A computational study on collaborative behavior of Self Help Groups

A THESIS

submitted by

BUBLI SAGAR AMBATI

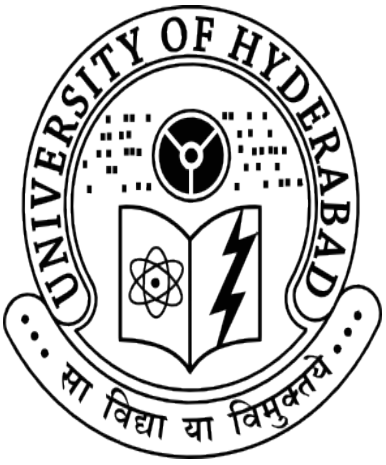
for the award of the degree

of

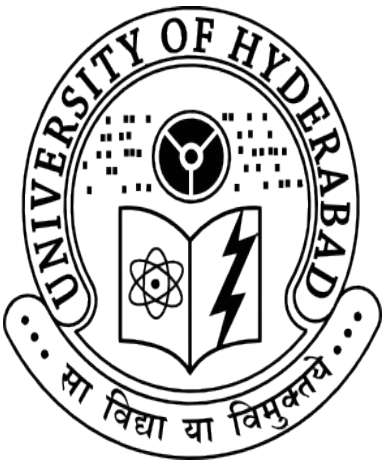
DOCTOR OF PHILOSOPHY IN
COMPUTERS & INFORMATION SCIENCES

Under the guidance of

Prof. HRUSHIKESHA MOHANTY,
University of Hyderabad



School of Computer & Information Sciences
University of Hyderabad
Hyderabad - 500046, INDIA



CERTIFICATE

This is to certify that the thesis entitled "**A computational study on collaborative behavior of Self Help Groups**" submitted by **BUBLI SAGAR AMBATI** bearing Reg. No. 08MCPC05 in partial fulfillment of the requirements for the award of **Doctor of Philosophy in Computer Science** is a bonafide work carried out by him under my supervision and guidance. The thesis has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Prof. HRUSHIKESHA MOHANTY,
Supervisor
School of Computer and Information Sciences
University of Hyderabad

Prof. ARUN K. PUJARI
Dean
School of Computer and Information Sciences
University of Hyderabad

DECLARATION

I, **BUBLI SAGAR AMBATI**, hereby declare that this thesis entitled "**A computational study on collaborative behavior of Self Help Groups**" submitted by me under the guidance and supervision of **Prof. HRUSHIKESHA MOHANTY** is a bonafide research work. I also declare that it has not been submitted previously in part or in full to this University or any other University or Institution for the award of any degree or diploma.

Date:

Name: **BUBLI SAGAR AMBATI**

Signature of the Student

Regd. No. **08MCPC05**

Abstract

According to data from the United Nations Development Programme 2010, an estimated 29.8% of Indians (400 million approximately) were living below the country's national poverty line in 2010. India's poor people can account for one-third of all poor people worldwide. The magnitude of the problem is quite staggering. Poverty eradication is one of the major objectives of planned development. As per the Planning Commission of India, poverty can effectively be eradicated only when the poor start contributing to growth by their active involvement in the growth process. This is possible through a process of social mobilization that encourages participatory approaches, institutions and the empowerment of the poor. In this, the voluntary organizations and community based Self-Help Groups(SHG) must be more closely involved. At microlevel, SHGs have shown success in providing assistance to people living in poverty. The time has come for its involvement at macro level for SHGs to take up grand objectives and large assignments so that the synergy available at SHGs can be collectively utilized for the speedy eradication of poverty. Our work on SHGs focuses on developing a framework for SHG system based on qualities such as collaborations, transparency and dependability. With reference to my engagement regarding the thesis objective, to the best of my knowledge there has not been any research done on the identified issues, particularly on the calculation and the ability to implement these ideas in the area of SHG management. Hence, the literature survey has included similar works carried out in other domains such as multi-agent systems, distributed systems, and e-commerce systems. Collaboration is important for SHGs as it enables them to execute larger projects that are otherwise impossible for execution at an individual SHG level. It enables sharing of skills and knowledge, and thus strengthens a society at large. To make a collaboration successful there would need to be some research performed, particularly in the context of the collaboration of SHGs. Transparency in a business can play an important role in building relationships among customers, investors, employees, peers and other stakeholders. Transparency is a key component in developing trust. In the proposed model, transparency is achieved by inculcating a mechanism that can deliver messages to stakeholders at different states during a task execution cycle. While satisfying a need is a factor for a collaboration, it is further qualified to a successful collaboration when a chosen collaborator is found suitable based on its history of performance. A collaborator found to be chronically well behaved would stand out for further collaborations. This feature of a collaborator is termed as dependability. From a group of possible collaborators, dependable ones are to be chosen for a purpose. Dependability

is computed from trust, competency and integrity. It is reasoned that an SHG can be counted as dependable for collaborations if it was ‘trusted’ in the past, has ‘competency’ for the present task, and has ‘integrity’ in its behavior. The present study also discusses the implementation of a prototype system.

Acknowledgments

First and foremost I want to give special thanks to my supervisor Prof. Hrushikesha Mohanty. He was not only my supervisor, but also my mentor and guide. His patience, flexibility, genuine caring and concern, and faith in me during the dissertation process enabled me to attend to life while also earning my Ph.D. His philosophical dimension is too great to be understood at first thought. He's been greatly motivating, encouraging, and enlightening. He has never judged nor pushed when he knew I needed to juggle priorities. We've laughed together and he's also been kind when I needed help. When I became too serious, his humor and friendly sarcasm allowed me to laugh and lightened my perspective. For this, I cannot thank him enough. I am forever grateful. Thank You Sir.

I am very grateful to the remaining members of my doctoral review committee, Prof. Chakravarthy Bhagvati and Dr. Siba Kumar Udgata. Their academic support and input and personal cheering are greatly appreciated. Thank you.

I am very grateful to my colleagues who have given a great cooperation. Deepak & Supriya, Vani Vathsala, H N Lakshmi, and K Ramesh Babu have been a source of great inspiration and fun.

The completion of my dissertation and subsequent Ph.D. has been a long journey. Much has happened and changed in the time I've been involved with this PhD. Finally, I have finished, but not alone. I could not have succeeded without the invaluable support of several supporters. Without these supporters, I may not have gotten to where I am today, at least not sanely.

Of course no acknowledgments would be complete without giving thanks to my parents. Both have instilled many admirable qualities in me and given me a good foundation with which to meet life. They've taught me about hard work and self-respect, about persistence and about how to be independent. Both have always expressed how proud they are of me and how much they love me. I too am proud of them and love them very much. After I lost my father, my mother has been a great role model of resilience, strength and character. I am grateful to her.

Finally, I must acknowledge with tremendous and deep thanks my wife, Ponni. Because of her I have been happier, calmer, and more content. Through her love, patience, support and unwavering belief in me, I've been able to complete this long dissertation journey. She is my biggest fan and supporter. She has taken care of whatever needed tending to without complaining, just so I could focus on completing my dissertation. She has patiently endured many long hours and days alone while I worked on my dissertation. At the same time, she has

also given me so many happy and beautiful memories throughout this journey. I could not have completed this journey without Ponni by my side. She went through every excruciating steps because of me. She loves me like no one else has and has changed me for the better. Thank you with all my heart and soul. I love you and am forever indebted to you for giving me your love and your heart. Your complete and unconditional love carries me through always.

To my beloved daughter Amrutha, I would like to express my thanks for being such a cute girl always cheering me up. Finally I thank my God, my good Father, for letting me through all the difficulties. I have experienced Your guidance day by day. You are the one who let me finish my degree. I will keep on trusting You for my future. Thank you, Lord.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

A Self Help Group [SHG] is a special kind of self-managed organization with an affinity for socio-economic development of the members. At the micro level, SHGs have shown success in providing succor to people in poverty. SHGs, for the most part in India, are poised to move to the next level of their engagement in nation building. The time has come for their involvement at macro level so that they can federate and be able to take up grand objectives and large assignments. This way, the synergy available at SHGs can be collectively utilized for the speedy eradication of poverty, particularly in India. SHGs in India are poised to next level of their engagement in nation building. In order to make a federation of SHGs work successfully, the administration of them will tend to formulate rules and monitor strict adherence to these rules. SHGs are socially responsive organizations, and therefore their governance needs also to be bestowed with social ethics. Hence the reason I have included three behavioral integrations — collaboration, transparency and dependability into the management of participating groups.

India lives in villages. In 1901, approximately 89% of India's population was living in villages while close to 11% lived in cities. Today, over a century later, roughly 69% of India's population lives in villages while 31% live in cities. The number of villages in India has increased from 6,38,588 in 2001 to 6,40,867 in 2011 [2] [3].

Rapid urbanization has given rise to the migration of poverty [4]. Though poverty has been evident to people in rural India before, now it has now started to pervade vehemently across India [5]. Furthermore, a massive increase in population has made poverty alleviation a herculean task. Though State, Non-Governmental Organizations (NGOs) and individuals are participating in poverty alleviation programs, the paradigm of 'self-help' has become a mantra for today's progress. Self Help Groups have become a successful tool in developing countries for the easing of poverty, of which India is in the forefront. India has contributed many

success stories to the utilization of SHGs[6]. But, there are many more unsuccessful stories that stare menacingly to the future of these groups which presents great challenge. We infer that automation of Self Help Group activities is just one dimension of the efforts required for their successful running. Their success also rests on socialization and therefore when involved in poverty alleviation programs, are required to be bound by social ethics and sensitivity.

The development of computing systems relentlessly intends and thrives to go beyond the input/output data processing paradigm to an intelligent decision making process, giving rise to the prospect of embedding individual cognition skills into software systems. Multi-agent systems are pragmatic implementations of such endeavors. Furthermore, efforts are being made on how to create systems that are responsive to social needs. The most current work in this direction is surrounded around the study of social networking. Drawing awareness from social networking systems such as Facebook or Twitter, which are in-tune with the social relations that exist among its users, emerges an exciting topic of research. Following the trend of system developments, I propose to incorporate social behaviors that will make SHG management systems to be socially responsive. In my present work, I have chosen collaboration, transparency, and dependability and have suggested design recommendations for further implementation.

1.1 Self Help Groups

A Self Help Group (SHG) is an association of people with similar social and financial habitats that are eager to achieve a common goal (either individually or communally) which leads to both individual and community development. This is a unique means by which individuals can be empowered to realize their inherent strengths. This concept has scripted a success story in Bangladesh and is now catching up the imagination of people, particularly in poor and developing countries. SHGs were started mostly by NGOs that had broad anti-poverty agendas. They were seen as instruments for a variety of goals including the empowering of women, developing leadership abilities among poor people, increasing school enrollments, and improving nutrition. SHGs have well-defined rules and by-laws and are self-managed institutions characterized by participatory and collective decision making. India has found itself engaged in experimentation on poverty eradication through the participation of SHGs. The preliminary experience of SHGs, in view of societal development, has been very encouraging. It seems that the SHG movement may be able to perform a few ‘frog-leap’ moves to catch up in societal development at a much faster rate. Also, now is the time to consider the sustainability

of SHGs. Initial experiments with these groups have focused on financial management. This has helped participants to cultivate a habit of making small savings and then by utilizing that knowledge to further their financial status. The movement has taken shape requiring SHGs to collaborate in order to perform better and more complex tasks by utilizing their intrinsic strengths which lie dormant and languishing under the heavy ruthless weight of poverty and social discrimination.

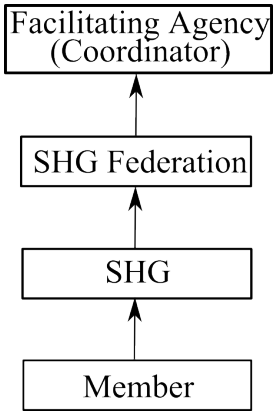


Figure 1.1 SHG Structure

The structure of a Self Help Group assumes a mostly hierarchical network (fig ‘1.1) that connects all entities to an overall figure head. However for more complex tasks at hand, occasionally the SHG must assume a different model for these different purposes. In the next section I will discuss some of these models.

1.2 Models

Basically there are three models for SHG formations and linkages. The strategy behind these models is to form small, cohesive and participatory groups. SHG-Bank linkage program has proved to be a major supplementary credit delivery system with a wide acceptance by banks, NGOs and various government departments [7]. The three broad models of linkage are [8]:

Model I: Bank — SHG — Members

Model II: Bank— Facilitating Agency — SHG — Members

Model III: Bank — NGO-MFI (Coordinator) — SHG — Members

The first model is introductory; it introduces the concept of SHGs to possible benefactors. First, at the start of self-help movement, banks play a pivotal role in gathering people to groups and nurturing them to play constructive roles in poverty eradication. In this model, the banks themselves take on the task of forming and nurturing these groups by opening bank accounts

for participants and providing them with loans after concluding that the participants are suitable to absorb credit. The banks would lend small amounts of money to the groups whilst promoting the activities of their livelihood and providing nurturing ways that enables the groups to pay back the loans on time. Thus, SHGs gain discipline in fiscal policies and enable themselves at the same time. Up until March 2006, about 20% of the overall number of SHGs that were given finances was in this category [9]. This shows an increase of 61.63 percent in bank loans to SHGs over the position in March 2005, which reflects an increased role that banks are playing in promoting and nurturing SHGs.

The first model, though helpful in a 'hand-hold' approach, it is un-scalable for incorporating necessary activities such as training SHGs in meaningful financial skills. The limitation of such a model can be alleviated through second model which introduces a facilitation agency between the bank and the SHG. In the second model, groups are formed by facilitating agencies (NGOs, in most cases) or by government agencies. The groups are then nurtured and trained by these agencies. The bank provides credit directly to the SHGs after observing their operations and maturity to absorb credit. While the banks provide loans to the groups directly, the facilitating agencies continue their interactions with the SHGs. Most linkage experiences begin with this model where NGOs play a major role. This model has also been popular and more acceptable to the banks, because some of the difficult functions of social dynamics are externalized. This model continues to carry most of the weight with approximately 70% of the overall number of SHGs being financed using it [7].

However, for various reasons, banks in some areas are not even in a position to finance SHGs promoted and nurtured by other agencies. In such cases, the NGOs act as both facilitators and micro-finance intermediaries. The present research is centered on this model where the role of the coordinator is to promote the groups, nurture them and provide financial means for their sustenance. In the next section I will discuss the sustainability of SHGs.

1.3 Sustainability of SHGs

Any social movement engaged in humanitarian, community and national development, needs a sustainable pursuance. An emotional engagement will most certainly have both high and low phases. The SHG movement is also vulnerable to the said phenomenon. In under-developed and developing countries, this movement has caught the interest of people, particularly for its tender care of the poorer populace. More specifically in India, the movement has shown to be successful and therefore has moved across societal sectors providing a flourishing means to a

greater target mass.

The following table (Table:1.1) shows the enormous amounts of bank loans issued to SHGs which portrays the interest of government in the growth and development of SHGs [1].

Year	SHGs linked to Bank	Bank Loan (in Rs Billion)
1992-94	620	0.01
1994-95	1,502	0.02
1995-96	2,635	0.04
1996-97	3,841	0.06
1997-98	5,719	0.12
1998-99	18,678	0.33
1999-00	81,780	0.36
2000-01	1,49,050	2.88
2001-02	1,97,653	5.45
2002-03	2,55,882	10.22
2003-04	3,61,731	18.56
2004-05	5,39,365	29.94
2005-06	6,20,109	44.99
2006-07	11,05,749	65.70
2007-08	12,27,770	88.49
2008-09	16,09,586	122.54
2009-10	15,86,822	144.53
2010-11	11,96,134	145.48
2011-12	9,23,000	165.34

Table 1.1 Self-Help Group-Bank Linkage Program (Amount in Rs Billion) [1]

As per NABARD's microfinance report in $March$ 2012, approximately **79.6** $lakh$ SHGs have an estimated membership of **9.7 crores** and have savings accounts in the banks with an aggregate bank balance of $Rs.$ **6,551** $crores.$ Over **43.54** $lakh$ SHGs have loan accounts totaling $Rs.$ **36,340** $crores$ loan outstanding. The total number of SHG federations that have been formed is **1.66** $lakh$, most of which are primary federations [10]. In proportion with the rising number of SHGs, the government believes that the size of the total loan disbursed to the sector will also increase in the future.

The involvement of NGOs is felt very strongly in the SHG movement. For example, the NGO called, 'Hand in Hand' (URL http://www.hihindia.org), works with a large number of SHGs which is summarized in the following table (Table:1.2).

Another NGO called SEVAI (URL http://www.sevai.in/shg.htm) has managed to form **6,073** Women Self Help Groups (WSHGs) and **331** Youth Self Help Groups (YSHGs), covering a total of **1,05,038** members. There are numerous other NGOs which are founding, nurturing, and funding thousands more SHGs.

The success of the SHG movement is not exclusive to India however, for it is known to be successful in any country where people are engaged with the movement. Hence, there is

Self-Help Groups	60,293
Members in Self Help-Groups	857,799
Family-based micro-enterprises started and/or strengthened	937,198
Micro enterprises	9,668
Women given Literacy Training	156,524
Women given Vocational Training	60,138
Credit disbursed through Hand in Hand	INR 3,923.84 million
Credit disbursed through Belstar	INR 3,722.39 million
Credit disbursed through banks	INR 2,816.74 million
Total credit disbursed	INR 10,462.97 million

Table 1.2 Hand in Hand - NGO (http://www.hihindia.org)

a search for a generic approach to see the movement be made sustainable and able to play in the 'final game' for the eradication of poverty which is threatening human kind by denying dignity of life. A broad view on sustainability is believed to have two dimensions, that is, structural and behavioral. For example, thousands of SHGs in India need an organizational structure be it hierarchical, federal, etc. and similarly, the functionality of a domain service decides behavioral aspects of SHGs. Policies, regulations and administrative rules are essential for the smooth functioning of SHGs, but more so for the encouraging of people to engage in the responsibility of the duties and obligations involved in ensuring of this smooth functioning. Adherence to rules is most desirable for administrative purposes, but it can be double-edged sometimes, as it mostly leads to bureaucracy. This state is utterly undesirable for SHGs as they are made for the poor and the vulnerable. There has been an adequate amount of study done on automation of organization activities, for example, workflow management, communication system, information management, and more, but there has been an insufficient amount of study on incorporating social aspects into automation of organization management. This has barred some studies on collaboration, belief and trust in multi-agent systems. And, coming back to automation of SHG management, no such study existed to the best of my knowledge. In the present thesis I am interested in studying the social aspects that will keep people engaged in such a program and also provide viable sustainability to the movement.

1.4 Social Role of SHGs

Self Help Groups represent an opportunity for social action and empowerment by considering, addressing and participating in matters that affect their members and their communities, including the particular issues that affect women [11]. Social activities that SHGs can take on can be found in several categories as follows: i) health, ii) social capital building and education, iii) social justice, iv) community infrastructure, and v) development activities. *Polity:* Working process of SHGs includes social problem analysis, decision making through regular meetings

and democratic procedures. Thus, in the long run, the working of SHGs fosters a healthy polity in a society. *Social harmony:* SHGs are beginning to bridge such divisions through mixed caste membership in some cases, and in others, through joint actions across groups of different castes. *Social justice:* SHGs seem to be specifically placed to support their members on issues of social justice affecting women and the downtrodden. It was observed that SHGs, whose members already enjoy some 'socio-economic' status, were able to assist their own members or extend support to other vulnerable women [12].

The success of initial SHGs in their objectives has generated an idea and offered hope that a proper and planned next level of development will also be successful. The next experiment with SHGs is to see how the groups can achieve higher objectives by solving large and complex tasks through federations. The hope is optimistic and achievable for two reasons which are the differing levels of expertise that SHGs would usher in and the participation of like-minded people working towards social uplifting. My understanding is that the evolving federations of SHGs would be desirable in addressing the larger social problems. But as the size increases, so does its entropy, bringing sustainability to center of our discussion. While strong administrative process is seen to be an obvious solution for the problem of sustainability, the same could turn to be detrimental for the SHG movement as the process could be repressive by eroding existing social values. A solution for this problem is the three dimensions for sustainability viz. collaboration, transparency and dependability. This work advocates the need of inculcating social values into systems and also formalizes these three social values. There is consistency among the three social values. For a given work, the SHGs that can collaborate are identified. During the execution of their roles while participating in the given work, both transparency and dependability are required for the maintenance of emulating a good working practice. A snapshot of the work reported in this thesis is given in the next section.

1.5 Overview of work

Following this introduction, the second chapter (Literature Survey) will make mention of the work available for the making of SHG management systems to be socially concerned. With reference to my thesis objective, and to the best of my knowledge, there has not been any study done to identify the issues, particularly on computation and the ability to implement these ideas in the area of SHG management. Hence, the literature survey has included similar works carried out in other domains such as multi-agent systems, distributed systems, and e-commerce systems.

Collaboration is important for SHGs as it enables them to execute large projects which are otherwise impossible for execution at an individual SHG level. It enables sharing of skills and knowledge, and therefore strengthens society at large. When working on a project, sometimes one SHG may not know everything it might need to, but by collaborating with other groups, skills can be pooled in order to make executing the project more successful. Also, collaboration can be a make or break component of a business. To make these collaborations successful, they need to be studied, particularly in the context of SHGs.

The third chapter of this thesis deals with collaboration — a process that results in selection of a set of SHGs that can perform a task that has subtasks. The process of collaboration essentially will be a match between the expertises of SHGs to a service/skill required by the task. For the matching or initiation of a collaboration, the use of publish-subscribe model is proposed. A match between specifications of tasks' requirements and skills of SHGs is the basic factor. Other factors that influence selection of collaborators are identified, and different structures of collaborations are also studied. Finally, a metric to measure the performance of a collaborator is proposed. It is realized that for the success of collaborations, transparency is a very essential component as well. Generally speaking, every SHG prefers to partner with another SHG which is transparent. If an SHG is not transparent in its behavior, then there is a possibility for failure or dissatisfaction over the collaboration which will lead to task execution failure.

The fourth chapter of this thesis deals with these notions of transparency — a value which defines the openness of an execution to stakeholders regarding matters that affect their interests. If a stakeholder (or an SHG) gives a task to another SHG, it requests regular reporting on how the given task is being executed. The report could include items such as; outcomes of the task, members involved in carrying it out, the current status of the task, any resources being utilized, and any problems that have been encountered. With transparency being a key component in developing trust between collaborative SHGs, it can play an important role in building relations among customers, investors, employees, peers and other stakeholders.

In the proposed model, transparency is achieved by inculcating a mechanism that can deliver messages to stakeholders at different positions within the task execution cycle, identifying the generic condition and messages corresponding to each condition. Messages are to possess the five aspects (relevance, timeliness, completeness, consistency and understandability). This model enables the inclusion of transparency for execution as well as the financial and organizational aspects. The task Execution Cycle [TEC] identifies generic states of executing a task while the SHG Behavioral Model [SBM] defines various processes of an SHG, making it

easier to define transparency.

Transparency messages are facilitated with the specification of '*state:context:action*' condition for each task. It specifies that when a specific task *state* is reached during task execution, and a *context* of the task state is encountered, then the system should generate a specific transparency message. This approach ensures transparency in task execution. The chapter also describes an algorithm for the process of reporting transparency messages detailing the different processes needed. It also provides a simple example that demonstrates computation of transparency.

Each collaborator will have a different level of collaboration performance. While satisfying a need is a beneficial factor for a collaboration, it is further qualified for successful collaboration when the chosen collaborator is found to fit based on its history of performance. A collaborator that is found to be consistently well-behaved obviously stands out for future collaborations. This quality of a collaborator is termed as dependability. From a group of possible collaborators, ones that are dependable are chosen for a particular purpose. The fifth chapter will explore this dependability trait.

It is reasoned that an SHG can be counted as dependable for collaborations if it is 'trusted' in its present status, and has 'competency' for the task in question along with 'integrity' in its overall conduct. Trust comprises of direct trust (which is a value computed based on interactions with the target SHG), indirect trust (which is computed from inputs from other SHGs which are associated with the target SHG), and recommender trust (which is a certification of SHG monitoring authority). Each type of trust is further explained by way of example, along with an explanation of the components that each type of trust comprises. Direct trust is computed from attributes such as quality, speed of completion, cost effectiveness and durability. Indirect trust is computed using behavioral attributes such as feedback, associations, social intimacy degree, etc. Recommender trust is computed using recommendations of experts who certify SHGs using their own criteria. Competency is of two types: technical and social. Technical competency is the competency acquired by an SHG based on skills, training, experience, resources & equipment, etc., whereas, social competency is based on social impact of an SHG. Integrity assessment is based on political, financial, organizational, and executional integrity. The chapter provides computation details for each of the attributes of direct trust, indirect trust, recommender trust, competency and integrity.

