic Manager, and f() is likely to be a variation of a moving or historical average.

The OTI and ITI values can be stored in the knowledgebase. The sensors in a computing system may be seen as event listeners to mouse clicks or keyboard hits which will modify the values of ITI. Since OTI is a function of ITI, every change in ITI, will change the values of OTI. Recomputing of values of ITI or OTI according to the formula discussed in [CSAW2005] may be seen as an analysis step. The actuators and effectors can be seen as the exchange of relevant information based on OTI and ITI values. This information exchange may be implemented using extensible languages based on XML such as Common Base Events [IBM2006]. Reinforcement learning processes can be used as a feedback loop from information extracted from user interaction to the autonomic managers [CSAW2005].

Autonomic systems at any point during trust management may need to 'self-configure' trust. For example an autonomic system may need to decide which pieces of

information need be presented to the user at any given instance to meet accountability towards the user which may not be relevant at other instances. Also, in case there has been a decline or complete loss in user trust, an autonomic system may need to 'self-heal' by presenting the user with more detailed pieces of information such as the current state of the system and expected actions for protecting after say a system crash to re-build trust. Also, with the varying levels of trust, the autonomic application will need to 'self-optimize' the level of detail in the information exchange between the user and the system. The system will need to be proactive and work towards 'self-protecting' itself against any losses in user trust. Hence, the process of development of trust in autonomic computing systems in itself follows the paradigm of autonomic computing.

2.4.3 Mapping Autonomic Computing maturity levels to different levels of trust

The road map for the autonomic computing architecture describes the following five levels of maturity, illustrating how businesses are constantly evolving their IT environment [Word2004]: Basic --> Managed --> Predictive --> Adaptive --> Autonomic.

These terms are defined as follows [Word2004]:

- **Basic**: The product and environment expertise resides in human minds, requiring consultation on even mundane procedures.
- **Managed**: Scripting and logging tools automate routine execution and reporting. Individual specialists review information gathered by the tools to make plans and decisions.
- **Predictive**: Early warning flags are raised as preset thresholds are tripped. The knowledge base recommends appropriate actions. The proposed resolution of events is leveraged by a centralized storage of common occurrences and experience.
- **Adaptive**: Building on the predictive capabilities, the adaptive system takes action itself based on the situation.
- **Autonomic**: Policy drives system activities such as allocation of resources within a prioritization framework.

In the field of trust in autonomic computing, few researchers have made an attempt to define levels of trust in an autonomic system. We see the above five levels of autonomic computing maturity directly analogous to the user selectable operation levels as defined by Chan and Lee which are minimal, partial and full trust [CSAW2005, LeSe2004].

- **No trust or minimal trust**: The OTI is close to zero (i.e., OTI ≈ 0). The user makes several modifications and the user expectations do not match the autonomic capabilities and actions performed by the autonomic manager. In this mode, the Autonomic Manager generated actions will not be executed and the user will examine them. If these actions are accepted without modification, an ITI of 1 is awarded. An ITI of 0 is assigned if these actions are completely rejected or replaced by the users' actions. Otherwise, ITI is assigned an amount specified by an expert-defined function of the amount of modification. We see minimal trust as a basic level of autonomic maturity when mapped to autonomic computing [CSAW2005].
- **Partial trust**: The value of OTI lies is in the range 0 and 1 (i.e., 0 < OTI < 1). In this case, the user trusts the Autonomic Manager generated actions but needs to examine the parameters of these actions. The ITI for this execution is computed as follows—if the action is accepted unchanged, an ITI of 1 is assigned; if the action is changed, then the ITI is assigned a value specified by a function with respect to the modification. We see partial trust as basic and managed levels of autonomic maturity when mapped to autonomic computing.
- **Full trust or calibrated trust**: The OTI is a close to one (i.e., OTI ≈ 1). There exists a correspondence between a person's trust in the automation and the automation's capabilities. In this case, the autonomic manager executes the actions without user intervention. A summary is generated for each autonomic manager action execution, and the user can examine this summary (or a digest of it) periodically and decide if he or she wants to switch the system back to partial trust mode or minimal trust mode. If the user does nothing, an ITI of 1 is awarded for the autonomic manager. If the user decides to switch back to other modes of operation, ITIs of 0 are assigned. An OTI is computed based on the long term performance of each autonomic manager or group of autonomic managers using user defined or other averaging techniques of all individual ITIs. An OTI close to 1 indicates a high level of trust, and similarly, an OTI close to 0 indicates no confidence of users with respect to the autonomic manager. In addition, this system could also use more advanced learning techniques to modify

its behaviour, based upon the actions of the user in response to suggested actions, in order to win the trust of the user (e.g. to increase the OTIs). Reinforcement learning processes [AHPR2003] can be used as a feedback loop from information extracted from user interaction to the autonomic manager. Full trust based on our understanding may be seen as an adaptive and autonomic level in the maturity of autonomic computing.

2.4.4 Trust in Automation

To understand the variation of trust and its dependencies that affect the level of trust, Lee *et al.* present a relationship among calibration, resolution, and automation capability in defining appropriate trust in automation as depicted in Figure 2.2 [LeSe2004]. They say that inappropriate reliance associated with misuse and disuse depends, in part, on how well trust matches the true capabilities of the automation. Supporting appropriate trust is critical in avoiding misuse and disuse of automation, just as it is in facilitating effective interpersonal relationships [WiBT1999]. According to Lee, calibration, resolution, and specificity of trust describe mismatches between trust and the capabilities of automation. Calibration refers to the correspondence between a person's trust in the automation and the automation's capabilities [LeMo2004].

Overtrust is poor calibration in which trust exceeds system capabilities; with distrust, trust falls short of the automation's capabilities. In Figure 2.2, good calibration is represented by the diagonal line, where the level of trust matches automation capabilities. The area above this line is overtrust, and below it is distrust. According to Cohen *et al.* [CoPF1999] resolution refers to how precisely a judgment of trust differentiates levels of automation capability. Figure 2.2 shows that poor resolution occurs when a large range of automation capability maps onto a small range of trust. Cohen *et al.* state that with low resolution, large changes in automation capability are reflected in small changes in trust [CoPF1999]. According to them, specificity refers to the degree to which trust is associated with a particular component or aspect of the trustee. They introduce a term called *functional specificity* which describes the differentiation of functions, sub functions, and modes of automation. With high functional specificity, a person's trust reflects capabilities of specific sub functions and modes whereas low functional specificity means the person's trust reflects the

capabilities of the entire system. Cohen also says that specificity can describe changes in trust as a function of the situation or over time. High temporal specificity implies that a person's trust reflects moment-to-moment fluctuations in automation capability, whereas low temporal specificity would mean that the trust reflects only long-term changes in automation capability. He points out that although temporal specificity implies a generic change over time as the person's trust adjusts to failures with the automation, temporal specificity also addresses adjustments that should occur when the situation or context changes and affects the capability of the automation. Temporal specificity reflects the sensitivity of trust to changes in context that affects automation capability. High functional and temporal specificity increase the likelihood that the level of trust will match the capabilities of a particular element of the automation at a particular time. Finally, as Cohen puts it, good calibration, high resolution, and high specificity of trust can mitigate misuse and disuse of automation, and so they can guide design, evaluation, and training to enhance human automation partnerships. In our thesis, we acknowledge the significance of Cohen's work but do not build on this. For this thesis, we use Chan [CSAW2005] and Cohen's [CoPF1999] work to determine trust levels and methods to measure trust.

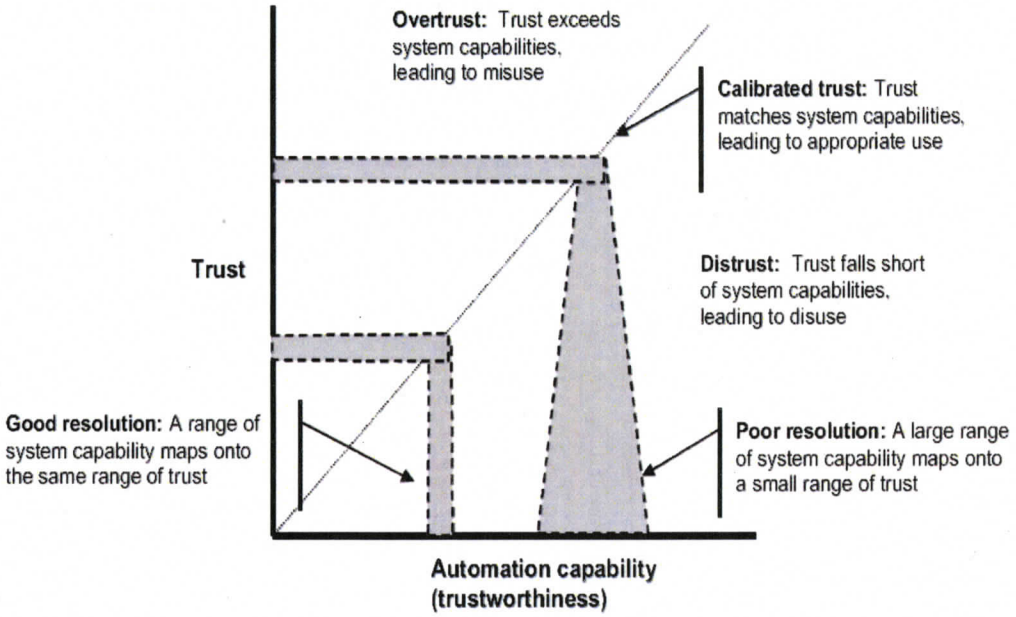

Figure 2.2: The relationship among calibration, resolution, and automation capability in defining appropriate trust in automation

Courtesy: Trust in automation, Lee and See, University of Iowa [LeSe2004]

Our understanding of trust levels is based on several papers [CSAW2005 and Rott1967]. In particular, we use Chan's [CSAW2005] and Lee's [LeSe2004] work to understand the possible levels of trust in autonomic systems. There is significant overlap with respect to the research domains of Chan and Lee because they both discuss trust levels of autonomic systems.

2.4.5 Existing Autonomic Applications

Having described how development of trust in autonomic applications follows the paradigm of autonomic computing itself, we now look at autonomic systems as an application. Since our research domain deals with overcoming the human-computer interaction and trust challenges that autonomic computing poses and hence, it is important to identify the features and modus operandi of existing autonomic systems deployed in the industry that may have led to its success. Studying Tivoli to determine the strengths and shortcomings with respect to trust issues may help generalize the basis for building trust for other autonomic systems. In this thesis, we also look at one other autonomic application called E-Marketplace to evaluate our findings. The following sub-sections provide details on these applications.

2.5 Autonomic applications

2.5.1 Tivoli—A Deployed Autonomic System

Tivoli systems management software helps traditional enterprises and e-businesses worldwide manage security, storage, performance, availability, configuration and operations. Tivoli's autonomic monitoring engine captures, analyzes and correlates key metrics to automatically detect resource outages and potential problems before they impact system performance or end-user experience. The monitoring engine has embedded self-healing technology to allow systems to recover from critical situations automatically. In a workshop on developing an e-Business policy framework, Joseph

presented how and what features helped users build trust in Tivoli [JoGe2004]. Borrowing from his ideas may help us make concrete recommendations to newly deployed autonomic systems and gain confidence in our trust framework.

2.5.2 The E-Marketplace Application—A Non-deployed Prototype of an Autonomic System

In order to apply our framework to a non-deployed autonomic e-Commerce application, we need to understand it. The system we looked at for this thesis is an autonomic prototype called the E-Marketplace Application, designed and developed as part of a Masters thesis at the University of Victoria, Canada [Zhou2006]. The E-Marketplace is a shopping portal which allows users to make requests for an item, search amongst the available sellers of the item to negotiate prices, and finally perform the most economically viable transaction for the user.

The application has autonomic capabilities such as self-protecting and self-healing. In this section, we briefly describe the E-marketplace application [Zhou2006]. The application achieves a set of desired features of autonomic systems such as autonomy, hiding complexity and self-healing. The E-Marketplace Application follows a mobile Multi-Agent Systems (MAS) approach and applies the technology to build an autonomic prototype. This system has not been evaluated or deployed in industry. In this section we discuss only those characteristics of the application which are relevant to our domain of research.

2.5.2.1 Architecture

The E-marketplace application is a multi-agent based application which comprises of the following components:

- User Interface Agents that provide a graphical interface that links the user with other agents.

- Middle Agents that work as a bridge to match the request to a class of potentially suitable candidates. These agents select the most qualified service provider from among those candidates, and assign a specific task to the agent to perform a particular function.
- Task Agents that support decision making by formulating problem solving plans. The agents carry out these plans through querying and exchanging information with other agents.
- Resource Agents that mediate access to a particular resource at a local level for an agent.
- Domain Agents that supervise and coordinate the activities that occur within a domain. A domain is a logical boundary used to delimit nodes, agents and resources into manageable and distinct entities.
- Monitor Agents that continuously monitor the status of each agent, detect agent failures, and execute protecting to achieve the self-healing property of autonomic systems.
- Mobile Agents that migrate between network nodes, reduce the network load, overcome network latency. These agents are naturally heterogeneous, robust and fault-tolerant.

The E-marketplace application is divided into modules. Table 2.1 shows the main modules in this electronic marketplace.

Table 2.1 List of design modules and descriptions

Courtesy: Masters Thesis - A Mobile Multi-Agent Autonomic Architecture for an Electronic Marketplace Application by Jing Zhou [Zhou2006]

Module Name	Description
Management Module	Responsible for managing the E-marketplace including all buyers and sellers in this market
Buyer Agent Module	Acts as a buyer to trade with sellers to obtain the best price
Seller Agent Module	Acts as a seller to sell products and realize profits
Monitor Agent Module	Continuously monitors the E-marketplace to detect if there is something wrong with any agent; recover from failures.

Each module contains a number of stationary agents and/or mobile agents that interact and are inter-related in some way to perform the tasks.

In order to cooperate effectively in this system, agents are required to communicate with one another. We appreciate that while agents communicate with each other, the application itself needs to communicate with the user based on the agent-agent interactions to develop trust in the system. In order to achieve our mission, we need to understand in detail how E-shopping works.

2.5.2.2 Autonomous E-shopping

This section introduces the autonomous transaction processing in the E-Marketplace application. The steps of shopping for buyers are described below.

The steps of E-shopping are as follows [Zhou2006]:

- The User Agent receives the "Go Shopping" message from a user action. It sends a purchase query to a Buyer Agent. The Buyer Agent creates a purchase order, which is an XML file. This purchase order is sent to the User Agent.
- The User Agent is required to send a confirmation to the Buyer Agent in terms of confirm or failure. Once the Buyer Agent has received the confirmation, it creates a Mobile Buyer Agent (MBA) and dispatches it to the Market Management Agent (MMA). After the MBA passes the identity verification, the Market Management Agent will check the configuration repository for the products that the buyer needs, and get the relevant information about all sellers and forward the sellers' address to the MBA.
- After the Buyer Agent receives the sellers' addresses, it gets to choose the sellers with which it wishes to transact. Based on the selection, the Buyer Agent then sends an MBA to appropriate seller.
- Upon arrival at a seller's site, the MBA starts to negotiate with the Seller Agent. Firstly, the seller checks his own inventory through an Inventory Agent, if they have the needed goods, the seller will accept the proposal from the MBA and

begin to bargain with the MBA by using three negotiation strategies; otherwise, it will send a "refuse" message to the MBA.

- Once the MBAs have visited the sellers, it returns with the purchase results. The MBA processes the results: compares all prices, and selects the best price. Finally, it generates a Mobile Report Agent and dispatches that to the relevant sellers so as to communicate the final purchase result. The successful sellers then update their inventories accordingly.
- After the Mobile Report Agent is done performing these tasks, it goes back to the parent Mobile Buyer Agent to report to the MBA and disposes itself.
- Finally, the Buyer Agent presents the purchase information to the user which is an XML file.

2.6 Concerns Affecting Trust in e-Commerce

Understanding and investigating the myriad areas of concern to build trust, as well as the concepts of trust itself is imperative. The universe operates on inter-dependence. No action may be performed in isolation. Analogously, building trust is not a stand-alone concept. Hence, while looking at trust and communication as a means to build trust it is obligatory on our part to consider other areas that may affect trust. An outline of some areas of concern to help build trust in e-Commerce systems such as informational contents, interface properties, pre-purchase knowledge, or usability in e-Commerce applications are already in place [Dekl2000, Egge2000].

Patrick suggests how interface design and comprehensive information, are contributing factors in building trust and how they need to be studied to build trustworthy agents [Patr2002]. Araujo states how embedded contextual help (field descriptions), customer support in the form of online agents, frequently asked questions, feedback, easy access to information, product descriptions on e-Commerce websites, privacy policy, pre-purchase and post-purchase terms and conditions, warranties, and informed consent on release of personal data, and so on are all responsible for building trust [Arau2003]. Anderson *et al.* stress how it is important to determine the nature and type of information in detail that would need to be exchanged with users but the nonchalance of focus in this direction [AHPR2003]. Hoffman *et al.* provide statistics on how users of

Internet commerce feel insecure while providing information to e-Commerce applications due to their lack of trust in the privacy offered [HoNP1999].

Hence, all this and more [LuNa2004] let us conclude that the communication (exchange of information between the application and the user) are the primary concern in the process of building trust. Also, another interesting finding from the above research indicates that an application may not be trusted—even if it is secure enough—if the user is not made aware of it. Hence, all concerns ultimately are based upon proper communication and accountability from the system to the user.

There has been immense research in identifying the concerns to build trust. However, the existing concepts only constitute a starting point for solving the problem at hand since they are mainly based on non-autonomic systems. These concepts therefore need to be investigated in detail and analyzed meticulously before applying to autonomic systems. These areas of concern have been established in pre-autonomic systems [Arau2003, Bruh2002, Egge2000, KaFJ2002] and few autonomic systems [Tivoli2006, AHPR2003, BMKB2005, RMDN2003]. It may be useful to consider and carefully weigh the consequences of using concepts in designing trust in non-autonomic systems to build trust in autonomic systems. We also need to determine how and to what extent these ideas can be useful. What we have not yet explored so far are the protocol, grammar and character of information exchange for establishing and implementing trust in autonomic systems.

Many theorists and experimentalists have offered ideas and some technical know-how with superficial details that may allow the establishment of trust with systems, specifically Internet applications. Design for trust requires enumerating the social assumptions (e.g., every user has a credit card) and examining how those assumptions can function to put some user of the system at risk [Camp2003]. Camp describes some technical implementations to achieve trust [Camp2003]. He builds his argument that the basis of trust is privacy (i.e., willingness to share information), reliance (i.e., a result of belief in integrity or authority of the party to be trusted, based on the concept of mutual self interest) and reliability. He argues that in order to design for trust we need to look into the privacy which further has several different perspectives to it. On the other hand, Kagal *et al.* advocate a distributed trust management security framework for multi-agents [KaFJ2002]. Such technical details may be useful while implementing trust agents and may provide us with a useful feasibility study. Also, an initial requirements list for designing our prototype may be obtained from existing research. Again, the

existing concepts only provide a starting point for the feasibility analysis and requirements engineering to the problem at hand since they are mainly based on non-autonomic systems.

2.7 Chapter Summary

This chapter distilled several important approaches, methodologies and concerns that have been a major influence on our Trust Framework for autonomic systems outlined in the next chapter.

Chapter 3: Trust Framework

Prior research on trust has taken into account human psychology and human-computer interaction issues [ANBL2002]. It is important to understand trust as a concept and as an abstract aspect to human behaviour before we can begin to apply it to autonomic systems in order to advance the present state of art. This section attempts to define trust and to classify trust into different categories. It also provides understanding, explanation and insight into the various facets of trust and constitutes the major contribution of this thesis. The result is a trust framework for autonomic computing.

3.1 Defining and Categorizing Trust

Trust has many dimensions, aspects and facets. Identifying trust categories that are orthogonal or non-overlapping is futile. Nevertheless, we submit that the following characterization of trust is useful for our purposes: type of trust, the basis or source of developing trust, and the definition of trust itself. In this section we attempt to classify these facets of trust to provide an understanding of trust. Figure 3.1 depicts our understanding of a trust framework. Sections 3.1.1 to 3.1.4 describe the framework and its applicability to autonomic computing in detail.

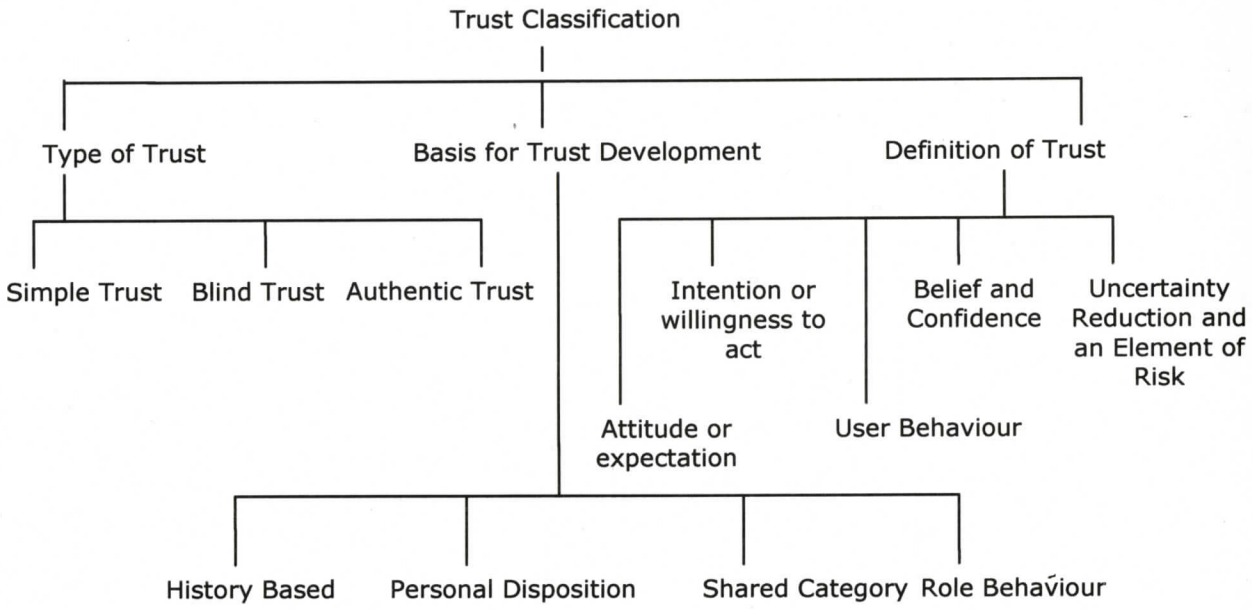

Figure 3.1: Framework of trust

3.1.1 Classification on the basis of Type

Trust can not only be classified on the basis of its definition, but also be categorized on the kind of trust being established. Bruhn identifies the following categories: Simple Trust, Blind Trust and Authentic Trust [Bruh2002]. Although we acknowledge this category as part of our trust framework, we appreciate that there may be less applicability of such a categorization with respect to autonomic computing. We, therefore, merely outline the components under this category without an in-depth analysis.

3.1.1.1 Simple Trust

Simple Trust is related to innocence and is seen to have no actual logic behind it. In other words simple trust is naivety. For example, the trust of a child with respect to strangers is simple trust. Simple trust in autonomic applications or software systems may be seen as the trust of a novice or naive user who may not have details on the

intricacies of autonomic computing capabilities and/ or is indifferent towards the application tasks and efficiency of operation for that reason.

3.1.1.2 Blind Trust

Blind trust can be understood as extreme faith beyond reasoning. Blind trust is often a consequence of extreme love or foolishness. Blind trust again has no logic behind it and is often a result of extreme passion rather than naivety. Blind trust often comes from the unwillingness to diverge from a particular viewpoint. For example, the trust in one's lover may be blind. Blind trust may be seen as a developer's trust in his or her application. Research shows that developers often times feel a sense of ownership with a piece of code they have written. We believe that a sense of ownership may encompass extreme faith and hence take the form of blind trust.

3.1.1.3 Authentic Trust

Authentic trust is trust wherein the risks and benefits are well understood. Authentic trust involves careful understanding and analysis of the pros and cons of a particular issue. Trust is established based on some logical basis. In sum authentic trust is rational assessment of reliability [KBCS1999]. For example, shareholders exhibit trust towards a company which may be seen as authentic trust because usually a study of the market share is done before investing. In terms of software systems and autonomic computing, we may understand authentic trust as the trust of an educated or informed user, who has computing experience, while using a third party application. For our thesis, we assume that authentic trust is the type of trust that needs to be instilled in users for autonomic systems.

3.1.2 Classification on the Basis for Development

Trust may be further classified on the basis of which trust was built. Trust may arise out of some past experiences, roles played in society, personal disposition, or feeling of shared category membership [Pysi2003]. This category of trust seems to have more

relevance and a direct relationship with autonomic computing systems. The following subsections give a brief description on how this part of our trust framework applies to autonomic computing systems.

3.1.2.1 History based source of trust

In an individual, trust is a factor of how they were treated previously [ReHZ1985]. Sometimes trust is built on the premise that a past experience in a similar situation was favourable. History based trust refers to establishing faith out of prior experiences. We assume history based trust as a basis for our trust framework.

In autonomic systems if the system recovers from say a failure, there are chances that the policy applied to overcome those issues will be considered a proven solution to resolve a similar issue in the future as well. Since the concept of autonomic computing entertains the concept rule based procedures, knowledge base and feedback loop, it seems feasible to develop trust based on prior experiences (history based source of trust) by reinforcement learning.

3.1.2.2 Personal disposition

Personal disposition means how signals are interpreted. For example, some people may be detail-oriented while others are brief and to the point. A concise description from a straightforward person may be interpreted as lack of knowledge in the minds of a detail-oriented individual. Hence, different people have different attributes and their personal disposition is based on that. People having a similar interpretation of signals may seem to trust each other more. Also, some people develop trust more easily than others hence personal disposition relates to our thesis. However, for our thesis we do not consider this as a factor while determining our trust framework but acknowledge it as a limitation of our research.

This seems to be worthwhile in view of autonomic systems and may apply directly because users may have varying levels of expertise and hence differences in personal disposition. Sousa in his Ph.D. thesis, mentions that users with different degrees of expertise on the demand side will ask for things at different levels: an inexperienced

user will try to get more abstract services, hiding internal details as much as possible, while an expert user may want to have more control over which, and how the parts are configured. The same user may want to have more control over the structure supporting a critical task, but be willing to take an off-the-shelf solution for a low priority task. On the supply side, it is to be expected that sophisticated environments, such as smart rooms, will have higher-level, well-tuned components, while poorer environments, such as handhelds, will have a collection of generic parts that can be assembled to deliver a similar function, but in a less polished way [Sous2005].

3.1.2.3 Shared category membership

Shared category membership refers to a sense of togetherness and feeling of working towards a common goal. For example, a team of workers developing software in an endeavour to meet deadlines tend to trust each other due to having a common goal.

We see this component as a vital force in building user trust with autonomic systems. An explicit layout of information by the system to the user; cause and effect of those actions shall re-affirm user belief and develop a sense of togetherness and feeling of working towards a common goal.

3.1.2.4 Role behaviour

Role behaviour refers to the established role definitions. For example, a patient would naturally trust a doctor to cure his or her ailment out of the role he or she plays and the position he or she holds.

Haubert enlists the different categories of users of a high performance storage system which includes system administrators, end users, and system architects [Haub2004]. We envision autonomic computing systems to be a subset of such a system. Hence, trust levels will vary for users performing varying roles due to differences in their task structure and requirements from the system. For example, a system architect is usually experienced, has a high degree of understanding of the system configuration requirements and has familiarity with the documents; hence he or she may seem to have more trust in an autonomic system and may seem to understand the system intricacies and details presented in a more precise manner. Conversely, he or she is

understood to be in a more eligible position to report to or query the system. The system may on the other hand, seem to be more responsive to such users for the aforementioned reasons. Hence, an autonomic system may seem to perform the role to accomplish a predefined goal for an architect in a more timely and efficient manner.

3.1.3 Classification on the basis of definition

The concept of trust is a broad one. Today there exist several definitions of trust with people taking their own stands with their own arguments in support of their theory. Grandison defines trust as multidimensional which concerns many different attributes such as reliability, dependability, honesty, competence, which may have to be addressed based upon the environment where it is specified [GrSl2000]. While some researchers see trust as a sociological topic as opposed to a psychological trait within the individual [Barb1983] others feel it is an attitude, belief and psychological trait rather than a sociological topic [Dict2005, Kini1998, Wrig1972]. Keeping these wide variety of concepts and opposing ideas, in view, Lee *et al.* make an endeavor to classify existing theories of trust into three main categories: Trust as an attitude or expectation, Trust as an intention or willingness to act, and trust as a behaviour [LeSe2004]. We develop and build on these theories. We also introduce new categories of trust based on these definitions. We see this classification of trust as most applicable to autonomic computing systems for the reasons outlined in the following sections. Here is a pictorial representation of our trust framework for autonomic systems based on definition.

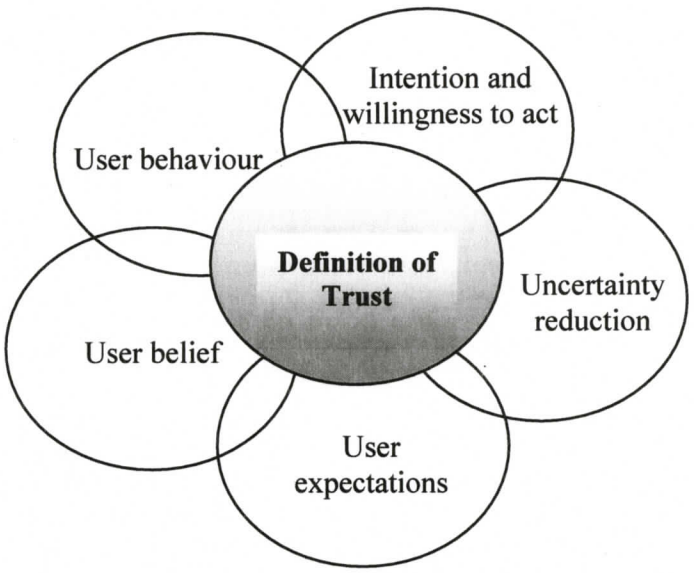

Figure 3.2: Representation of our trust framework for autonomic systems based on definitions

Note that the circles depicted in Figure 3.2 overlap. For example, although we have acknowledged the differences between user belief and user behaviour, we appreciate that there is a point of intersection between the two and hence the overlap. Similarly, user belief and user expectations are different yet not mutually exclusive and hence overlap. The gradient applied to the circle in the center of the figure indicates that the concepts represented by the lower circles contribute more to our trust framework of autonomic computing than the concepts of the upper circles.

We acknowledge that the concept of trust is a vast domain and complete analysis of trust in all dimensions may seem to be outside the scope of this research. We, therefore propose the framework of trust as a combination of user belief, user behaviour, user intention and willingness to act, user expectation and trust as an element of risk and uncertainty reduction. The central gray circle overlaps each of the five components mentioned in the preceding sections. Note that the central gray (trust) circle also falls in an area where there is no overlap from any one of the five components based on the classification of trust as a definition. This non-overlapping free space area signifies the possibility of having other components in the trust

framework. We believe that this composition is necessary, but not sufficient to establish a solid trust framework.

3.1.3.1 Building on the existing categories from Lee

Trust as an attitude or expectation

While trust has been advocated as an expectation held by an individual due to a promise or written communication of another that can be relied on [Rott1967], trust can also be seen as expectations related to a subjective probability an individual assigns to the occurrences of some set of future events [ReHZ1985]. Trust is also described as an expectation of fiduciary obligation and responsibility, that is, the expectation that other people in our social relationships have moral obligations and responsibilities to demonstrate a special concern for others' interests above their own [Barb1983]. Although Lee *et al.* rightly acknowledge the 'expectation' category of trust, they miss some related definitions of trust that belong to this category. For example, Bickmore *et al.* define trust as people's abstract positive expectation that they can count on partners to care for them and be responsive to their needs now and in the future [Bick2003]; this notion of trust has not been acknowledged by Lee as a part of this category. Also, Deutsch's contribution, which explains trust as an expectation of an event requiring the existence of both positive or desirable outcomes and negative or undesirable outcomes, has been missed.

Sousa underlines how expectations and/or needs of users with respect to computational support for their tasks are significant and applicable to task models [Sous2005]. For autonomic systems, situation awareness is important [BMKB2005]. Fulfilling users' expectations regarding what the system is doing, why it is doing that, and what will it be doing next may prove to be worthwhile when building trust.

Autonomic systems do not display cause and effect relationships and are not transparent. Hence, the user will not know if the operation performed by the autonomic application actually matches the 'expectation' the user had in mind. This suggests that for autonomic systems to be trusted, it would be worthwhile that users be informed of the probability or success rate of a particular action based on prior knowledge and state

changes that have occurred before. This seems feasible for autonomic systems provided there is sufficient history stored in the knowledgebase to analyze patterns.

Trust as an intention or willingness to act

Lee aptly supports the approach to categorize trust as an intention or willingness to act rather than portraying it as an expectation [LeSe2004]. He pleads that trust is the willingness to place oneself in a relationship that establishes or increases vulnerability with the reliance upon someone or something to perform as expected. Other researchers campaign that trust is the willingness to rely on an exchange partner in whom one has confidence [MoDZ1993]. Yet other researchers take the stand that trust is the willingness of a party to be vulnerable to the actions of another party based on the expectancy that the other will perform a particular action important to the trust or, irrespective of the ability to monitor or control that party [MaDS1995].

With respect to autonomic systems, we assume that users of our e-Commerce prototype are open to the idea of building authentic trust based on the provided evidence that the system itself cooperates. Thus, the autonomic system must also communicate to the user in explicit terms that it is willing to perform the assigned tasks.

Trust as a user behaviour

Behaviour is the manner in which something or someone functions or operates [Dict2005]. Although, Lee *et al.* recognize this category of trust, they fail to provide evidence and support to defend this classification [LeSe2004]. Wrightsman provides evidence that people tend to develop a personal philosophy about their interaction with people [Wrig1972]. As such it is important to appreciate trust as behaviour. Trust as behaviour is a conscious regulation of one's dependence that will vary with the task, the situation or the person. Judgments made by a user based on the experience gained from being a consumer and from the perception of a particular merchant [KBCS1999] is another definition of trust which may be placed in this category. Note that judgments and perceptions may vary from person to person.

Patrick states that the factors contributing to trust of a system also include the user's individual ability to trust and past experience [Patr2002]. He quotes a survey wherein 27% people in one group only marginally were concerned with online privacy, while 17%

in another group were labeled as 'privacy fundamentalists' and were very particular about privacy. The third group (56%) was labeled the 'pragmatic majority' because they had some concern about privacy, but also had developed tactics to deal with those concerns.

While this thesis focuses on building trust based on comprehensive information provided to the user, we also appreciate that in order to build trust in totality we need to touch upon the user behaviour part. We also acknowledge that although it may not be feasible to change a user's behaviour per se, it seems possible to roughly gauge a user's behaviour based on patterns and preferences. Making the autonomic system adapt to user behaviour is possible. Reeves and Nass show that users prefer agents that match their own personality [ReNa2003]. Having autonomic systems make adjustments to user behaviour and thereafter conveying the same to the user will make it appear more appealing, trustworthy and convincing. Moon demonstrated that a computer which uses a strategy of reciprocal, deepening self disclosure in its text based conversation with the agent will cause the user to rate it as more attractive particularly in e-Commerce applications [Moon1998].

This definition of trust is important since it brings into limelight the user ought to be assessed personally in order to build trust with autonomic software. Scenario questions and questions to gauge his or her temperament before using autonomic software may help the system decide the level of risk the system can take. Some researchers have recommended that users play a game before they get to use software to test their temperaments and their willingness to act in different situations. Sousa stores user preferences and user knowledge (level of user expertise) to build truly usable adaptive systems [Sous2005]. We understand that user interfaces are more often than not a characteristic of user behaviour and preferences. Available devices are often a function of the task behaviour. This reassures our understanding that defining trust as a behaviour is justified.

3.1.3.2 Introducing New Trust Categories

The three categories of trust as identified by Lee *et al.* provide a good starting point to help understand trust as a concept by itself [LeSe2004]. However, based on our research, these three categories of trust seem to be insufficient to fully cover the

concept of trust. There seem to be many more than just three dimensions to the definitions. The following sections introduce additional categories for our trust framework based on definition.

Trust as a belief and confidence

Belief is different from expectations. While belief is a degree of conviction of the truth of something especially based on a consideration or examination of the evidence, expectation is more related to prospects, especially of success or gain. Expectation is a mental picture of the future with the feeling that something is about to happen.

Belief is different from behaviour. Behaviour is more motor related or related to physical action while belief is more psychological in nature. Behaviour is the manner in which something or someone functions or operates [Dict2005]. Belief is the mental act, condition, or habit of placing trust or confidence in one another. The mental acceptance of and conviction in the truth, actuality, or validity of something may be understood as a belief [HMKM2005]. Trust may thus, be justified as belief in the system characteristics, specifically belief in the system's competence, dependability and security of the system, under conditions of risk [Kini1998]. Ring [RiVa1994] says that trust is the confidence of goodwill in others when you are vulnerable. Trust at the individual level is conceptualized as a belief, expectancy or feeling that is deeply rooted in the personality, with its origins in the individual early psychological development [LeBu1996]. Trust is a positive belief about the perceived reliability of, dependability of, or confidence in a person, object, or process [TsFo1999]. Webster's dictionary defines trust as reliance on character, ability, strength, or truth of someone or something.

Since we appreciate the distinction between *belief*, *expectations* and *behaviour*, it is important to acknowledge trust as a belief. Researchers have stated that autonomic decisions must be correct to the user's belief [RMDN2003]. Hence, our classification of trust as a belief and its application to autonomic systems seem to be worthwhile. Based on our analysis and general experiences with life, we understand that providing information to the user about the system state, intensions and operation modes in part will help reassure the user's belief. Hence, our work which is an explicit layout of the nature and type of information to be expressed to the user fits well.

Trust as a process of uncertainty reduction and an element of risk

Trust is a process of uncertainty reduction, the ultimate goal of which is to reinforce assumptions about a partner's dependability with actual evidence from the partners' behaviour. Reliance upon information received from the other person about uncertain environmental states and their accompanying outcomes in risky situations [ScHT1973]. This category of trust may also be related to Sousa's [Sous2005] second category of task models where task descriptions play the role of guiding users along complex tasks. Task descriptions and guides help reduce uncertainty in the minds of the user.

We understand that providing the user with relevant information thereby allowing the user to remain in the loop during operation of an autonomic system will help the user in making informed decisions. We perceive that communication and transparent autonomic systems will help reduce uncertainty and aid in risk assessment.

3.2 Chapter Summary

This chapter discussed our framework of trust. This chapter will help designers understand the basics of trust which will in turn help in the development of trustworthy autonomic applications.

Next, we present an explicit layout of a subset of the nature and type of information to be exchanged between the user and the autonomic system to gain the users confidence in its actions.

Chapter 4: Type and Nature of Information Exchange to Build Trust

Trust is formed by demonstrating to the user that the system makes correct and expected decisions [CSAW2005]. In this chapter we present an explicit layout of a subset of the nature and type of information to be exchanged between an autonomic system and its users to gain the users' confidence in its actions. While outlining the nature and type of information exchange, we try to make corresponding relations to our framework of trust and classify each piece of information into a class of trust as introduced and discussed in the previous chapter.

In order to lay out the pieces of information that build trust with users, we inspect the information exchanged in Tivoli, a successfully deployed autonomic system. Tivoli was inspected by the author of this thesis over a period of three hours. The following is a summary of the exercises performed over these three hours:

- **Exercise 1**: Taking a tour of the Tivoli Provisioning manager
  - Step 1: Signing on to the Tivoli Provisioning manager
  - Step 2: Accessing help
  - Step 3: Finding objects in the Management interface
- **Exercise 2**: Creating a Customer Application
  - Step 1: Creating a new application
  - Step 2: Adding a tier
  - Step 3: Assigning workflows to the tier
  - Step 4: Adding a dedicated server to the tier
  - Step 5: Adding an overflow server to the tier

- **Exercise 3:** Creating a Workflow
    - Step 1: Creating a new automation package
    - Step 2: Creating a new workflow file
    - Step 3: Creating the MyFirstWorkflow workflow
    - Step 4: Compiling and running your workflow
    - Step 5: Viewing workflow results
- **Exercise 4:** Enabling Access Control
    - Step 1: Enabling access control
    - Step 2: Creating an access group
    - Step 3: Creating a new user
    - Step 4: Adding access permissions
    - Step 5: Adding objects to an access group
    - Step 6: Logging on

To identify the missing pieces in the information exchange between Tivoli and its users we employed Sousa's user task models for adaptive systems [Sous2005]. In this chapter, we also offer recommendations on how to augment the Tivoli user-system interaction to improve the incremental trust building. These recommendations might help the designers of Tivoli in making the system more trustworthy.

The following sections describe the nature and type of information exchange required to build trust in detail.

4.1 Type of Information

From Chapter 1, we know that the characteristic of an autonomic system is that it possesses any subset of the following characteristics: self-configuration, self-protecting, self-healing, or self-optimization.