A metric on dependability is proposed which measures the dependability of an SHG on a scale of zero to sixteen. SHGs that get a rating in the range of 0 - 5 are regarded as not dependable, while those in the range of 6-10 are moderately dependable, and those in the range

of 11 - 16 are highly dependable.

The sixth chapter presents the implementation details of the work. Application logic of the system and of various modules is described using Use Case, Class, Activity, and Interaction diagrams. System architecture is given in the figure labeled 'System Architecture', which presents a broad view of the proposed system. The three proposed features (transparency, collaborations and dependability) are shown to be consistent with each other. Figure (Fig 6.2) depicts the relationship between these three features. The constraints of the system are specified using OCL (Object Constraint Language). Constraints for transparency, collaborations and dependability are outlined separately.

Use Case diagrams are provided for all the interactions of the system. This diagram is a representation of users' interaction with the system and can portray the variety of users of the system and the various ways they interact with it. Class diagrams are provided for each of the features along with a description of each class and its significant methods. Activity diagrams show the overall flow of control and describe how a group of objects collaborate on a number of behaviors - typically a single Use Case. These diagrams show several example objects with the messages that are passed between these objects within the Use Cases. Interaction diagrams are presented for each feature. They picture a control flow with nodes that can contain interaction. A communication diagram is provided specifying the collaboration procedure and the monitoring of the progress of task execution. While the system is in operation, several input and output interfacing screens will be generated. The user interfaces of the system are presented in the 'System Development' section of the chapter. These screens depict various forms containing input options and output data. They portray how transparency, collaboration and dependability are implemented within the system. Transparency message templates give an overview of how these messages are to be designed for the state of various tasks. The data section presents snapshots of Publications, Subscriptions, Task, Dependability and Transparency tables. Transparency message templates give an overview of how these messages are to be designed for the state of various tasks. The data section presents snapshots of Publications, Subscriptions, Task, Dependability and Transparency tables.

Presented in the seventh chapter is a summary of the completed work and a discussion on the possible aspects for future research. There is a brief summary of each chapter highlighting its contributions to the thesis.

Chapter 2

Literature Survey

Self Help Group (SHG), a variant of social network, has been successful among the poor participating in rural economic development in recent years. This movement is rooted from people's desire to meet their needs and determine their own destinies through the principle, "by the people, for the people and of the people" [13]. Coleman studied the impact of group lending programmes in North East Thailand. He found that rural banks that provide group-loans in rural areas have positive impact on women's development [14] and claimed that in order to improve rural economic conditions and enhance social inclusion, social networks (such as SHGs) have to be studied in a greater extent [15]. Several survey studies were carried out on SHGs in Andhra Pradesh, Odisha, West Bengal, etc. [16] [17] [18].

Information systems for microlevel management of SHGs have already been in operation. But now, as told in the previous chapter, time has come for SHGs to take up complex tasks that require combined expertise of several SHGs. Since SHGs are social groups, automation of their information systems need to be socially sensitive. This need is the motivation for the work reported in this thesis. The present work proposes a framework on which groups of SHGs collaborate to accomplish complex tasks. And, we study three social values — collaborations, transparency and dependability, in the gambit of a framework that formalizes associations of SHGs for a task. Many works reported have studied the issues related to SHGs from the social scientists' point of view rather than computing scientists. In the absence of such works in computing, we resort to review related works on multi-agent systems. Multi-agent model is very close to the proposed framework of SHGs. An agent in mulit-agent model can be defined as a component of a system capable of acting exactingly in order to accomplish tasks. Thus, agents directly correspond to SHGs that together accomplish tasks.

Multi-agent systems (MASs) are an increasingly popular paradigm for the study and

implementation of distributed systems that are highly complex and dynamic. Agents within such systems are generally assumed to be autonomous, self-interested, and goal-driven, interacting with each other as necessary in the pursuit of those goals. Multi-agent system design has emerged as a powerful approach to perform tasks or solve problems in a decentralized environment [19] and such systems are expected to solve problems that may be too large for a single agent, and provide enhanced speed and reliability [20]. MAS, when applied to SHGs, can be characterized by the following properties [21]: (1) each SHG has incomplete capabilities to execute a task; (2) an SHG executes a part of a task matching its skills; (3) SHGs execute tasks asynchronously. These properties imply that coordination among multi-agents is a critical aspect in making SHGs work in multi-agent framework. In MAS design, the designer first creates a local plan for each agent, and then forms a unit called $coordinator$ to integrate the inter-agent resource sharing behaviors from the local plans. Finally, the local plans and the $coordinator$ are concatenated to build a global plan - an integrated model for the whole MAS system [22]. A single SHG has limitations of visibility, mobility and skills. However, if several SHGs with varied skills can come together to form an association or a network, their strengths and skills can be multiplied, and activities can be carried out cost-effectively. Thus, looking at strong similarity between MAS and network of SHGs, I will review works of system architecture, collaboration, transparency and dependability in the domain of agent based computing.

In order to develop architecture in a systematic way, we need to analyze the system in terms of its ultimate goals and design the system both abstractly and concretely by mapping the goals to components. The implementation of a system is only as good as its design; hence, it is critical that correct design decisions are made [23]. A well thought out architecture for the system provides an effective blueprint for the system and leads to the right implementation with little error. Thus, the architecture of the system is its backbone and offers guidelines for its development. Every system has its own framework and right establishment of this framework can lead to a right system, and even right analysis and design for extending the system in future [24]. Furthermore, this is an efficient way to improve a system's reliability and performance [25]. Hattori et al. have proposed an MAS architecture for supporting network communities [26] [27]. But they did not address how to make a coalition of agents, and how humans and agents can effectively collaborate as a system. But this work proposes an architecture-based method for the systematic development of MAS for SHGs and also address the issue of collaboration. A goal-based approach is used for task execution. In order to support the coordination and autonomy of agents (SHGs), which are considered as main properties of

MAS, architectural styles and patterns are utilized in representing the system. UML (Unified Modeling Language), BNF (Backus-Naur Form) and OCL (Object Constraint Language) are used in modeling and formalizing the architecture.

To execute complex tasks, agents (SHGs) collaborate with each other and share their resources. Multi-agent collaboration is studied by many researchers in various domains [28] [29] [30] [31]. Ultimately, in all domains, collaboration means working together to achieve a goal [32]. It generally refers to individuals or organizations working together to address problems and deliver outcomes that are not easily or effectively achieved by working alone [33]. Two SHGs are said to be collaborating whenever they make a collaboration agreement and share skills and tasks, regardless of whether the sharing is productive or not. Multi-agent systems offer an efficient means for collaboration among the partners in a business [34]. Given the general benefits of multi-agents, scholars have also recognized the importance of having a central coordination agent to support and enhance the multi-agent collaboration [35] [36]. For this purpose, the current research employs SHG Coordinator to oversee collaborations and coordinate SHGs in their resources sharing. Most collaborations require leadership, although the form of leadership can be social within a decentralized and egalitarian group [37]. Of course, collaboration has its own risks for lack of cohesion among the collaboration parties [38].

Williamson further categorized business transactions into competition (market transaction), governance (internal transaction), planning (contract), and promise (collaboration) [39]. Contractor et al. believed that collaboration and competition provide alternative or simultaneous paths to success [40]. Much of the recent research, for example, Hagedoorn's research on technology partnership [41] [42] [43], Gilroy's work on networking benefits for multinational enterprises [44], Roos' paper on the cooperating strategies for wider business [45], Kay's research on innovation and trust [46], and Chen and Shih's research on high-tech development [47] signify the importance of collaborations.

The incentives for business groups to collaborate may be external or internal, but they all target the impact on final net profits or utilities in different business and political environments. Collaborations show great variety in their motivation. The external reasons or pressure that make business group collaborate may include — rapid economic and technological change, declining productivity growth and increasing competitive pressures, resources interdependence, blurring of boundaries between businesses, overcoming government mandated trade or investment barriers, labor, lack of financial resources, facilitating initial international expansion of inexperienced business groups, and dissatisfaction with the judicial process for solving problems [40], [48], [49]. As per [40], [49] and [50], on the other hand,

there are many benefits from collaborations that push businesses that are chasing profits into collaborations. Collaborations give — access to producers of information, new markets, lower coordination costs throughout the industry value chain, lower physical distribution costs, redistribution and potential reduction in total profits, reduction of innovation lead time, technological complementary, strategic market structure, rationalization of production, monitoring technological opportunities, specific national circumstances, basic R&D, and advantages of linking the complementary contributions of the partners in a "value chain".

Before entering into a collaboration, agents will make a collaboration agreement. Unfortunately, agents cannot assume that the statements of their peers regarding their competencies and capabilities are accurate. Worse still, agents must accept the possibility that their peers may be intentionally deceptive, masquerading, spreading false information, or otherwise behaving in a malign manner, in the pursuit of their own goals [51]. The fact that agents can make such assumptions greatly increases the uncertainty in their business and social interactions, which in turn introduces a significant degree of complexity to decision-making [52] and resource sharing [53]. This brings us to the theme of this thesis — the need for ethics in SHGs.

Choosing a collaborator is an important decision which directly affects the efficiency of task execution. While choosing a collaborator, several attributes need to be considered. A collaborator SHG of bigger size may have better capability, scope, process, structure, behavior, and decision making capability compared to smaller sized groups. Larger groups are more likely to possess more specialized assets, business networks, and skilled labors [54]. Size of a group has significant affect on collaboration formations and performance during task execution [55], [56], [57]. Size of a group also affects performance and success of a collaboration [58], [59], [60], [54]. Collaboration with larger-sized groups enhances an SHG's operations, process, product quality, reputation, and market position [61].

Most researchers defined size in terms of group's resources, sales, revenue, turnover, etc. [62] [63] [64]. On the other hand, definition of a group's size varies as per locality and business type. It should be argued that a group with a profile such as '*20 members - labor intensive business - located in rural area*' is significantly different from a group with a profile as '*20 members - technically intensive business - located in urban area*'.

The term "transparency" appeared in many areas, and had different meanings depending on the context it applies to. Transparency in science refers to the degree to which a medium allows radiation to pass through [65]. Transparency in philosophy refers to the phrase "referential transparency", which expresses the semantics of a given object being unaffected by the manner

of referring to it [66]. Transparency in business and law refers to information openness which allows stakeholders to see and use the information of the organization [67] [68]. It also has the notion of being easily and clearly understood or recognized, which has been emphasized in governments and organizations [69] [70], as well as in ethics [71]. Transparency in computer networks often implies that the users of a network are unaware of the computational process or artifact [72]. But in the present work, transparency implies disclosure of information and full access to it.

More information transparency in businesses facilitates better analysis, monitoring, and evaluation of events that are significant for economic and social well-being [73]. Every organization requires information to be produced, distributed and shared among stakeholders for achieving its goals through the support of stakeholders. Heald states that transparency of information allows an organization to perform its mission and periodic release of information about its performance can be used for assessment purposes [74]. Several authors have explored this concept, developing several ideas based on different approaches to transparency and exploring different varieties of transparency. The Oxford Dictionary of Economics defines transparent policy measures as "policy measures whose operation is open to public scrutiny; transparency includes making it clear who is taking the decisions, what the measures are, who is gaining from them, and who is paying for them" [75]. The present work develops transparency in line with the definition of [75].

Transparency in financial information is studied and several aspects were defined by many researchers. Transparent financial information need to be objective and understandable, and should be offered in a timely manner [76]. Park et al. stated that transparent accounting information need to be accessible [77]. Choi indicated that financial transparent information is to be offered timely [78]. Han et al. emphasized timeliness, completeness, consistency, reliability, accessibility, visibility and understandability of information for financial transparency [79]. The present work agrees with [76] [77] and [78], and the proposed framework does not contradict the above ideas. We consider the definition of [79] for financial transparency, and also extend the same to other aspects such as organizational and executional transparencies. In terms of business information, Williams et al. argued that consistent, universal representations of data were highly desirable [80]. Bushman et al. [81] presented timeliness of financial information disclosure as a variable to measure organization's transparency. Zhu [82] stated visibility and accessibility of information as the requisites for information transparency. Information disclosure for transparency needs to be timely, reliable, relevant and sufficient [83].

Elorrieta documented that high quality information means comparable, comprehensive and reliable information that is prepared under common financial reporting principles and represents economic reality, with balanced discussion of risks [84]. Damodaran presented that too many data in financial statements and their complexity make users evaluate an organization's value, and organizations that disclose understandable business information rise in stakeholder's estimation [85]. The present work agrees with the views of [84] and [85], and considers transparency as the key to win more investors, customers and other stakeholders.

Since agent-based technology focuses more on executing complex tasks to achieve defined goals, we believe that collaborating agents, besides being transparent, also need to be dependable [86]. Researchers tend to define dependability based on specific context or field. For example, dependability definition might vary among the field of psychology, sociology, and economics. Psychologists defined dependability from the perspective of personal trait, while sociologists view dependability from social structure and economists define dependability from the viewpoint of economic-choice mechanism [87]. Although a variety of dependability models are available [88] [89] [90] [91], there is not a general scheme that can be followed. In the present work I defined dependability as a combination of trust, competence and integrity and proposed a framework that fits for SHGs.

Every business depends heavily on trust. Trust serves as a mechanism to reduce collaboration complexity [92]. Trust allows agents to resolve some of the uncertainty in their interactions, and form expectations about the behaviors of others. Good interactions between agents builds trust, which in turn allows future interactions to proceed as if certain unknown (or unknowable) quantities were known. While a great number of definitions of trust have been proposed in various disciplines, the present study defines trust intuitively as the belief that others will not harm or betray, that they will behave as expected, and that they will consider others' interests as if they were their own, when relied on them for some issue. Because of the facilitating role trust plays in multi-agent systems, much has been written on the subject in recent years. Many general-purpose trust models exist, and it is not the intention of the present work to replicate this body of existing work. Rather, the study focuses on dependability by making trust as a component of dependability. In this work, trust is between pairs of agents engaging in collaboration relationships, where tasks are transferred from one agent (the trustor) to another (the trustee) depending on the level of trust between them.

There are various trust models applied for solving different issues in literature. Some trust models are based on sound theories, e.g. PKI [93], some on reputations and recommendations [94] [95], and some on Probability or Bayesian network [96] [97]. Many trust models have been

constructed for special computing environment such as ad hoc networks, peer-to-peer systems, and E-commerce [94] [95] [96] [97] [98]. One of the most widely used definitions of trust in both human-organizational and multi-agent systems domains is given by Gambetta as, "Trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action " [99]. It is useful to mention this definition here, as it is generally well accepted in both human and multi-agent systems, and allows us to avoid negotiating the ongoing sociological debate on the precise meaning of trust.

One of the most widely cited models of trust in organizations is that of Mayer et al. [100]. In this work, Mayer et al. draw together a number of subtle distinctions prevalent among theorists, such as the relationship of trust with notions such as confidence, predictability, and trust propensity. Importantly, however, this work clarified the dimensions of trust that are semantically distinct and arise from different beliefs. The two most important of these are the dimensions of competence (ability) and integrity. It is necessary to make distinction between these two, as beliefs about competence and integrity are formed from different sources of evidence, and must be addressed in different ways.

Competence is defined as the ability of an individual to perform a task to an expected standard [101]. For example, we may evaluate the competence of builders in terms of their performance, as their work can be compared with prior expectations. A builder who builds unsound homes may be considered incompetent with respect to building tasks. Competence is generally considered an innate property that the trustee has no control over, i.e., it does not involve intention on the part of the trustee. While competence may change over time (for example, by gaining experience or training), it is not the result of an intentional choice on the part of the trustee. Some agents perform similar tasks, but work with different mechanisms. So, competence is measured for the SHGs with respect to tasks. If an SHG has high competence for executing a task, it does not matter what mechanism it follows.

Competence is the most commonly modeled dimension in multi-agent systems. Many approaches consider either competence or integrity, but not both. But our approach uses both competence and integrity as components of dependability. Competence is certainly most immediately amenable to representation as a subjective probability. Integrity, on the other hand, implies some modeling of the intentions of the trustee.

Integrity is defined broadly as the belief that one individual holds about another, regarding their adherence to norms, standards, laws, or other behavioral expectations. In contrast to the

competence dimension, integrity implies some intentional choice of the trustee to behave in a certain way. For example, in a given society there may exist a social norm, such as, “it is obliged to be on time for meetings”. If a particular trustee is often late for meetings, trustors may conclude that the trustee does not consider this norm important, or is bound by another, conflicting norm. The trust in the trustee’s integrity, with respect to the social norm, will be reduced by non-compliance. Integrity beliefs can be formed by comparing an agent’s behaviors in the context of the organizational norms and regulations of the organization in which the agent is embedded. Techniques exist for specifying and reasoning with norms [102], resolving conflicts [103], and monitoring the normative behavior of agents within a system [104]. These techniques can be used to build integrity in the same way as competence, by providing a framework in which trustors can form normative behavioral expectations about their partners. With respect to rational agents, the simplest kind of integrity belief we may form is the belief that another agent will behave in such a way as to maximize their own expected utility, at all times.

The present work considers agent’s integrity (with respect to a particular norm) as a belief about that agent’s expected behavior with respect to that norm, based on prior observations. For example, behavior can be immediately evaluated with respect to norms, such as, “it is obliged to arrive five minutes before the beginning of a lecture”, because the trustor knows exactly when and how to detect a violation. Norms which persist are more difficult, and require some notion of constant observation; for example, evaluating an agent’s behavior with respect to a norm, such as, “it is forbidden to open the door”, may require constant surveillance of the door, or some alarm mechanism to indicate its opening.

The notion of functional integrity, i.e. executional integrity, may be understood as the ability to fulfill functional requirements in a complex system [86]. In agent-based environments, it may be really difficult to check whether the set of cooperating but autonomous agents is able to achieve the global goal of the system (solve the problem) [105]. The integrity check of the proposed system guarantees that the collaborating agents (SHGs) have high integrity in their behaviors. Social integrity was studied by Yu Wang et al. [106]. The present model has social integrity, but differs in the approach of computation relying only on the social impact of an SHG.

Thus, the present chapter gives an overview of the various works related to SHGs, multi-agent systems, transparency, collaborations and dependability.

Chapter 3

Collaborations

Collaboration means working together to achieve a goal [32]. It generally refers to individuals or organizations working together to address problems and deliver outcomes that are not feasible or effectively achieved by working alone [33]. The majority of collaborations will require leadership, even if the form of leadership is social within a decentralized and egalitarian group [37]. Given that SHGs have leadership at different levels, it is possible for them to have organized collaborations. Collaboration involves identifying the right groups with the most relevant skills, personalities, knowledge, work-styles, ethical values, etc. to ascertain whether they could share the same commitment to carry out the task at hand, sharing the same environment, tools, knowledge, training, process and facilitation to guarantee they work together efficiently.

SHGs, by definition, ought to be collaborative, which would result in groups being empowered to execute tasks, which otherwise would not be feasible to be done by individual groups. Collaboration is important for SHGs as it enables them to execute complex tasks by sharing skills and knowledge. When working on a project, sometimes an SHG may not have the knowledge needed to complete a particular task. This would be when collaboration with other SHGs would be most helpful in completing the project. The process of collaboration essentially is identifying a match between the expertises of an SHG to the service/skill required to perform a task. Also, it can be a make or break component of a business. To make a collaboration successful there needs to be some research done particularly in the context of SHG collaboration.

3.1 Organization of the chapter

This chapter discusses collaborations among SHGs and cites the benefits of such collaborations. Firstly, an outline is presented for the structure of tasks in relation to collaborations. Secondly, there is a discussion about the process of finding collaborators. Usually, several collaborators may be available for any given task, and therefore a selection criteria is defined in order to choose the most suitable collaborators available. After the selection of such collaborators, a collaboration process is defined to actually become engaged in collaboration. Thirdly, a discussion is put forward regarding task monitoring for when the collaborated task execution has been initiated. Finally, the last section gives definition to the method of rating a collaborator on completion of a collaboration to assist other SHGs judge the performance of an SHG as a collaborator.

3.2 Tasks

Let us first discuss the format of specification and other aspects of a task. A task is a basic requirement for collaboration and task execution is one of the chief functions of an SHG. Each SHG receives tasks either from the coordinator or from peer SHGs. When an SHG receives a task, it checks whether it has enough resources to carry out and complete the task. If the necessary resources are available, it then can execute the task. But if it does not have enough resources, it will then be required to look for collaborators. A task executed with the collaboration of other SHGs is referred to as $coltsk$ (collaborated task). Usually, all collaborations occur over $coltsks$ and for each $coltsk$, skills/resources are shared. The structure of a task can be represented as follows in BNF:

$< task >::= < task\_id > \quad < task\_desc > \quad < body > \quad [< other\_info >] \ [< task >].$

$< task\_id >::=< string >< integer >$

$< task\_desc >::= < task\_type >< req\_spec >$

$< task\_type >::=< string >$

$< req\_spec >::=< string >$

$< body >::= < serv\_needed > [< skill\_level >][< duties >] < task\_proc >$

$< serv\_needed >::= < string >$

$< skill\_level >::=< string > | < integer >$

$< duties >::=< string >$

$< task\_proc >::=< string >$

$< other\_info >::= \ [< task\_priority >] < task\_distance >< notif\_recp > [< risks >]$

$[< cond >]$

$< task\_priority >::= < string > | < integer >$

$< task\_distance >:= < integer >$

$< notif\_recp >:= < string >$

$< risks >:= < string >$

$< cond >:= < string >$

$Task\_Id$: Each task is identified by a unique identifier. $Task\_Type$: Describes the business type to which the task belongs. Req_Spec: Specifies the details of materials to be used and the required outputs of the task. $Serv\_Needed$: Specifies what services or skills are needed for the task in question. $Skill\_Level$: Specifies the level of skills needed for the task. $Duties$: This gives a clear description of the responsibilities required of the skilled personnel. $Task\_Proc$: Contains a detailed description of how the task is to be executed and the steps to be followed. $Risks$: This provides a list of risks and problems associated with the task. $Task\_Priority$: Specifies whether the task is of high priority or normal importance. $Distance:$ Gives a numerical value of the task, deduced using complexity and number of subtasks. $Notif\_Recp$: Lists the stakeholders that need to be notified about the task updates. $Cond$: Specifies additional constraints associated with the task, such as duration, execution site, speed, social impact, etc.

3.2.1 Task Procedure

Let us study task procedure in more detail here. A task in execution has a *start* state and an *end* state. We assume that each state of a task is observable. A complex task can be split into smaller subtasks for easier execution, and hence, the execution time of a subtask will be less than its parent. The steps to be followed for executing a task are elaborated on in $task\_proc$. Each step details the instructions in simple terms specifying how the work is to be carried out. It also specifies the states at which updates are to be reported. On successful completion of a step in $task\_proc$, an update report is sent to the specified notification recipients. This is how transparency in coltsk execution can be achieved.

$Task\_proc$ can be elaborated as follows:

$< task\_proc >::= < step > < workDesc > < preCond > < postCond > < workRpt >$.

$Task\_proc$ is basically a composition of steps, each marked with at least one sign of success. Also, a work report ($workRpt$) is associated with each step to mark the task status. Each $workRpt$ is packed with additional information and sent to the respective notification

recipients. These $workRpts$, once modified into 'transparency messages', provide transparency in task execution.

For example, let us consider two steps of a task procedure for making a door. *Step 1* - Cut the stiles and rails 1/2 an inch longer than the height and width of the door. The door size must be 4ft wide and 7ft tall. Use 2 inch wide boards. *Step 2* - Round over the front inside and outside edges of each piece with a 3/8 inch round-over bit.

Illustrated here we can see that there are two $workRpts$ for these steps. After successful completion of each step, a $workRpt$ is to be sent to the appropriate notification recipients informing them of the successful completion of each step, along with the task status and other details. Thus, task procedure makes it easy for the performers to complete a task, and also informs those facilitating and monitoring that the task is executed correctly.