Therefore, while identifying the 'type' of information to be exchanged with the user of an autonomic system, we are directed to consider, the information exchange which would make the user aware of the system's capabilities with respect to these characteristics. Therefore, the type of information to be exchanged between an autonomic application and the user of the application may be divided into the following categories: self- configuration, self-protecting, self-healing, and self-optimization.

4.1.1 Self-Configuration

Autonomic software systems may need to configure and re-configure resources both at load time and at run time under varying and unpredictable conditions. This action may be initiated by the need to adjust the allocation of resources based on the current optimization criteria or in response to hardware or firmware faults. If a system is capable of self-configuration, it is significant to apprise the user of such capabilities so as to gain user confidence. Information that would reflect self-configuring capabilities would include the information that communicates to the user, for example, the ability to concurrently add or remove hardware or software resources in response to commands from administrators, service personnel, or hardware resource management software. In the last few years, we have become accustomed to regular self-configuration by updating the components of our operating systems, desktop applications and virus database automatically.

4.1.2 Self-Optimization

Self-optimizing capabilities refer to those capabilities of computing systems that allow the autonomous measurement of the performance or usage of resources and then tune the configuration of hardware resources to deliver improved performance. In an autonomic system, which is capable of self-optimization, we perceive that the exchange of information that reflects these capabilities to the user will foster confidence and build trust in autonomic systems. "An autonomic system never settles for the status quo—it always looks for ways to optimize its workings" [Horn2001]. For example, QoS requirements (e.g., for data or compute servers) can be satisfied and optimized using autonomic computing solutions.

4.1.3 Self-Healing

With self-healing capabilities, platforms can detect hardware and firmware faults instantly and then contain the effects of the faults within defined boundaries. This allows platforms to recover from the negative effects of such faults with minimal or no impact on the execution of operating system and user-level workloads. Hence, information exchange that reflects these capabilities to the user is understood to foster confidence and build trust in autonomic systems. Just like our nervous system, an autonomic system ought to recover from routine and extraordinary events that might cause some of its parts to malfunction.

4.1.4 Self-Protection

Self-protecting allows computing systems to protect against internal and external threats to the integrity and privacy of applications and data. Hence, information exchange that reflects these capabilities to the user will foster confidence and build trust in autonomic systems. As we all know by now, the virtual world is as dangerous as the physical one. Thus, an autonomic computing system must be an expert in self-protection [Horn2001].

4.2 Nature of Information Exchange

Having laid out the type of information required for building trust with autonomic applications, we now characterize the nature of information exchange. This section discusses each piece of information in detail. Table 4.1 presents an overview of our trust framework (as introduced in the previous chapte) in tabular format for reference purposes.

Table 4.1: Trust framework in autonomic systems

Trust framework	Trust framework (sub-classification)
Type of trust	Simple
	Blind
	Authentic
Basis for development	History-based
	Personal disposition
	Shared category membership
	Role behaviour
Definition of trust	Attitude or expectation
	Intention or willingness to act
	User behaviour
	Belief and confidence
	Uncertainty reduction and An element of risk

Table 4.2 below outlines the nature and type of information required to be exchanged. The expansion of the nature and type of information falling under the Type of Trust and Basis for Development of Trust categories has been left as areas for future work. For the purpose of this research, we delve into the definition of trust and enlist the nature and type of information that fall under it. This section specifies the levels of trust at which the respective piece of information should be exchanged. In Tables 4.2a-4.2e, we list each type of information and map it correspondingly to our trust framework. Table 4.2a-4.2e also specifies the levels of trust for information exchange.

Table 4.2a: Level of trust and nature and type of information exchange to build trust with autonomic systems for the attitude or expectation category

Trust framework (sub-classification)	Type of information	Nature of information
Attitude or expectation	All	Expectations with regards to functionality and goal oriented performance assessment
	All	Modus operandi or the plan to solve an issue in hand
	Self-configuration	User-perceived state
	Self-optimization	User expectations towards QoS such as throughput

Table 4.2b: Level of trust and nature and type of information exchange to build trust with autonomic systems for the intention or willingness to act category

Trust framework (sub-classification)	Type of information	Nature of information
Intention or willingness to act	All	Service awareness
	All	Context awareness
	Self-optimization	Resource awareness

Table 4.2c: Level of trust and nature and type of information exchange to build trust with autonomic systems for the belief and confidence category

Trust framework (sub-classification)	Type of information	Nature of information
Belief and confidence	Self-configuration	Environment details such as the set of suppliers, materials (an information asset such as a file or data stream) and resources accessible to a user at a particular location
	All	Context such as set of human-perceived attributes like physical location.
	Self-configuration	Configuration details
	Self-optimization	Status indicators
	Self-configuration	Displaying data already entered
	Self-optimization and self-configuration	Continuous visibility via tracking
	Self-protection	Recourse (recall or undo)
	Self-protection	Trial runs

Table 4.2d: Level of trust and nature and type of information exchange to build trust with autonomic systems for the user behaviour category

Trust framework (sub-classification)	Type of information	Nature of information
User Behaviour	Self-configuration	User-attributes should be saved in the environment model.
	Self-configuration	Task-specific preferences with respect to alternative configurations for supporting the task.
	Self-optimization	Supplier
	Self-optimization	QoS
	All	Specification

Table 4.2e: Level of trust and nature and type of information exchange to build trust with autonomic systems for the uncertainty reduction category

Trust framework (sub-classification)	Type of information	Nature of information
Uncertainty reduction and an element of risk	Self-protection	Alternative suppliers to support a service
	Self-optimization	QoS aspects such as response time, accuracy, image resolution, frame rate.
	Self-configuration	Changes in user task
	Self-configuration	Changes in environment
	Self-healing	Error messages
	Self-healing	Unforeseen circumstances with recommendations to correct them

LEGEND

At all levels of trust	
At levels of full trust	
At levels of partial trust	
At levels of partial and full trust	
At levels of no trust and partial trust	
At level of no trust	

Sections 4.2.1 to 4.2.5 describe each piece of information in detail and how it relates to autonomic software like Tivoli. We also consider user-task models for adaptive systems to analyze pieces of information exchange that may help build trust and identify the ones that are missing in autonomic systems such as Tivoli.

4.2.1 User expectations

Mayer *et al.* state that in order to build trust, the trustee must communicate and demonstrate to the other party that it is performing an important action as is expected [MaDS1995]. Autonomic systems must communicate with the user so that he or she can decide if the system is meeting the expectations or not. Also, there would be pieces of information that users would expect the system to communicate to them. Hence, the importance of information to fulfill or conform to user expectations needs to be taken into consideration. With respect to autonomic applications, users may have several expectations from the system as listed in the sections below.

4.2.1.1 Functionality and goal-oriented performance assessment

Lee *et al.* state that automation has a perspective that is related to the goal-oriented nature to assess performance [LeSe2004]. While working with software applications users are goal-oriented. Thus, it is worthwhile to acknowledge the communication of functionality of a system (or module), as an expectation from the user.

Functionality of a system could be with regards configuration, optimization, healing or protection. Hence, this piece of information falls under the 'All' type of information category.

Level of trust for information exchange

At levels of no trust and partial trust, the user is understood to doubt the doings of the autonomic system. Hence, the user might like to be informed of the goals of the system.

With the increase in the level of user trust in the system, the user would be more interested in the results than the goal of the system. Hence, we feel that the need to communicate the functionality of the system should tend to decline with increasing levels of trust.

Application with respect to Tivoli

In the Tivoli Storage Manager, there exist brief descriptions of the functionality and goal of each module on top of the controls to perform the tasks (cf. Figure 4.1). Communicating information of this sort may be seen as a user experience goal because it helps the user. Users can recognize rather than having to recall the purpose of the module. Also, this adheres to Nielsen's principle of usability heuristics because a description of functionality provides the user with help and documentation [Nielsen 1994b].

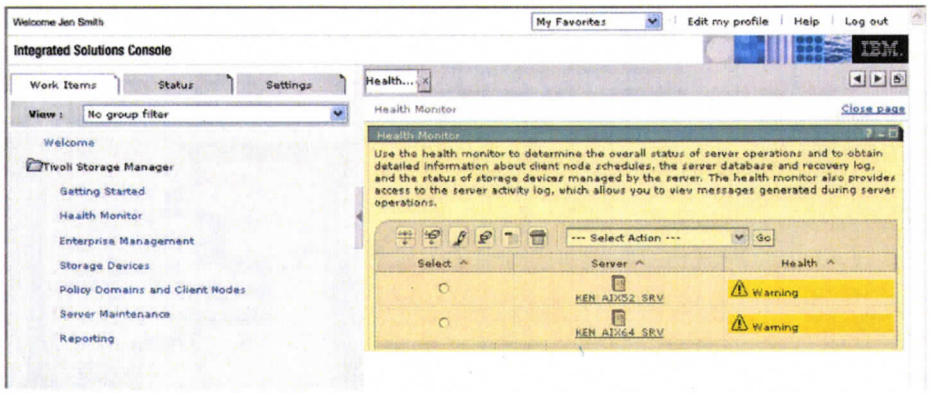

Figure 4.1: Tivoli Store Manager: health monitor

4.2.1.2 Modus operandi

Users may also like to know in some cases, the modus operandi or the plan to solve an issue on hand (depending on the level of interest and expertise of the user). Modus operandi may be seen as the mode of operation of a user to accomplish a certain task in an autonomic system. In addition to the system communicating the purpose and/or procedure of a certain action on its part, the system may also like to inform the user of the steps he or she needs to take to cooperatively perform a given task when working towards a common goal, and indicate when human intervention may be required.

System adaptation and implementation of this information is described in Section 4.1.4 in detail. Satisfying and communicating this expectation to the user seems to be feasible. This is because autonomic systems are policy-driven and have a knowledgebase. Hence, the communication of the plan of how a task will be carried out should not be hard to achieve. During our research we discovered surveys that demonstrate how the communication of modus operandi is important. For example, a survey demonstrated by Whitten and Tygar revealed that the majority of users of PGP software were unable to use the system to perform the desired task [WhTy1999]. This was because the PGP interface failed to provide comprehensive information about how the public key encryption works and the roles and uses of the public and private keys. 25% of the people surveyed emailed secret information without any protection. An analysis of errors and an evaluation of the interface led these researches to conclude that the major source of the problems was that the users did not understand the public key model used in PGP.

Modus operandi of a system could be related to system configuration, optimization, healing or protection. Hence, this piece of information falls under the 'All' type of information category.

Level of trust for information exchange

At levels of no trust and partial trust, we believe users will be more skeptical about how the system operates. Once the user gets a fair idea of how the system operates, the user should be more interested in the final results. Hence, our understanding is that when a user starts to believe in the capabilities of the autonomic system, the system will not need to communicate modus operandi details under normal conditions.

Application with respect to Tivoli

In the Tivoli Storage Manager, the system presents *Hints and Tips* to the user to accomplish a given task (cf. Figure 4.2). This helps fulfill users' expectation of being communicated of the modus operandi. Such a communication is expected to reduce human errors and allow the required results to be achieved more quickly and easily. Autonomic systems today are seen as intelligent software applications, capable of making decisions while offering recommendations to users. Therefore, autonomic systems are

expected to demonstrate its capabilities while providing full support to the user when human intervention or attention is required.

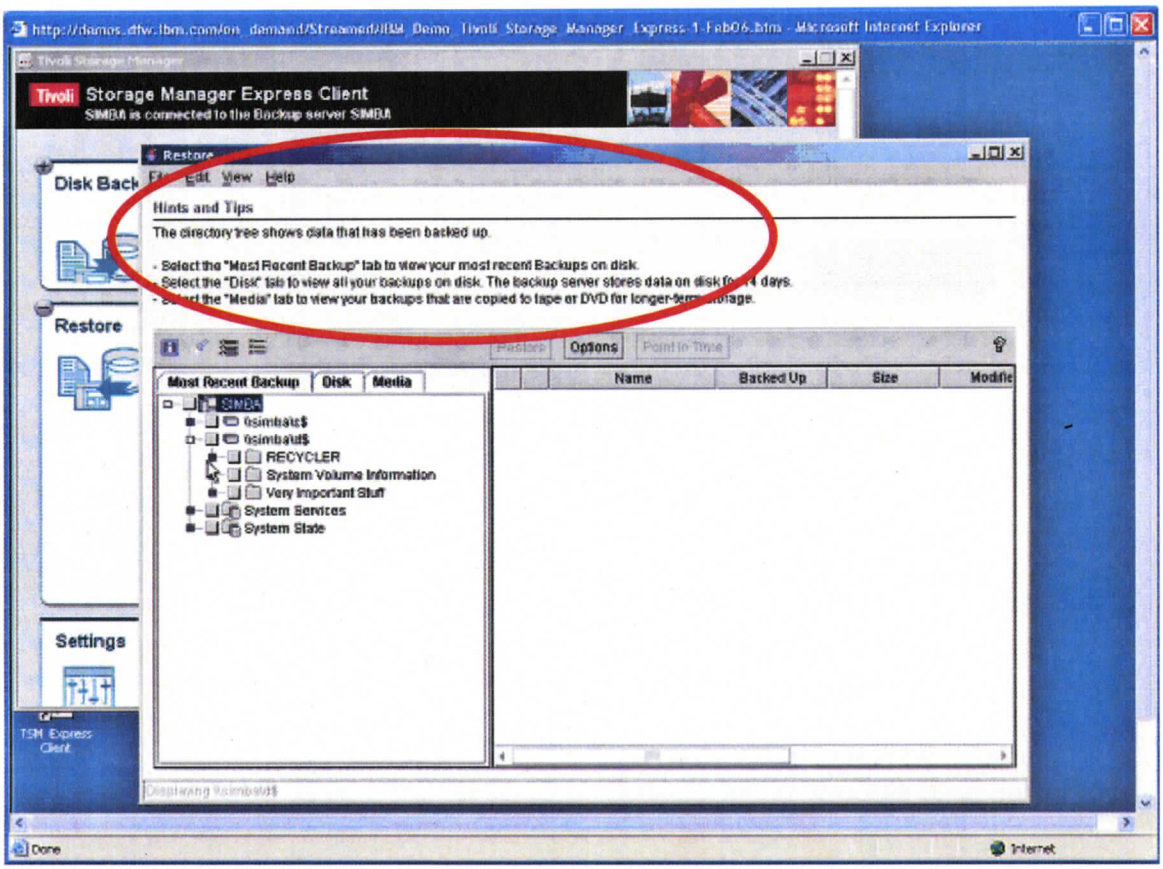

Figure 4.2: Tivoli Storage Manager: the restore feature with Hints and Tips

4.2.1.3 User-perceived state

In the present context, user perceived state would refer to a set of properties in the environment that characterizes the support for the task. Specifically, user perceived state is seen as the user-level settings (e.g., preferences, options) associated with each of the services supporting the task, the materials being worked on, user-interaction parameters (e.g., window size, cursor), and the user preferences for the task. Reeves and Nass also show that users prefer agents that match their own personality [ReNa1996]. System adaptation and an implementation of the mode of communicating this information are described in Section 4.1.4 in detail. Initially and ideally, users will be

required to define their preferences with respect to state. With time, the system must learn to adapt to these preferences. Once users have defined their preferred state(s), they will expect the system to display it. In other words, an autonomic system is expected to demonstrate its capabilities to 'autonomically' adjust to user-perceived state(s).

Information to communicate the state as perceived by the user may require adding or removing resources. Self-configuration by example is the ability to concurrently add or remove hardware resources in response to commands from administrators, service personnel, or hardware resource management software. Hence, this piece of information falls under the self-configuration type of information category.

Level of trust for information exchange

At levels of partial trust and full trust, we understand that the user will start to expect things from the autonomic application. The user may start developing expectation from the system with regards to the environment settings and so on. In such a scenario, the user is also likely to make perceptions on the system state. Thus, demonstrating that the system conforms to the user perceived state at levels of partial and full trust seems to be beneficial.

Application with respect to Tivoli

In the Tivoli Storage Manager, the system allows the user to configure and specify user preferences for accomplishing a given task (cf. Figure 4.3). The user can make modifications as per his or her requirements and desires. They may filter out the pieces of information deemed irrelevant or of relatively low importance at a particular point in time. We believe that saving and storing such preferences made by the user, and allowing the flexibility to change the same at a later point, should not be a difficult task. The system may learn to adjust to these preferences once they have been set by the user. This functionality in the autonomic application offers flexibility and control over the system by the human allowing the required results to be achieved more quickly and easily. This is also a user experience goal because it supports creativity. For example, configuring certain parameters may allow alternative views to present a report. This feature is emotionally fulfilling and hence again a user experience goal because the user

can control the way they want the environment to look. Freedom, flexibility and user control are included in the Usability Heuristics Principles of Nielsen for good design [Nielsen 1994b]. Hence this piece of detail also conforms to principles from Human Computer Interaction.

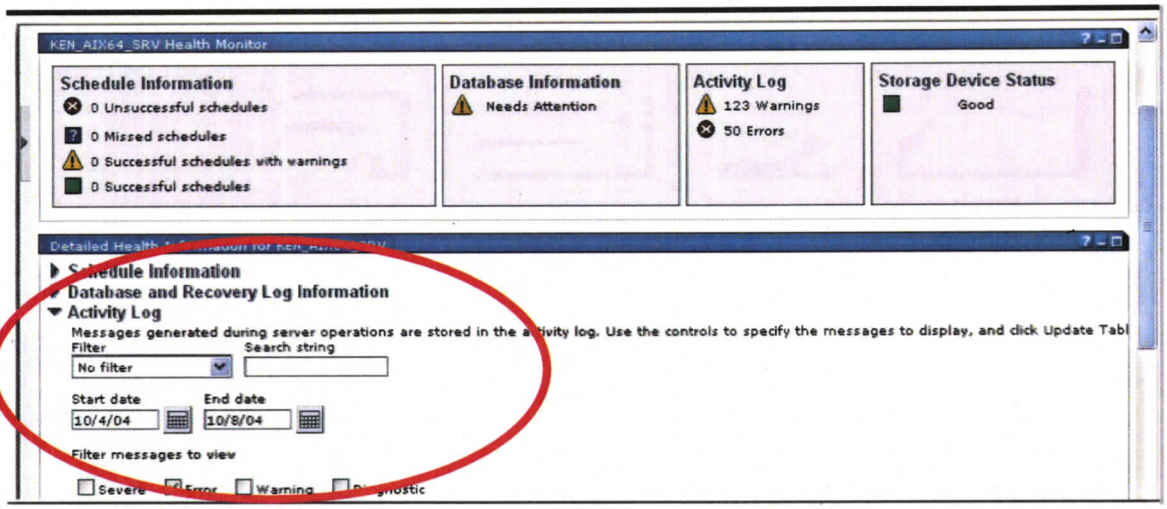

Figure 4.3: Tivoli Storage Manager: setting preferences and filters to suit user perceived state

4.2.1.4 Quality of Service

In the present context, QoS refers to the acceptable levels of QoS and the preferred tradeoffs. User's expectations with regards to QoS typically relate to response time. QoS preferences play a key role in guiding the adaptation policies within resource-adaptive applications while users carry out their tasks [HIPD2003]. Additionally, during automatic configuration, QoS preferences play a role in choosing among alternative service suppliers, and in determining the optimal resource allocation among the service suppliers involved in a task [LMRW2005].

Information on QoS would include details such as throughput or turnaround time. QoS in itself is a measurement of performance. Self-optimizing capabilities refer to those capabilities of computing systems that allow the autonomous measurement of the

performance of hardware resources to deliver improved performance. Hence, this piece of information is in sync with the self-optimization type of information.

Level of trust for information exchange

We recommend QoS details to be exchanged at all levels of trust. This is because the QoS preferences for a user may change with time and tasks. Users may prefer to have different QoS tradeoffs for a given service in different tasks [Sous2005]. Sousa in his Ph.D. thesis gives an instance wherein Fred is watching a video over a network link and the bandwidth suddenly drops. Now, reducing the frame rate or the image quality will depend on Fred's preferences for the current task. If Fred is watching a sports event, he may prefer frame rate to be preserved at the expense of image quality. On the other hand, for watching a documentary on painting, the reverse may be preferable. Furthermore, the preferred QoS tradeoffs may change during the task. Sousa cites an instance where there is a task that involves automatic translation of natural language. The users in this case may prefer faster responses over accuracy of translation during the introductory part of the conversation, but they may prefer the opposite once the conversation gets more involved. According to Sousa, while the simplest form of expressing a tradeoff is to indicate which dimension is preferred, this form has very limited expressive power. For instance, a user might indicate that response time is preferred over accuracy of translation. However, how short of a response time will satiate the user? And even if accuracy is less important, what if it degrades so much that the translations become unusable? A clearly more powerful form is to express the thresholds that characterize a tradeoff. For example, if Fred requires highly accurate translations, he may be willing to wait up to 30 seconds for an answer. In Sousa's work, QoS tradeoffs are set by expressing user happiness with the level of quality provided along each QoS dimension. For example, when faster responses are preferred over accuracy of translation, the QoS preferences set stricter happiness thresholds for response time and looser thresholds for accuracy. The tradeoff can be reversed by relaxing the thresholds on response time and tightening the thresholds on accuracy. Therefore, QoS aspects keep changing over time and, hence, are recommended to be communicated to the user at all times.