A role is a service offered by an SHG. In other words, it is equivalent to a skill or resource. Collaborations over tasks can be managed well if they visualize each SHG as a pool of resources. Each SHG has members with different skills and abilities, enabling them to be classified into different roles. Besides material assets, these roles are the resources of SHGs that are shared in collaboration. Every SHG is made up of a set of members and therefore every group will carry out context specific roles. This is to be seen in correspondence with the professional expertise a member of an SHG has been endowed with. A member for a particular role performs certain actions corresponding to that role. This approach helps us to bring SHGs and tasks to a common platform because those tasks that have specific sets of activities requiring the participation of specifically skilled persons will need to be matched according to the activities required for the task. This matching is the foundation for all collaborations.

3.2.2 Task as distance

Let us now examine how a task can be converted into distance. Evaluation, monitoring and visualization of task progress are easier when a task is represented as a distance. For this representation, consider the properties of a task such as duration, resources, and skills. As per the proposed framework, a task is comprised of subtasks, which may in turn also contain smaller subtasks. Each subtask may be of different type, and each subtask type may have several instances. While converting a task into distance, we use the complexity of each subtask type and the number of subtask instances. Considering the complexity of each subtask gives the benefit of differentiating between difficult subtask instances from easier ones. If such differentiation is not made, then there is a possibility of failure to grasp specific information about a task's

progress.

Complexity can be computed from factors such as — number of hours of work for each instance, total members required for each instance, requirement of technical skill level, requirement of machinery, duration of work for each instance, amount of resources and skills for each instance. Later on, when the system is implemented in real life, more factors may need to be added to give more relevance to complexity.

$$Task\_Distance, D = \sum_{i=1}^{n} (numInstance(subtask_i) \ * \ instanceComplex(subtask_i))$$

Suppose, a subtask is to make instances of $complexity$ 5, and the number of instances is 100, then the subtask's $distance$ can be computed as:

$$\begin{aligned} subtask\_distance, \ d &= numInstance \ * \ instanceComplex \\ &= 100 \ * \ 5 \\ &= 500 \ units \end{aligned}$$

This distance can also be used for calculation of an SHG's capacity.

In order to specify an SHG's $capacity$, we can use this $task\_distance$. The capacity of an SHG is the time taken by that SHG to cover that distance. This capacity is computed using its available skills and resources. Thus, this $capacity$ is specific to a subtask type or skill. And because, there there will usually be many subtask types that an SHG could execute, an SHG can therefore have several $capacity$ values. There will be different capacity values for the different subtask types. Also, an SHG's capacity is not a permanent value because it keeps changing with the addition and/or loss of members, upgrading of their skills, and resource availability, etc.

3.2.3 Speed of task execution

Let us now evaluate the speed of task execution. Because we have the task_distance, this allows us to calculate the speed of execution required to complete it. Computing the speed of execution gives SHGs the possibility to verify whether they need collaborations or not. Also, before giving a task to an SHG, the giver can verify whether the SHG's speed is sufficient for the task or not. Speed of execution can be computed as follows.

$$speed\,of\,execution = \frac{task\_distance}{duration}$$

The speed of execution and the SHG's capacity can be used for monitoring progress of the task execution. For monitoring and analyzing task completion rate, the below visualization (fig: 3.1) can be used. The visualization shows that each task has several subtasks, each subtask has several instances and each instance has a task_distance. When an instance is completed successfully the task will move toward the goal by a distance corresponding to that instance's task_distance. And for task completion rate we can use basic usability metric such as $binary\ task\ completion\ rate.\ WorkRpts$ in the task procedure give a clear definition for successful completion of each instance. When an instance is completed successfully, it's allocated a value 1, else 0. After testing all the instances the average of all the 1's and 0's will give the task completion rate. For example, if 9 out of 10 instances were completed successfully, the completion rate is 90% when expressed as a percentage. Since an SHG's capacity is known, and speed of task execution can be monitored in real time, it is then possible to compare them and monitor the task progress accurately.

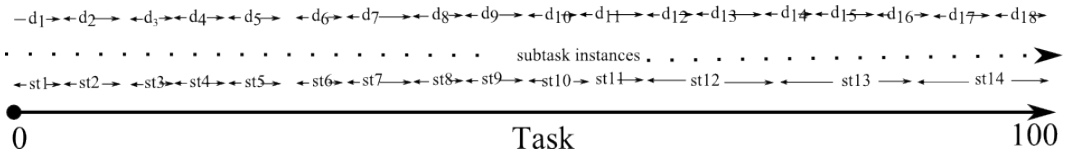


Figure 3.1 Task and Distance

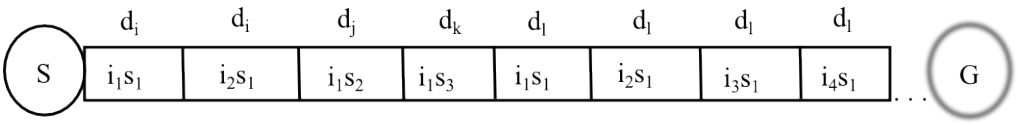


Figure 3.2 Task Completion

This model enables us to monitor task execution progress despite the order of completion of subtasks and instances. It also enables us to indicate the slightest progress in task execution as instances also have intermediate milestones earmarking progress in execution in addition to completion of the task. Instead of waiting until the end, or for long periods of time to discover that a task execution is not progressing as required, this model instantly identifies any delay.

3.3 Collaborations In SHGs

Let us now study the collaboration procedure in more detail. Occasionally SHGs receive tasks that they cannot complete with their existing resources [i.e. skills, materials, machinery, workers, etc]. In such cases, they need to collaborate with other SHGs that can provide the necessary resources. SHG collaboration is a joint effort of multiple SHGs to execute a complex task (or tasks). Hence, collaboration has more credibility, influence, and ability to accomplish goals compared to an individual SHG working alone. Other advantages of collaborations include reduced production costs, access to less expensive labour, increased creativity and innovation, better products/services, improved revenue opportunity, and shrinking distances and time. In general, collaborations between business parties are chiefly profit oriented, whereas SHG collaborations have broader objectives. Besides profits and business growth, SHG collaborations are meant for the uplifting of socially and economically backward SHGs. For this reason, during the process of choosing collaborators, SHGs that are socially and economically backward are given preference. Thus, making SHG collaborations not only benefit SHGs but also society at large.

3.3.1 Collaboration Representation

To represent collaborations we need three basic entities. The first being an SHG that needs collaboration, secondly, a task over which collaborations are sought, and thirdly is a list of SHGs that are participating in collaboration. If 'G_i' is an SHG that is in need of collaboration, 'T_a' is a task over which collaboration is sought, and G_j, G_k, G_l, G_m and G_n are the collaborating SHGs, then we can represent collaboration basically as seen in fig: 3.3.

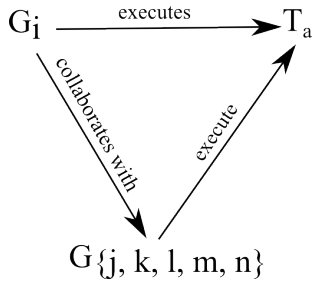


Figure 3.3 Collaboration Representation

This representation is intuitive and provides basic information. However, it could be observed that it does not address some of the other important aspects of collaborations, such as collaboration types, association of collaborators with subtasks (if the task has subtasks), distances of subtasks, task progress, etc.

Furthermore, when considering a task that has several subtasks and each subtask requires a different type of collaboration, this representation of collaboration needs to be enhanced. For example, consider a task T having different types of subtasks - t_1, t_2, t_3, t_4, t_5, and t_6. Subtask t_1 can be performed by the SHG not requiring a collaborator and subtask t_2, needs collaboration. Also the collaboration for t_2 has to be simultaneous. That is, the collaborator and the task owner SHG have to be physically present at the site. Subtask t_3 requires three collaborators, but they do not have to be simultaneously present at the site. Subtasks t_4, t_5 and t_6 have to be performed together and the collaborators are also required to be present at the site location.

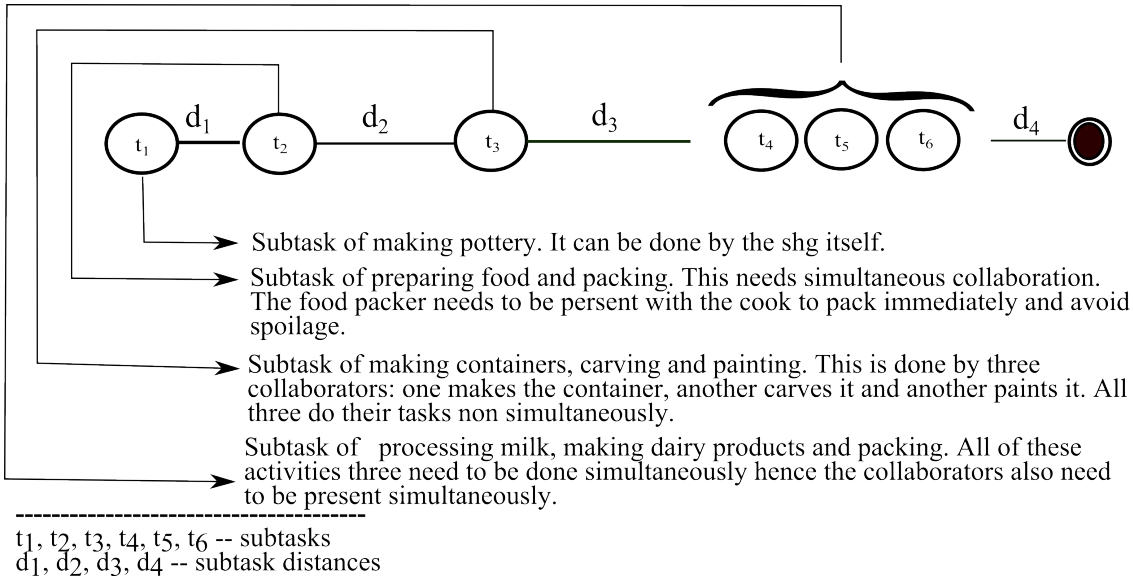


Figure 3.4 Example for different types of collaborations

The representation found in fig: 3.3 could not accommodate all the different types of subtasks and collaborations as required by the task found in fig: 3.4, therefore let us improve the representation by introducing two new symbols ([ ] & { }) to indicate different collaboration types. Now '[ ]' denotes simultaneous execution and '{ }' denotes non-simultaneous execution. Collaboration scopes such as 'Partial' and 'Complete' along with collaboration site locations such as 'Local' and 'Remote' attributes are denoted using symbols P, C, L and R respectively. The enhanced representation can be seen in fig 3.5.

Now, Fig: 3.5 provides more details about collaborations of a task. We have 'd_1', 'd_2', 'd_3', 'd_4' and 'd_5' to represent subtask distances, which can be summed to give the task's total distance. Also, it is now easier to understand that subtask 't_1' is being executed by the SHG itself locally, without the need of a collaborator, and the collaborators 'C2' and 'C6' are executing the subtask 't2'. And since 'C_2' and 'C_6' are enclosed in brackets '[ ]', we understand that these collaborators are working on the subtask 't_2' simultaneously. We observe

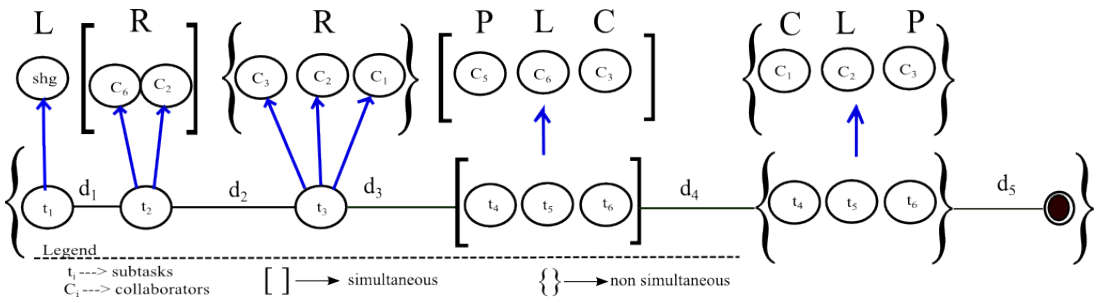


Figure 3.5 Modified collaboration representation

that the subtask 't_3' is being executed by 'C_1', 'C_2' and 'C_3'. They are encapsulated in braces '{ }', and therefore we understand that these collaborators are not working simultaneously. It means that while one collaborator is working on subtask t_3, the other collaborators may be away and need not be physically present along with the collaborator(s). Then, we observe that the subtasks 't_4', 't_5' and 't_6' are being executed by 'C_3', 'C_5' and 'C_6'. Both the collaborators and subtasks are enclosed between brackets '[ ]', which implies that all three collaborators have simultaneous participation at the simultaneous execution of the three subtasks. And finally, we observe that the collaborators 'C_1', 'C_2' and 'C_3' are executing the subtasks 't_4', 't_5' and 't_6' non-simultaneously. The letters 'L' and 'R' specify whether a task is being executed locally (at the SHG's site) or remotely.

The letters 'C' and 'P' indicate whether collaborators are partially or completely involved with a task's execution. If a collaborator is partially involved, it implies that a collaborator will not be responsible for the complete execution of a task, but only for an agreed portion of a task. If a collaborator is involved completely, it implies that it is the collaborators responsibility to see the task through to completion. This representation is extensible because additional details about collaborations can be added to the representation as per requirement. Each SHG keeps track of its task or coltsk progress with the help of this representation.

3.3.2 Choosing Collaborators

Now let us will study the criteria for choosing collaborators. While choosing a collaborator, in addition to checking an SHG's socio-economical backwardness, several other attributes need to be considered. Choosing a collaborator is an important decision that directly affects the efficiency of task execution. A collaborator SHG of larger size may have better capability, scope, process, structure, behaviour, and decision-making compared to smaller groups. Larger groups are more likely to possess more specialized assets, business networks, and skilled labours [54]. Collaboration with larger-sized groups enhances an SHG's operations, process,

product quality, reputation, and market position [61]. A larger sized SHG will have alternate members to do a job in case a member fails or leaves. The size of a group has significant affect on collaboration formations and performance during task execution [55], [56] & [57]. It will also effect the success of a collaboration [58], [59], [60] & [54]. There is another dimension to size as well and that is that the group size does not necessarily mean strength in numbers. Most researchers defined size in terms of a group's resources, sales, revenue, turnover, etc. On the other hand, the definition of a group's size varies as per locality and business type. It could be argued that a group with a profile as '*20 members, labor intensive business, and located in rural area*' is significantly different from a group with a profile as '*20 members, technically intensive business, located in urban area*' because an SHG in a rural area tends to have less revenue and turnover compared to an SHG in an urban area.

The following is a list of some more attributes to be considered when choosing collaborators:

- *Locality*: For some types of tasks it is necessary that a collaborator should be located geographically closer. For a rural SHG, collaborating with an urban SHG gives more scope for business development. And for an urban SHG, collaboration with a rural SHG gives advantages such as minimizing labor costs, goods and services. For some tasks it is necessary for a collaborator to be located in such a neighborhood where it can have easy and inexpensive access to any necessary or backup resources.

- *Transaction Costs*: Transaction costs are the costs associated with the exchange of goods or services that were experienced in overcoming market imperfections. Transaction costs cover a wide range such as communication charges, transportation charges, traveling charges, etc. [107]. These are critical in deciding the selection of collaborators. Transaction costs affect the net profit; if they are high, then profits will be less.

- *Services*: The services required by a task of an SHG should match with services available with a collaborator. The degree of matching between a collaborator's available services and a task's required services should be high. The higher the degree of matching, the better the chances will be for a successful collaboration.

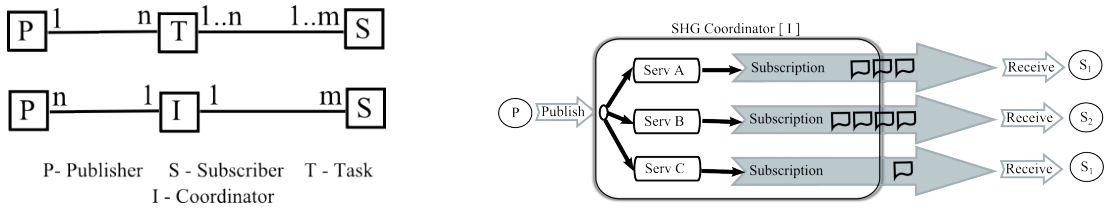
- *Speed feasibility*: In terms of task as a distance, the distance covered by a collaborator of a task i.e. capacity of a collaborator (or, the effective speed of execution when a collaborator joins the SHG) should be equal to or greater than the speed required by the task.

3.4 Pub-Sub model for SHG Collaborations

In this section let us study the Pub-Sub model and discuss how it is employed for seeking collaborators for SHGs. "Publish and Subscribe (PubSub) model is a well-established communications paradigm that allows any number of publishers to communicate with any number of subscribers asynchronously and anonymously via an event channel". In this PubSub model, message producer is the publisher and message consumer is the subscriber. One publisher can send a message to multiple subscribers. Publishers and subscribers may not know each other. There could be a number of publishers publishing, and a number of subscribers receiving those publications.

Subsequently this PubSub model is very consistent with the scenario of SHGs looking for collaborations, and it can be used for implementing collaborations in SHGs. Each SHG with a task and looking for collaborators is a $publisher$, and each SHG with available resources and willing to provide collaborations to a task is a $subscriber$. Coordinator maintains one database, which contains information of all $publishers$ and their corresponding publications, and another database with all of the $subscribers$ and their subscriptions. The coordinator receives tasks from $publishers$ and matches them against subscriptions of $subscribers$. All the matched $subscribers$ are notified about the publications. Interested $subscribers$ will initiate an agreement process with the $publishers$ to become collaborators.

Each publication is addressed to a virtual channel known as a $service$. Service-to-subscribers is a one-to-many relationship. Each $service$ may be subscribed by several subscribers and therefore, one publication may be consumed by multiple $subscribers$. Meanwhile a $service$ can have multiple $subscribers$ with every $subscriber$ receiving a copy of all matching publications. PubSub is a push-based model which implies that all publications are broadcast to several $subscribers$ without them having to poll for the subscribed content. $Subscribers$ that are disconnected and not listening actively would not receive any publications available on the $service$ during the time they remain disconnected. Thus, any publication available on a $service$ when a $subscriber$ was inactive would be lost. SHGs are mostly based in rural areas, however, and may have technical difficulties receiving the publications. These technical limitations should not hinder the competent SHGs from becoming collaborators. Hence we need the publications to persist when a $subscriber$ is inactive and deliver the publication whenever that $subscriber$ becomes active. For this reason SHGs should have a PubSub model that allows durable subscriptions, allowing $subscribers$ to disconnect, reconnect again and collect all publications that were delivered when they were not active.



(a) PubSub entities (b) PubSub operation

Figure 3.6 PubSub Model

Each $publisher$ [P] publishes a task [T] to the SHG coordinator [I]. But, all $publishers$ publish to one coordinator and all $subscribers$ subscribe to that same coordinator.

We can define PubSub model as a 6-tuple.

$< PubSub >::= P, S, I, T, PubDB, SubDB$

where

P: Publisher

S: Subscriber

I : SHG Coordinator

T : Task

$PubDB$: Publications Database

$SubDB$: Subscriptions Database

PubSub model uses other entities such as publication message, subscription message, unsubscribe message, etc., and their definitions are given below:

1. *Publication Message (PM)*

   PM is a message used for publishing the task of a publisher to the coordinator. The format of PM is as follows:

   $PM(Task, TTL)$

   The message (PM) comprises of the $task$ for collaboration, TTL (Time To Live). TTL specifies a time limit for the validity of a collaboration request. TTL is required to keep publications from being valid forever.

2. *Subscription Message (SM)*

   SM is a message sent by a subscriber to register with the coordinator. A subscriber informs the coordinator about its available services/resources and also specifies how long they will be available. An SHG's services/resources cannot have the same availability all the time, so there is a Time To Live (TTL) attribute to specify the limit of availability. We define the SM's format as follows:

$SM(SHG\_id, serv/resources, TTL)$

3. *UnSubscribe Message (USM)*

   USM is a message sent by a subscriber to cancel its subscription at the coordinator. An un-subscribe message contains the details about the service that has to be cancelled. We define USM's format as follows:
   $USM(SHG\_id, serv)$

4. *Publishers Database (PubDB)*

   PubDB is the publications database maintained by the coordinator. It contains listing of all publishers and the corresponding tasks they have published. Whenever an SHG publishes a new task, a new entry of the task corresponding to the publisher is added to the PubDB. Whenever a publisher sends a $Stop$ message, or a message that a collaborator was found, and a collaboration agreement was successfully made, then the corresponding publication is removed from the PubDB.
   $\{p, t\} \in$ PubDB

5. *Subscribers Database (SubDB)*

   SubDB is the subscriptions database maintained by the coordinator. It contains the list of all subscribing SHGs and the services with which they have subscribed. Whenever an SHG subscribes with new roles or resources, a new entry of subscription corresponding to the subscriber is added to the PubDB. Likewise, whenever a subscriber sends a USM message, or a message that a collaborator was found, and a collaboration agreement was successfully made, the corresponding subscription is removed from the SubDB.
   $\{ s, serv\} \in$ SubDB

3.4.1 Basic Functions in SHG's PubSub Model

In SHG's PubSub model, the following basic functions are used.

1. $publish()$: This method is used by publishers to send tasks to the coordinator. Each publication contains the tasks specifying the services required for collaboration, task descriptions, and other conditions related to the task. This method will add the publication to the PubDB.

2. $subscribe()$: This method is used by subscribers for registering their available services with the coordinator. This method will access SubDB and add a subscription

corresponding to the subscriber. If the subscriber has subscriptions already, then it will update the subscription accordingly.

3. $unsubscribe()$: Availability of the subscriber's services are subject to time. If the services of a subscriber become unavailable at any time, then it can unregister its services with the coordinator using the unsubscribe function. This method removes the corresponding subscriptions and updates the SubDB accordingly.

4. $notify()$: This method is used by the coordinator to send different types of messages. Mostly the coordinator uses this method to inform subscribers about a published task that matches, or is relevant, to their subscriptions. By using this method for multicast messaging, all SHGs whose subscriptions match the task's requirements can be informed simultaneously without delay. Because this scheme notifies all matching subscribers and provides an equal opportunity to become collaborators, this scheme can be regarded as a fair scheme.

5. $ack()$: This method is used for sending acknowledgments. Ack confirms the receipt of items and it also adds transparency to the system. When the coordinator notifies subscribers about matching, subscribers acknowledge the receipt of notification using the ack method. Also coordinator uses ack to inform publishers or subscribers that their items were received and added to the databases successfully.

6. $agree()$: After a subscriber receives notification about a task, it checks whether the task is feasible or not. If the task is infeasible, it drops the task. But if it finds that it has enough resources and the task is feasible, then it will inform the publisher about its willingness using the $agree()$ function. Sometimes the subscriber may accept with some conditions. And if the publisher accepts the conditions, then it will convey acceptance through the $agree()$ function, and they proceed to collaborate. Thus, the $agree()$ method is a negotiation method between the publishers and the subscribers.

7. $stop()$: After a publisher finds a collaborator, or it wishes to drop a publication, then it uses the $stop()$ method. This method will stop the notification of the task to subscribers.

Interaction diagram (Fig:3.7) below shows the process of collaboration using the Interaction diagram.

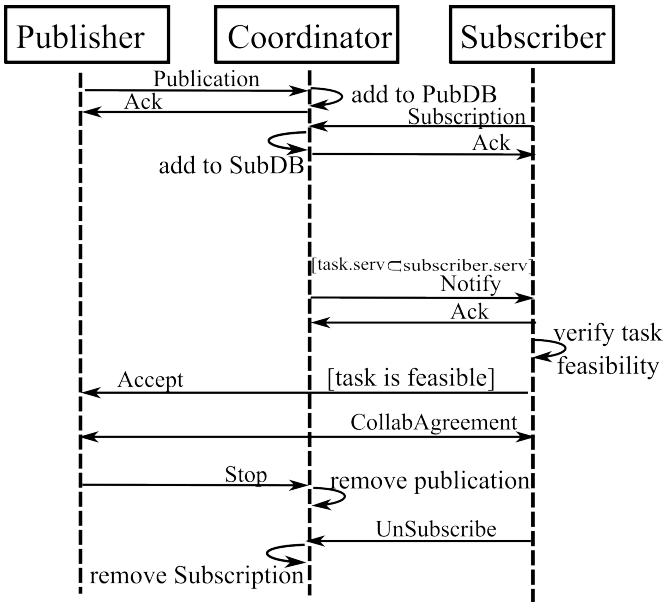


Figure 3.7 Interaction diagram

Following are some rules in PubSub model:

- *Publish rule:* Each publication adds a publisher and its task to the PubDB.

  $pub(p,t) \Longrightarrow PubDB \longleftarrow \ PubDB \ \cup \ (p,t)$

- *Subscribe rule:* Every subscription adds a subscriber and its services to SubDB.

  $sub(s,serv) \Longrightarrow SubDB \longleftarrow \ SubDB \ \cup \ (s,serv)$

- *Unsubscribe rule:* Unsubscription of an SHG implies removal of the subscription from the SubDB.

  $unsub(s,serv) \Longrightarrow SubDB \longleftarrow \ SubDB \ - \ (s,serv)$

- *Notify rule:* In the event of a publisher publishing a task it is always followed by the matching subscribers in SubDB being notified.

  $notify(s,t) \Longrightarrow \exists s,t \,|\, (t \in PubDB) \wedge (s \in SubDB) \wedge (t.req\_serv = s.avail\_serv)$

- *Agree rule:* A subscriber's acceptance to collaborate for a task implies that the published task's required services match with its available services and the conditions of the task are acceptable to the subscriber.

  $agree(s,p,t) \Longrightarrow (t.req\_serv = s.avail\_serv) \wedge feasible(t.conditions) \wedge notify(p,t)$

- *Stop rule:* When a publisher finds a collaborator or withdraws its task, it asks the coordinator to stop advertising. Stopping an advertisement implies the removal of the task from the PubDB.

$$stop(p,t) \implies PubDB \longleftarrow PubDB - (p,t)$$

Algorithm for PubSub Model

The following is the algorithm of PubSub model.

The $Publish$ method first splits the task into subtasks according to the services required by a task and those that are not present at the publisher. After creating subtasks it will send them to the coordinator to be advertised. The $CreateSubtask$ is a sub function in the $publish$ method which splits a larger task into smaller subtasks. The $Subscribe$ method is for SHGs to subscribe to the coordinator with their available services. Each SHG that wants to collaborate will provide the available services to the coordinator. If a subscriber does not want to collaborate, or some of the subscribed services become unavailable at some point in time, then it cancels its subscriptions. Unsubscription means that the subscriber and its services are removed from the SubDB. The collaboration process starts with the coordinator informing relevant subscribers about the matching tasks. Subscribers check the services, locality, size, transaction costs, and speed required by the publisher. If the subscribers find that the received task and its conditions are feasible, then they notify their willingness to the publisher so that they can proceed to make a collaboration agreement. If a collaboration agreement was successful then the subscriber unsubscribes its services at the coordinator so that it is not notified of future tasks until its services become available again. A publisher, after finding a collaborator, will ask the coordinator to stop notifying other subscribers about the task.

PubSub Algorithm

Task t,tt;
SHG p,s;
Coordinator I;
message unsubmsg, cancelmsg;
service r[ ];
SubDB S; PubDB P; TaskDB T;
Publish(p, t)
{
while($t.serv\_req > 0 \wedge t.serv \notin p.serv\_avail$) { {
tt = CreateSubtask(t);
send(p,I,tt);
$t.serv\_req = t.serv\_req - tt.serv\_req;$

} }

}

Subscribe(s, r)

{

$if\ (s.serv\_avail \neq null\ )$

then

{

$r \longleftarrow s.serv\_avail$

send(s,I,r)

$if\ (s.serv\_avail \sim s.serv\_subscr) \neq null$ then

$update(s, r)$

}

}

Unsubscribe(s, r)

{

$if(s.serv\_avail = null)\ then$

$send(s, I, unsubmsg)$

}

Notify(s, t)

{

for every t in T do {

for every s in S do {

$if\ ((t.serv\_req \subseteq s.serv\_subscr\ )$

then

$Send(s, t)$

}

}

}

Agree(p,s,t,r)

{

$if\ ((s.serv\_avail = t.serv\_req)\ \wedge (feasible(t.cond)))\ then$

{

send(p,s,t,r)

unsubscribe(s,I,r)

Stop(p,t)

}

}

Stop(p,t)

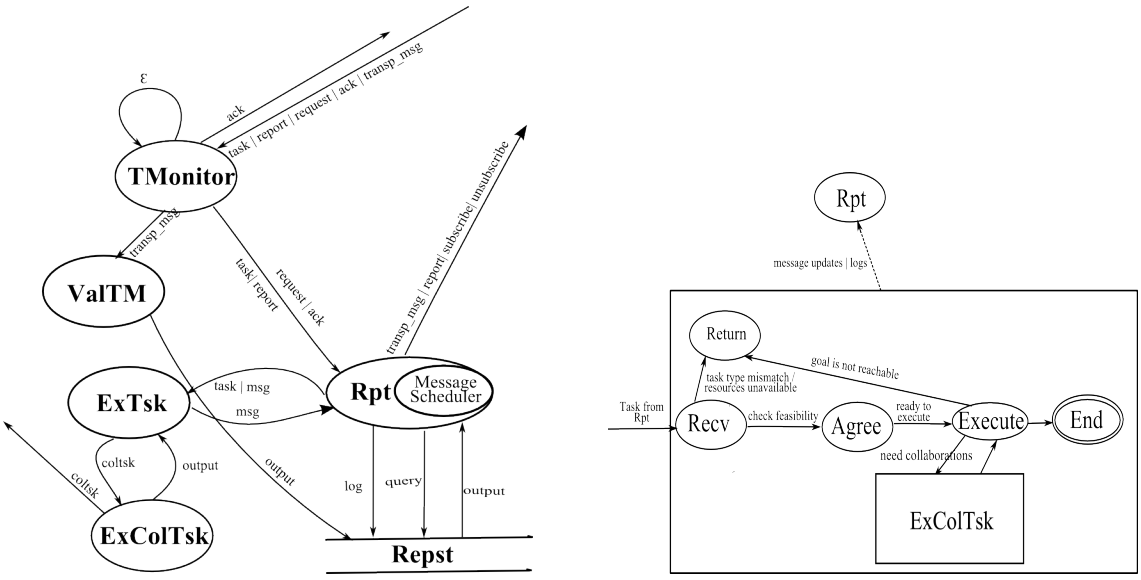
{ $if((p.hascollab(t)) == true)then$

$send(p, I, t, cancelmsg)$

}

3.4.2 Collaborated Task Execution

Let us now study how a collaborated task is executed and what the various states are that a collaborated task attains during its execution.



(a) SBM (b) States of a Task in ExTsk

Figure 3.8 SBM and Task Execution States

SHG Behavioral Model [SBM] gives a generic view of SHG behaviour and details how items such as tasks, reports, requests, etc. are dealt with at each SHG. TMonitor is a continuously running process that keeps monitoring for all items. When a task is received at TMonitor it sends acknowledgment to the task sender and forwards the task to Rpt process. The Rpt process logs the details of the received task in the repository (Repst) and forwards the task to ExTsk process. At ExTsk the task is analyzed and a decision is made whether it can

be executed or not. If it is decided that a task cannot be executed then the task is dropped. But if it was found that a task could only be executed with the help of collaborations, then the task or part of the task is forwarded to the Excoltsk process. A task that is performed using collaborations is called a coltsk and the ExColTsk process monitors it. The Coltsk process has the following states: CInit, CAgree, CRunMonitor, CSuspend, and CEnd as depicted in fig: 3.9.

CInit: When the collaboration process is initiated for coltsk, it will be in CInit state. In this state an SHG starts searching for collaborators by using the PubSub model. After subscriber SHGs show willingness to execute a task, the agreement process is initiated, and the state changes to the CAgree state. In the subscriber SHGs side when they receive a task it is referred to as the Recv state. After they respond to the corresponding publisher, it then moves to the CInit state. When they start the agreement process it moves to the CAgree state.

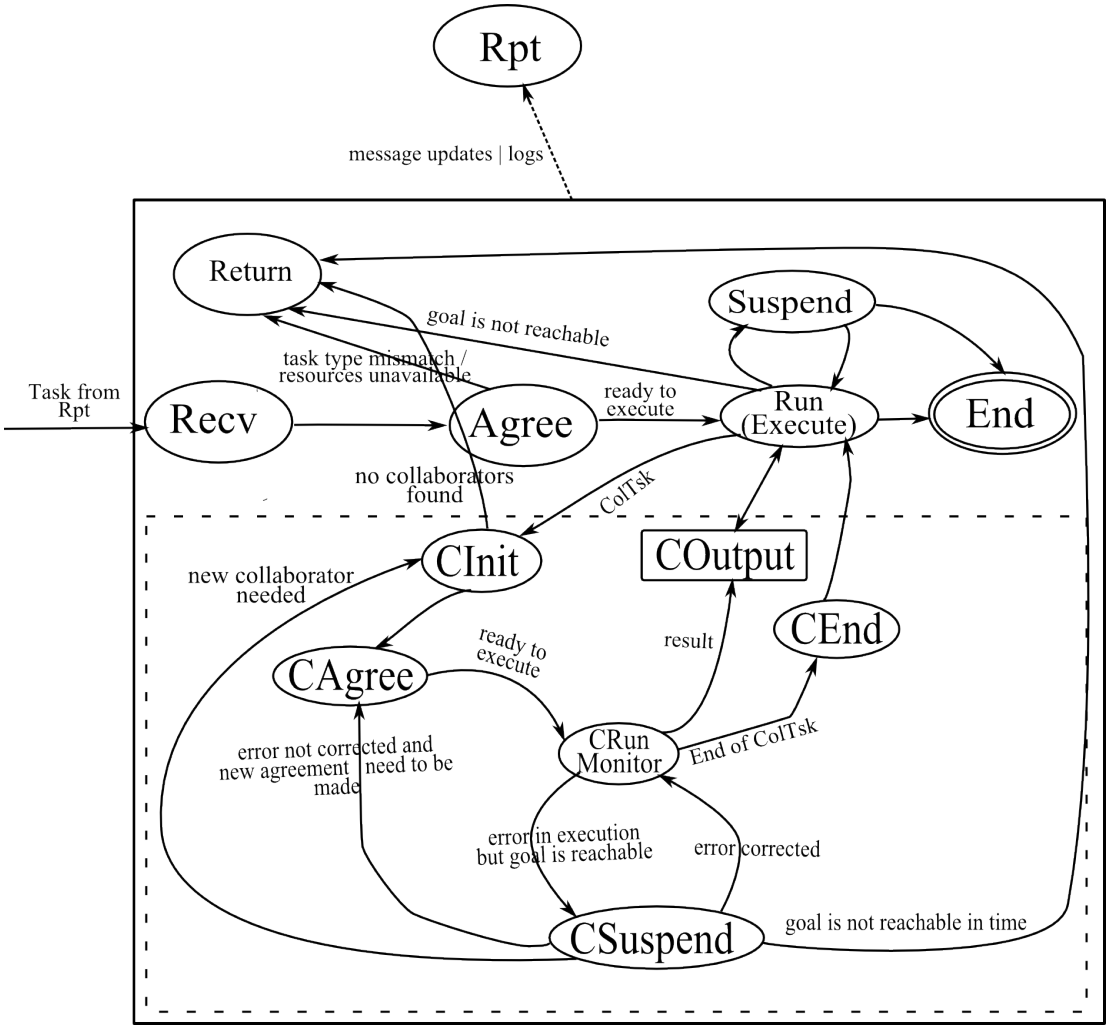


Figure 3.9 Collaborated Task Execution diagram

CAgree: When a coltsk is in the CAgree state, both the publisher and subscriber SHGs make an agreement with each other's conditions for collaboration over the execution of the coltsk. The Coltsk will stay in this state until the publisher and subscriber SHGs make a

successful agreement and are ready to begin execution. Once they set out to execute the coltsk, it moves to the CRunMonitor state where its progress is monitored and reported.

CRunMonitor: Coltsk remains in this state throughout the execution. Here several updates become available and they are all sent to Rpt process for logging and reporting. The CRunMonitor state contains information pertaining to the present state of coltsk, status of collaboration, progress, status of resources, estimated time for completion, audit information, and so on.

CSuspend: During execution of a coltsk, problems such as misunderstandings in collaboration, unavailability of resources, weather conditions, business conditions, and technical problems, for example, may arise and hinder the execution. In such cases coltsk is suspended and its state changes to CSuspend. Later when it is restarted it moves back to the CRunMonitor state. CSuspend is a sleeping state meaning that there will be no progress in coltsk execution and hence no updates will be available. If favourable conditions for restarting the coltsk did not appear, and it was decided that execution of coltsk couldn't be continued, it then is returned to the task sender. Afterwards coltsk moves to the Return state.

CEnd: When a $coltsk's$ execution is completed, it moves to the $CEnd$ state. The next state to $CEnd$ is again $Execute$ because any remaining portion of work in the task has to be completed before moving to the End state.

Risks in Collaboration

Generally all collaborations will have some associated risks and SHG collaborations are no exception. Choosing to collaborate with other SHGs is definitely a reasonable option for an SHG to execute complex tasks. In spite of several advantages, collaborations also have many considerable risks. Some of the risks are as follows: (i) outcomes may not justify the time and resources invested (ii) beneficiary confusion (iii) loss of flexibility in working practices (iv) complexity in decision-making and loss of autonomy (v) cultural mismatch between groups (vi) the diversion of energy and resources away from the core aims (vii) change management challenges (viii) lack of consistency and clarity on roles and responsibilities (ix) dilution of brand (x) damage to the group if the collaboration is unsuccessful and (xi) legal obligations. Risk levels such as technical complexity, benefit distribution, etc. increase as the depth of collaboration increases. With the dynamic changes in markets it is hard to predict whether a business is safe when contributing to a new collaboration or not. The "bad" collaborators are those that occupy some resources but do not contribute any tangible or intangible benefit.

But on the same lines, we can't say that the profit making groups can definitely be "good" collaborators. The learning process is a huge work and also beyond an SHG's ability. It needs several real cases to clarify the blurred boundary between the "good" and "bad" collaborators [50]. However, one near solution to this problem was proposed in the next section. The idea is to evaluate each collaborator at the end of a collaboration by giving it a value reflecting its performance in the collaboration. The $CollabRating$ metric helps in making a rational judgment about the performance of an SHG as a collaborator.

3.4.3 Collaboration Rating ($CollabRating$)

Let us understand how a collaborator can be rated over a task and study the attributes of $CollabRating$. This is a fairly realistic metric system to help make a reasonable judgment. For this purpose, first the attributes that are required in each collaborator are identified and a specific range of values is defined for each attribute. Each collaborator is rated for the following characteristics.

Timeliness and Speed

The timeliness of a collaborator is about reaching a goal in a specified time. If a collaborator reaches a given deadline in a specified time, then it is said to have timeliness in task execution. Values for timeliness can range from negative to positive. Timeliness can be calculated using speed of task execution. Every SHG has different speeds corresponding to different task types. If an SHG keeps up the required speed for task completion then it is an SHG with timeliness. We measure timeliness on a scale of -2 to +2. If an SHG maintains the speed of task execution as agreed during the collaboration agreement, then the value for timeliness is 1. If an SHG executes a task faster than its usual speed, then its timeliness value ranges between 1 and 2. But if the speed of task execution is slower than agreed speed, it then gets negative values ranging between 0 and -2 corresponding the amount of delay.

$$Agreed\ Speed\ of\ Execution\ (CollabAgree) = Speed_{agreed}$$

$$Speed\ During\ Execution = Speed_{exec}$$

$$\therefore Timeliness\ of\ SHG\ for\ task, Time_{task} = (Speed_{agreed} \sim Speed_{exec})$$

$$on\ a\ scale\ of\ -2\ to\ 2$$

Productivity

Productivity measures the relationship between inputs into the production process and the resulting outputs. There are some problems in measuring SHG productivity. For example, by simply observing the activities of the group you could not make an adequate judgment about productivity. For SHGs that make physical products, measuring productivity becomes a problem when several SHGs work on each unit. It would be difficult to articulate who is helping and who is hindering. You can hardly hold an SHG accountable for low productivity when it spends its time waiting for others to finish their parts of the job. As a result, we measure an SHG's productivity based on the fulfillment of the conditions necessary for productivity. If the conditions are satisfied, and the collaborator does not give enough output as agreed, it would then get a negative value corresponding to the amount of reduced output. It would get a value of 1 if it gives the expected output, and a positive value of more than 1 if it gives more (or better) output than agreed.

The range of values for productivity is -2 and +2.

$$Productivity\ agreed = Prod_{agreed}$$

$$Actual\ productivity\ = Prod_{exec}$$

$$\therefore Productivity\ of\ SHG, Prod_{task} = (Prod_{agreed} \sim Prod_{exec})$$
$$on\ a\ scale\ of\ -2\ to\ 2$$

Communication

This measurement is the responsibility that an SHG has in communicating during collaboration. According to the proposed model, at every stage of task execution there is a workRpt that is to be sent to the specified notification recipients. A collaborator is rated for its responsibility in conveying workRpts to the stakeholders. The range of values for responsibility in communicating is -2 to +2. If a collaborator sends all the required workRpts, then it has been good in its communication and the value given is 1. And if it sends more workRpts than specified or agreed, it would then get a value of more than 1. But if it does not send enough workRpts, then it gets a 0. If the number of workRpts is far less, then it gets a negative value.

$$Communication\ required\ = comm_{req}$$

$$Communication\ actually\ maintained\ during\ execution\ = comm_{exec}$$

$$\therefore Communication\ of\ collaborator, Comm_{task} = (comm_{req} \sim comm_{exec})$$
$$on\ a\ scale\ of\ -2\ to\ 2$$

Commitment

Commitment is the state or quality of being dedicated to a cause or activity. A collaborator must be committed to the task given to it. If a collaborator makes an agreement and accepts a task but later fails to execute it, we regard this as 'lacking commitment' to complete the received task. But if a collaborator persists in its task execution and completes the task successfully despite all problems, then it is considered as a '*committed collaborator*'. The range of values for commitment can be from -2 to 2. If a collaborator completes the task it gets a '2', but if it returns the task without completing it, it would get a '-1'. If it completes it but with a delay then it is either given a 1 or 0 corresponding to the amount of delay. It also is given a '-2' if it fails to complete it and also does not even notify regarding its failure to complete the task.

$$Status\ of\ task, TaskStatus = \{\ failed,\ completed\ with\ delay,\ completed,$$
$$failed\ and\ not\ reported\ \}$$

$$Commitment\ \ of\ collaborator = Commit_{task}$$

$$Commit_{task} = -1, if\ \ Taskstatus = failed$$

$$Commit_{task} = 2, if\ \ Taskstatus = completed\ in\ time$$

$$Commit_{task} = 0\ or\ 1, if\ Taskstatus = completed\ with\ delay$$

$$Commit_{task} = -2,\ if\ Taskstatus = failed\ and\ neither\ returned$$
$$the\ task\ nor\ informed\ notification\ recipients\ about\ failure$$

CollabRating

Finally, using all of the above attributes, the $CollabRating$ of a collaborator j computed by i over a task with $task\_id$ can be given as follows:

$$CollabRating_{ij}^{task\_id} = Time_j^{task\_id} + Prod_j^{task\_id} + Comm_j^{task\_id} + Commit_j^{task\_id}$$

While the range of values for each of these attributes is -2 to 2, the range of CollabRating is -8 to +8. CollabRating value allows us to have an idea about an SHG's performance as a collaborator. If the CollabRating of an SHG is between -8 and 0 it implies that it is one of the worst collaborators. Similarly if it is between 4 and 8 then it is considered one of the best collaborators. Likewise one could make reasonable assumptions for the intermediate values between 0 and 4 of CollabRating value.

3.5 Conclusion

This chapter has set out to discuss the concept of Collaborations in SHGs and has identified the importance and benefits of collaborations. The study has tried to find a means to bring tasks and executors to a common platform by converting tasks and capabilities into distance. The format of the structure of tasks has been specified using BNF. The study resulted in creating a new representation for collaborations. This representation allowed us to identify most of the aspects of collaboration in the form of diagrams. The study employed the PubSub model for finding collaborators and has discussed various methods used in the PubSub model along with presenting an algorithm showing the functionality of its basic functions. Since there could be more than one collaborator for a task the study identified certain criteria for choosing the most suitable collaborators. It also explained the process of engaging in a collaboration and execution of collaborated tasks. The study explored various task states during the execution of a task. It showed that collaborations are very advantageous, however, there are some risks associated with them, and some such risks were highlighted. To assist others with gaining an understanding about the quality of collaboration of a certain collaborator, the study proposed a collaboration rating measure for each collaborator. This enables other SHGs to judge the performance of an SHG as a collaborator. There are still several other issues that need to be studied and resolved in real time. This chapter investigated some of the fundamental and noteworthy aspects related to collaborations in SHGs.

Chapter 4

Transparency

Transparency means openness of decisions and actions. Put another way, transparency means a free flow of information from the source to the recipient [108] [109]. Transparency can also be defined as "*the openness of information for stakeholders of a business, regarding matters that affect their interests*" [110]. Doubts and uncertainties that stakeholders have can lead to a loss of faith and trust in a business. A lack of transparency, which affects collaborations and results in loss of faith and trust in a business, ultimately leads to the collapse of a business. Recent observations revealed that there is an increase in societal attention to the issue of transparency and it is also predicted that transparency will become the most required premise in the future for gaining and maintaining stakeholder trust and collaborative relationships. To achieve the trust of stakeholders, a business unit must provide timely, accurate, relevant and sufficient information to them. The information should be qualitative and quantitative, enabling stakeholders to make a rational assessment of the businesses activities [111]. At present, there are studies about information disclosure that are qualitative [112] [113], but this chapter proposes a quantitative transparency and provides a framework for its implementation in the SHGs' system.

4.1 Motivation

SHGs have a huge potential to excel in business which leads to the achievement of any social objective [114]. They can collaborate with each other, forming *business networks* or *chains of businesses,* and compete with large firms in production and performance. Foreseeing this potential of SHGs, several banks and government have invested millions of dollars for the development of SHGs. But the potential of SHGs may not be materialized if they become victims of corruption and the stakeholders eventually lose trust on them. In this connection,

significance of transparency in businesses is to be realized. Transparency has the potential to minimize corruption, improve trust, and maximize chances for business success and growth.

Information asymmetry is another reason which makes transparency necessary in a business. Information asymmetry is defined as a situation in which one party in a transaction has more information than the other. Practically, this could be a potentially harmful situation because the party with more information can take advantage of the other party's ignorance. Hence, increasing information transparency can minimize information asymmetry between the parties. In the present study, the parties are SHGs, funding agencies (investors), government bodies, coordinator, expert groups, etc. SHGs receive funds from government or funding agencies, and tasks from coordinator or peer SHGs. SHGs execute the tasks and use the funds for task execution, skill development, acquiring resources & equipment, etc. Hence, SHGs are the hubs of information and they possess all information about task status and expenditures. Stakeholders only have the information that is provided by these task-executing SHGs. Stakeholders completely rely on SHGs' reports and updates for their information. Thus there is an information asymmetry. It is necessary to reduce this information asymmetry and strengthen relationships between stakeholders and SHGs, and also improve governance by NGO-MFI (Coordinator) [115].

Another component that affects the success of a business is trust. Trust, which results from transparency, is considered a critical factor in determining the success of a business [116]. In the case of SHGs, where the investors and task executors are usually separated by large geographical distances, trust is extremely essential.

Also, funding agencies do not fully know of SHG's' actual skills and capabilities in executing tasks, since their knowledge is only based on the history of the SHGs. Owing to these reasons, requirement of transparency is strongly felt in the SHG system [117].

4.2 Organization of Chapter

The chapter begins with a brief introduction of transparency and cites the motivation for study of transparency in the SHG system. Some of the important previous works of researchers who have studied transparency, with respect to different types of business and social units are cited. It is also observed that the present work agrees with most of their views and that it extends the scope discussed by them. It identifies different stakeholders and their interests and represents them using a Use Case diagram. Transparency is to be maintained for financial, organizational, and

executional aspects. But in the present study, the scope of discussion was limited to executional transparency only. However, the same scheme can be employed for financial and organizational aspects. The present work employs transparency messages for achieving transparency. So, this chapter discusses various transparency messages and their message formats. It then discusses transparency quantification and also presents a representation for assessment of transparency. It also discusses the process of reporting transparency messages, corresponding algorithm, and an example for computing the transparency value.