Application with respect to Tivoli

In the Tivoli Storage Manager, the system displays the response time the user may expect (cf. Figure 4.4). This is satisfying and hence also a good experience for the user since the system is accountable and transparent in its doings. However, the application does not allow the user to set or change QoS preferences for scaling over user tasks and time. We suggest incorporating capabilities as suggested as discussed above.

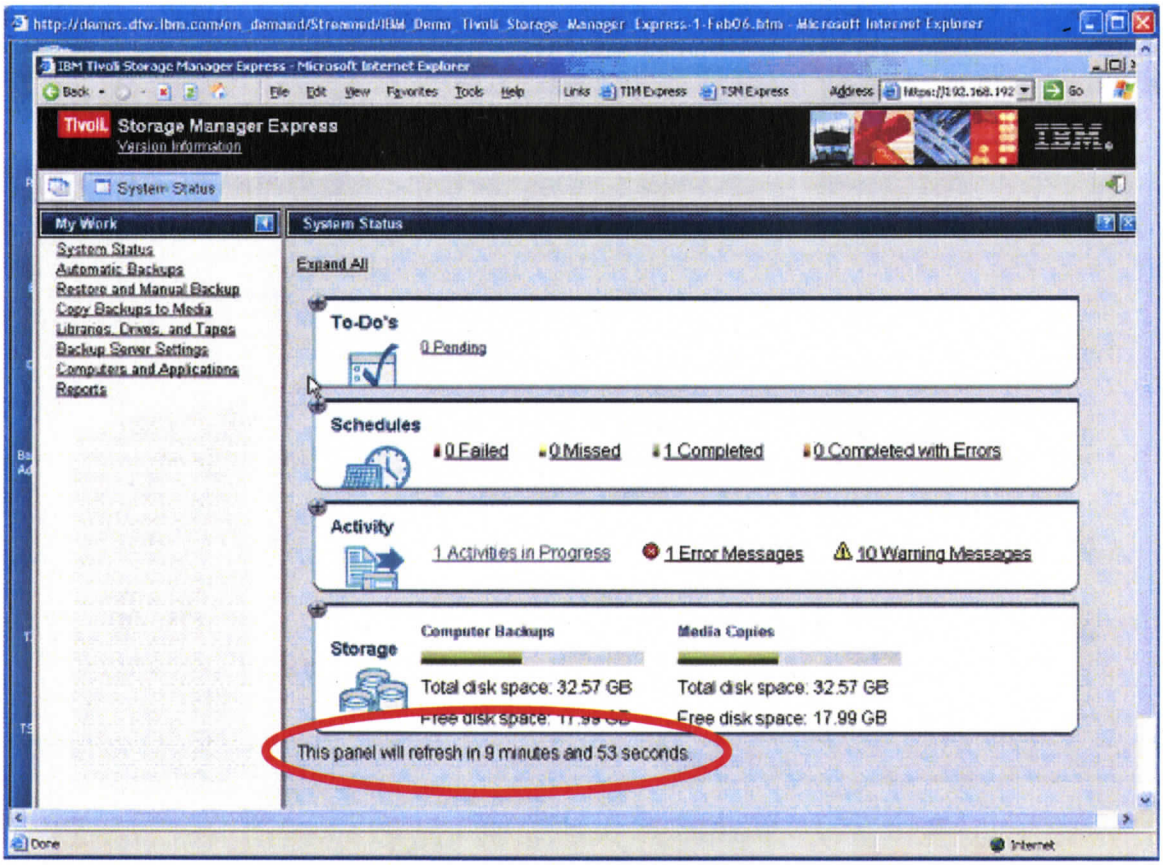

Figure 4.4: Tivoli Storage Manager: the system status feature depicts expected refresh rate

4.2.2 Intention and willingness to act on tasks and services

4.2.2.1 Service awareness – tracking tasks or services

In the present context, service awareness refers to the engagement or the employment of the system's tasks or processes. According to Sousa [Sous2005], the fault tolerant and load balancing systems are the most primitive forms of service aware systems. He says that today service awareness includes suppliers providing a service for a task, qualitative models of QoS to guide adaptation policies, and so on. Under the following umbrella, we see users to naturally expect their autonomic system to be service aware.

Service awareness could cater to system needs such as configuration, optimization, healing or protection. Hence this piece of information falls under the 'All' type of information category.

Level of trust for information exchange

The ability to track tasks or services should be made possible at all instances. At level of no trust, the user will be extra conscious of the progress of tasks being performed and is therefore likely to be interested in such information. At levels of trust partial trust, the user is understood to be in a transition state from controlling the system to 'letting go'. Hence we perceive that there might be instances where the user may like to confirm that the system is making progress. At the level of full trust, the user may still need to track tasks or services. This information may be helpful for the user to schedule other tasks. Hence at the level of full trust, the user may still like to know that the system is self-aware and willing to account to the user.

Application with respect to Tivoli

In the Tivoli Storage Manager, the system displays service awareness (Figure 4.5) on the aforementioned terms. The TODO list, the Scheduler and the Activity information as depicted in Figure 4.5, all communicate the engagement or the employment of the system's tasks or processes. In Figure 4.5, the QoS aspects to service may be seen

under schedule which displays the accuracy or success rate with respect to tasks completed.

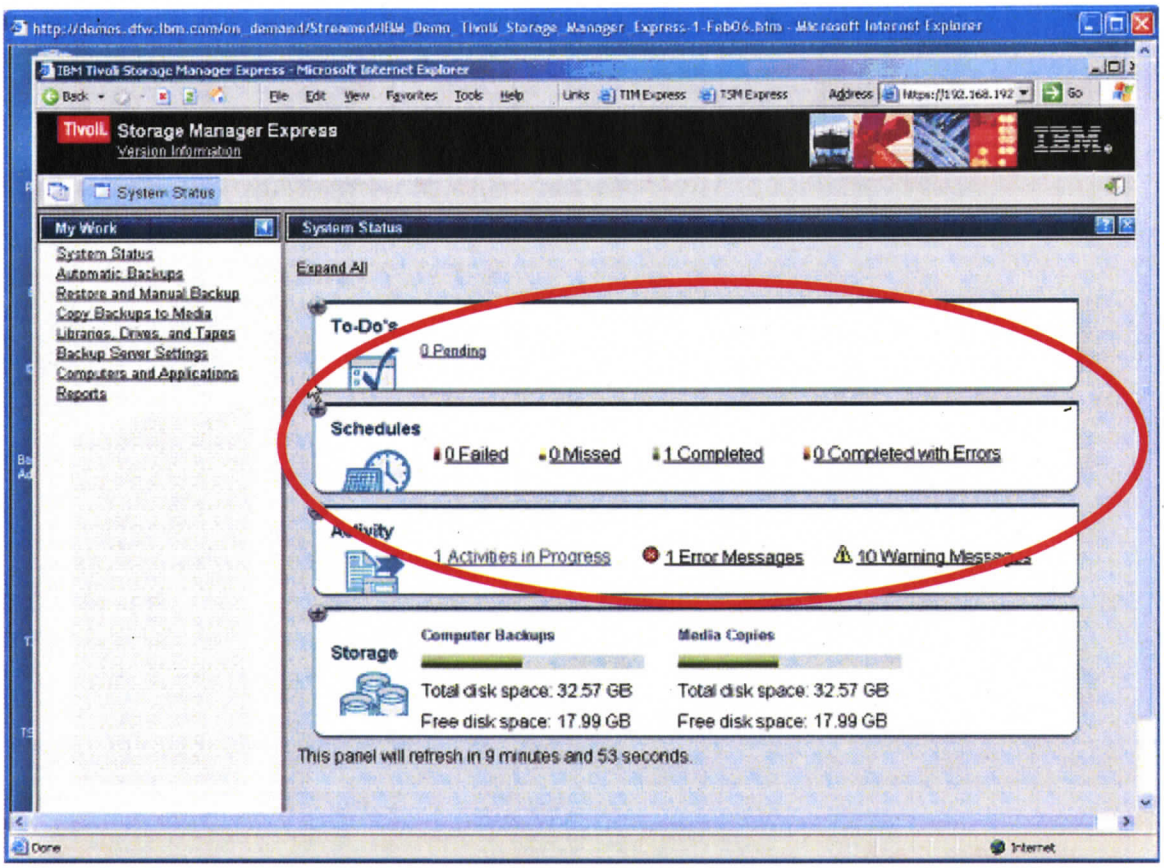

Figure 4.5: Tivoli Storage Manager: the system status feature depicts state of the system

4.2.2.2 Context awareness – event detection

In the present context, context awareness refers to circumstances in which an event is placed or occurs. Context-aware systems react to variations in the physical context around users [CSAW2005]. Examples of observed variables are: user location, attention focus, physical activity, emotional state (relaxed, working, in distress), privacy (who else is in the vicinity), and so on. Typical context-aware systems represent such awareness as collections of interpreted rules (clauses of the form "if context then action"), or embedded logic in the code [CSAW2005].

Context awareness of a system could be referred to in relations to system configuration, optimization, healing or protection. Hence this piece of information falls under the 'All' type of information category.

Level of trust for information exchange

Information to demonstrate the system's context awareness should be made possible at all instances. At level of no trust, the user will be extra conscious of what is being done and why. Therefore, the user is likely to be interested in context information at the level of no trust. As mentioned above, at levels of trust partial trust, the user is understood to be in a transition state wherein he or she is moving from the notion of controlling the system to simply 'letting go'. Hence we perceive that there might be instances where the user may like to confirm that the system is making progress in the right context. At the level of full trust, the user may still need to detect events and contexts in which they occur. This information may be helpful for the user to schedule other tasks.

Application with respect to Tivoli

In Tivoli Storage Manager, and some Windows applications as well, we see the hierarchical display of file structures and directories as a mode of presenting context awareness (Figure 4.6). Each file or subsection is contained within a super-file which gives meaning and context to the existence of the file itself. However, Tivoli does not seem to be context aware in view of the above definition. Hence, we believe there is scope for improvement in Tivoli in this regard. Sousa presents an architectural framework describing ways to sense changes in context and adapt to them [Sous2005].

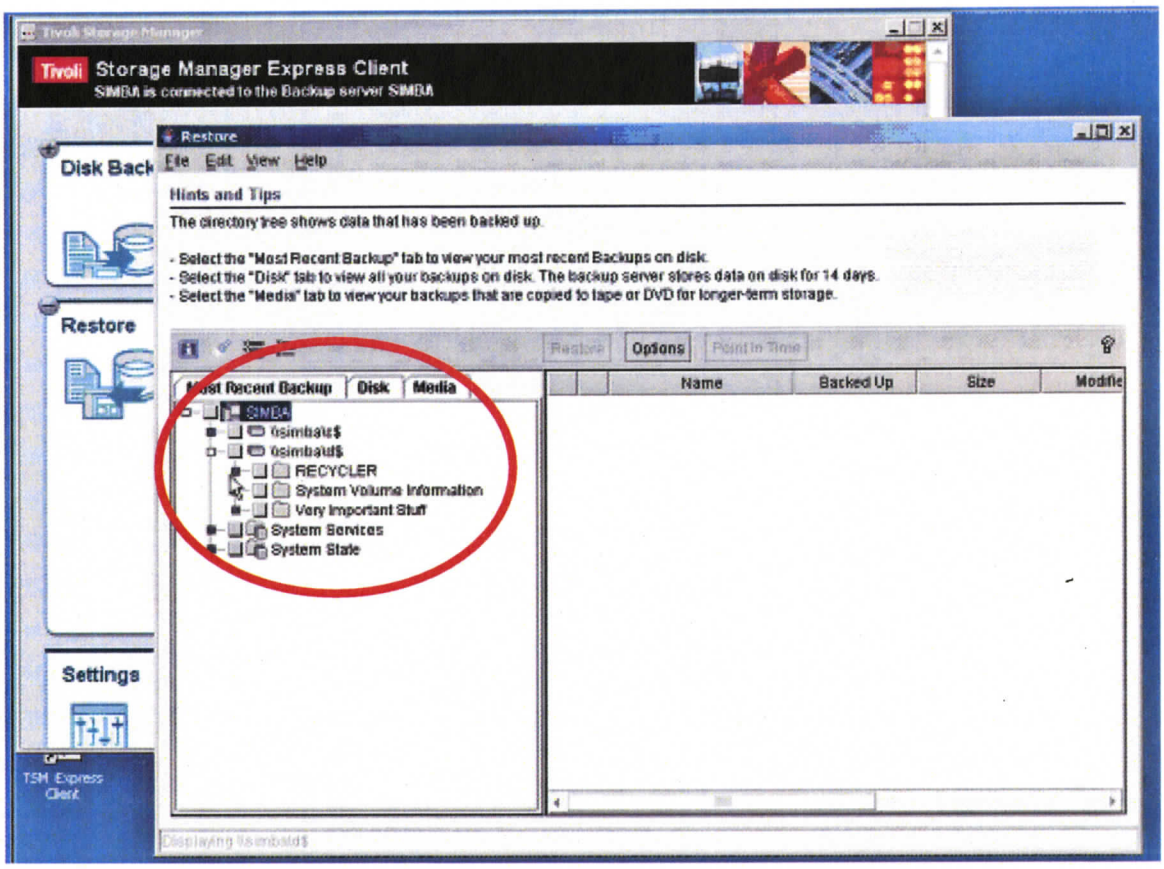

Figure 4.6: Tivoli Storage Manager: the restore feature depicts file hierarchy

4.2.2.3 Resource awareness – support and maintenance of services

In the current context, a resource can be seen as something that can be used for support or help [Dict2005]. Resource-aware systems react to resource variation: components adapt their computing strategies so they can function optimally with the current set of resources (bandwidth, memory, CPU, power, and so on) [CSAW2005]. The adaptation policies in such resource-aware systems are often hard-coded. However, which adaptation policies are appropriate at each moment depends on the user preferences for the current task. And user preferences change dynamically, as users switch between tasks, or even in the middle of a task. Sousa's research associates user preferences to each task, and provides mechanisms to communicate those preferences dynamically to all the components supporting the task. Such preferences determine the appropriate resource allocation, and adaptation policies.

Resource awareness of a system is generally related to the hardware and software resources available to it. Self-optimization refers to those capabilities of computing systems that allow the autonomous usage of resources and then tune the configuration of hardware resources to deliver improved performance. Hence this piece of information falls under the 'self-optimization' type of information category.

Level of trust for information exchange

The adaptation policies in resource-aware systems are often hard-coded. However, which adaptation policies are appropriate at each moment depends on the user preferences for the current task. And user preferences change dynamically, as users switch between tasks, or even in the middle of a task. Hence we recommend exchanging information related to resources at all levels of users trust.

Application with respect to Tivoli

In the Tivoli Storage Manager, the system displays the available means of storage space which shall give the user an idea of the resources at his disposal. See Figure 4.7. Hence, Tivoli displays information related to resources. However, based on the above mentioned definition of resource aware systems, we find information exchange in Tivoli to be lacking.

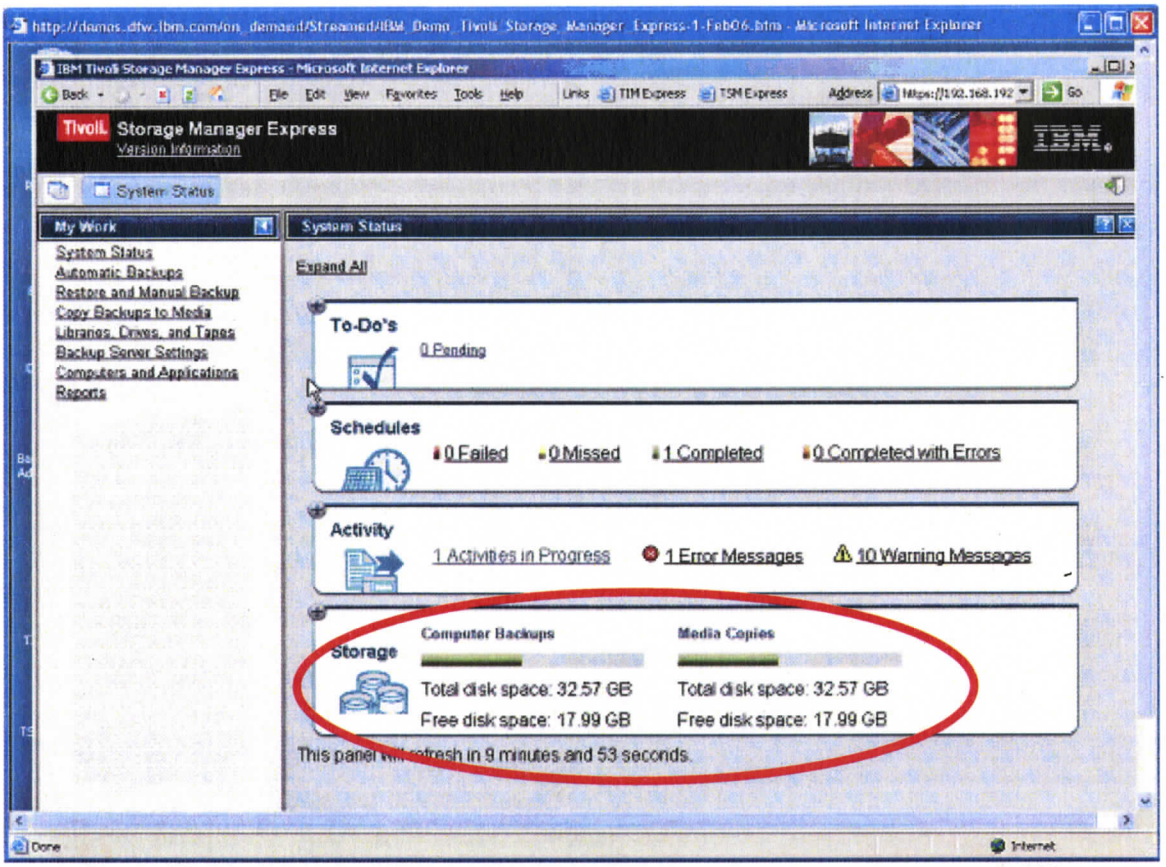

Figure 4.7: Tivoli Storage Manager: the restore feature communicates resource availability

4.2.3 Users' belief and confidence

4.2.3.1 User environment

In the current context, environment details refer to the set of suppliers, materials such as an information asset (a file or data stream), and resources that are accessible to a user at a particular location. An autonomic system that makes dynamic changes to the environment to suit user tasks and communicates the same will be seen as a system affirming to users' belief with regards to the efficiency of its operation.

Environment details in the current context refer to resources available for use. The purpose of exchanging this piece of information with the user is to allow the system to reassure the user's belief that the system is adapting to the environment to suit user tasks. Adapting to the environment to suit user tasks can require tweaking several

parameters, addition or removal of several resources. Hence this piece of information falls under the 'self-configuration' type of information category.

Level of trust for information exchange

User environment is expected to change with change in user task, system status, and time. Information that conveys changes with respect to changes in user environment is recommended to be visible to the user at all levels of trust.

Application with respect to Tivoli

The resource accessible to a user is communicated in Tivoli. The system displays the available means of storage space which shall give the user an idea of the resources at his disposal (cf. Figure 4.7).

4.2.3.2 Context awareness

We acknowledge that presenting information demonstrating that the system is context aware not only demonstrates the system's willingness to act but also reaffirms user belief. Hence we repeat this piece of information. Section 4.2.2.2 describes context awareness in detail.

4.2.3.3 Configuration details

Carrying out a task may require configuring the environment, and accessing several materials. A configuration means a set of possibly interconnected services that together support a task. The system may require the user to provide this piece of information early on to adjust itself so that it may reassure the user's belief at a later stage.

As the name would suggest, this piece of information falls under the 'self-configuration' category.

Level of trust for information exchange

Configuration details are expected to change with change in user task, system status, and time. Information that conveys changes with respect to changes in configuration is recommended to be visible to the user at all levels of trust.

Application with respect to Tivoli

During our experience with the Tivoli Storage Manager, we did not discover the software to offer such capabilities. Although Tivoli Storage Manager may have built-in characteristics and functionalities to encompass configuration management of services for an environment; we are convinced by the ideas suggested by Sousa [Sous2005] and find it more appealing. The configuration management of a multimedia adaptive application called Aura as suggested in his Ph.D. thesis is addressed as follows.