4.3 Literature Survey

The term "transparency" appears in many areas, and has different meanings depending on the context it applies to. Transparency in science refers to the degree to which a medium allows radiation to pass through [65]. Transparency in philosophy refers to the phrase "referential transparency", which expresses the semantics of a given object being unaffected by the manner of referring to it [66]. Transparency in business and law refers to information openness which allows stakeholders to see and use the information of the organization [67] [68]. It also has the notion of being easily and clearly understood or recognized, which has been emphasized in governments and organizations [69] [70], as well as in ethics [71]. Transparency in computer networks often implies that the users of a network are unaware of the computational process or artefact [72]. In Information Technology, transparency implies to disclosure of information and full access to it. The Oxford Dictionary of Economics defines transparent policy measures as "policy measures whose operation is open to public scrutiny; transparency includes making it clear who is taking the decisions, what the measures are, who is gaining from them, and who is paying for them" [75]. The present work considers transparency as implied by Information Technology and the definition of [75].

Transparency in financial information is studied by many researchers who defined its several aspects. Transparent financial information needs to be objective and understandable, and that should be offered in a timely manner [76]. Park et al. stated that transparent accounting information needs to be accessible [77], and Choi defined that financial transparency should be offered timely [78]. The present work agrees with [76] [77] and [78], and the proposed framework does not contradict the above ideas. We consider the definition of [79] for financial transparency, and also extend the same to other aspects such as organizational and executional transparencies.

4.4 SHG Transparency

In this section we shall discuss transparency with respect to SHGs. To achieve transparency, an SHG needs to disclose information to its stakeholders. The figure (Fig: 4.1) presents the stakeholders and the transparency aspects. The basic stakeholders of SHG are executor, manager, and financier. Executor represents the task executing SHG. collaborator, or any other entity connected to task execution.

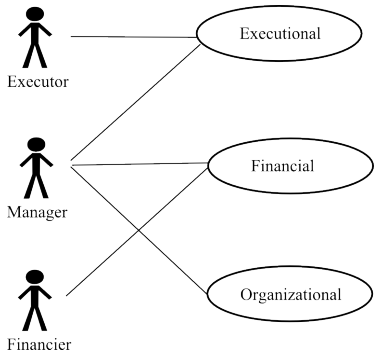


Figure 4.1 Transparency information for stakeholders

Transparency, to be holistic, is required in executional, financial and organizational aspects of an SHG. In order to achieve transparency in all workings of an SHG, there is a need to make transparency parameters observable during all its processes and process states. Since task execution is considered the most important functionality of an SHG, we shall limit the scope of discussion to executional transparency. Executional transparency involves bringing transparency to all entities related to task execution. In the present framework, transparency is made possible through reporting of updates or status of each entity in each process. Task execution involves several processes, and each process has different states where the values of entities are modified or updated. Now, in order to explain the idea of achieving executional transparency, we first have to carve out some of the related contours of the behavior of SHGs, thus resulting in an abstraction of SHG Behavioral Model [SBM] (Fig:4.2).

4.4.1 SHG Behavioral Model [SBM]

Let us discuss SHG Behavioral Model [SBM] and its processes in more detail. SBM is a generic abstract model. It is the way of determining the functioning of each SHG in the system. The table (Table 4.1) lists the processes in the SBM. There are several processes in the SBM and each process has predefined roles and specifications. SBM (Fig:4.2) lists only a few processes, and these processes are only meant for the present discussion of executional transparency. But

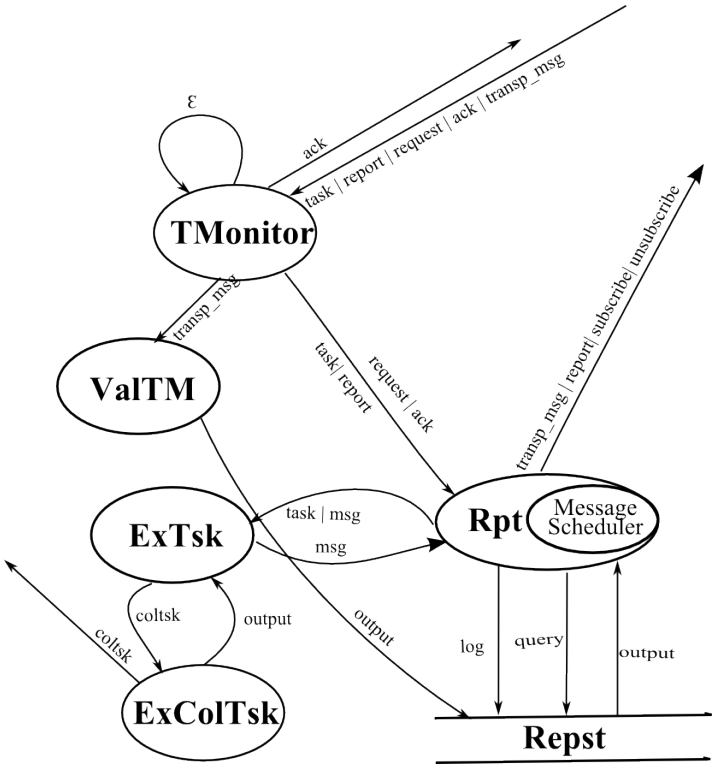


Figure 4.2 SHG Behavioral Model

when we discuss organizational and financial transparencies, more processes are to be added to this SBM. Each of the five processes i.e. TMonitor, ValTM, Rpt, ExTsk and ExColTsk have varied and well defined functionality. Each process, in its own way, contributes to the executional transparency of an SHG. Details of each process is discussed in the subsequent part of this chapter.

Process	**Function**
TMonitor	Receive tasks and other items
Rpt	Report data
ValTM	Validate transparency messages
ExTsk	Monitor task when it is in execution
ExColTsk	Monitor coltsk when it is in execution

Table 4.1 SHG Behavioral Model Processes

In the proposed system, transparency is achieved through exchange of transparency messages. Transparency is of two forms: external transparency and internal transparency. External transparency can be defined as the disclosure of information of an SHG in its workings to others. Internal transparency can be defined as the transparency built into the system during system development and implementation. This is comparable to a hygiene worker. The two aspects of consideration are: what hygienic methods the worker is using in the society, and the other is, how personally hygienic is the worker. Similarly, being transparent externally

implies that it provides to others the full amount of information when it is required. Being transparent internally implies that it is built in a way for facilitating transparency. Internal transparency is obtained by using transparency concepts in the operation of the system of the SHGs. It is related to concepts such as database storage, message scheduling, logging, information retrieval, etc. Thus it is mostly the hardware and software specifications of the system. Internal transparency needs to be addressed specifically only if SHGs use different hardware and software in a distributed system. But if the SHGs use a centralized system, and hence share the same hardware, software, and databases, then internal transparency is not relevant and need not be calculated separately because all SHGs will acquire the same value. However, the interest is on assessing the transparency behavior of SHGs occurring because of its choices and actions. Therefore, I shall discuss less of internal transparency and focus more on 'external transparency'. Also, hereafter I shall refer to 'external transparency' as 'transparency' and omit the word 'external'.

Each process in the SBM has varied functionality, and each process goes through different states during the life cycle of task execution. So, at each state of a process, relevant transparency messages are defined. By sending these transparency messages to relevant stakeholders, transparency is achieved. Let us now breifly define functionality of each process, its states, and corresponding transparency messages.

TMonitor: It is a process which monitors and receives all correspondence for an SHG and also acknowledges each item it receives. It is like an antenna for an SHG's communication. TMonitor receives items such as tasks, coltsks, requests, reports, acknowledgments, etc. Item senders need confirmation that their item was received at the receiver's end. This is the first step to transparency. A sender is to be notified that its item was received successfully. TMonitor offers this first step of transparency by sending an ack to the sender to confirm the receipt of an item. Transparency of an SHG with respect to TMonitor process is inferred based on the number of acknowledgments sent in response to the number of items received.

During execution of an assignment i.e. a task allocated to an SHG, several processes are involved, and Rpt (Reporting) is one such process. Rpt is designed to communicate with external parties by sending reports, transparency messages, reply to queries, etc. It also interacts with repository (Repst) for storage as well as retrieval of information. TMonitor forwards the received items such as request, ack, task, coltsk, report, etc. to Rpt, and items such as transp_msg to ValTM.

ValTM: ValTM is a process which validates all transparency messages. All received messages are verified for the properties such as relevance, timeliness, completeness, consistency

and understandability. If any such message is received that does not satisfy these properties, then it is regarded as an invalid transparency message. If an SHG sends invalid transparency messages, it will negatively affect its transparency. To ensure internal transparency, ValTM assesses the outbound messages and sees that they have the transparency message properties.

Rpt : Rpt has varied functionality depending on the type of item it receives. If the received item is a task, it forwards the item to ExTsk and logs all information regarding the task. To facilitate internal transparency, log information includes details about the task sender, notification recipients, milestones, deadlines, skills required, etc. If the received item is an ack or report, it stores the details in the repository (Repst). And, if the received item is a request, it queries the Repst, generates a report and dispatches it to the requester.

During the execution of a task, ExTsk sends all updates to Rpt. Rpt first logs the updates in the repository and then dispatches them as transparency messages to the notification recipients. Rpt has a Message Scheduler to alert an SHG to send messages at regular/specified intervals. Sometimes stakeholders make requests for reports concerning certain tasks. In such cases, Rpt generates the reports and sends them to the requesters. Message dispatch occurs on two occasions. First is when a request is received from stakeholders, and second is when the Message Scheduler alerts. So, transparency here is about sending update messages at the right time and to all the required recipients.

Rpt process does not monitor any tasks, it only prepares transparency messages and sends them to the respective notification recipients. Rpt forwards tasks to ExTsk, which is the process that monitors executing tasks. During the execution of a task, transparency is maintainable if, as intended, during change of states during task execution, the specified transparency messages are reported to stakeholders as required.

ExTsk : ExTsk process receives tasks from the Rpt process. ExTsk first checks a received task for its feasibility by matching it with the existing skills/resources. If a task is found feasible, then a message of acceptance [AgreeTsk] is sent to Rpt to be forwarded to the task sender. In the case that a task is found infeasible, it then checks whether the task can be completed with the help of collaborations. In such a case, the task is forwarded to ExColTsk process which deals with collaborated tasks. ExColTsk looks for collaborators to determine if the task can be completed using collaborations. But if no collaborators are available, then the task is returned to the task sender. In ExTsk, after sending the *AgreeTsk* message to the task sender, the SHG waits for the *StartTsk* message to start executing the task. Once this *StartTsk* message is received, the SHG starts executing the task and ExTsk process starts monitoring the task status. Whenever there is a change in the task status the task state and all associated information is sent to the Rpt.

Transparency at ExTsk is determined by promptness in reporting the task status and information to Rpt. ExTsk should notify changes in the task states such as *initiated, running, suspended, ended, aborted,* etc. accompanied by financial and non-financial transparency information. This information has to be manually provided into the system by the SHGs. For example, if the task state is $Agree$, then the update message contains information specifying the terms of agreement, members involved in the task, leader/supervisor of the task, resources and skills to be used by the task, expected date of completion, expected outputs of task, etc.

If the state is $Running$ (executing), then the update message contains information specifying progress of the task, expected date of completion, financial information, etc. If the state is $Suspended$, then the update message contains information specifying the reason for suspension, people or circumstances responsible for suspension, loss incurring because of suspension, probable length of suspension, factors that may revive the task to it returning it to the $Running$ state, the possibility for the task to never recover from suspension and be abruptly terminated, etc. If the state is $Ended$, then the update message contains information specifying detailed report of the executed task. It specifies the number of days it took for completing the task, number of days the SHG worked, and the number of days it was idle, the reasons for being idle, the problems encountered during execution, the contribution of other SHGs, the lessons learned from executing the task, and whether the SHG is willing to do another similar task. However, not every aspect of information is of interest to all the notification recipients. So, each update message contains information which is specific to notification recipients. So, transparency of an SHG with respect to ExTsk implies that updates *must* contain *all* the information as required by the notification recipients. ExTsk process monitors the tasks that are being executed solely by that SHG. But if any task is to be executed with collaborations, then such a task is forwarded to ExColTsk. A task which is executed with collaborations is called coltsk. Coltsk execution is more complex and hence a separate process called ExColTsk is created. It has more states than ExTsk.

ExColTsk : ExColTsk process receives collaborated tasks (coltsks) from ExTsk. It is responsible for monitoring the execution of all collaborated tasks and reporting their details. After receiving a coltsk, an SHG searches for collaborators. It uses Publish Subscribe model to find collaborators, and after finding a collaborator it will enter into an agreement for collaboration. From the beginning of a collaboration to the end of it, a coltsk undergoes several changes in its states. Transparency of an SHG with respect to ExColTsk implies updating the Rpt about all the state changes and associated transparency information. ExColTsk has to report details regarding – collaborators selected, resources and skills being shared, conditions of

collaboration, projected profits of collaboration, foreseeable risks, members' involvement, etc. in all the states of coltsk. Every time there is a change in the state of coltsk, a lot of important information becomes available for reporting.

Suppose the state is $CInit$. It means that ExColTsk is in the activity of finding collaborators. The update message contains information specifying the type of collaboration being sought, resources/skills required, length of collaboration, target SHGs, etc. If coltsk's state is $CAgree$, it means that a negotiation is happening and an agreement is being made with collaborator(s) for that coltsk. So, the update message contains information specifying details of the collaborators, agreement details, etc. If coltsk reaches $CRunMonitor$ state, it implies that coltsk's execution has started. So, the update message contains information specifying progress of coltsk, present milestone reached, next milestone and predicted date of reaching it, feedback on collaborator's behavior and performance, new skills acquired during collaboration, trainings held or taken by the members, resources utilization, funds required for further execution, problems at hand, possible solutions, etc. Throughout the execution period of coltsk, it remains in $CRunMonitor$ state only.

If coltsk reaches $CSuspend$ state, it implies that the coltsk execution was suspended. So, the update message contains information specifying why the coltsk was suspended, any issues with the collaborator, status of the resources, probable date when the coltsk may be resumed, any losses due to suspension, members activity during suspension, probable solution from coordinator or funding agency to get the coltsk running again, conditions for resumption of coltsk, necessity of new agreements, possibility of changing the collaborator, etc.

If coltsk reaches $CEnd$ state, it implies that the collaboration was completed successfully. So, the transparency message contains information specifying utilization of funds, benefits received by members, new skills obtained by members, willingness of the SHG to take up a similar task again and the same collaborator, behavior and performance of the collaborator, feedback, etc. If coltsk reaches $Return$ state, it implies that the collaboration and coltsk execution has failed. So, the update message contains information specifying why the task has failed, who is responsible, what evidence there is, what would have prevented the failure, loss incurred by failure, feedback, etc.

The table (Table 4.2) details the processes names, associated transparency messages and purpose of the messages. However, it should be noted that this is not the complete list of the messages. This is only a small set of prototype messages provided for discussing the idea of transparency messages. When the system is implemented in real life, there will be larger sets of messages. The formats of each of these messages are given in the subsequent section.

Process	MsgName	Msg Description	Purpose of Transparency Msg
Tmonitor	Ack	acknowledgement	receipt of item by the receiver
	Nak	negative acknowledgment	non-receipt of an item
Rpt	Transp_msg	update	SHG and task updates regarding
	Report	report on task	completed tasks' details
	SentTsk	update about sent task	dispatch of task
	RequestAck	request an ack for a sent item	item dispatch
ValTM	InvalMsg	reporting message	transparency message is invalid
ExTsk	AgreeTsk	willingness to execute a task	agreements for task execution
	ExecutingTsk	update about task execution	task execution and progress
	SuspendTsk	intimation & reason for suspension	task status & reason for suspension
	EndTsk	notifies when a task is completed	status of task
	TaskReturn	return task if infeasible	failure of task
ColExTsk	CInit	to initiate collaborations	collaboration initiation
	CAgree	collaboration agreement details	terms of collaborations
	CRunMonitor	updates of collaborated task	status of collaborated task
	CSuspend	collaboration suspension message	coltsk status of suspension
	CEnd	notifies collaboration completion	completion of coltsk

Table 4.2 Summary of Transparency Messages

4.4.2 Task Specification

Task specification is already discussed in the Collaborations chapter. Let us now discuss only specification pertaining to the concept of transparency.

As discussed earlier, a task is represented as follows in BNF:

$< task >::=< task\_id >< task\_desc >< body >$ **<other_info>**$[< task >]$.
$< task\_id >::=< string >< integer >$
$< task\_desc >::= < task\_type >< req\_spec >$
$< task\_type >::=< string >$
$< req\_spec >::=< string >$
$< body >::= < serv\_needed > [< skill\_level >][< duties >] < task\_proc >$
$< serv\_needed >::= < string >$
$< skill\_level >::=< string > | < integer >$
$< duties >::=< string >$
$< task\_proc >::=< string >$

<other_info> $::= [< task\_priority >] < distance >< notif\_recp > [< risks >]$**[<cond>]**
$< task\_priority >::= < string > | < integer >$
$< distance >:= < integer >$
$< notif\_recp >:= < string >$
$< risks >:= < string >$
<cond> := **<msg_cond> <other_cond>**
<msg_cond> := **<state>, <context>, <action>**
$< state >:= < string >$
$< context >:= < string >$
$< action >:= < string >$
$< other\_cond >:= < string >$

In the above task representation, there is a task attribute called 'msg_cond' (message condition). In addition to other specifications, it defines the states at which transparency messages have to be dispatched, the information they should provide, time schedules, etc. 'Msg_cond' is represented as a triplet with three components — $state, context,$ and $action$. $State$ refers to 'execution state reached by task', such as, $Agree$, $Suspend$, End, etc. $Context$ refers to a 'situation' that arises during task execution (such as, resources unavailable, new skills acquired, new agreement required, etc.). $Action$ refers to what an SHG has to do in response to the corresponding $context$. Usually $action$ would be about sending a transparency message or alert.

Therefore, for each task, transparency is well established with each process having multiple states and each $state$ having multiple $contexts$ and definite $actions$ defined for each $context$.

For example, consider a task of making 1000 articles. Then, the task specification's

<msg_cond> ($state$: $context$: $action$) might then be defined as:

$TaskRun$: 300 $articles\ completed$: $notify\ coordinator, task\_owner, funding\_agency$

$TaskRun$: $'equipment\ failure'$: $notify\ coordinator$

The idea behind the concept of '$state$: $context$: $action$' is that, when the system is implemented, it keeps monitoring the status of a task and resources. During task execution, when a defined $context$ is encountered in a defined $state$, then the system carries out an action such as generating a transparency message, thus ensuring transparency at each important event in task execution. Different message formats are defined in the next subsection. However, it will be the responsibility of SHG to 'send' the message with the relevant additional information. Transparency of an SHG is measured based on the sent transparency messages only. If a message is generated, but the SHG does not send it, then it negatively affects the transparency of that SHG.

4.4.3 Transparency Messages Formats

Transparency messages are the means to achieve transparency. Let us outline the formats of some of the important transparency messages. Transparency messages are generated at different states of processes during task execution to convey organizational, financial and executional information.

Notification recipients vary from task to task and are usually comprised of SHG coordinator, collaborating SHGs, third party agencies, funding agencies, government bodies, and other stakeholders. Different messages carry different type of information and have varied formats. The specific format for each type of message ensures that each message has necessary fields that ensure transparency. Following are some of the basic templates of some of the messages.

Ack: Ack message is a signal to acknowledge the receipt of items such as task, request, etc.

$< Ack > \ for \ < item\_id > \ from \ < item\_recv\_SHG > \ on \ < date\&time > \ to \ < notif\_recp > < /Ack >$

Nak: Nak is used to negatively acknowledge or reject a previously received item or to indicate some kind of error.

$< Nak > \ for \ < item\_id > \ from \ < item\_recv\_SHG > \ on \ < date\&time > \ to \ < notif\_recp > < /Nak >$

SentTskMsg: This message is to inform the SHG coordinator and other notification recipients that a task has been sent to an SHG.

$< SentTskMsg > < tsk\_Id > \ to \ < recvr\_SHG\_Id > \ by \ < sender\_SHG\_Id > \ on \ < date\&time > \ informing \ < notif\_recpnts > < /SentTskMsg >$

RequestAck: After sending a report or task, if an SHG does not receive any

acknowledgment correspondingly, then it sends RequestAck message to remind it to send acknowledgment.

$< RequestAck > \ for \ < tsk\_id/report\_id > \ sent\ on \ < date\&time > < /RequestAck >$

AgreeTskMsg: This message is to inform the task sender, SHG coordinator and other supervising bodies that the SHG agrees to execute a task or collaborate to execute a given task.

$< AgreeTskMsg > < agreeing\_SHG\_Id > \ agrees\ to\ execute \mid collaborate\ with \ < collab\_id > \ for \ < tsk\_id > \ from \ < date\&time > \ on\ the\ following\ terms \ < collab\_cond > \ informing \ < notif\_recpnts > < /AgreeTskMsg >$

StartTskMsg: This message is to inform the collaborator or task executor to start executing a given task.

$< StartTskMsg > < tsk\_id > \ on \ < date\&time > \ message\ issued\ by \ < issuer\_id > < notif\_recp > < /StartTskMsg >$

ExecutingTskMsg: In response to the StartTskMsg, if an SHG has started executing a task, it will inform the notification recipients using ExecutingTskMsg.

$< ExecutingTskMsg > < SHG\_id > \ started\ execution\ of \ < tsk\_id > \ on \ < date\&time > \ notifying \ < notif\_recpnts > < /ExecutingTskMsg >$

TskSuspendedMsg: If an SHG has suspended a task, it informs the notification recipients using TskSuspendedMsg.

$< TskSuspendedMsg > < SHG\_id > \ suspended \ < task\_id > \ on \ < date\ \&\ time > \ due\ to \ < context > \ notifying \ < notif\_recpnts > < /TskSuspendedMsg >$

CInitMsg: In case of coltsk, if collaboration is being initiated, then the SHG coordinator is notified using CInitMsg.

$< CInitMsg > < coltsk\_id > < publish\_to\_coord > < date\&time > \ notifying \ < Coord > < /CInitMsg >$

CAgreeMsg: When a collaborator is found, then collaboration can be established after the CAgreeMsg message is sent. In this agreement process, the coordinator, supervisors, etc. are notified of the agreement details.

$< CAgreeMsg > \ collaboration\ agreed \ for \ < coltsk\_id > \ with \ < collab\_id(s) > \ on\ terms < collab\_agrmnt\_details > on \ < date\&time > \ notifying \ < notif\_recpnts > < /CAgreeMsg >$

CRunMonitorMsg: When the coltsk is being executed, the task owner, SHG coordinator and other supervising bodies are notified of updates.

$< CRunMonitorMsg > < transp\_msg > < date\ \&\ time > \ notifying \ < notif\_recpnts > < CRunMonitorMsg >$

CSuspendMsg: If a collaboration with an SHG is suspended, it is reported to the coordinator, supervisor, and others connected to the coltsk.

$< CSuspendMsg > \ collaboration\ with \ < collab\_id > \ is\ suspended\ on \ < date\&time > \ for \ < coltsk\_id > \ due\ to \ < prblm\_details > \ notifying \ < notif\_recpnts > < CSuspendMsg >$

CEndMsg: After successful completion of a collaboration, the coordinator and other stakeholders are intimated.

$< CEndMsg > \ collaboration\ completed\ successfully\ by \ < collab\_id(s) > \ for \ < coltsk\_id > \ notifying \ < collab\_feedback > \ to \ < notif\_recpnts > < CEndMsg >$

TskEndedMsg: If a task is successfully completed then this message is sent to the notification recipients.

$< TskEndedMsg > < task\_id > \ is\ completed\ by \ < SHG\_id > \ on \ < date\&time > \ with\ feedback$

$< feedback\ for\ task > \ notifying \ < notif\_recpnts > < /TskEndedMsg >$

TskReturnMsg: When a task cannot be executed, it is returned and the reason for failure of the task execution is specified.
$< TskReturnMsg > returning < task\_id > on < date\&time > with < reason\_for\_failure(s) > to < notif\_recpnts >< /TskReturnMsg >$

4.4.4 Task Execution Cycle [TEC]

Let us discuss the Task Execution Cycle (Fig: 4.3) which is a refinement of ExTsk and ExColTsk processes described in the SHG Behavioral Model. Transparency regarding the state of a task and state transitions is achieved using Task Execution Cycle [TEC]. TEC defines all the states of a task when it goes through ExTsk and ExColTsk. During the state transitions and inside the states, values of various variables related to financial, organizational and executional aspects will usually get updated. Update messages (i.e. pre-transparency messages) are prepared using these variables and are sent to Rpt to then be sent on to the notification recipients.

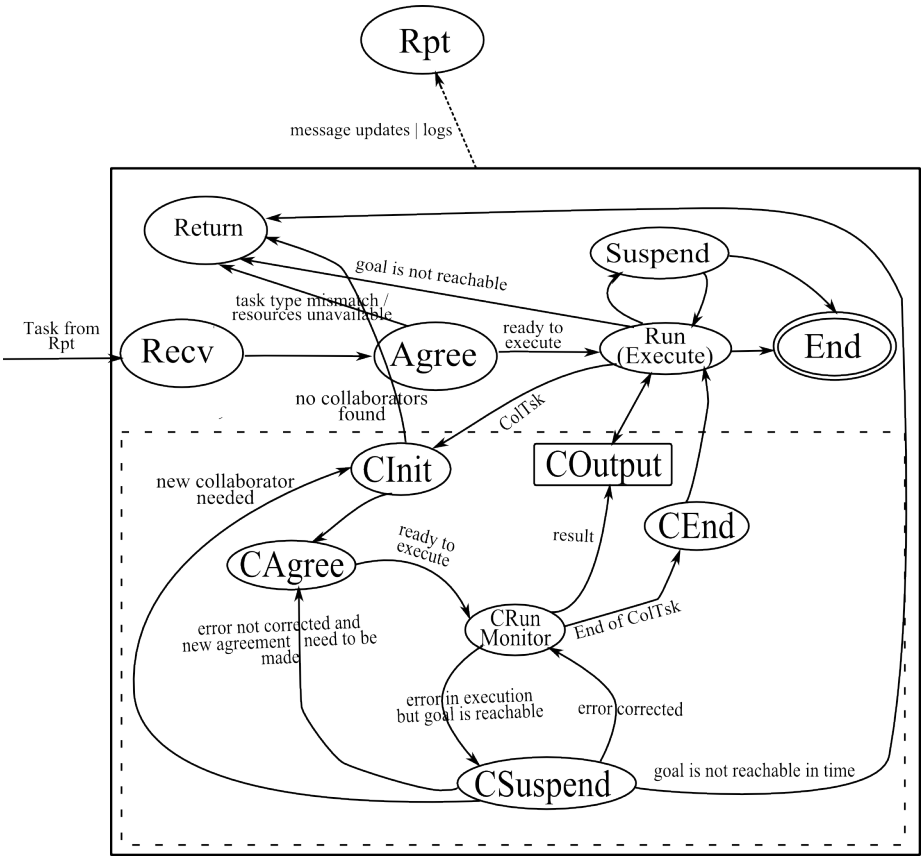


Figure 4.3 Task Execution Cycle

Following are the different states of a task. *Recv:* When a task is received from Rpt process, it enters into Recv state. *Agree:* When it is time for a decision to be taken, it moves to the Agree

state and remains there until it is examined for its feasibility and a decision is made whether it can be executed or not. If an agreement was made successfully, then it moves to the Execute state, otherwise it moves to the Return State. There may be several tasks waiting for execution in the task pipeline, and executing a task immediately after agreement may not be possible. Also, an SHG needs to wait for StartTsk message before executing a task. When task execution begins, then it moves to the Executing state. *Return:* If a task was found to be infeasible or collaborations fail or SHG cannot continue execution, then it moves to the return state. *End:* Finally, if a task execution is completed successfully, then it moves to the End state.

CInit: If it was decided that a task can be executed with the help of collaborators, then the collaboration process is initiated. At this stage, task moves to CInit state. The task is now referred to as coltsk (collaborated task) and the process monitoring is the ExColTsk process. An SHG initiates a collaboration process through PubSub model. Some of the subscribers will eventually agree for collaboration and will send AgreeTsk message to the publisher. Thus a publisher SHG obtains a list of probable collaborators in CInit state. But if no subscribers respond, then the task state changes to Return and that task is returned.

CAgree: When a publisher and subscriber agree with each other's conditions for a coltsk, they will become collaborators. During this process, task state is CAgree. *CRunMonitor:* When execution of a coltsk is started, it enters CRunMonitor state. During coltsk execution several values of variables keep changing and all of these variables are monitored and reported to Rpt. CRunMonitor has information pertaining to current status of a coltsk, status of collaboration, progress of coltsk, status of resources, estimated time for completion, etc. Some updates may also be written to COutput. Rpt can fetch updates from COutput and dispatch them to respective notification recipients at scheduled times.

CSuspend: During execution of a coltsk, problems may arise such as collaboration issues, unavailability of resources, weather conditions, business conditions, etc. When such things happen, execution may be suspended to be revived again later. During this scenario, coltsk moves to CSuspend state. And, when favorable conditions for resuming the execution arrive, coltsk is restarted and its status again moves to CRunMonitor. *CEnd:* When a collaboration ends or the part of the coltsk that can be executed using collaborations is completed, it moves to CEnd state. The next state is again Execute state so that any pending works may be completed by the SHG before moving to End state.

Repst: Repository (Repst) is the storehouse for all the log information and facilitates internal transparency. It hosts all the information of members, SHG, and other data provided by the SHG coordinator and stakeholders. Whenever a report is needed/requested, Rpt queries the

Repst and generates the necessary reports. Internal transparency is achieved when Repst stores all the necessary data.

4.4.5 Transparency Representation

In addition to qualitative transparency, we also quantify transparency. As stated earlier, transparency messages are the update messages of an SHG sent to stakeholders regarding a specific task. We measure transparency between two SHGs using the number of transparency messages sent by an SHG.

Let us represent transparency as $Transparency(A, B, tsk\_id, transp\_val)$. It is read as transparency assessed by SHG A for SHG B over task_id is equal to the transp_value. If we want total transparency between two SHGs A and B over all tasks, then it can be given as $Transparency(A, B, transp\_value)$. Depending on $transp\_value$, it is possible to understand whether A is fully transparent to B, or A is opaque to B. Also, transparency is not commutative. So, it means that if A is transparent to B, it does not automatically imply that B is also transparent to A. Sometimes A may be fully transparent to B, while B may be fully opaque to A. For this reason, transparency will be computed by every SHG for each of its peers.

4.4.6 Transparency Messages Properties

Structured messages ensure that necessary fields of transparency are present in each transparency message but it could not ensure that the fields contain transparency information. Marcia et al.[118] stated that the use of a ranking scale helps to evaluate the degree of transparency but does not say much about the quality of the information published. The present model overcomes this limitation through 'validity checking'. If a stakeholder receives an invalid message, i.e. a message without the required transparency information, it declares the message as invalid. Messages which were declared invalid do not count as transparency messages and thus they negatively affect the transparency of the SHG that sends them. This serves a motivation for all SHGs to send messages that are really transparent.

To see that the messages ensure transparency, in addition to structured message formats, they should possess some transparency messages properties. The essential properties of all transparency messages are timeliness, completeness, consistency, relevance and understandability. Fig: 4.4 gives a detail of the idea being discussed in this section.

Timeliness implies that a transparency messages should be sent to the recipients on or before the deadline. Suppose it was agreed that a transparency message will be sent on first day of every

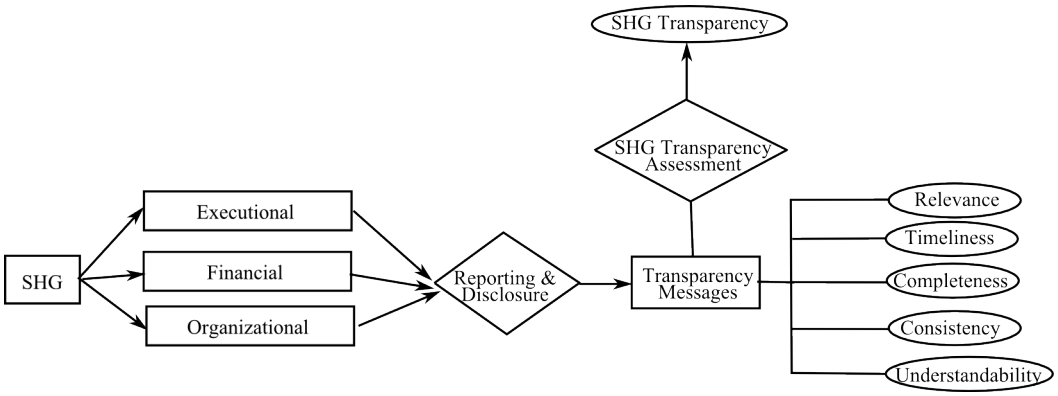


Figure 4.4 Transparency Messages

week, then the transparency message must be sent on the first day. Sending it on the second or third day makes a transparency message fail in timeliness. Completeness implies that a message should have all the data that is required by the message receiver. Completeness is as important as timeliness. A message sent 'in time' but 'incomplete information-wise' is not beneficial. So, to ensure completeness, we require the receiver to validate each message and report to the sender and coordinator if a message does not contain necessary information. Consistency implies that a message must be logically correct. For example, if a transparency message ($transp\_msg\_1$) indicates that 50 items were completed, then the next transparency message ($transp\_msg\_2$) should indicate any number more than 50 as completed. But if $transp\_msg\_1$ indicates that 50 items are completed and then at a later time $transp\_msg\_2$ arrives indicating that only 25 items were completed, it would then imply that one of the messages is false. Either $transp\_msg\_1$ is reported falsely or $transp\_msg\_2$ has an error, and the message fails in consistency.

Relevancy property of messages implies that a stakeholder should receive information that is most related to it. For each task there are several stakeholders and each of the stakeholder's interests are different. For example, a financier is interested in information about utilization of funds (financial transparency), SHG coordinator is interested in knowing the impact of a task on SHG and members (organizational transparency), peer and collaborator SHGs are interested in knowing information about task execution and profits and losses (executional and financial transparency). Hence relevancy property ensures that each stakeholder receives information that is most related to them. Finally we require that a transparency message is understandable to the recipients. That is, messages have to be in comprehensible format and the content unambiguous.

When a receiver receives a transparency message satisfying all these five properties, then it considers the transparency message as valid. Invalid messages are reported to the coordinator and sender, and they do not count during transparency quantification. Transparency is evaluated

based only on valid transparency messages. Also, all the information that is sent to stakeholders will also be made available to all members of the SHG [119] to preserve internal transparency of the system.

4.4.7 Transparency Messages Reporting Process

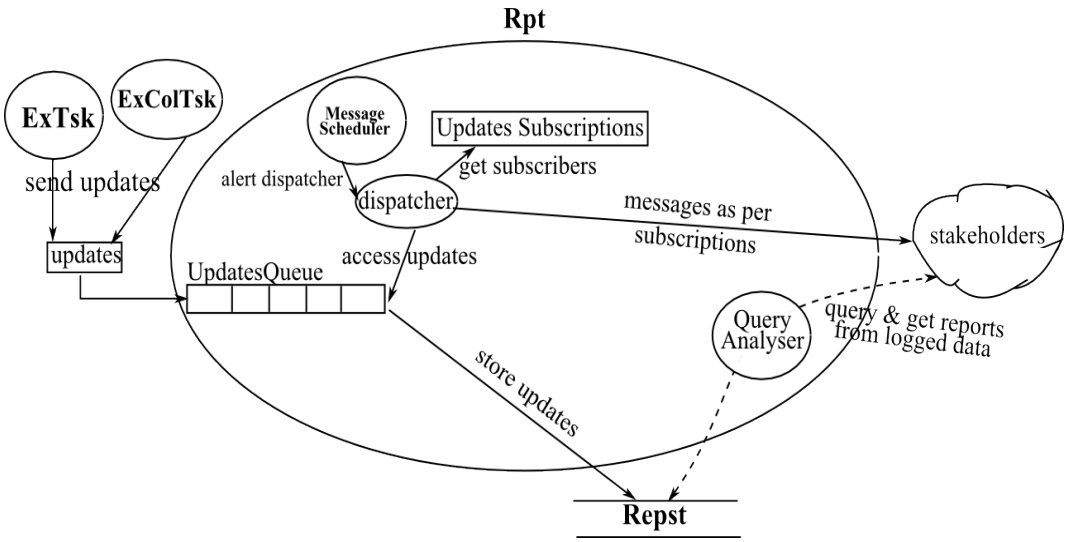


Figure 4.5 Reporting Transparency Messages

Let us now discuss the process of reporting the transparency messages. In order to receive notifications, SHGs subscribe to task executing SHGs. As discussed earlier, all SHGs and stakeholders may not be interested in the same type of information. Some may be interested in organizational information, while some may be interested in financial details, and some in task execution details. In order to avoid receiving all the updates, stakeholders subscribe as per their preferences. A list of notification recipients is maintained, listing stakeholders and their subscriptions for updates. Whenever an update is available, the SHG checks the list of notification recipients, picks the relevant subscribers, and dispatches them respective updates; this is 'push' messaging. As we see, push messaging provides enough transparency regarding updates of an SHG. Transparency can still be improved by adding a 'pull' mechanism to the process. An SHG can make its updates available at all times by storing them and providing options for stakeholders to request and obtain any information of their choice at any point of time. Thus, by using push and pull mechanisms, the system increases information availability, reduces information asymmetry, and paves the way for high transparency.

Algorithm 1: Rpt_Process() \∗ Algorithm for messaging process ∗\

input : $update$;
output : $transp\_msg$;
var : msg;
$list[]$: $subscribers$;

$If(ExTsk(avail\_update)||ColExTsk(avail\_update)||MsgSched(alert))$
{
$AddUpdate(UpdQueue)$;
$Invoke(Dispatcher())$;
}

$Dispatcher()\{$
$if(UpdQueue.isNotEmpty())$
{
$msg = UpdQueue.front$;
$DelFront(UpdQueue)$;
$AddToRepst(msg)$;

$subs = ExtractSub(subscrip, tsk)$
$Send(msg, subs)$;
}
}

$if(Request(tsk, upd\_type))$
{
$msg = Query(Repst, tsk, upd\_type)$;
$Send(msg, req)$;
}

4.4.8 Executional Transparency Computation Example

Let us consider a task execution scenario and compute its executional transparency. Suppose, a task (tsk_101) is given to an SHG (SHG_007). The task is about farming five acres of land and producing rice. The conditions are that the SHG should only use organic fertilizers prepared, and also the costs should be kept as minimal as possible. The SHG coordinator (SHGcord) will pay for seeds, expenses and labor wages. When SHG_007 is notified of this task, it first checks whether it has enough members who have agricultural skills and are available for executing the task. If enough members are available, then it will notify its acceptance to SHGcoord. If some of the required skills are not available with the SHG, then it finds collaborators that can help with the missing skills. SHG_007 may also consider to collaborate with other SHGs for obtaining organic fertilizers. To find collaborators, SHG_007 publishes the task of preparing

organic fertilizers along with some conditions such as price limit, speed of preparation, amount of preparation, etc. Then, several SHGs that have experience in making organic fertilizers will respond with their quotations and specifications. SHG_007 selects one (or some) of the responding SHGs as collaborator(s). Let us say that SHG_007 selects SHG_075, SHG_037 and SHG _207 as collaborators for making organic fertilizers. Let us say that the supplier of seeds is SeedsCo company and funds are given by SHG Development Organization (SHGDO). To the above scenario let us add one more detail, that is, SHG_099 has experience in doing the similar task and is an expert. So SHGcoord appointed SHG_099 as the supervisor over SHG_007 for task_101.

Updates Subscriptions: Updates subscriptions are the registrations by different stakeholders for receiving notifications from a task executing SHG. These updates subscriptions are different from the skill subscriptions which SHGs do at the SHG coordinator. Those subscriptions are skill/resources availability registrations and they are stored in SubDB database. But these updates subscriptions can be made by any stakeholder and are stored at the SHG coordinator and task executing SHG. Different stakeholders have interests in different types of notifications. Suppose SHGcoord is interested in the organizational information of the executing task. So SHGcoord subscribes to organizational update messages of tsk_101 of SHG_007. SeedsCo supplied seeds, so it subscribes to executional updates. SHG_099 wants to offer advice based on its experience and expertise, so it subscribes to executional updates. SHGDO financed the project, so it subscribes to financial information. SHG_075, SHG_037, SHG_207 are collaborators for organic fertilizers; they subscribe to executional updates to know about the status of crop, the effectiveness of fertilizers, feedback, and also to provide necessary fertilizers in time. SHG_050, SHG_051 and SHG_059 are peers and they also have a similar task to execute, so they subscribe to executional updates to keep track of each other's progress and problems.

Computing Transparency: SHGcoord specifies the task states and the number of messages to be sent at each state. SHG_007 may send more messages than prescribed and to more stakeholders than specified in the list, but they are not considered during transparency computation. That is, if SHG_007 has to send 100 messages at given time periods, transparency is computed based on these 100 messages only. Message Scheduler of SHG_007 copies the information about message time intervals and alerts the SHG to send required transparency messages. If a message is reported invalid, then the SHG coordinator checks the message if it satisfied all the five transparency properties, i.e. timeliness, completeness, consistency, relevance and understandability. If it was found that the message does not satisfy the properties,

Update Period	Message Info	Notification Recipients
germination -month 1	crop condition, feedback on fertilizers	SeedsCo, SHG_075, SHG_037, SHG_207, SHG_050, SHG_051, SHG_059
MS leaf production-month 1	crop condition, feedback on fertilizers	SeedsCo, SHG_075, SHG_037, SHG_207, SHG_050, SHG_051, SHG_059
tiller production-month 1	crop condition, feedback on fertilizers	SeedsCo, SHG_075, SHG_037, SHG_207, SHG_050, SHG_051, SHG_059
MS node production (stem elongation)-month 1	crop condition, feedback on fertilizers	SeedsCo, SHG_075, SHG_037, SHG_207, SHG_050, SHG_051, SHG_059
booting -month 2	crop condition, feedback on fertilizers	SeedsCo, SHG_075, SHG_037, SHG_207, SHG_050, SHG_051, SHG_059
heading -month 2	crop condition, feedback on fertilizers	SeedsCo, SHG_075, SHG_037, SHG_207, SHG_050, SHG_051, SHG_059
anthesis -month 2	crop condition, feedback on fertilizers	SeedsCo, SHG_075, SHG_037, SHG_207, SHG_050, SHG_051, SHG_059
grain milk stage -month 3	crop condition, feedback on fertilizers	SeedsCo, SHG_075, SHG_037, SHG_207, SHG_050, SHG_051, SHG_059
germination -month 3	crop condition, feedback on fertilizers	SeedsCo, SHG_075, SHG_037, SHG_207, SHGs SHG_050, SHG_051, SHG_059
grain dough stage -month 3	crop condition, feedback on fertilizers	SeedsCo, SHG_075, SHG_037, SHG_207, SHGs SHG_050, SHG_051, SHG_059
ripening -month 3	crop condition, feedback on fertilizers	SeedsCo, SHG_075, SHG_037, SHG_207, SHG_050, SHG_051, SHG_059

Table 4.4 Transparency Messages Schedule

then the message is regarded as invalid.

For example, a time interval for the messages may be specified as given in the table (4.4). Here the task's lifetime is three months and has nine phases which are as follows: germination, main shoot (MS) leaf production, tiller production, MS node production (stem elongation), booting, heading, anthesis, grain milk stage, grain dough stage, and ripening. SHGcoord verifies if SHG_007 has sent all the prescribed messages, and then computes the transparency value. Suppose, total messages to be sent to SHG_037 are 100, but SHG_007 has sent only 90 valid transparency messages. Then, transparency of SHG_007 to SHG_037 for task_101 is:

$Transparency(SHG\_007, SHG\_037, tsk\_101, 0.9)$

Also, SHG_007 sends transparency messages to other stakeholders such as SeedsCo, SHGDO, SHG_075, , SHG_207, SHG_050, SHG_051, etc. Transparency values for the stakeholders are computed same as above.

Now, let us consider that SHG_007 has executed several tasks such as task_101, task_232, task_214, task_114, and task_219 and their corresponding transparency values computed as above be 0.7, 0.6, 0.8, 0.5, and 0.9. Then, the total transparency value of SHG_007 to SHG_037 is : $Transparency(SHG\_007, SHG\_037, 0.7)$

Thus we compute the transparency of an SHG to any SHG/stakeholder.

4.5 Conclusion

This chapter set out to define transparency in SHGs and proposed a quantitative approach for transparency measurement. It discussed SHG Behavioral Model [SBM] and Task Execution Cycle [TEC], as they are the keys to realize transparency of an SHG. Transparency has three aspects: executional, financial and organizational. This chapter discussed executional transparency, and it is argued that same methodology can be applied to assess financial and organizational transparencies. Transparency representation was discussed. Transparency messages carry transparency information and they are the means for achieving transparency. So, transparency message formats and transparency properties that are required for ensuring transparency were also discussed. The study also detailed a simple example for transparency computation. It becomes very long-winded if each and every task state is considered and all possible message formats are discussed. This study considered basic states of a task and a few transparency message formats to familiarize the concept.

Chapter 5

Dependability

Self Help Groups [SHGs] publish their particulars on the Internet for collaborators and other stakeholders to make use of their services. It's usual that the stakeholders need to judge the dependability of a service provider before signing up an association. The process of building a dependent association with an Internet community suffers from usual drawbacks such as lack of proximity, anonymity, misrepresentation, proxy/masquerading, etc. Research of online cyber community and social networking, addresses some of these problems. In our case, SHGs on the Internet are not exactly similar to social networking online communities. An SHG is a formal entity with a defined behaviour and structure. In spite of its defined formality, endowed with its registration with government agencies, stakeholders are usually in search of dependable SHGs for availing services.

This chapter introduces the concept of dependability of SHGs and quantifies it. The subsequent sections define dependability and identify its attributes and properties with respect to the SHGs. They provide methods to quantify each attribute and finally, propose a metric for dependability.

5.1 Dependability of SHGs

SHGs are geographically distributed business groups with limited resources. They need to collaborate with each other for mutual benefits such as resource sharing, business growth, skills improvement, achieving larger goals, improving efficiency, etc. In such a distributed business environment where one SHG needs to utilize the services and resources provided by other SHGs, dependability is an important factor.

Dependability is of concern to SHGs or coordinators when there is a task to be assigned.

An SHG's dependability is time and task-dependent. It means that dependability of an SHG is not constant at all times and for all tasks. Dependability may increase or decrease over time. Also, dependability is subject to the type of the task. Some tasks may require high skill levels and some may require normal skill levels. If a task's skills requirements match correctly to an SHG's available skills, then there is a probability of that SHG performing well, and then its dependability for that task will be high. Otherwise if they don't match, then that SHG may not perform well and its dependability value may be much less. Considering these aspects of dependability, we propose a framework for quantification of dependability of an SHG from its features i.e. *trust, competency,* and *integrity*. While trust will account for the past behaviour and performance, competency will account for capability to the current task execution, and likewise, integrity will account for overall adherence to principles of the system. Therefore, if an SHG has high trust, competency and integrity, then it is regarded as a dependable collaborator. Let us study these attributes in detail and see how they are consistent with each other.

5.1.1 Trust

Trust is primarily a cognitive issue and is perceived to vary from person to person and context. Research on trust has become necessary for activities on the Internet. In particular, the topic has become interesting for study because the trustee and trusted are not aware of each other and sometimes are located far apart. In such environments, computing trust has become a challenging problem. Considering these different problems, there have been several trust computing models. Before proposing a suitable trust model for SHGs, let us touch upon the features that bear weight when making a model for computing the trust between SHGs.

Trust, being essentially a matter of cognition and abstraction, still having practical uses that need to be made tangible. The need is more in Internet based associations where there is an inherent difficulty among users because of anonymity and in-between remoteness. To be specific to the problem in hand, the proposed model for computing the trust of an SHG is based on three aspects — direct, indirect and recommender trusts.

The trustworthiness of an SHG is usually sensed from its activities and performance. An SHG that has been consistently working with perfection as per the specification is usually deemed the most trustworthy. Of course, another source for trust information about an SHG could be other users who had availed some service(s) earlier and also have something to say about the service providing SHG. And the third means to gather trust information on an SHG is from recommendations. A recommender could be thought of as an agency or expert committee that monitors SHGs' performance. They judge an SHG by observing its skills, professionalism,

or scrutiny by visit. Before computing trust, let us study some of the properties of trust with respect to SHGs.

Properties of Trust

Some important properties of trust that are to be true in case of SHGs [120] [121].

- Context dependence: A trust relation concerns a precise action or a particular entity and cannot be generalized to other actions or entities. In other words, trust is specific to a service being provided by a trustee. For example, SHG A may trust SHG B on a service such as *'providing raw material'*, but not on *'processing it'*. Thus, trust is specific to services that SHGs offer in the context of a particular business.

- Measurability: Trust is also a measurable dimension of one's behavior. There can be several degrees of trust. For example, A may trust B more than C for a specific service. This property emphasizes the need for a suitable metric to represent trust. The measurement can be quantitative (e.g., a probability assignment) or relative (e.g., by means of a partial order).