By finding the available components that best fit the user's needs for each task, the infrastructure enables users to take full advantage of diverse environments. Furthermore, by keeping track of user preferences with respect to QoS, of resources demanded by alternative computing modalities, and resource availability, the infrastructure can select the optimal configuration and carry out dynamic adaptation to changes in the environment.
By automatically configuring environments, on demand, and by continuously adapting to changes in the environment, the infrastructure reduces the burden of routine configuration and reconfiguration tasks for the user.

We believe that these ideas may be applied to any autonomic application, as is.

4.2.3.5 Status indicators

A status is a position relative to others or a state of affairs. A system wide view of aspects such as QoS would satisfy user's belief. Information on status as mentioned above can include QoS. QoS in itself is a measurement of performance. Self-optimizing capabilities as discussed on Page 42 refer to those capabilities of computing systems

that allow the autonomous measurement of the performance of hardware resources to deliver improved performance. Hence this piece of information is in sync with the 'self-optimization' type of information.

Level of trust for information exchange

The perceived reliability of, dependability of, and confidence in an object or process will vary depending on the circumstances. A status indicator is itself a position relative to the state of affairs. Hence, the status indication of an operation should be accessible to the user at all levels of trust.

Application with respect to Tivoli

In Tivoli, status indicators are available in abundance (e.g. Figures 4.8 and 4.9).

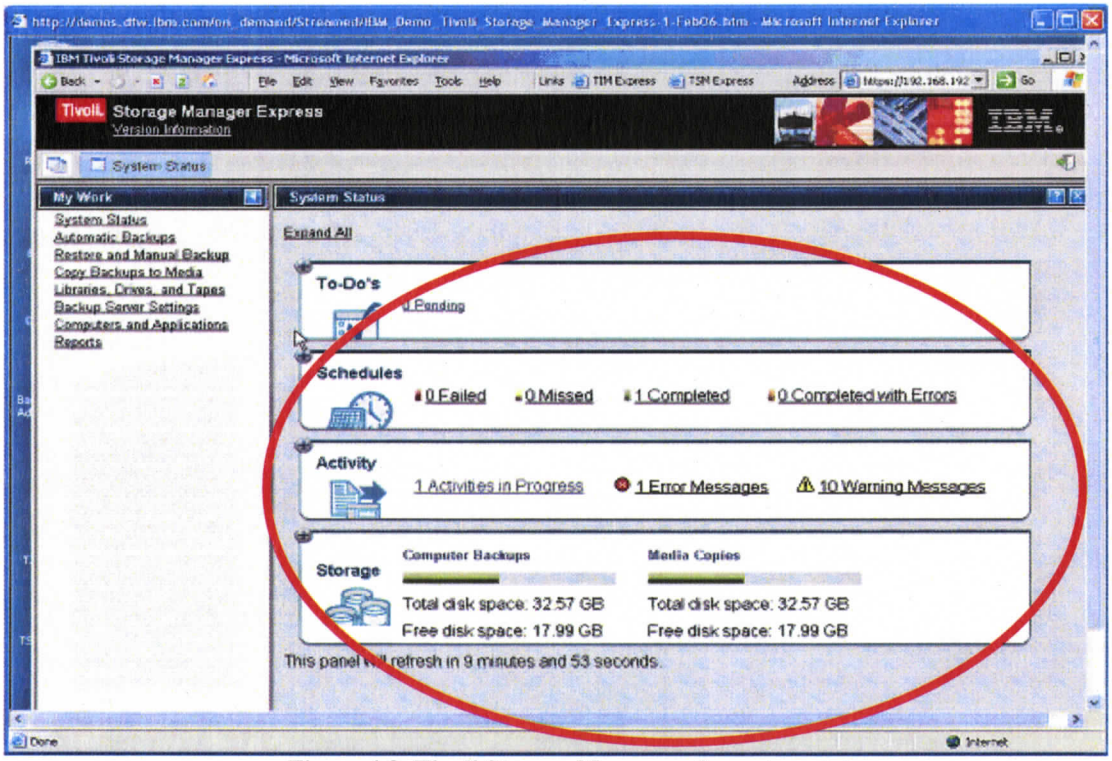

Figure 4.8: Tivoli Storage Manager: the system status

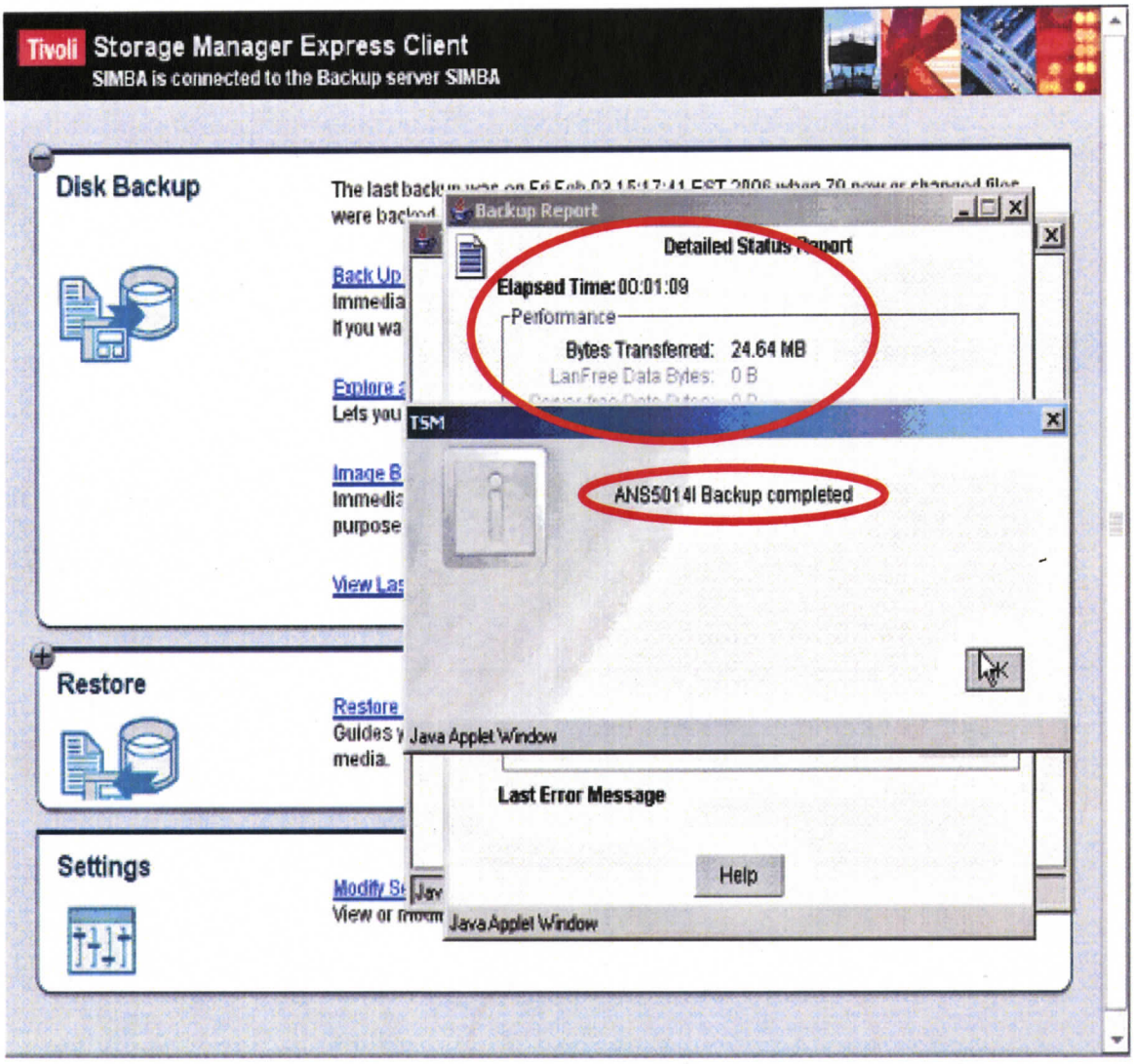

Figure 4.9: Tivoli Storage Manager: message boxes communicate the current state of the system

4.2.3.6 Data review

Displaying data or information already entered provides reassurance towards users' belief. Hence we consider this as a relevant piece of information exchange to build trust with autonomic systems.

Data review requires saving and retrieving data from a memory space. Self-configuration capabilities as discussed on Page 42 may also involve addition or removal of resources. Information may also bee seen as an intangible resource. Hence, this piece of information is in sync with the 'self-configuration' type of information.

Level of trust for information exchange

Data review must be allowed at all times. If users have no trust in the system, they will want to make sure that the system is being receptive of user input. Hence data review is important. Data review may not only mean presentation of text. In the present context, we refer data review to the display of any user activity such a created folder, resource access or modification, or (text) input in the past. Irrespective of the level of users trust in the system, not allowing users to look up what they did before makes things appear mysterious. Hence data review must be allowed at all levels of trust.

Application with respect to Tivoli

Figure 4.10 depicts a file that was once manually backed up in the Tivoli Storage Manager. Hence Tivoli allows the capability of data review. Based on our experience, other forms of data review also seemed to be possible with Tivoli.

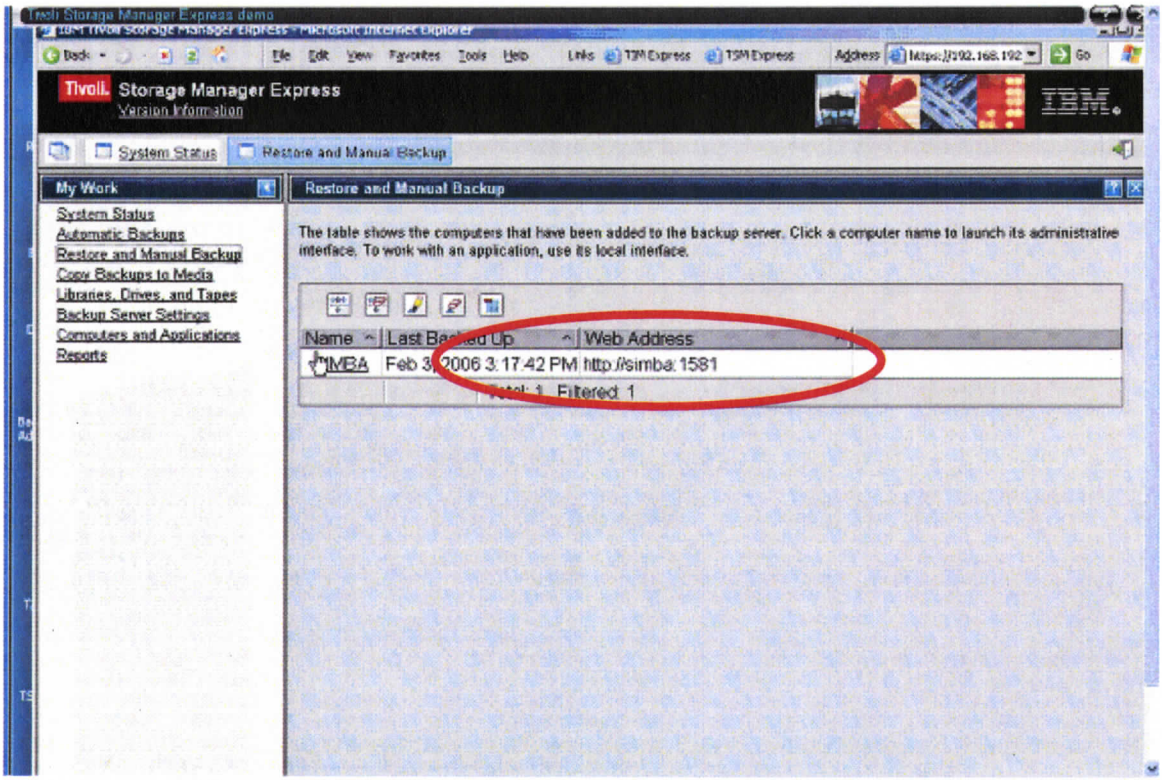

Figure 4.10: Tivoli Storage Manager: displaying data already entered

4.2.3.7 Visibility of information and tracking of processes

Russel *et al.* suggest that information required should be readily and easily accessible to the users of autonomic software. This piece of information is in sync with data review and status indicators as discussed in the previous sections.

Because, this piece of information is in sync with status indicators and data review, it falls under the 'self-configuration' and 'self-optimization' category. For more information about why this piece of information falls in the category mentioned, see Section 4.2.3.5 and 4.2.3.6.

Level of trust for information exchange

Visibility of information and tracking of processes must be exchanged at all levels of trust for reasons discussed in Sections 4.2.3.5 and 4.2.3.6.

Application with respect to Tivoli

Figure 4.8 shows how visibility via tracking is imposed in Tivoli Storage Manager.

4.2.3.8 Recourse and undo

User actions should be saved in their profiles and undo capabilities should be possible. Recall or undo options should be visible and accessible to users. Russel *et al.* also advocate autonomic systems with the capability to allow users to back out of changes easily. This reassures user belief that the system is working towards a common goal and is willing to cooperate. This will build user trust with autonomic systems.

Recourse or undo is usually done to avoid an unfavourable situation. Hence this piece of information seems to be a good fit for the 'self-protection' category of information.

Level of trust for information exchange

Users may want to back out of operations at any instance. Barring few restrictions, recourse should be allowed at all times. Therefore user steps and information entered must be visible at all levels of trust.

Application with respect to Tivoli

Based on our experience with the Tivoli Storage Manager, the system does not have recall and undo features.

4.2.3.9 Trial runs

Trial runs refer to demos and documentation. Allowing trial runs build the users' confidence in the system.

Trial runs are usually done to gain a hand over something before doing the actual thing. Hence, trial runs minimize the chances of failure or breakdowns. Hence this piece of information seems to be a good fit for the 'self-protection' category of information.

Level of trust for information exchange

We see trial runs to be more a source for building trust rather than re-enforcing trust. Hence trial runs are good to be accessible to the user at levels of no trust.

Application with respect to Tivoli

The Tivoli Storage Manager provides several online demos and has good documentation. An inexperienced user may perform trial runs on demo versions in case he or she is not confident and wishes to gain some proficiency before operating on the actual software.

4.2.4 User and task behaviour

4.2.4.1 User-attributes

User attributes should be saved in the environment model. Users with different degrees of expertise will ask for things at different levels: an inexperienced user will try to get more abstract services, hiding internal details as much as possible, while an expert user may want to have more control over which, and how the parts are configured. The same user may want to have more control over the structure supporting a critical task, but be willing to take an off-the-shelf solution for a low priority task [Sous2005]. This piece of information is in accordance with the classification of trust as an expectation.

Exchanging pieces of information to demonstrate adaptability to and match user-attributes may require hiding or presenting of tasks, capabilities, environments, information and so on. This can be seen to involve a great deal of configuration overhead on the part of the system. Hence this piece of information falls under the 'self-configuration' type of information category.

Level of trust for information exchange

We feel that with an increase in level of trust in someone or something, the level of expectation rises. Hence at the levels of partial and full trust, the user is understood to more often expect that the system adapts to its attributes. Therefore, information exchange that provides reassurance that user attributes are being stored or saved (for system adaptation to user behaviour) is required at levels of partial and full trust.

Application with respect to Tivoli

In the Tivoli Storage Manager as well, each user is assigned a login name (cf. Figure 4.11). As such, user-attributes are saved in the environment model.

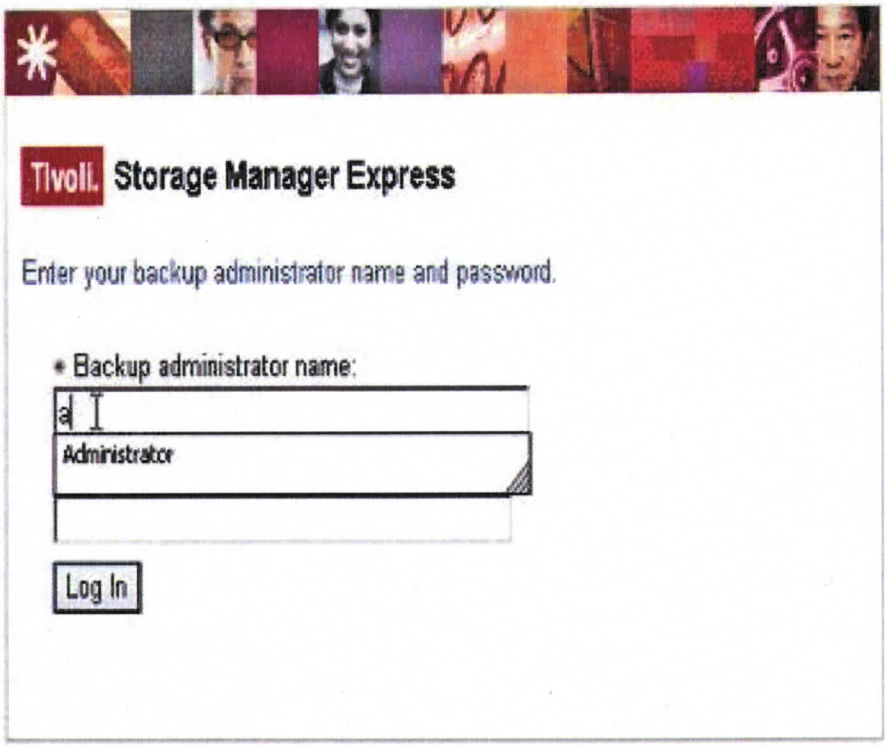

Figure 4.11: Tivoli Storage Manager: feature that facilitates logon to specific user profiles

4.2.4.2 Task-specific preferences

To address the heterogeneity of ubiquitous computing environments, and the fact that their resources are subject to frequent variation, user preferences relative to functionality and QoS must be exchanged with the system. Sousa's thesis [Sous2005] cites an instance: if a task involves editing a text document and two distinct text editors are available in the environment, the answer to which should be activated depends on how the user appraises the functionality of each editor for the task at hand. For editing natural language, the user may prefer an editor with automatic spell checking, but for editing a computer program that feature may hinder more than help [Sous2005]. This also depends a lot on the users' behaviour. Different users tend to give priorities to different tasks. Hence this piece of information falls under the user and task behaviour category.

Exchanging pieces of information to demonstrate adaptability and matching of user's task-specific preference may require hiding or presentation certain system capabilities, environments, information and so on. This can be seen to involve a great deal of configuration overhead on the part of the system. Hence this piece of information falls under the 'self-configuration' type of information category.

Level of trust for information exchange

At no level of trust, we believe the user would not expect the system to adjust or be receptive to task specific preferences of a user. However, with time as users will start to work on several tasks, their preferences will change perpetually. Hence users' task-specific preferences should be communicated to the system at levels of partial and full trust.

Application with respect to Tivoli

During our experience with the Tivoli Storage Manager, we discovered that task preferences may be specified and are communicated to the user but only at a low level. Figure 4.12 shows how a user may select an existing path for new volume or set a path reference afresh. The user may also go back and change these settings at a later stage. However, we feel it is worthwhile for Tivoli to better address task specific preferences of users. User should be allowed to specify their preferences relative to functionality and QoS.

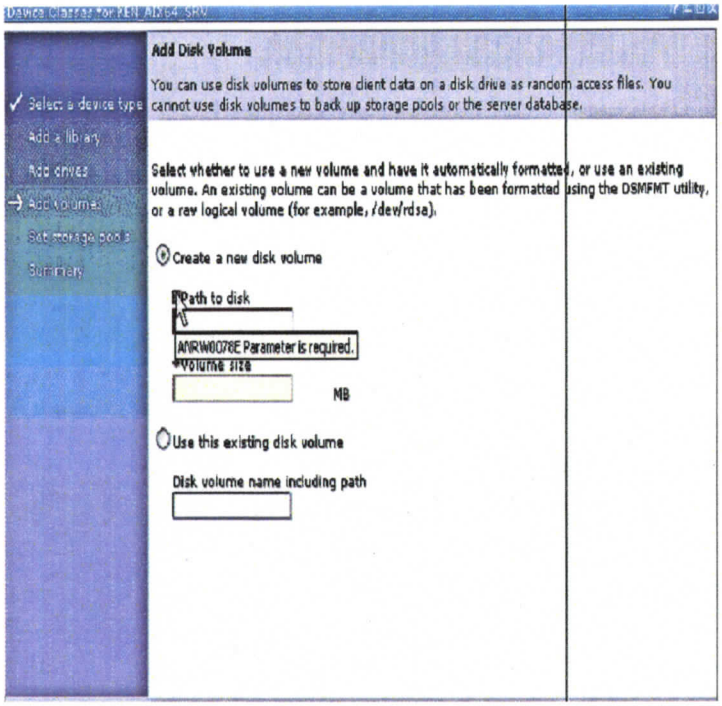

Figure 4.12: Tivoli Storage Manager: task specific preferences

4.2.4.3 Supplier information

In the present context, supplier refers to the provider of resources. Users may have preferences with respect to suppliers. Therefore, supplier information has been placed under user behaviour category.

Supplier information of a system is related to resources as mentioned above. Hence as described in Section 4.2.2.3 this piece of information falls under the 'self-optimization' type of information category.

Level of trust for information exchange

As discussed in the previous section, tasks change with time. Depending on the nature of the task, the suppliers to provide resources to complete those tasks change. Also, users' preferences keep changing from time to time. Therefore information that the system must be communicated of users preferred supplier at all instances. Information

that demonstrates to the user, the alternative suppliers of a resource needs to be communicated at all levels of trust.

Application with respect to Tivoli

Although Tivoli Storage Manager may have built-in characteristics and functionalities to encompass configuration management of services from an environment, we as researchers are more inspired by the ideas suggested by Sousa for an adaptive multimedia application called Aura [Sous2005]. Sousa suggests a Managed Environment layer (ME) to be responsible for monitoring the availability of suppliers and resources, and for optimally matching the incoming requests to the available alternatives. Upon resuming a task, the ME maps the service requests to the concrete suppliers that best match the user preferences. While tasks are being carried out, the ME monitors the environment for failures and opportunities for improvement. Should an alternative configuration become more attractive, the ME is the first to reason whether to replace one or more suppliers to reach the desired configuration. According to Sousa, a cost of change is factored into this reasoning, since users may perceive a cost whenever they are interacting directly with a supplier targeted for replacement. The Aura framework as described by Sousa, takes advantage of the knowledge about user tasks and preferences captured by an element called a Prism [Sous2005] to guide adaptation policies inside resource-aware applications. We envision that a similar functionality may be implemented in autonomic systems and will be useful to build trust.

4.2.4.4 Preferences relative to QoS

Preferences relative to QoS play a key role in guiding the policies for adaptation to the variation of resources. For that, however, we need to take into account that QoS is seldom expressed along a single dimension. For instance, suppose that the user is watching a video over a network link and that the bandwidth suddenly drops. Should an adaptive video player reduce the frame rate, or the image quality? The answer depends on the user's preferences for the current task. If the user is watching a sports event, he may prefer frame rate to be preserved at the expense of image quality. For watching a documentary on painting, the opposite might be preferable. Task models should capture the QoS tradeoffs that are appropriate for each task [Sous2005]. For reasons

mentioned in Section 4.2.1.4, this piece of information is in sync with the 'self-optimization' type of information.

Level of trust for information exchange

User's QoS preferences change with the task on hand. The system needs to make adjustments in QoS with changes in the environment (resources available, task being performed) as well as with user's preferences. Communicating this to the user at all levels of trust QoS will build the users confidence in the system.

Application with respect to Tivoli

Figure 4.13 depicts how the Tivoli Storage Manager allows the setting of QoS preferences.

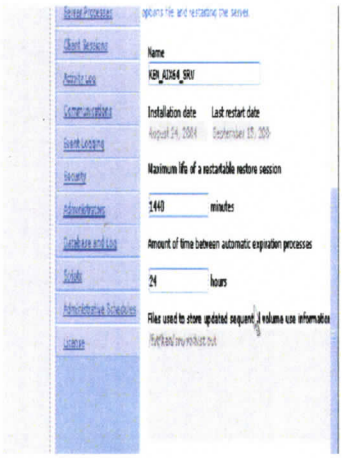

Figure 4.13: Tivoli Storage Manager: preferences with respect to QoS

4.2.4.5 Specification

In the context of daily task management, users constantly switch between tasks, incrementally change their needs for the ongoing task (e.g., by adding or removing services), may change their QoS preferences midway through a task. For instance, during a task that involves automatic translation of natural language, a user may prefer faster responses over accuracy of translation during the introductory part of the

conversation, but may prefer the opposite once the conversation becomes more involved [Sous2005]. Allowing the user to specify task based and other preferences and demonstrating to the user that those preferences are applied by the system, should help in gaining users confidence.

The specification of could relate to system configuration, optimization, healing or protection. Hence this piece of information falls under the 'All' type of information category.

Level of trust for information exchange

Specifications indeed change with time and task. Hence, exchanging this piece of information at all levels of trust is recommended.

Application with respect to Tivoli

Although Tivoli Storage Manager may have built-in characteristics and functionalities to encompass configuration management of services from an environment, we as researchers are more inspired by the ideas suggested by Sousa for an adaptive multimedia application called Aura [Sous2005]. Sousa, in his Ph.D. thesis presents a dashboard for adaptive systems in which users can save and select their task based preferences.

4.2.5 Reducing uncertainty in the mind of the user

This piece of information is in accordance with the classification of trust as an element to reduce complexity. The autonomic applications need to provide reliable user assessments of the system and the world to ensure proper operation in an unpredictable environment [Camp2003].

4.2.5.1 Alternative suppliers to support a service

This may be seen as an overlap with the information presented on suppliers in Section 4.2.4.3. Providing information on alternatives reduces uncertainty in the mind of the user.

Being able to provide alternative suppliers is a form of protection for the system. Hence this piece of information falls under the 'self-protection' category of information.

Level of trust for information exchange

We recommend exchanging this piece of information at the levels of partial and full trust. For details, see Section 4.2.4.3.

Application with respect to Tivoli

The application to Tivoli remains almost the same as discussed in Section 4.2.4.3.

4.2.5.2 QoS

QoS cover aspects such as response time, accuracy, image resolution, or frame rate. This may be seen as an overlap with the information presented on QoS in Section 4.2.4.4. A good rate of QoS could possibly eliminate user doubts and uncertainty levels to some extent.

For reasons mentioned in Section 4.2.1.4, this piece of information is in sync with the 'self-optimization' type of information.

Level of trust for information exchange

This piece of information is good to be exchanged at all the levels of trust. For details, see Section 4.2.4.4.

Application with respect to Tivoli

The application to Tivoli remains almost the same as has been discussed in Section 4.2.4.4.

4.2.5.3 User task-specific preferences

This may be seen as an overlap with the information presented on task-specific preferences in Section 4.2.4.2.

For reasons mentioned in Section 4.2.4.2, this piece of information is in sync with the 'self-configuration' type of information.

Level of trust for information exchange

This piece of information is good to be exchanged at all the levels of trust. The reasoning behind this may be seen as an overlap with the information presented in Section 4.2.4.2.

Application with respect to Tivoli

The application to Tivoli remains similar to what has been discussed in Section 4.2.4.2.

4.2.5.4 Changes in environment

In the present context, environment refers to an aggregation of user attributes, resources and/or suppliers available, tasks being performed or in the queue, QoS settings, and so on.

Exchanging pieces of information to depict adaptability to and match changes in the environment can require the hiding or presenting certain system capabilities, environments, information, and so on. This can be seen to involve a great deal of

configuration overhead on the part of the system. Hence this piece of information falls under the 'self-configuration' type of information category.

Level of trust for information exchange

We recommend exchanging this piece of information at levels of partial and full trust. This is because if the system informs a skeptical user of changes in environment, it will only increase the user's skepticism. Hence at levels of no trust, we believe it is not a good idea to present details that reveal changes in environment to the user.

Application with respect to Tivoli

Although Tivoli Storage Manager may have built-in characteristics and functionalities to encompass configuration management of services from an environment, we as researchers are more inspired by the ideas suggested by Sousa for an adaptive multimedia application called Aura [Sous2005]. This may be seen as an overlap with the information presented on configuration in Section 4.2.3.

4.2.5.5 Error messages

Communicating error messages is a vital part of any system-human interaction. Autonomic computing systems while being at the best of behaviours is still understood to throw exceptions at some point or the other during its operation. So long as the error message is communicated effectively, the user will know what to expect. Communicating faults to the user will help users to make informed decisions with regards to future operations. This will reduce the risk involved in an operation and is hence understood to reduce uncertainty in the mind of the user. Error messages by definition are a form of 'self-healing' information.

Level of trust for information exchange

We propose exchanging error messages, when they occur, at all levels of trust. The error message must contain some explanation of the exception, background information on the error and the effect of the error on the environment. This will reflect that the system is self-aware. Hence, this should help reduce user anxiety, especially at low levels of trust.

Application with respect to Tivoli

Based on our research and existing literature, we believe that the Tivoli Storage Manager exchanges messages when an error occurs. However during our hands on experience with Tivoli, this piece of information was not communicated. We believe this was because an error event did not occur during our operation.

4.2.5.6 Unforeseen circumstances with recommendations

Autonomic systems are policy based and learn by experience. Therefore, autonomic systems are expected to face circumstances that it may not have experienced before. In such conditions, the autonomic system must communicate the state of affairs to the users with some background information so that the user can take over control. Communicating recommendations to fix the state, if possible, will help reduce uncertainty in the minds of the user.

Communicating unforeseen circumstances and offering recommendations are a form of 'self-healing' information.

Level of trust for information exchange

Reflecting the systems' unfamiliarity to something at levels of no trust is understood to back track the level of trust. If a user does not trust a system, he or she will tend not believe in the recommendations offered by the system. At levels of partial trust, exchanging this information may be somewhat more useful. At level of full trust

providing users with recommendations on unforeseen situations should strengthen user-confidence in the system. However, we believe that at levels of overtrust, the exchange of such information is most fruitful. This is because overtrust is a sensitive level of trust, the damage of which may lead to no trust. Hence in unforeseen circumstances, the system needs to be more transparent.

Application with respect to Tivoli

Based on our research and existing literature, we believe that the Tivoli Storage Manager exchanges messages and offers recommendations when an unforeseen circumstance is met with. However during our hands on experience with Tivoli, this piece of information was not communicated. We believe this was because such a scenario did not occur wherein it would need to offer recommendations during our interaction with the system.

4.3 Development of the trust framework

Studying Tivoli helped us identify the nature and pieces of information exchange that can help build trust and come up with our trust framework. Table 4.3 summarizes our trust framework upon investigation of Tivoli.

Table 4.3: Development of the trust framework by studying Tivoli

Trust framework (sub classification based on the definition of trust)	Type of information exchange	Nature of information exchange
User expectations	All	Functionality
	All	Modus operandi
	Self-configuration	User perceived state
	Self-optimization	QoS
Intention and willingness to act	All	Service awareness
	Self-optimization	Resource awareness
User belief	Self-configuration	User environment
	Self-optimization	Status indicators
	Self-configuration	Data review
	Self-optimization and self-configuration	Visibility of information via tracking
User task and behaviour	Self-configuration	User attributes
	Self-configuration	Task specific preferences at a low level
	Self-optimization	QoS
Reduce the uncertainty in the minds of the user	Self-optimization	QoS
	Self-configuration	Task specific preferences

4.4 Evaluation of Tivoli

Conversely, in the process of developing our framework using Tivoli, we were also able to evaluate Tivoli and determine the strengths and shortcomings with respect to trust issues in the application. Table 4.4 lists the plus and minus points of Tivoli with respect to information exchange, based on our trust framework. The plus sign indicates that the respective piece of information is exchanged with the user whereas the information pieces listed in the minus sign column indicate that those pieces of information exchange are not available (based on the author's study).

Table 4.4: Evaluation of Tivoli with respect to our trust framework

Table 4.4: Evaluation of Tivoli with respect to our trust framework **Trust framework (sub classification based on the definition of trust)**	+	–
User expectations	Functionality Modus operandi User perceived state QoS	None
Intention and willingness to act	Service awareness Resource awareness	Supplier information
User belief	User environment Status indicators Data review Visibility of information via tracking	Context awareness Configuration details Recourse or undo Trial runs
User task and behaviour	User attributes Task specific preferences at a low level QoS	Specification
Reduce the uncertainty in the minds of the user	QoS Task specific preferences	Supplier information Error messages Recommendations in unforeseen circumstances

4.5 Chapter Summary

This chapter outlines the nature and type of information for building trust with autonomic applications. Tables 4.2a – 4.2e outline the nature and type of information exchange to help build trust with autonomic applications.

This chapter also helps us evaluate Tivoli with respect to trust issues. Based on our study, Table 4.3 lists the plus and minus points of Tivoli with respect to the pieces of information it exchanges that fall under our trust framework.

Next, we apply our trust framework to a case study and apply the nature and type of information mentioned in this chapter to an autonomic prototype.

Chapter 5: Evaluation

Evaluating this thesis entails demonstrating that (i) the proposed framework of trust is useful with respect to autonomic systems; and (ii) the nature and type of information to be exchanged fits into our proposed trust framework. Ideally, we would conduct a case study in industry to assess the value of the framework for developers of autonomic systems. However, a better tactic is to perform some internal case studies first to further refine the trust framework before deploying it in industry.

Thus, to evaluate (i) we cite literature and demonstrate how our trust framework holds with respect to autonomic systems (cf. Section 5.1). To evaluate (ii), we apply our framework to a non-deployed autonomic e-Commerce prototype (cf. Section 5.2).

5.1 Trust Framework with respect to Autonomic Systems

In the following sections we perform a documentation review to evaluate how our trust framework is useful with respect to autonomic systems.

5.1.1 User expectations and autonomic systems

For autonomic systems, situation awareness is important [BMKB2005]. Sousa underlines how expectations and/or needs of users with respect to computational support for their tasks are significant [Sous2005]. We have mentioned previously that there is little difference between Sousa's specific notion of an adaptive system and the more general notion of a self-managed system (cf. Section 2.3). Fulfilling a user's expectations regarding what the system does, why it does that and what it will be doing next may prove to be worthwhile for building trust [ANBL2002]. This suggests that for autonomic systems to be trusted, it would be worthwhile to communicate the probability of success of a particular action based on past experiences. This seems feasible because autonomic

systems are primarily policy based. Calculations to determine the success or failure rate of a particular action should be possible with information from the knowledgebase and by observing the pattern of behaviours of the system.

5.1.2 Willingness to act and autonomic systems

Lee *et al.* aptly categorize trust in automated systems as an intention or willingness to act [LeSe2004]. In this thesis, we assume that the user is willing to trust the system provided that the system is accountable to him or her. In fact, our thesis is based on the premise that users of our e-Commerce prototype are receptive to building authentic trust based on the past experiences (cf. Section 3.1) provided the system itself is willing to cooperate.

5.1.3 User belief and autonomic systems

We appreciate the distinction between belief and behaviour and acknowledge trust as a belief (cf. Chapter 3). Autonomic decisions must be correct to the user's belief [Camp2003]. Hence, our classification of trust as a belief and its application to autonomic systems seems worthwhile. Based on our research, we argue that providing information to the user about the system state, intentions and operation modes in part will help reassure the user's belief. Hence, our work which is an explicit layout of the nature and type of information to be expressed to the user fits well in place.

5.1.4 User behaviour and autonomic systems

The definition of trust, as a willingness to act, is important since it brings into the limelight the fact that in order to build trust with autonomic software, the user himself or herself needs to be assessed personally. Scenario questions and questions to gauge his or her temperament before use of autonomic software may help the system decide the level of risk it can take. Some researchers have recommended that users be presented a game to play before using the software so that their temperaments and their willingness to participate in different situations may be gauged. Sousa in his Ph.D. thesis mentions that it is important to store user preferences and user knowledge (level of user expertise) in order to build adaptive systems that are truly functional [Sous2005]. Lee *et*

al. also recognize that trust in automated systems is affected by behaviour to some extent [LeSe2004]. This reassures our understanding that defining trust as a behaviour is justified with respect to autonomic software.

5.1.5 Reducing uncertainty and autonomic systems

We understand that providing the user with relevant information will keep users in the autonomic operation loop. This will help the user to make informed decisions. Communication and transparent autonomic systems will help reduce uncertainty in the minds of users.

5.2 Nature and Type of Information Exchange with respect to the E-Marketplace Application

We have described the basics of our E-Marketplace autonomic e-commerce prototype in Section 2.4.2.2. The E-Marketplace application was inspected over a period of four hours by the author. During our inspection, the author performed transactions with the system to buy products from various fictitious sellers. In this section, we present an evaluation of the E-Marketplace application. In turn, we also apply the nature and types of information, which we described in Chapter 4, to the E-Marketplace application to evaluate the usefulness of our trust framework from a developer's perspective.

5.2.1 User expectations

5.2.1.1 Functionality and goal-orientated performance assessment

This piece of information exchange is missing in the E-Marketplace application. We propose an 'About E-Marketplace' option to be visible to the user which would specify the functionality of the application.

5.2.1.2 Modus operandi

In some cases the users may like to know the modus operandi or the plan to solve an issue at hand (i.e., depending on the level of interest and expertise of the user). This display of information is apparently missing from the E-Marketplace application and recommended. For example, when a User Agent receives the "Go Shopping" message from a user action, it sends a purchase query to a Buyer Agent (cf. Section 2.5.2.2). The Buyer Agent immediately creates a purchase order, which is an XML file, and sends it to the User Agent to get confirmation. The user does not know what the system is up to until it receives the confirmation. Therefore the user is kept in the dark and may not know what to expect as a result or may expect something completely different. A tool tip above the 'Go Shopping' button that states 'Sends a purchase query and creates a purchase order' may be useful. Also, upon obtaining confirmation from the User Agent, the Buyer Agent creates a Mobile Buyer Agent (MBA) and dispatches it to the Market Management Agent. After the MBA passes the identity verification, the Market Management Agent checks the configuration repository based on the products that the buyer needs, and gets the relevant information about all sellers and gives the MBA the sellers' address. This information is not communicated to the user. We recommend that this be communicated to the user so as to depict the mode of operation the application follows to achieve user expectations. Displaying the mode of operation to perform a task increases the credibility of the application and makes the system efforts sound more prudent, believable and visible.

5.2.1.3 User perceived state

The E-Marketplace application does not adapt to user preferences with respect to user level settings and interaction parameters. The system does not allow the user to set his or her preferences in the first place. Hence, the display of user perceived state information is apparently missing from the E-Marketplace application and is recommended.

5.2.1.4 Quality of Service

This display of information is apparently missing from the E-Marketplace application and recommended. For example the expected turnaround time after a user hits the 'Go Shopping' button should be communicated to the user.

5.2.2 Intention and willingness to act on tasks and services

5.2.2.1 Service awareness

Fault tolerant and load balancing systems represent the earliest form of service aware systems [Sous2005]. Modern service aware systems would also include QoS details. This display of information about the systems willingness to be service aware is apparently missing from the E-Marketplace application and is recommended.

5.2.2.2 Context awareness

Context refers to the circumstances in which an event occurs. The E-Marketplace is not receptive to and does not exchange information such as user location, attention focus, emotional state, and so on. This piece of information exchange should be considered to depict the system's willingness to act on tasks and services.

5.2.2.3 Resource awareness

This display of information is apparently missing from the E-Marketplace application and is recommended. For example upon arrival at a seller's site, the MBA (cf. Section 2.5.2.2) starts to negotiate with the Seller Agent. Firstly, the seller checks its own inventory through an Inventory Agent, if it has the needed goods, the seller will accept the proposal from the MBA and begin to bargain with the MBA by using negotiation strategies; otherwise, it will send a "refuse" message to the MBA. This information is not

communicated to the user. We recommend that this be communicated to the user so as to depict the willingness of the MBA to be aware of the available resources and operate accordingly. However, the E-Marketplace application does display the name and the location of the sellers and the buyers. This allows the user to be aware of the context in which the transaction has been processed. For example, a single seller in the seller's list will be indicative that there has been little or no negotiation with respect to the price of the product.