- Dynamic: Trust is dynamic since it may evolve over a period of time. The fact that A trusted B in the past does not guarantee that A will continue to trust B in the future. B's behavior, performance and other relevant information may lead A to re-evaluate the trust on B from time to time. During a collaboration, the more A is satisfied with B's performance for service X, the more A trusts B. On the other hand, A's trust in B may decrease if B proves to be less trustworthy than anticipated by A.

- Sensitivity: Trust is very sensitive to malicious behavior. Trust declines very quickly with the detection of malicious behavior in an SHG. Once lost, an SHG needs many good interactions to regain a good level of trust.

- Asymmetry: A trust relation is asymmetric. That is, if A trusts B, it does not imply that B also trusts A equally. A may have high trust for B, but B may have much less trust or even distrust for A.

- Transitivity: Though controversial, in SHGs, trust relation can be transitive. That is — if A trusts B, and B trusts C, then A can also trust C — to a certain level. But this is subject to constraints, and this property can hold true to a larger extent in SHGs. This property is essential for computation of indirect trust.

Context dependence property is satisfied by calculating trust with respect to a service. Measurability property is satisfied by attributing a quantitative value to the trust. Dynamic property is satisfied by calculating trust at random intervals. Sensitivity property is satisfied by defining penalties for malicious behaviors. Asymmetry property is satisfied by computing trust with respect to each SHG. Transitivity property is satisfied by using recommendations from other SHGs and third parties.

Trust is to be calculated in three parts to satisfy all the above properties. That is, when SHG B calculates trust of SHG A, the first part is computed using the direct experiences with SHG A, the second part is obtained from associated SHGs which had direct experiences with SHG A, the third part is from the recommenders (expert committee). So, in essence, trust is of three forms — direct trust, indirect trust, and recommender trust.

Direct Trust

Direct trust is based on self experiences and interactions with a target SHG. It is computed for behavioral features such as quality, speed of completion, cost effectiveness, and durability. Let us suppose that SHG_i computes the direct trust value of SHG_j. If SHG_j has interacted with SHG_i before, then SHG_i will have an opinion about the behavior of SHG_j. So, quantifying these behavioral features will help SHG_i compute the direct trust of SHG_j. Let us represent direct trust value of SHG_i on SHG_j as $DT_{ij}^{a}(t)$, where 'a' represents the service and t represents the time at which trust is being calculated, since trust is time dependent and dynamic (a property of trust). As trust is also context dependent, trust is calculated for each service of SHG_j.

Trust is dynamic and therefore the trust value is to be updated each time an SHG encounters a situation that affects the trust value. When SHG_j interacts with SHG_i at time t, then SHG_i updates its trust value for SHG_j as follows:

$$DT_{ij}^{a}(t) = \begin{cases} DT_{ij}^{a}(t' + \delta t) + \Delta_{DT}(t) & \text{if } SHG_j \text{ is not new } SHG_i \text{ and } t' < t, \\ \Delta_{DT}(t) & \text{if } j \text{ is new to } i \end{cases}$$

Here, 't' represents time, and $t = t' + \delta t$. Here, δt represents the *'time elapsed'* since last update of trust value, and t' represents the time of last update. $\Delta_{DT}(t)$ represents the direct trust value that resulted due to direct experiences with SHG_j at time t. On occasion, $\Delta_{DT}(t)$ could also be negative. Thus the trust measurement preserves the property of dynamic nature of trust [122].

Let us compute values for the components that comprise the direct trust.

$DT_{ij}^{a.quality}$: This is the rating of SHG i to SHG j about the quality of service provided by j for the service/product/task a. SHG i estimates $DT_{ij}^{a.quality}$ by verifying/observing/experiencing the quality provided by j at time t. Usually SHGs i and j make an agreement regarding the standard of quality that is to be maintained by SHG j. The range of values for quality component is {0,1}. If SHG j maintains quality standards and provides the service/product at an agreed quality, then $DT_{ij}^{a.quality} = 1$. Otherwise, if there was a compromise in the quality, then $DT_{ij}^{a.quality}$ value lies between 0 and 1. If the supplied quality is too inferior to the agreed quality, then $DT_{ij}^{a.quality} = 0$.

$DT_{ij}^{a.complSpeed}$: This is the value given by SHG i to SHG j about the speed of task execution by SHG j. This value is based on i's direct observation of j's performance. Usually SHGs i and j have an agreement regarding the speed of completion and deadlines to be followed. The range of values for complSpeed component is { 0,1 }. If SHG j adheres to the deadlines perfectly, then $DT_{ij}^{a.complSpeed} = 1$, otherwise, if the speed of completion is very slow, then $DT_{ij}^{a.complSpeed} = 0$. In other cases, it will be any intermediary value between 0 and 1.

$DT_{ij}^{a.costEffect}$: This is the value given by SHG i to SHG j about cost effectiveness of j with respect to service/task a at time t. This value is based on i's research on cost of 'a' quoted by j compared to other providers/executors. The range of values for costEffect component is { 0,1 }. If j has quoted less than the others, then the cost effectiveness of j is good, which is represented as $DT_{ij}^{a.costEffect}$ = 1. If SHG j is found to be not cost effective, then $DT_{ij}^{a.costEffect}$ = 0.

$DT_{ij}^{a.durability}$: This is the value given by SHG i to SHG j about durability of service/ product 'a' of j. This value is based on i's direct observation of j's product/service. Durability is the measure of life of a service/product. Durability does not always depend on quality. Sometimes one may use high quality components in making an item, but the durability of such an item may be much less. Mechanical systems characteristically fail from wear out mechanisms, while electronic systems more closely exhibit a constant failure rate over their useful life. Durability testing is the duration of time a product, part, material, or system can meet its performance requirements e.g. lifetime span. Depending on the type of material or product, SHG i simulates environmental factors or conditions to determine the operating limits or life span of the products, parts, and materials. Besides others, some of the tests are aging, operational limits, component degradation, failure points, material testing, tensile testing, burst testing, environmental testing, load testing, etc. The range of values for durability component is { 0,1 }. If the durability of an item a is high, then $DT_{ij}^{a.durability} = 1$, else, $DT_{ij}^{a.durability} = 0$.

Now that the values of all the attributes were quantified, SHG_i can now compute direct trust

value for SHG_j. Since the range of values for each attribute is {0, 1}, the range of values for direct trust will be {0, 4}.

$$DT_{ij}^{a}(t) = DT_{ij}^{a.quality}(t) + DT_{ij}^{a.complSpeed}(t) + DT_{ij}^{a.costEffect}(t) + DT_{ij}^{a.durability}(t)$$

Indirect Trust

The second part of trust is indirect trust. This value is computed by an SHG for a target SHG with the help of the peers. An SHG can ask the peers to offer an indirect trust value for the target SHG. It is good if the peers are common friends to both the evaluating SHG and the target SHG. If such common peers are not available, then any SHGs that have direct experience(s) with the target SHG may be considered. Indirect trust value can be computed using feedback, associations, or social intimacy degree. Let us detail the strategies to calculate indirect trust from the behavioral features of SHGs.

Feedback

This is one of the methods for obtaining the indirect trust value of a target SHG. The peers of a target will be asked to give feedback for the target SHG. Indirect trust obtained from feedback can be represented as $IT_{ij}^{a.Feedback}(t)$ where IT represents Indirect Trust, 'a' represents the service/product for which Indirect Trust is being calculated, Feedback is the method employed, ij indicates that SHG i is computing Indirect Trust for SHG j. Feedback is usually taken for the following five aspects — cost, quality, transparency, timeliness and response time. As per [123], feedback for an SHG can be calculated as follows. Each aspect is measured on a 3-point scale calibrated as high, average, and low, and numerically each aspect may assume a value from set (1,2,3). For example, when cost is low, this aspect scores a value of 3; but for quality, low indicates a value of 1.

Let the components of feedback i.e. cost, quality, transparency, timeliness and response time, be represented by x_i. Then, feedback on SHG j to be obtained by SHG i for the service a can be computed from $\sum_{i=1}^{n} a.Feedback.x_i$. It is the sum of the scores of the components provided by the peers about the target SHG. Let us say that there are n components and let S_{mx} represent the 'maximum value of the scale' used to calibrate each satisfaction level. Currently, we considered only five aspects, each calibrated by a scale with S_{mx} value 3. Now we can compute the indirect trust value resulting from feedback using the below equation.

$$IT_{ij}^{a.Feedback} = \sum_{i=1}^{n} \frac{a.Feedback.x_i}{n * S_{mx}}$$

Associations

Feedback is not an *'always preferred'* approach. In society, trust on a person can be ascertained to a certain degree by judging the quality of his/her associations. Usually, a person is trusted if his/her associations are trusted [123]. Similarly, an SHG can infer an indirect trust value for a target SHG by looking at its associations for service/resource. So, it is possible for an SHG to choose alternate approaches to feedback due to circumstances, such as, peers not responding with feedback, the SHG does not have time to wait for the peers to respond, the SHG could not rely on the peers' responses, etc. One advantage of associations over feedback is the aspect of time. Inferring from associations has less delay compared to feedback. To compute feedback, an SHG needs to wait until the peers reply. But in the case of associations, there is no waiting time; an SHG itself can infer an indirect trust value by checking how good the associates of the target SHG are.

Let n be the number of associated SHGs, and CR_r^a represent the CollabRating value of an SHG r associated with SHG j for a service 'a'. Then, the association value that can be obtained by SHG i for SHG j can be represented as:

$$IT_{ij}^{a.Assocn} = \frac{\sum_{r=1}^{n}(CR_r^a)}{n}$$

5.1.2 Recommender Trust

The third part required for computing the total trust value for an SHG is *Recommender Trust*. In practice, recommendation has been a successful approach in building trust [123]. The SHG coordinator has a set of expert recommenders for each type of service in the SHG system. These recommenders are experts in specific service/product domains they represent. When an SHG seeks recommendation regarding a target SHG, the recommenders check their data or use their custom methods to return a recommendation value for a target SHG. If $RT_{ij}^a(t) \in [0, 1]$ indicates the recommender trust assigned to SHG j and revealed to SHG i for the service/product a, then $RT_{ij}^a(t) = 1$ implies that the recommenders fully recommend SHG j to SHG i regarding service a, and $RT_{ij}^a(t) = 0$ implies that the recommenders do not recommend SHG j to SHG i regarding service a. And, intermediate values show the degree of recommendation.

Trust Scale

On detailing the methods to compute the said three forms of trust, we can compute a trust value (T_{ij}^a) for an SHG as follows:

$$T_{ij}^{a} = DT_{ij}^{a} + IT_{ij}^{a} + RT_{ij}^{a}$$

Using the above computation, it is possible to arrive at a final trust value for an SHG with respect to a particular service/product a. The range of values for trust scale is {0,6}. This is because, the maximum value for IT_{ij}^{a} and RT_{ij}^{a} is 1, and for DT_{ij}^{a} is 4, and the minimum value for all three is 0. Now, T_{ij}^{a} = 0 implies that SHG j cannot be trusted for the service a, and T_{ij}^{a} = 6 means that the SHG j can be fully trusted for the service a. If it is any intermediate value, then level of trust for the specified service/product can be assumed accordingly.

5.1.3 Competence

As per our definition, trust is a measure of how good an SHG has been as a collaborator. Trust value mostly focuses on things of the past, and not of the present. Sometimes, such situations could be true when an SHG had a high level of trust for a service/task but it does not have enough competency to execute a task at the moment. So, for dependability to be applicable to the current task of an SHG, its competency is to be computed. Competency is defined as an underlying characteristic which enables one to deliver required performance in a given job, role, or situation. High competency implies being able to execute and complete given task(s) efficiently and in specified time period.

Competency is of two types: social competency and technical competency. An SHG needs to be competent in both forms. Training, skill, experience, awareness, resources & equipment, etc. are classified under technical competency. Social relationships/collaborations, social responsibility, social impact, etc. are classified under social competency. Technical competency gives the technical capability of an SHG to perform a task whilst social competency gives the social behavior and attitude.

Technical Competency

The following components constitute the technical competency. They directly affect the way an SHG executes a task.

Skills competency: An SHG should have technical skills necessary for executing a task.

Let T_s denote a set of skills required by current task.

$T_s = \{s_a |\ s_a$ is a skill required by current task$\}$

Let J_s denote a set of skills possessed by SHG J.

$J_s = \{s_k |\ s_k$ is a skill possessed by SHG J$\}$

If skills competency of J is denoted by $Cskills_J$, then for the current task T,

$$J_s \not\subset T_s \Longrightarrow Cskills_J = 0$$
$$T_s \subseteq J_s \Longrightarrow Cskills_J = 1$$

If only a few of the skills in J_s match with skills in T_s, then $Cskills_J$ value will be equal to the percentage of matching and lies between 0 and 1.

Experience: It refers to an SHG possessing hands on experience in executing an identical task. The history of an SHG should indicate successful completion of a task similar to the current task. Failed experiences indicate that an SHG is not effective in handling tasks such as the current task. Following the above notation of skill competency, if experience competency of J is denoted by $Cexp_J$, then for the current task T,

$$J_{exp} \not\subset T_{exp} \Longrightarrow Cexp_J = 0$$
$$T_{exp} \subseteq J_{exp} \Longrightarrow Cexp_J = 1$$

If SHG J has less experience then $Cexp_J$ value will be equal to the percentage of matching and lies between 0 and 1.

Training: It refers to an SHG's training background for a task. Sometimes an SHG may not have experience executing a task similar to the current task but may have received training for it. Usually, there will be a considerable difference in the performance between a trained SHG and an untrained SHG as a trained SHG could perform better because it would've obtained more competencies compared to other SHGs which do not have any formal training. Following the notation of skill competency, if training competency of J is denoted by $Ctrain_J$, then for the current task T,

$$J_{train} \not\subset T_{train} \Longrightarrow Ctrain_J = 0$$
$$T_{train} \subseteq J_{train} \Longrightarrow Ctrain_J = 1$$

If SHG J has less training than required, then $Ctrain_J$ value will be equal to the percentage of matching and lies between 0 and 1.

Awareness : An SHG's reply to a checklist of problems and risks, makes a basis for computation of its risk awareness. It refers to an SHG being aware of the problems and risks involved with a task execution. An SHG that is aware of the problems and still willing to take a task is better than an SHG that is ignorant of the risks associated with the current task. This is because, after a task was assigned and execution has begun, a collaborator, which is unaware of the risks, might withdraw in the middle of execution when it faces problems. But in case of an

SHG that is aware of the risks and problems, the scenario could be different. Such an SHG may not be discouraged by the problems it faces and does not attempt to withdraw, but executes and completes the task as it knows the problems and anticipates them with preparedness. Following the previous notations, if awareness competency of J is denoted by $Caware_J$, then for the current task T,

$$J_{aware} \not\subset T_{aware} \Longrightarrow Caware_J = 0$$
$$T_{aware} \subseteq J_{aware} \Longrightarrow Caware_J = 1$$

If SHG J has less awareness of risks than required, then $Caware_J$ value will be equal to the percentage of awareness possessed and lies between 0 and 1.

Resources & Equipment: An SHG possessing its own resources and equipment necessary for a service or current task execution, usually performs better than other SHGs which have to borrow equipment/resources from others. Following the notation of skill competency, if resources competency of J is denoted by $Cresequip_J$, then for the current task T,

$$J_{ResEquip} \not\subset T_{ResEquip} \Longrightarrow CResEquip_J = 0$$
$$T_{ResEquip} \subseteq J_{ResEquip} \Longrightarrow CResEquip_J = 1$$

If SHG J has less resources or equipment than required, then $CResEquip_J$ value will be equal to the percentage of resources/equipment possessed and lies between 0 and 1.

Social Competency

In contrast to technical competency, social competency measures an SHG's social behavior and attitude. An SHG might be technically very competent but if it is not sociable in its dealings with other SHGs, then certainly, it could not become a good collaborator. Hence, in SHGs social competency is equally important as technical competency. We believe that an SHG that is socially responsible will also be sociable with its fellow SHGs and collaborators. Since the objective of the system is to build better collaborations and improve social relations among SHGs, a responsible SHG will endeavor to keep up the objectives. So, in computing the social competency of an SHG, we assess the endeavors of SHGs in keeping up the system objectives. We consider the social responsibility of an SHG as a measure of its social competency.

Social Responsibility: Social responsibility is an ethical notion that an entity, be it an organization or individual, has an obligation to act to benefit society at large. To check whether

an SHG is socially responsible or not, we check its associations with social work organizations and social welfare programs.

Let $T_{SocResp}$ denote a set of social work activities required by the current task.

$T_{SocResp} = \{SocResp_a |\ SocResp_a$ is a social work activity required by task$\}$

Let $J_{SocResp}$ denote a set of social work activities in which SHG J is involved.

$J_{SocResp} = \{SocResp_k |\ SocResp_k$ is a social work activity in which SHG J is involved. $\}$

If social competency of J is denoted by $CSocResp_J$, then for the current task T,

$$J_{SocResp} \not\subset T_{SocResp} \Longrightarrow CSocResp_J = 0$$
$$T_{SocResp} \subseteq J_{SocResp} \Longrightarrow CSocResp_J = 1$$

If SHG J has less social work activities than required, then $CSocResp_J$ value will be equal to the percentage of the social work activities possessed and will lie between 0 and 1.

The competence of $SHG\ J$ for executing the current task may be obtained by adding technical and social competencies together.

Competence of SHG_J, $C_J = Cskills_J + Ctrain_J + Cexp_J + Caware_J + CResEquip_J + CSocResp_J$

Since the range of each component of Competence is $\{0,1\}$, the range of Competence is $\{0,6\}$.

5.1.4 Integrity

Barbara Killinger defines integrity as, "a personal choice, an uncompromising and predictably consistent commitment to honour moral, ethical, spiritual and artistic values and principles" [124]. Integrity is a concept of consistency of actions, values, methods, measures, principles, expectations, and outcomes. In discussions on behavior and morality, an individual is said to possess the virtue of integrity if the individual's actions are based upon an internally consistent framework of principles. These principles should uniformly adhere to sound logical axioms or postulates.

One can describe a person as having ethical integrity to the extent that the individual's actions, beliefs, methods, measures and principles all derive from a single core group of values. An individual must therefore be flexible and willing to adjust these values in order to maintain consistency when these values are challenged.

In SHGs, the perception of integrity complements the perception of performance. It is possible to believe that an SHG is competent and has the potential to perform well but to not

believe that it has integrity [125]. Integrity is an agreement of the actions and stated values to the governing authority and it creates the foundation for trust [125] [126]. Since integrity is being consistent with the principles set by higher authorities, in SHGs, it can be interpreted as conforming themselves to all the principles set by the SHG coordinator. Therefore, we measure the integrity of an SHG by its degree of conformance in action to the directives of SHG coordinator. Let C_p denote the set of directives laid down by SHG Coordinator and let J_p denote a set of directives to which SHG J adheres to. If Integrity is denoted by I, then
$J_p \not\subset C_p \implies I = 0$, and $C_p \subseteq J_p \implies I \leq 4$

Let us look at practical example for Integrity. Let I_j represent integrity of SHG j and suppose that the coordinator specifies the following directives. First directive: there should not be more than 20 members in an SHG. Second directive: there must be a leader for each SHG. Third directive: all members' personal and skill information must be provided in corresponding SHG's profile. In similar lines, there may be several directives specified by the SHG coordinator.

These directives can be categorized into four types — organizational, financial, executional, and political. Let I_j^{org} represent SHG j's integrity resulting from commitment to the organizational directives laid down by the coordinator. Then, $I_j^{org} = 1$ implies that SHG j fully complies with all the organizational directives, and $I_j^{org} = 0$ implies that j is non-compliant to the directives. Any other level of commitment to the directives will have I_j^{org} having an intermediate value between 0 and 1. Let I_j^{fin} represent SHG j's conformation to the financial directives. The financial directives might be given as, for example, every SHG must disclose financial information of members, wages of every member; every SHG must provide all profit and loss information, etc. $I_j^{fin} = 1$ implies that SHG j fully conforms to the directives of financial requirement and discloses all financial information as required. $I_j^{fin} = 0$ implies that SHG j does not conform to any of coordinator's financial directives. Any other level of commitment to the directives will have I_j^{fin} having an intermediate value between 0 and 1.

Let I_j^{task} represent SHG's conformation to the coordinator's directives regarding tasks of the SHG. The task directives might state that every SHG should provide details of all tasks it has executed, all tasks that it is executing currently, commitments for future tasks, details of its current collaborations, etc. $I_j^{task} = 1$ implies that SHG j fully conforms to all the coordinator's directives of task. $I_j^{task} = 0$ implies that SHG's task execution does not conform to any of the coordinator's directives. Any other level of commitment to the directives will have I_j^{task} having an intermediate value between 0 and 1. Let I_j^{pol} represent SHG's conformation to the coordinator's directives on political aspects. Coordinator's political directives might state that SHGs should not receive funds from some of the listed organizations, not should sign contracts

with political parties, etc. $I_j^{pol} = 1$ implies that SHG j fully conforms to all the coordinator's directives regarding political aspects, and $I_j^{pol} = 0$ implies that it does not conform to any of the political directives laid down by the coordinator. Any other level of commitment to the directives will have I_j^{pol} having an intermediate value between 0 and 1.

Finally, total integrity of SHG j can be obtained by summing all integrity aspects' values.

$$\text{Integrity of SHG } J,\ I_J = I_J^{Org} + I_J^{Fin} + I_J^{Task} + I_J^{Pol}$$

Since the range of each component is [0,1], the range of integrity of an SHG is [0,4].

Transparency vs. Integrity: There is a considerable difference between transparency and integrity. Transparency of an SHG specifies whether an SHG reveals the required information or not, but *integrity* is more than that. Integrity requires that the information should conform to the principles of the authorities. For example, an SHG is regarded as transparent if it reveals all the information about its members. But integrity is about checking conformance of organizational directives, for example, whether the *'count of members is in accordance with the directives or not'*. That is, if an SHG discloses that it has 30 members and provides details of all the members, then we say that the SHG *has* transparency. But we have to agree that the SHG *lacks integrity*. This is because the coordinator's organizational directive says that an SHG should not have more than 20 members. So, transparency and integrity are different, and we can say that the considering integrity for computing dependability is a rational idea.

5.1.5 Dependability

Now that we have computed trust, competency, and integrity values, we can now compute dependability by summing up all the three components. Dependability of SHG j computed by SHG i for service 'a' can be represented as, D_{ij}^a.

$$D_{ij}^a = T_{ij}^a + C_j^a + I_j$$

An observation of the above equation reveals that dependability value varies with time and is also evaluator dependent. This is because of trust, which is evaluator specific. Trust computation includes 'direct trust' — which is based on individual experiences with a target SHG. Different SHGs can have different experiences with the same target SHG. Hence direct trust value differs for each evaluator. Competency of a target SHG may be evaluated in a similar process by all SHGs, but its value varies because competence is task, time and constraint dependent. Competency for a task may be high at one time, and low at another. This is because the same constraints which are formerly applicable may not be applicable later. Also, an SHG

may be highly competent for one type of task and poorly competent for another. Integrity is a measure of an SHG's adherence to the directives of the system/coordinator. So, its value will be same for all evaluators. SHG competency is task dependent and integrity is dependent on its behavior and attitude towards principles of the system and Coordinator.

5.2 Dependability and the SBM

Let us see how dependability is mapped onto the SHG Behavioral Model (SBM). Direct trust is calculated whenever there was an interaction with a target SHG. Rpt has the subprocesses defined in it to calculate direct trust of other SHGs and store the reports in the Rpst. The subprocesses — assessQual, assessSpeed, assessCost and assessDura will compute the values of quality, speed, cost effectiveness, and durability of the service/product supplied by a target SHG and will store the result in the Rpst.