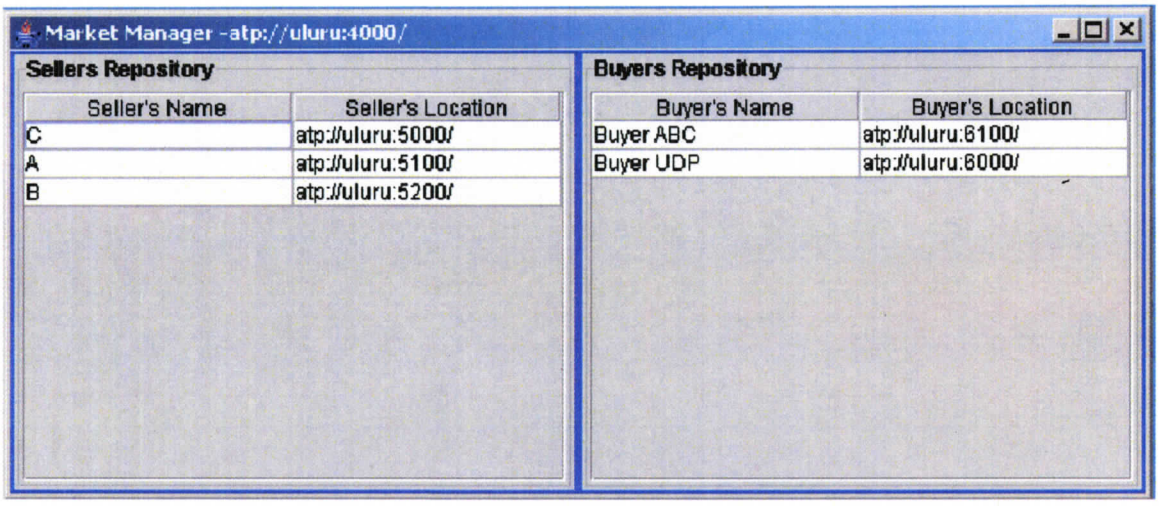

Figure 5.1: User interface of the Market Manager agent in the E-Marketplace application

5.2.3 Users' belief and confidence

5.2.3.1 User environment

This display of information is apparently missing from the E-Marketplace application and recommended. For example, environment details such as the set of suppliers, materials (e.g., an information asset such as a file or data stream), and resources accessible to a user at a particular location, should be communicated if available. This communication to the user shall reassure the user that there are alternative suppliers for a product and, hence, the transaction has better chances of success. Also, depicting that the system is aware of its options will increase the users' confidence and hence the user will appreciate that he or she got a good deal.

5.2.3.2 Context awareness

In the present context, context awareness refers to circumstances in which an event is placed or occurs. Context-aware systems react to variations in the physical context around users [CSAW2005]. Examples of observed variables are: user location, attention focus, physical activity, emotional state (e.g., relaxed, working, or in distress), or privacy (i.e., who else is in the vicinity). Typical context-aware systems represent such awareness as collections of interpreted rules (i.e., clauses of the form "if context then action"), or embedded logic in the code [CSAW2005]. The E-Marketplace is not a context-aware system.

5.2.3.3 Configuration details

Carrying out a task may require obtaining a configuration of services from an environment, and accessing several resources. In this case, configuration means a set of possibly interconnected services that together support a task. The presentation of such information is apparently missing from the E-Marketplace application and is recommended. We envision that configuration details may not be desired by a user who is using the E-Marketplace for shopping. However, an administrator or maintainer of the system may need pieces of information such as details on the constituent components of the system or network details.

5.2.3.4 Status Indicators

In the E-Marketplace application, Monitor Agents continuously monitor the status of each agent, detect agent failures, or restart an agent to achieve self-healing. However, the presentation of this information to the user is missing and is recommended. For example, a progress bar indicating the status of a negotiation process will keep the user in the loop and reassure the user of the system's endeavour to work towards his or her goal. Hence, the capabilities of the system need to be expressed to the user to reassure the user's belief and build user's confidence in the system.

5.2.3.5 Data review

The E-Marketplace allows the display of information already entered. Figure 5.3 shows how the application remembers input text from the user for future references.

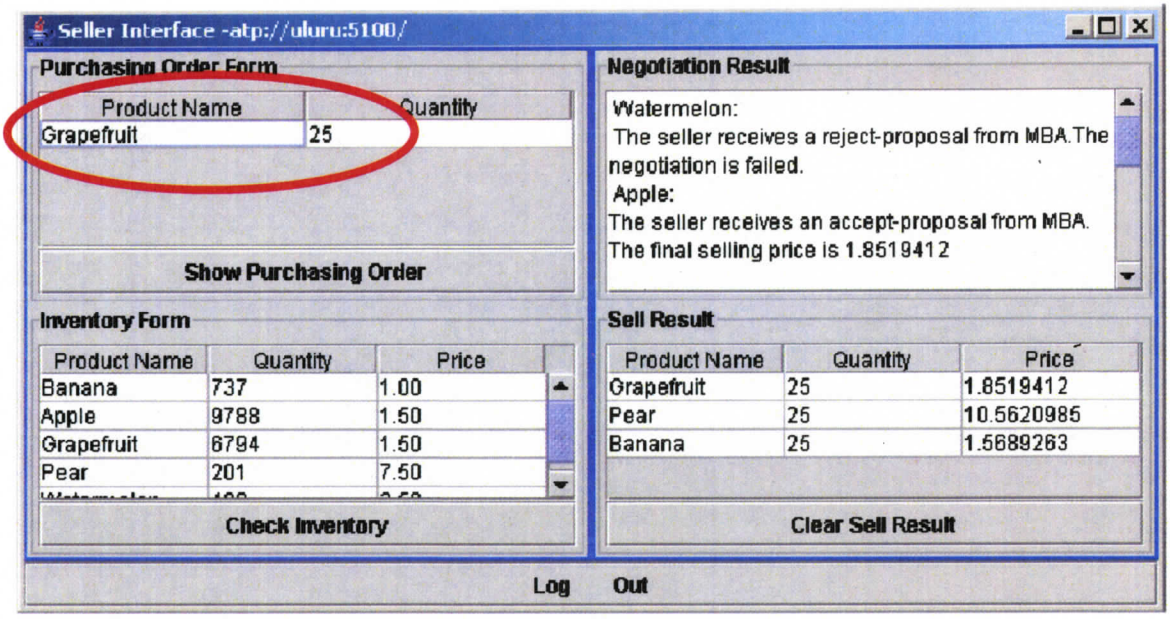

Figure 5.2: User interface for the Seller Agent in the E-Marketplace application

5.2.3.6 Visibility of information and tracking of processes

Figure 5.3 exhibits how, in the E-Marketplace application, the Negotiation Result space (cf. Figure 5.2) displays the failed status of the negotiation process in addition to the final result. However, the application needs to be more responsive to the user during the shopping process. In the E-Marketplace application, the system does not communicate to the user what it is doing and where exactly the Mobile Agents are wandering at a specified point in time. For example, the system should allow the user to be able to keep track of its activity and allow transparency during its operation. Communicating information during the negotiation process such as 'Reached seller A and awaiting response', 'Seller A refused to transact, approaching Seller B', 'Seller B does not have the product, searching for other suppliers', 'No more suppliers found, transaction cancelled' and so on will facilitate continuous visibility of operations to the user and reassure him or her that the system is working towards the goals specified.

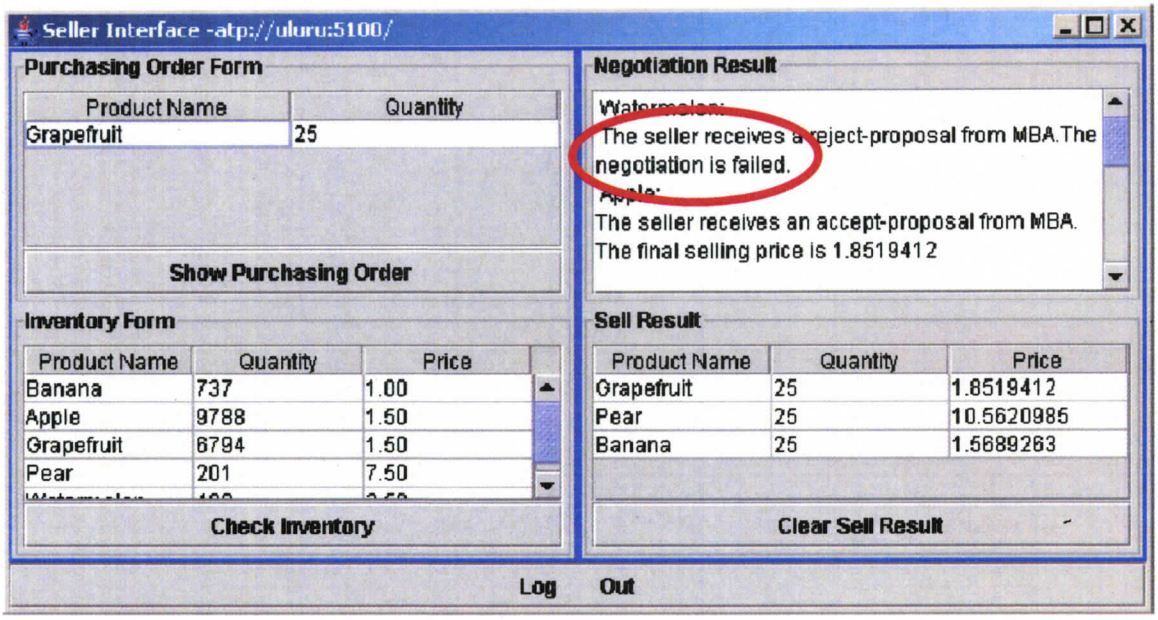

Figure 5.3: User interface for Seller Agent in the E-Marketplace application

5.2.3.7 Recourse or undo

Figure 5.4 depicts how the E-Marketplace allows recall or undo. However, once the user hits the 'Go Shopping' button, the shopping command cannot be aborted. Allowing users to undo such an action will reassure the users' belief that the system is receptive to users discretion and flexible enough to adapt to users wishes.

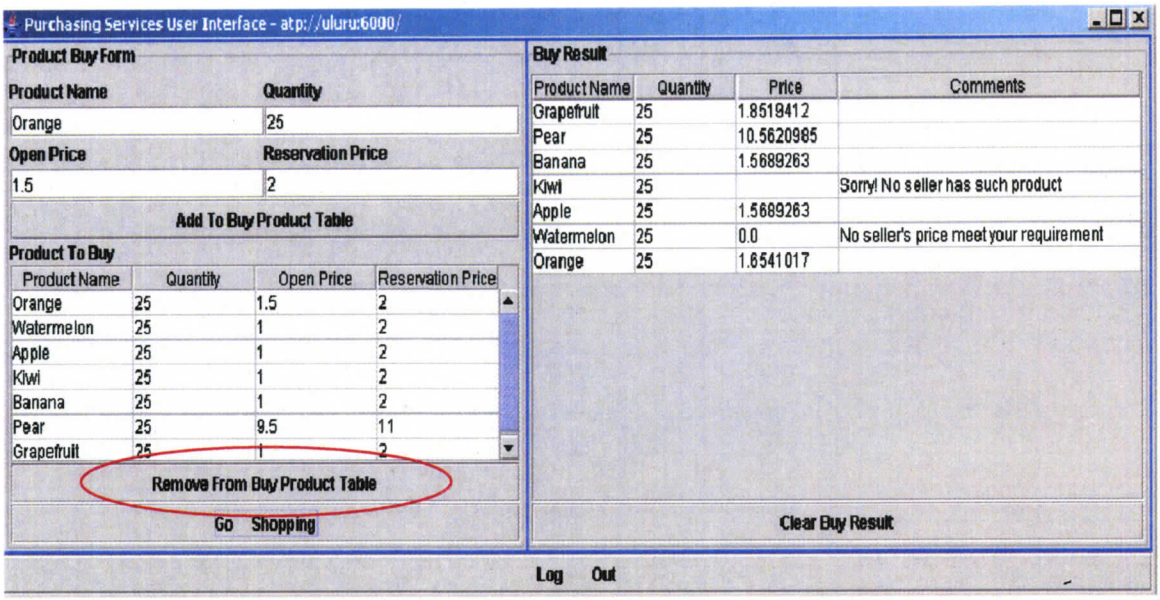

Figure 5.4: User interface for the Buyer Agent in the E-Marketplace application

5.2.3.8 Trial runs

The E-Marketplace application does not provide the feature to allow users to perform trial runs. The user is expected to know how the system works. However, no documentation or guidance is available for reaffirming the user's belief that the action he or she may be taking will be correct and cause the expected result. We recommend the provision of trial runs to reassure the user's belief in the system.

5.2.4 User behaviour

5.2.4.1 User-attributes

Section 4.2.4.1 mentions how users with different degrees of expertise may have different levels of expectations from the system and react differently. The E-Marketplace application does not allow user-attributes to be saved in the environmental model. We recommend the set up of user accounts and profiles so that users can customize the environment as per their preferences. This will also provide the system with information on user-attributes and allow the system to adjust accordingly.

5.2.4.2 Task-specific preferences

User-attributes should be saved in the environment model task-specific preferences with respect to alternative configurations for supporting the task as discussed in Section 4.2.4.2. The E-Marketplace does not exchange user preferences relative to functionality and QoS. The system therefore is not receptive of user task specific preference information and does not display information to demonstrate adaptability in this regard. Hence it is recommended that in the E-Marketplace application, user task specific preferences be considered to build user trust in the autonomic prototype.

5.2.4.3 Supplier information

The E-Marketplace application does not allow the user to specify his or her preferences with respect to the supplier they are obtaining the services from. For example, if the user wishes to exclude a particular seller from the market when the user 'Goes shopping', it is not possible. We recommend that the E-Marketplace allow users to rank sellers based on their preference or exclude a particular seller. This can be made possible by saving users preferences under the users' identity in the user profile. When a user logs on with their identity, these preferences can be retrieved from the users profile and loaded for the user. This will allow the system to make adjustments with the user's behaviour and the communication of such system adjustments to the user will help build trust with the system.

5.2.4.4 Preference relative to QoS

The presentation of such information is apparently missing from the E-Marketplace application and is recommended. Section 4.2.4.3 explains how users may have different preferences over QoS for different tasks. Also QoS covers several aspects of service including turnaround time, failure rate etc and the user may be interested in only a particular set of QoS preferences. The user should be allowed to make these specifications and save them in their profile so that the application may make adjustments accordingly. When a user logs on with her identity, these preferences can be retrieved from the users profile and loaded for the user. This will allow the system to

make adjustments with user's behaviour and the communication of such adjustments to the user will help build trust with the system.

5.2.4.5 Specification

Section 4.2.3 describes how users may have specifications for different tasks. The E-Marketplace application does not allow specification of QoS related issues. Looking beyond QoS in the E-Marketplace application, the user may like to specify the currency, the number of decimal point for the cost of a product or simply its features in specific terms. This is not currently possible with the E-Marketplace application. However, the user can make specifications to a certain extent in the application. Figure 5.5 shows the fields where users can specify the price range and abstract requirements for the product that they would like to buy. However, users should have the better flexibility in making specifications. The system should adjust to these specifications and the adjustments made by the system should be communicated to the user. This reflects the system's willingness to make adjustments according to the user's behaviour which will help build the user's trust in the system.

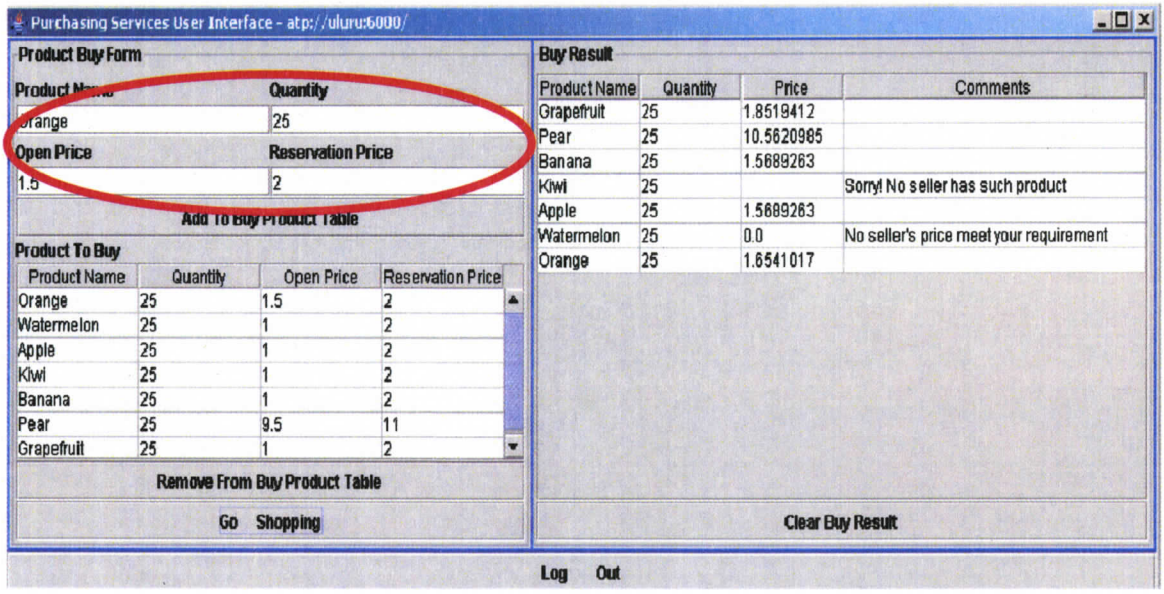

Figure 5.5: User interface for the Buyer Agent in the E-Marketplace application

5.2.5 Reducing uncertainty in the mind of the user

5.2.5.1 Alternative suppliers to support a service

The presentation of such information is apparently missing from the E-Marketplace application and recommended. In the E-Marketplace application, after the Buyer Agent receives all the relevant sellers' addresses from the MBA, it can decide which sellers to transact with. Then the Buyer Agent sends the MBA to visit each seller (cf. Section 2.5.2.2). However the alternative of the supplier available is not communicated to the user. Displaying this piece of information will help reduce uncertainty in the mind of the user.

5.2.5.2 QoS

The presentation of such information is apparently missing from the E-Marketplace application and is recommended. QoS aspects include issues such as response time, accuracy, image resolution, or frame rate. The E-Marketplace application does not communicate the QoS setting of the system. Allowing the user to view the QoS settings will check the user from making unreal expectations from the system and the system violating such expectations. Also, the user will then know the state of art and, hence, this will reduce uncertainty in the mind of the user.

5.2.5.3 User task-specific preferences

Changes in users' tasks are the same as task specific preferences as discussed in Section 5.2.4.2. Expressing system adjustments to accommodate changes in user's tasks will help reduce uncertainty in the mind of the user.

5.2.5.4 Changes in environment

The presentation of such information is apparently missing from the E-Marketplace application and is recommended. Environment details include information such as the

set of suppliers, and resources accessible to a user at a particular location. Expressing the system adjustments made to accommodate changes in the environment will help reduce uncertainty in the mind of the user.

5.2.5.5 Error messages

The presentation of such information is apparently missing from the E-Marketplace application and is recommended. Any deviation from the expected output is considered an error [CSAW2005] from a user's point of view. A possible list of user expectations has been listed in Section 5.2.1. Once the system has processed the information based on the user's expectations, if any deviation is found it should be communicated to the user.

5.2.5.6 Unforeseen circumstances with recommendations

The presentation of such information is apparently missing from the E-Marketplace application and is recommended. The E-Marketplace application does not offer recommendations to the user in unforeseen circumstances.

5.3 Evaluation of the E-Marketplace application

By evaluating the E-Marketplace application using our trust framework, we were able to recommend a set of improvements to the developer of E-Marketplace. This demonstrates that this systematic inspection of trust issues can be utilized by developers during the design of an autonomic system to improve the design with respect to trust. Moreover, the same systematic inspection techniques can be used to analysis operational autonomic systems. Thus, the E-Marketplace case study gives us confidence that the nature and type of information exchange described in Chapter 4 are useful.

Conversely, in the process we were able to evaluate E-Marketplace application and determine the strengths and shortcomings with respect to trust issues. Table 5.1 lists the positive and negative aspects of the E-Marketplace application with respect to information exchange, based on our trust framework. A plus sign indicates that the respective piece of information is exchanged with the user whereas the information pieces listed in the minus sign column indicate that those pieces of information are not

available. These findings are based on the author's case study of the E-Marketplace application.

Table 5.1: Evaluation of E-Marketplace application with respect to our trust framework

Trust framework (sub-classification based on the definition of trust)	+	–
User expectations	None	Functionality Modus operandi User perceived state QoS
Intention and willingness to act	None	Supplier information Service awareness Resource awareness
User belief	Data review Visibility of information tracking Recourse or undo (available in some situations)	User environment Status indicators Context awareness Configuration details Recourse or undo Trial runs
User task and behaviour	Specification (available in some situations)	User attributes Task specific preferences QoS Specification
Reduce the uncertainty in the minds of the user	None	Supplier information Error messages Recommendations in unforeseen circumstances QoS Task specific preferences

5.4 Chapter Summary

In this chapter, we applied our trust framework in a case study to the E-Marketplace autonomic computing prototype. We are pleased that our ideas readily apply to this prototype. We concluded that the autonomic prototype has significant issues with building user trust. It turns out trust issues were not really considered during the design of the E-Marketplace application. As a result, this chapter offers recommendations to the developer of the E-Marketplace application on how to build user trust into her application. In the next chapter, we discuss the limitations and contributions of our research and outline avenues for future research.

Chapter 6: Conclusions

6.1 Summary

The increasing heterogeneity, dynamism and interconnectivity in software applications, services and networks lead to complex and difficult-to-manage systems. Coping with such a complexity directs us to investigate a new computing paradigm, namely autonomic computing. Of course, autonomic computing poses significant research challenges. Two of the most critical research challenges are trust and adoptability. Autonomic systems, in an endeavour to hide complexity, give up accountability to the user to a certain extent. The system itself takes over full control whereby it may or may not operate as per the user's expectations during its operation. Consequently, autonomic systems engender trust and adoption issues since the system by definition hides cause and effect relationships. Researchers have stressed the fact that building trust is critical when automating tasks. Increasing trust will allow the proliferation of autonomicity in existing autonomic systems [LoLi2002]. Newly deployed autonomic applications on the other hand will be more readily accepted and adopted if trust issues are addressed. In fact, the issue of trust is in many ways at the core of self-managing systems, and the issue is very complex [LMRW2005]. There is not only a need, but also an absence of mechanisms, to develop a rapport with autonomic systems [MiLa92]. Until now, there has been no concrete understanding, specification and layout of methodologies, and steps to describe how trust can be built with autonomic systems.

We advocate information exchange and communication as a tool to build trust with autonomic systems. Using our trust framework as a base, we discuss how the tool of communication helps build trust with software systems. In our thesis, we advocate an exchange of information between the user and the autonomic application in order to build trust. Improved interaction will allow autonomic systems to be more autonomous, exhibiting increased initiative without losing the users' trust. Higher levels of trust and usability should in turn lead to improved adoptability. This thesis addresses the trust problem by enabling an in-depth understanding of trust and applying the understanding to autonomic systems thereafter.

In order to achieve the above mentioned goal, we followed an exploratory approach for our research. In an exploratory approach, usually little is known of the subject under study at the outset of the research. The subject under study is investigated from different viewpoints, and the researcher, in the process, gains 'insights' of the subject under study (cf. Figure 1.3). Hence, the researcher deepens his or her understanding and this method may therefore, reveal valuable new aspects about the subject. In our case, the insight led to a trust framework for autonomic computing systems. We looked at trust from the different viewpoints—from the way trust is defined to how trust is formed. We also reviewed the literature on trust to gather information on key trust topics, issues, nomenclatures and taxonomies and then derived our trust framework.

A framework is a simplified description of a complex entity or process. In our case, the complex entity was trust itself. Our framework provided a simplified description of trust by:

Presenting a nomenclature (naming conventions) of trust
Presenting a taxonomy (classification) of our nomenclature
Presenting a Venn diagram of the definition of trust
Identifying the different levels of trust
Reviewing a method to measure trust [CSAW2005]

We related the concepts of trust and autonomic computing. In this thesis we distilled and pruned the key topics of our trust framework. We investigated the Tivoli provisioning system (i.e., an IBM application that has autonomic capabilities) to determine the nature and pieces of information exchange to build trust. To evaluate the strengths and shortcomings with respect to trust in Tivoli, we looked at the nature and type of information exchanged in an adaptive system [Sous2005].

After identifying the nature and type of information to be exchanged, our approach involved meticulous planning and decision making regarding which pieces of information among these should be inherently built into or provided by an autonomic system. We proposed a three-stage trust model to describe the varying levels of trust in autonomic applications. We described the existing methods to measure trust in autonomic systems and outlined which pieces of information need to be exchanged with each level of trust.

In order to evaluate our trust framework, we applied our findings in a case study to an autonomic e-Commerce prototype called the E-Marketplace application. This application was designed at the University of Victoria by Jing Zhou [Zhou2006]. The system is based on a Mobile Multi-Agent Autonomic Architecture to achieve its desired e-Commerce functionalities. The system implements autonomic properties such as self-protecting, and self-healing from failures. The case study produced a set of recommendations for the developer of the E-Marketplace application to improve the application with respect to trust issues.

Based on our experiences gained with the analysis of the commercial Tivoli system and the non-deployed e-Commerce prototype, we feel that our findings will be able to guide the developers of trustworthy autonomic applications to some extent. We are confident that the trust framework can be used by other researchers in a step-by-step fashion. It is our hope that our trust framework will form a base for other researchers to build upon.

6.2 Contributions

We cannot claim that we have completely solved the problem of building incremental trust for autonomic systems by providing guidelines for communication and interaction with such systems. However, we believe that our approach is a step in the right direction. The following list outlines our main contributions to the research in this specific field.

- Investigated trust from various viewpoints, organized and categorically presented the different dimensions of trust.
- Proposed a trust framework for autonomic computing systems, while outlining a subset of the nature and type of information exchange required for building trust. Our trust framework
    - Presents a nomenclature (naming conventions) of trust;
    - Presents a taxonomy (classification) of our nomenclature;
    - Presents a Venn diagram of the definition of trust;
    - Identifies the different levels of trust; and

    - Reviews a method to measure trust [CSAW2005].
- Studied a deployed autonomic application, Tivoli, to discover how or how our trust framework does or does not conform to a successful solution.
- Identified the trust strengths and shortcomings in Tivoli using our trust framework by exposing concepts from user task models for adaptive systems to determine how autonomic applications may violate these ideas.
- Applied our concepts to a non-deployed autonomic prototype to evaluate the system itself and also our trust framework to autonomic systems.

It is our hope that our research will benefit developers and researchers in the following ways:

- Aiding designers in understanding and designing incremental trust in AC systems.
- Guiding the development of autonomic computing systems with respect to trust.
- Proposing a trust framework for other researchers to follow and build upon.

6.3 Limitations

As stated earlier in this thesis, trust is multi-dimensional in that it is affected by several parameters (cf. Section 2.5). Communication alone cannot completely solve trust issues within a software system. For example, trustworthiness in an e-Commerce system also depends on the vendors behind the scene who are selling the products. Financial risk in online transactions is also a major consumer concern; vendors may or may not provide guarantees on transactions. Hence, there are several aspects to trust which may be beyond the reach of software engineering researchers. Nevertheless we argue however that communication is the key tool to help building trust. Hence, our trust framework and listing of the nature and type of information exchange may not be a complete solution to the issue of trust with autonomic systems—but it is a step in the right direction.

The concepts and ideas presented in this thesis are based on the researchers' study and understanding. Therefore there may be a bias in the pieces of information identified as a part of our trust framework.

Also, trust levels may depend on a user's basic nature, personality and mood. Patrick [Patr2002] quotes a survey wherein 27% people in one group only marginally were concerned with online privacy, while 17% in another group were labeled as 'privacy fundamentalists' and were very particular about privacy. The third group (56%) was labeled the 'pragmatic majority' because they had some concern about privacy, but also had developed tactics to deal with those concerns. We understand that it is difficult to gauge a user's basic nature, personality, and mood. Therefore the framework may or may not prove to be effective in all circumstances. This thesis is based on the assumption that all users will ultimately develop authentic trust with the system if the system is accountable. Autonomic systems are rule based and include a knowledgebase in its autonomic element. Therefore our assumption is that history based trust is the basis for building trust with autonomic systems. Although we see this to be a reasonable argument, we cannot guarantee that our assumptions will be true at all instances.

6.4 Future Work

This research does not address all aspects related to trust issues with Autonomic Systems. Several interesting but unexplored research avenues are as follows:

- Modify the E-Marketplace application so as to exchange the pieces of information that are missing amongst the ones we described.
- Conduct a user study before and after the modification of the applications. The comparison of the results will help in determining the effectiveness of our approach.
- Present a complete and comprehensive listing of the pieces of information that should be exchanged with users to build trust with autonomic computing systems.
- Outline the character and grammar of the information being exchanged.
- Explore the nature and type of information that fall in the 'type of trust' and 'basis for development' dimensions (i.e., Table 4.1, Sections 3.1.1 and 3.1.2) in detail.

Bibliography

[AHPR2003] Stuart Anderson, Mark Hartswood and Rob Procter, et al., Making Autonomic Computing Systems Accountable: The Problem of Human Computer Interaction, *Proceedings of the 14th International Workshop on Database and Expert Systems Applications,* pp. 718-724, 1-5 Sept. 2003.

[ANBL2002] Adams, Sam S., Nancy Alvarado, Steve Burbeck, Craig Latta, Bootstrapping in an Autonomic Computing System, Fourth *International Workshop on Computational Semiotics for Intelligent Systems, Joint Conference on Information Systems (JCIS)*, Chapel Hill, NC, 2002.

[Arau2003] Ildermaro Araujo, Trust in Internet Commerce, *Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research,* pp. 1-15, 2003.

[Barb1983] B. Barber, Logics and Limits of Trust, pp. 14, 1983.

[BMKB2005] R. Barret, P. Maglio, E. Kandogan and J. Bailey., Usable autonomic computing systems: The system administrators' perspective, *Proceedings International Conference on Autonomic Computing, 2004,* Vol. 13, No. 3, pp. 18-26, 30 September 2005.

[Bick2003] Timothy W. Bickmore, Relational Agents: Effecting Change through Human-Computer Relationships, Ph.D. Thesis, Media, Arts and Sciences, Massachusetts Institute of Technology, pp. 38, 2003.

[Bowe2005] Nick Bowen, Autonomic Computing: an overview, IBM, Texas, USA, 2005.

[Bruh2002] J. Bruhn, Trust and the Health Organizations, Clinical Sociology: Research and Practice, pp. 234, New York, 2002.

[Camp2003] L. Jean Camp, Springer-Verlang Rino Falcone, Designing for Trust, Reputation and Security: Theories and Practice, Berlin, 2003.

[CaMu2006] M. Capra and L. Musolesi, Autonomic Trust Prediction for Pervasive Systems, *20th International Conference on Advanced Information Networking and Applications, 2006.*, Vol. 2, pp. 5, 18-20 April 2006.

[CoPF1999] Marvin S. Cohen, Raja Parasuraman, and Jared T. Freeman, Trust In Decision Aids: A Model and Its Training Implications, Technical Report USAATCOM TR 97-D-4, Arlington, VA, Cognitive Technologies.

[CSAW2005] Hoi Chan, Alla Segal, Bill Arnold and Ian Whalley, How can we trust an autonomic manager to make the make the best decisions, *Second International Conference on Autonomic Computing,* pp. 351-352, 2005.

[Dekl2000] Sasa Dekleva, Electronic commerce: a half-empty glass? *Communications of the AIS,* Vol. 3, No. 4, Atlanta, GA, USA, June 2000.

[Dict2005] Dictionary.com, http://dictionary.reference.com

[Egge2000] F. N. Egger, "Trust me, I'm an online vendor": towards a model of trust for e-commerce system design, *Conference on Human Factors in Computing Systems,* pp. 101 - 102, New York, NY, USA, 2000.

[Egge2001] Florian N. Egger, Affective design of e-Commerce User Interfaces: How to maximize perceived trustworthiness, *Proceedings on International Conference on Affective Human Factors Design,* pp. 317-324, June 2001.

[Falc2004] Rino Falcone and Cristiano Castelfranchi, Trust Dynamics: How Trust Is Influenced by Direct Experiences and by Trust Itself, *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems,* Vol. 2, pp. 740 - 747, Washington, DC, USA, 2004. [GaCo2003] A. Ganek and T. Corbi, The Dawning of the Autonomic Computing Era, *IBM Systems Journal,* Vol. 42, No. 1, pp. 5-18, Riverton, NJ, USA, January 2003.

[Gidd1991] Anthony Giddens, The consequences of modernity, Stanford University Press, pp. 200, 1991.

[GrSl2000] Tyrone Grandison and M. Sloman, A Survey of Trust in Internet Applications, *IEEE Communications Surveys & Tutorials,* Vol. 3, No. 4. Oct-Dec 2000.

[Haub2004] Elizabeth Haubert, Threats of Human Error in a High Performance Storage System: Problem Statement and Case Study, pp. 13, December 2004.

[HIPD2003] L. Herger, K. Iwano, P. Pattnaik, A. Davis and J. Ritsko., Special Issue on Autonomic Computing, *IBM Systems, Journal,* Vol. 42, No. 1, pp. 3-188, 2003.

[HoNP1999] Donna L. Hoffmann, Thomas P. Novak and Marcus Peralta., Building consumer trust online, *Communications of the ACM,* Vol. 42, No. 4, pp. 80-85, New York, NY, USA, April 1999.

[Horn2001] Paul Horn, Autonomic Computing: IBM's Perspective on the State of Information Technology, IBM T.J. Watson Labs, NY, 15th October 2001.

[HMKM2005] A. R. C. Hussin, L. Macaulay, K. Keeling and P. McGoIdrick, A Trust Agent for E-Commerce: Looking for Clues, *Proceeding, The 2005 IEEE International Conference one-Technology, e-Commerce and e-Service.,* pp. 286- 289, 2005.

[IBM2006] IBM, An Architecture Blueprint for Autonomic Computing, White Paper, June 2006.

[JoGe2004] Gerald A. Joseph, Building trust for users and consumers, Regional Workshop on Developing e-Business Policy Framework, 2004.

[KaFJ2002] Lalana Kagal, Tim Finin and Anupam Joshi, Developing Secure Agent Systems Using Delegation Based Trust Management, Security of Mobile Multi-Agent Systems Workshop, Autonomous Agents and Multiagent Systems (AAMAS 2002), July 2002.

[KAMK2005] Piotr Kaminski, Priyanka Agrawal, Holger Kienle and Hausi Müller, <username>, I Need You! Initiative and Interaction in Autonomic Systems, *Proceedings of the 2005*

workshop on Design and evolution of autonomic application software, International Conference on Software Engineering archive, Vol. 30, No. 4, pp. 1 - 4, May 21, 2005.

[KBCS1999] Peter Keen, Craig Balance, Sally Chan and Steve Schrump, Electric Commerce Relationships: Trust by Design, pp. 247, November 1999.

[Keph2005] Jeffery Kephart, Research Challenges of Autonomic Computing, *Proceedings of the 27th international conference on Software engineering,* pp. 15 - 22, New York, NY, USA, 2005.

[Keph2005b] J. O. Kephart and W. Walsh, An artificial intelligence perspective on autonomic computing policies, *Fifth IEEE International Workshop on Policies,* pp. 3-12, Washington, DC, USA, June 2004.

[KeWa2004] Jeffrey O. Kephart and David M. Chess, The Vision of Autonomic Computing, *Computer,* Vol. 36, No. 1, pp. 41-50, January 2003.

[Kini1998] Anil Kini, Trust in Electronic Commerce: Definition and Theoretical Considerations, *Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences,* Vol. 4, pp. 51, Washington, DC, USA, January 1998.

[LeMo2004] J. D. Lee and N. Moray, Trust, Self-Confidence, and Operator's Adaptation to Automation, *International Journal of Human-Computer Studies,* Vol. 40, pp. 153-184, 2004.

[LeKM2000] J. Lee, J. Kim and J. Y. Moon, What makes Internet users visit cyber stores again? key design factors for customer loyalty, *Proceedings of the SIGCHI conference on Human factors in computing systems,* pp. 305 - 312, New York, USA, 2000.

[LeSe2004] J. D. Lee and Katrina A. See, Trust in automation: Designing for Appropriate Reliance, *Human Factors,* Vol. 46, pp. 50-80, 2004.

[LMRW2005] Rogerio de Lemos, Julie A. McCann (editor), Omar F Rana, Andreas Wombacher, Academic Panel: Can Self-Managed Systems be trusted, in *International Workshop on Self-*

Adaptive and Autonomic Computing Systems (SAACS), 2005, ISBN: 0-7695-2424-9, pp. 1171- 1175, 22-26 Aug. 2005.

[LeBu1996] R. J. Lewicki and B. B. Bunker, Developing and Maintaining Trust in Work Relationships, *Trust In Organizations: Frontiers of Theory and Research,* pp. 114 --139, Thousand Oaks, CA, 1996.

[LeWe1985] J. D. Lewis and A. Weigert, Trust as a Social Reality, *Social Forces,* Vol. 64, No. 4, pp. 271-286, June 1985.

[LoLi2002] Guy M. Lohman and Sam S. Lightstone, SMART: Making DB2 (More) Autonomic, *Proceedings of 28th Int. Conf. on Very Large Databases,* pp. 877-879, August 2002.

[LuNa2004] Wenhong Luo, Mohammad Najdawi, Trust-building measures: a review of consumer health portals, *Communications of the ACM,* Vol. 47, No. 1, pp. 108 - 113, New York, NY, USA, January 2004.

[MaDS1995] R. Mayer, J. Davis and F. D. Schoorman, An integrative model of organizational trust, *Acedemy of Management Review,* Vol. 20, No. 3, pp. 709-734, 1995.

[MiLa1992] C. A. Miller and R. Larson, An Explanatory and Argumentative Interface for a Model-Based Diagnostic System, *Proceedings of User Interface Software and Technology,* pp. 43-52, 1992.

[Mode2004] Models in the Research Process http://www2.uiah.fi/projects/metodi/177.htm

[MoDZ1993] C. Moorman, R. Deshpande and G. Zaltman, Factors affecting trust in marketing relationships, *Journal of Marketing,* Vol. 57, pp. 81-101, January 1993.

[Moon 1998] Y. Moon, Intimate self-disclosure exchanges: Using computers to build reciprocal relationships with consumers, working paper No. 99-059, Cambridge, MA: Harvard Business School, (1998).

[MuMy2004] H.A. Müller, J. Mylopoulos, IBM Collaborative Research and Development Grant Proposal, 2004.

[Nielsen 1994b] Ten Usability Heuristics by Jakob Nielsen http://www.useit.com/papers/heuristic/heuristic_list.html

[Norm1998] Donald A. Norman, The Design of Everyday Things, pp. 256, August 1998.

[Patr2002] Andrew Patrick, Privacy, Trust, Agents & Users: A Review of Human-Factor Issues Associated with Building Trustworthy Software Agents, Canada, March 2002.

[Pysi2003] Jarkko Pysiaiinen, Building Trust in Global Inter-Organizational Software Development Projects: problems and practices. *Proceedings of the International Workshop on Global Software Development,* Vol. 26, No. 6, Shanghai, China, pp. 80-86, 2006.

[ReNa2003] Byron Reeves, Clifford Nass, The media equation: how people treat computers, television, and new media like real people and places. Cambridge University Press, New York, U.S.A, pp 305, 1996.

[ReHZ1985] J. K. Rempel, J. G. Holmes and M. P. Zanna, Trust in close relationships, *Journal of Personality and Social Psychology,* Vol. 49, pp. 95-112, 1989.

[RiVa1994] P. S. Ring and A. H. Van, Development processes of cooperative inter-organizational relationships, *Academy of Management Review,* Vol. 19, No. 1, pp. 90-118, January 1994.

[Rott1967] Julian B. Rotter, A new scale for the measurement of Interpersonal Trust, *Journal of Personality,* Vol. 35, No. 4, pp. 651-665, University of Connecticut, December 1967.

[RMDN2003] M. Russel, P. P. Maglio, R. Dordick and C. Neti, Dealing with ghosts: Managing the user experience of autonomic computing, *IBM Systems Journal,* Vol. 42, No. 1, pp. 177-188, March 2003.

[ScHT1973] B. R. Schlenker, B. Helm and JT Tedeschi, The effects of personality and situational variables on behavioural trust, *Journal of Personality and Social,* Vol. 25, No. 3, pp. 419-427, March 1973.

[Sous2005] João Pedro Sousa, Scaling Task Management in Space and Time: Reducing User Overhead in Ubiquitous-Computing Environments, Ph.D. Thesis, Carnegie Mellon University, May 2005.

[THLM2003] R. Telford, R. Horman, S. Lightstone, N. Markov, S. O'Connell and G. Lohman, Usability and design considerations for an autonomic relational database management system, *IBM Systems Journal,* Vol. 42, No. 4, pp. 568 - 581, October 2003.

[Tivoli2006] http://www-306.ibm.com/software/tivoli/solutions/smb/index.html

[TsFo1999] Hsiang Tseng and B. J. Fogg, The elements of computer credibility, *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit,* pp. 80 - 87, New York, NY, USA, 1999.

[Word2004] Daniel Worden, Understand autonomic maturity levels, http://www-128.ibm.com/developerworks/library/ac-mature.html, 2004.

[WhTy1999] A. Whitten and J. D. Tygar, Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0, Proceedings of the 8 USENIX Security Symposium, pp. 169–184, Washington D.C., August 1999.

[WiBT1999] Andrew C. Wicks, Shawn L. Berman and Thomas M. Jones, The Structure of Optimal Trust: Moral and Strategic Implications, *Academy of Management Review,* Vol. 24, No. 1, pp. 99-116, January 1999.

[Wrig1972] L. S. Wrightsman, Social Psychologies in the Seventies, Brookes/ Cole, California, USA. 1972.

[Zand1972] D.E. Zand, Trust and Managerial Problem Solving, *Administrative Science Quarterly*, Vol. 17, pp. 229-239, Johnson Graduate School of Management, Cornell University, June 1972.

[Zhou2006] J. Zhou, A Mobile Multi-Agent Autonomic Architecture for an Electronic Marketplace Application, pp. 1-112, Victoria, B.C., Canada, June 2006.