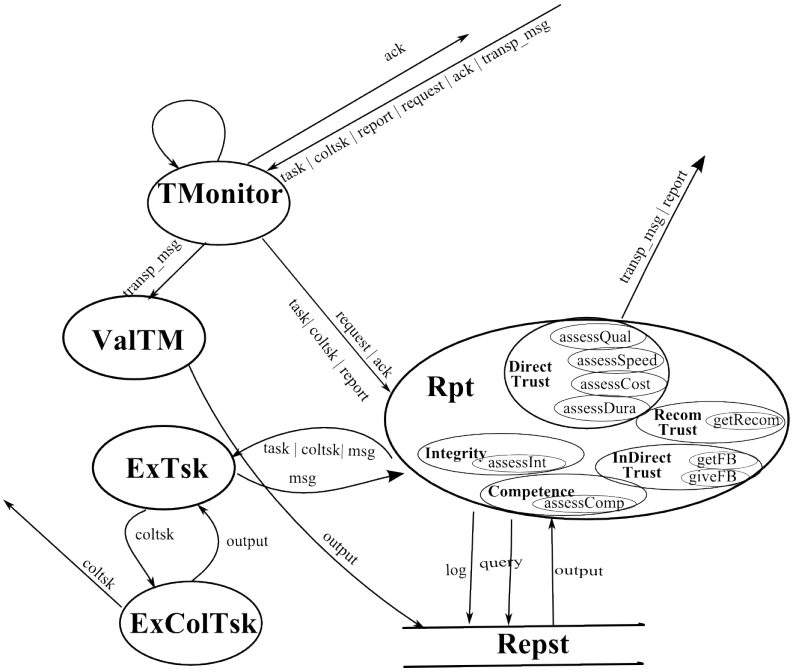


Figure 5.1 Dependability mapping onto SBM

5.3 Dependability Computation

The strategy for computing dependability in an SHG, is by beginning with the computation of trust followed by computation of competency and integrity. Trust is a composition of direct, indirect, and recommender trusts. So, let us first outline computation for direct trust and then proceed to indirect and recommenders trusts, and afterwards calculate competency and integrity. Because the maximum values for trust, competence and integrity are 6, 5 and 4 respectively, and

the minimum value is zero, the range for dependability value of an SHG for a service will be [0,14].

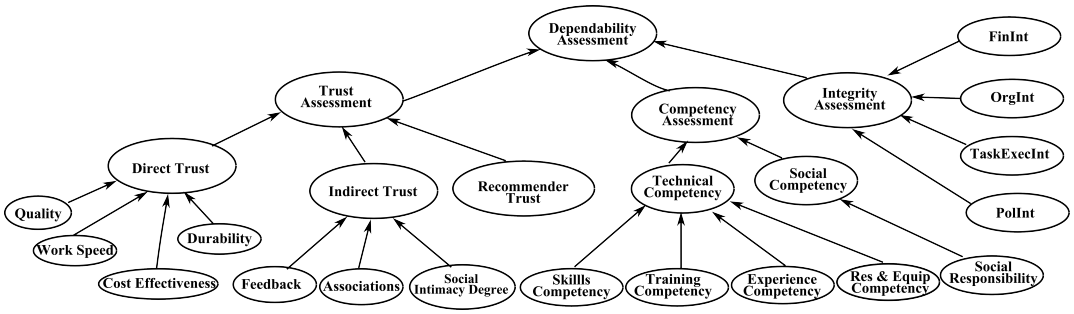


Figure 5.2 Dependability Framework

5.3.1 Direct Trust Computation

To calculate direct trust for a service/product of a target SHG, we first need to compute values for quality, work speed, cost effectiveness, and durability.

Quality

Quality is assessed by an SHG for a target SHG's service/product. Quality evaluation can be made using a quality table. The attributes in the quality table are — Service/Product, SHG_id, Material Quality [MQ], Design Correctness [DC], Timeliness [TB], and Compliance to Standards [CoS]. Upon observation of the previously availed services from the target SHG, the evaluating SHG assigns values to each of these attributes of a service. Each attribute is given a 1 or 0 or any intermediate value between 0 and 1 such that 1 implies good/true and 0 for bad/false. The final value for quality is obtained by dividing the sum total of all the attribute values by 4.

$$\therefore Final\ quality\ value = \quad \frac{MQ+DC+TB+CoS}{4}$$

Service / Product (a)	**Material quality [MQ]**	**Design correctness [DC]**	**Timeliness [T]**	**Compliance to standards [CoS]**	**Quality assessed**
a	1	1	1	1	1
b	0	0	0	0	0

Table 5.2 Quality assessment for Direct Trust

If a supplied service/product has good material quality, design correctness, timeliness, and compliance to standards, then MQ=1, DC =1, T=1 and CoS =1, and the quality value for that service/product is 1. But if a supplied service/product is poor in material quality, design correctness, timeliness, and compliance to standards, then MQ=0, DC =0, T=0 and CoS =0,

making the quality value for that service/product to be 0. For intermediate values of these attributes, quality will also get an intermediate value between 0 and 1.

Work Speed

Based on the previous experiences, it is possible to compute work speed of a target SHG for a task using the following attributes — No. of days allotted for task completion, No. of days actually taken for completion, ratio of difference in days (±), Speed of Completion.

No. of days allotted for task completion (N)	**No. of days actually taken for completion (M)**	**(±) Delay ratio $\frac{(N-M)}{N}$**	**Speed of Completion**
10	8	0.2	1
10	12	- 0.2	0.8
10	10	0	1

Table 5.4 Speed of Completion of Task

If $\frac{(N-M)}{N}$ = 0, it means that the SHG has completed the task in perfect time and speed of completion = 1. If $\frac{(N-M)}{N} > 0$, it means that the SHG has completed the task ahead of time and speed of completion = 1 + $\frac{(N-M)}{N}$. But this is rounded to 1 as the maximum value for speed of completion is 1. If $\frac{(N-M)}{N} < 0$, it means that the SHG has delayed the task execution and speed of completion = 1 - $\frac{(N-M)}{N}$. The minimum value it can have is 0, since the minimum value of speed of completion is 0. For intermediate values of these attributes, work speed will also get an intermediate value between 0 and 1.

Cost Effectiveness

An SHG measures the cost effectiveness of a target SHG by comparing the cost claimed/levied by target SHG against the cost quotations of others. The attributes required for checking cost effectiveness are: Lowest cost quoted by other SHGs, cost quoted by target SHG, cost effectiveness, etc.

Service/ Product	**Lowest cost quoted by other SHGs (N)**	**Target SHG**	**Cost levied by target SHG (M)**	**Difference ratio $\frac{(N-M)}{N}$**	**Cost effectiveness**
a	100	SHG j	80	0.2	1
b	100	SHG j	120	- 0.2	0
c	100	SHG j	100	0	0

Table 5.6 Cost effectiveness of service/product

For a service, if the costs levied by the target SHG are more than what is quoted by the others, then the target SHG is not cost effective.Hence cost effectiveness in such a case is 0. However, if the target SHG quotes less than the cost compared to other SHGs for the same service, then the target SHG is cost effective and the value of cost effectiveness is 1. For intermediate values of these attributes, cost effectiveness will also get some intermediate value between 0 and 1.

Durability

An SHG computes the durability of a target SHG's service/product by testing/observing the supplied service/product. To compute durability it uses a table with the following attributes — Service/Product, Type of Durability Test, Result, and Durability. Each service/product is tested with the relevant procedures.

Service/ Product	Durability Tests	Result	Durability
a	Pressure Testing, Thermal Testing, Load Testing	success	1
b	Aging, Burst Testing, Operational Limits	failure	0

Table 5.8 Durability of a product

Having computed values for Quality, Work Speed, Cost Effectiveness, and Durability of a service/product, Direct Trust for that service/product of the target SHG is computed by summation of all the four attributes.

5.3.2 Indirect Trust Computation

Indirect trust is computed with the help of the peers of the target SHG.

Feedback: Feedback is computed from the feedback values obtained from the peers of the target SHG. The obtained feedback values are fed into a feedback table. The attributes in the feedback table are — Servicename, SHGid, Cost, Quality, Transparency, Timeliness, ResponseTime, Trust, and Timestamp. Each peer of the target SHG is sent a feedback form in which it has to respond to questions relating to quality, cost effectiveness, transparency, timeliness, and responsiveness of the target SHG. Each tuple in the feedback table represents the feedback by an SHG which has some direct trust value to the target SHG.

Service/ Product	Target SHGid	Cost	Quality	Transp arency	Timeli ness	Response Time	Feedback giving SHG	Time stamp	Feedback Value
a	SHG098	3	3	3	3	3	SHG567	12-2-2012:04:50	1
a	SHG098	0	0	0	0	0	SHG433	22-2-2012:08:44	0

Table 5.10 Feedback on the product

Feedback value of a target SHG can be computed as follows:

$$IT_{ij}^{a.Feedback}(t) = \sum_{i=1}^{n} \frac{a.FB.x_i}{n * S_{mx}}$$

$$= \sum_{i=1}^{n} \frac{a.FB.x_i}{15}$$

$FB.x_i$ indicates the attributes of the feedback table. S_{mx} represents the maximum value of the scale used for measuring each attribute, and n is the number of attributes. Since, n = 5 and S_{mx} = 3, we have $n * S_{mx}$ = 15.

Associations

Computation of an association also needs the peers of the target SHG. For computation of association, an SHG uses the association table. The attributes in the table are — Target-SHG-id, ServiceName, Stakeholder1-CollabRating, Stakeholder2-CollabRating,.., AvgCollabRating. If a is the service, n is number of associated SHGs, then, association trust value of a target SHG is given by,

$$IT_{ij}^{a.Assocn}(t) = \frac{\sum_{i=1}^{n} a.CollabRating_i}{n}$$

Computation of trust due to association is not time consuming because it avoids waiting time for receiving feedback from all the peers; it only considers the CollabRatings of the peers and makes an average of it. Hence it is a time saving process.

Service/ product	target SHGid	Trusts of stakeholders	AssociatedTrust
a	SHG007	1,1,1	1
b	SHG007	0,0,0	0

Table 5.12 Associations Trust

5.3.3 Recommender Trust Computation

To compute recommender trust, an SHG uses the Recommenders Table which is managed by the coordinator or recommenders. It consists of information regarding all the expert recommenders nominated by the coordinator, and their areas of expertise. The attributes of the recommenders table are Recommenders, Service/Product, RecommendedSHG, RecommendationValue. The Recommendation value is given on the basis of the confidence of the recommenders and their discretion about the service/product of the target SHG. The highest value for recommendation is 1, i.e. $RT_{ij}^{a} = 1$ and the lowest value is $RT_{ij}^{a} = 0$. Intermediate values carry the corresponding level of confidence in recommendation.

Recommenders List	Service / Product	Recommended SHG	Location	Recommendation Value
rec_100, rec_34, rec_78	carpentry	SHG_007	Guntur	1
rec_120, rec_233, rec_484	garments	SHG_908	Hyderabad	0.5

Table 5.14 Recommender Trust

5.3.4 Competence & Integrity

The value for competency of a target SHG is computed from the *Competency Table* of the target SHG. It is assumed that each SHG maintains a competency table which portrays the competency information of that SHG. The attributes of the Competency Table are SHGId, services/products, skills, trained/not-trained, experience, awareness, resources & equipment, friend SHGs. Every SHG updates its own competency table. Competency information is updated each time a new member joins or leaves, skills added or removed, resources acquired or lost, participated in social work schemes or withdrawn from social work schemes, etc. Basically, it is updated whenever an event happens that affects the technical or social competencies of an SHG.

Based on the values in the target SHG's competency table, an SHG computes a competency value for a service/product of the target SHG.

Service/ Product	SHGid	Skills	Training	Experience	Awareness	Res & Equip	Soc_ Resp	Competence Index
prod1	SHG_007	1	1	0	0	1	0	3
prod2	SHG_007	1	0	1	0	1	1	4

Table 5.16 Competency of the SHG

To compute integrity of a target SHG, an SHG uses the Integrity Table. This table is also the same as the Competency Table and is maintained at each SHG. It contains information of an SHG with respect to the directives of the coordinator. Each SHG has to provide information about its adherence to each and every directive of the coordinator and the system. Adherence is tested against organizational, financial, executional and political (operational) aspects. Suppose there are ten directives about organizational integrity, and an SHG adheres to all the directives, then organizational integrity of that SHG will be 1. The same method of computation is used for the other three aspects also. The attributes in the integrity table are: SHGid, Org_Integrity, Fin_Integrity, Exec_Integrity, Pol_Integrity, Total Integrity.

SHGid	Org_Integrity	Fin_Integrity	Exec_Integrity	Pol_Integrity	SHG_Integrity
SHG_007	1	0	1	0	2
SHG_011	1	1	1	0	3

Table 5.18 Integrity table

5.3.5 Dependability metric

After computing trust, competency, and integrity values, dependability of a target SHG is computed using,

$$Dependability = Trust + Competency + Integrity$$

The range of dependability is [0,15] because the ranges of trust, competency, and integrity are [0,6], [0,5], and [0,4]. For a specific service/product, if a target SHG scores a value in the range of [0,5], then it is not a dependable SHG for that service/product; if it scores in the range of [6,10], then it is moderately dependable; but if it scores in the range of [11,15], then it is highly dependable. Even in the case of moderately and highly dependable SHGs, a more reasonable judgment would be to have a base value for each of the factors of dependability. This solves the issue for such cases where one factor is a zero. The base value requirement ensures that each factor will be a non-zero, positive and a certain value.

5.4 Conclusion

This chapter introduced the concept of dependability in SHGs and discussed its framework in the SHG system. It also proposed a strategy for computation of dependability of an SHG. The concept of dependability arises when an SHG wants to collaborate with another SHG or when the coordinator wants to assign a task to an SHG. In both cases, verification is done for SHGs to determine if they are dependable or not. This chapter defined a dependability framework which uses a summation of trust, competence, and integrity for dependability assessment. The properties and types of trust are discussed with relevance to SHGs. Trust, which is of three types — direct trust, indirect trust and recommender trust, is explained with a simple example. Also, the components of each type of trust are explained. Direct trust is computed from attributes such as quality, speed of completion, cost effectiveness and durability. Indirect trust is computed using one of the behavioural attributes such as feedback, associations, or social intimacy degree. Recommender trust is computed using recommendations of an expert committee. Competency and integrity are defined along with the processes of quantifying them have also discussed. The computation details for each of the attributes of direct trust, indirect trust, recommender trust, competency and integrity were also given. The Dependability metric and its implications were also discussed. This chapter considered most of the attributes that have a significant influence on dependability. It is possible to find even more attributes, but that is not required. The present attributes are good enough to determine whether an SHG is good enough for collaboration or not. Instead of working more on this concept of dependability, it is felt that it would be more beneficial to have more research done on other factors that could make collaboration more successful.

Chapter 6

Implementation

6.1 Introduction

In the previous chapters, we discussed transparency, collaborations and dependability concepts of SHG system at a generic level, stating them in theory. This chapter describes the implementation details of the proposed concepts in a prototype system. In the present context, the term "system" refers to the prototype of SHG management system being developed.

UML for Specification

The proposed system is specified using Use Case, Class, Activity, Interaction and Communication diagrams of UML (Unified Modeling Language), and constraints are specified using OCL (Object Constraint Language). UML is a graphical language, offering a set of diagrams and notations to describe a system. It is a standard notation for modeling of real-world objects and systems and is now the de facto modeling language for software development. Several features account for its popularity: it's a standardized notation, rich in expression, and provides 14 diagram types that enable modeling several different views and abstraction levels [127]. It provides the modeler of object-oriented systems with ways to express the semantics of an object-oriented model in a precise manner [128].

A Use Case diagram is a graphic depiction of the interactions of external users with a system. A Use Case is a methodology used in system analysis to identify, clarify, and organize system requirements. Class diagrams provide platform independent view of the system and are probably the most important and well-established of UML models. They play an essential role in the analysis and design of complex systems and allow specification of constraints of cardinality, class hierarchy, and inter-association[129].

This chapter also provides Activity diagrams to give graphical representations of work flows of stepwise activities and actions. Activity diagrams provide support for choice, iteration and concurrency. They provide a view of the behavior of a system by describing the sequence of actions in a process. They are similar to flowcharts as they show the flow among the actions in an activity; however, they can also show parallel or concurrent flows and alternate flows. Interaction diagrams are models that describe how a group of objects collaborate in some behavior - typically a single use-case. The diagrams show a number of example objects and the messages that are passed between these objects within the use-case. In other words, it can be said that the working together of objects to implement a behavior is specified by interaction diagrams. Such a diagram shows the sequence messages exchanged among objects in implementing a behavioral aspect of a system. Communication diagram, formerly called a collaboration diagram, is an interaction diagram showing similar information to sequence diagrams but with primary focus on object relationships.

UML defines several models. The Class model captures the static structure, Use Case model describes the requirements of the user, Interaction model represents the scenarios and messages flows. UML defines nine different types of diagrams. Of all these nine types, this chapter uses only four types of diagrams to give a clear understanding about the prospects of the proposed system and its implementation. Use Case diagrams represent the functions of a system from the user's point of view, Class diagrams represent the static structure in terms of classes and relationships, and Activity diagrams represent the behavior of an operation as a set of actions. Communication diagrams represent the interactions between objects or parts in terms of sequenced messages. Communication diagrams represent a combination of information taken from Class, Sequence, and Use Case Diagrams describing both the static structure and dynamic behavior of a system.

Object Constraint Language (OCL) is a language that has been defined as part of the UML. It is a standardized constraint and query language supported by the OMG and a number of modeling environments. OCL is generally used to specify — invariants on classes and types in the class model, type invariants for stereotypes, pre- and post-conditions on operations and methods, guards, and constraints on operations. Invariants: the basic idea of an invariant is a "condition which always holds", so it is a truly static constraint. Also, constraints are expressed not globally, but locally, i.e. a constraint is always "attached" to a single object, to follow object-oriented principles. For OCL invariants, a class is used as the "hook" for the invariant. This class is called the context for the invariant and represented as:
context <class name> inv: <Boolean OCL expression>.

Pre-conditions: a pre-condition is a condition that is assumed to hold before an operation is executed. To any operation in a UML class diagram, a pre-condition (and a post-condition) in OCL may be attached, which are Boolean-valued expressions on the object configuration, using additional parameters that correspond to the operation parameters. Post-conditions: post-conditions are the way, how the actual effect of an operation is described in OCL. Some of the notations used in OCL expressions are: m- member, t- task, g- SHG, SHGC- SHG Coordinator, msg- message, coltsk- collaborated task.

Format: context <class name> :: <operation> (<parameters>)
pre: <Boolean OCL expression>
post: <Boolean OCL expression>

6.2 System & Use Cases

6.2.1 System Architecture

The system architecture (Figure: 6.1) contains three main modules: Interface, TskExec and SOCIALIZE. The first two modules are common in all systems that execute tasks. Task execution was well studied for a long time. So, we do not re-work on task execution, but we bring the socializing attributes into the system. Though we have discussed through out the work, task and its states during execution, the primary aim remains the same, bringing social attributes into the system. This introduces a new module, called SOCIALIZE, for this purpose. SOCIALIZE contains three sub-modules — Transpa, Collab, and Depen — which are abbreviated forms for transparency, collaborations and dependability. The system is scalable, i.e. more sub-modules can be added in future work. These three sub-modules have a consistent relationship among them and it is shown in the figure (Figure: 6.2).

Transparency is ascribed to the SHG system by incorporating ‘Transpa’ sub-module in to the SOCIALIZE module. Transpa is designed to bring transparency to an SHG through the tasks under execution by that SHG. Transpa interfaces to TskExec, which in turn has interface to the stakeholders. So, TskExec is a conduit between stakeholders and the SHG system. Transpa has two primary goals. First is to make an SHG transparent, and second is to use the transparency entity in choosing SHGs as collaborators. The functionality of Transpa is discussed in the Transparency section of this chapter.

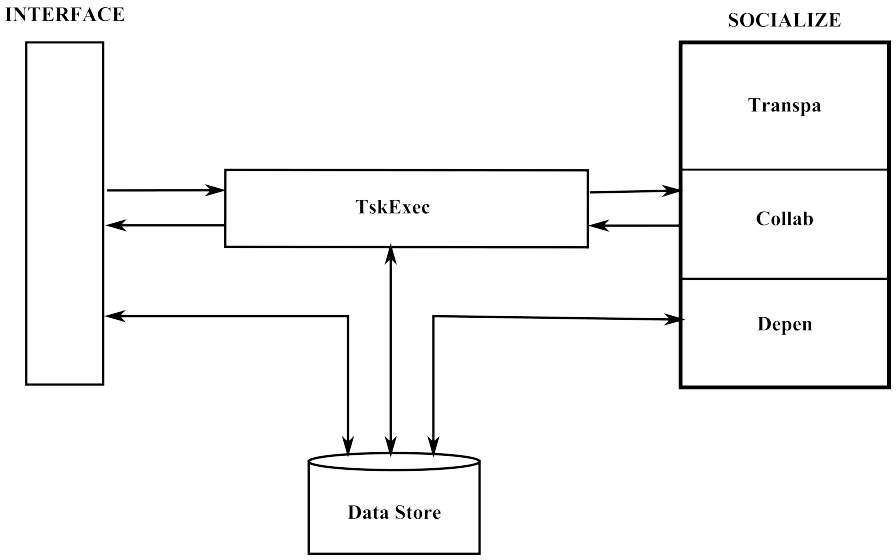


Figure 6.1 System Architecture

The hardware system architecture is not of significance in this work. The system is a typical web application and runs on any modern computer system. So we discuss only the logical system architecture and its top-level components. The logical system architecture contains eight fundamental components. A higher level implementation of framework with the software components is given in the figure (Figure: 6.2).

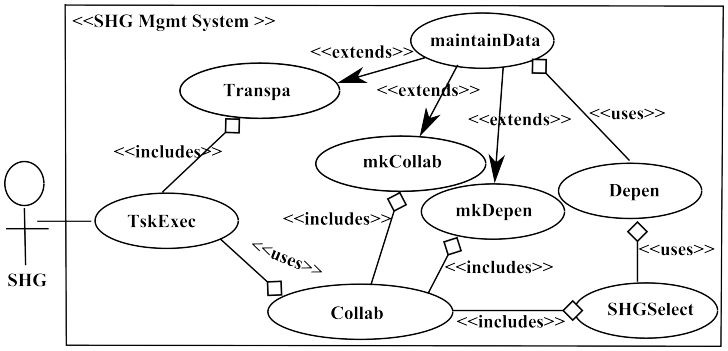


Figure 6.2 Relationship between sub-modules of SOCIALIZE

.

TskExec component monitors tasks during execution, updates the status of tasks regularly, and reports accordingly. It is connected to two other components - Transpa and Collab with ‘includes’ and ‘uses’ relationships correspondingly. The ‘includes’ relationship implies that while executing a task, the system is also making the SHG transparent. The ‘uses’ relationship to the Collab implies that while executing a task, it uses collaboration of other SHGs.

The aim of Transpa component is to make an SHG transparent in its task execution.

Transparency in task execution is achieved through transparency messages. The objective of Collab component is to search and identify potential collaborators for a task. It uses PubSub model to find collaborators and uses specified criteria to sort and select the best ones of them. This component has 'includes' relationship to three other components: mkCollab, mkDepen and SHGSelect. SHGSelect component identifies and selects some collaborators based on criteria such as locality, transaction costs, services, and speed feasibility. Collab component has also got an 'includes' relationship to mkCollab component which helps an SHG to keep itself up in its CollabRating. This component keeps monitoring and alerting the SHG regarding its timeliness, productivity, communication and commitment. Also, SHGSelect component is related to Depen component through an 'includes' relationship. Depen component provides necessary information to the SHGSelect regarding trust, competency and integrity attributes of the collaborator SHGs. This component uses the data which was collected and maintained by the maintainData component. The data maintained by the maintainData component would be the data helping to evaluate indirect trust of a collaborator SHG. Collab component has also got an 'includes' relationship to mkDepen component whose role is to help an SHG become more dependable. This component monitors and alerts an SHG about its quality, work speed, cost effectiveness, durability, social impact, skills, training, social responsibility and integrity. The mkTranspa, mkCollab and mkDepen components have an 'extends' relationship to maintainData. This means that they write certain information which helps other SHGs and stakeholders to understand about an SHG's transparency, collaborations and dependability.

And, only TskExec component provides interface to the users. All the other components do not have any interface to the users. They only interact with each other and the TskExec component.

OCL constraints of the system:

1. **context** Member **inv**: SHG → nonEmpty( ) implies
   5 <= Members → size() <= 20 , and
   member.gender = Gender::male | Gender::female, and
   skills → size() >= 1.
   forAll(m1, m2 : member| m1.memberid <> m2.memberid)
   Each SHG should have member count between 5 and 20, and members can be either male or female, and also each member should possess at least one skill. Every member has a unique memberId.

2. **context** SHG **inv**: UniqueId

   forAll(g1, g2 : SHG | g1.shgid <> g2.shgid)

   forAll(t1, t2: task | t1.taskid <> t2.taskid)

   member → forAll(m:Person | m.responsbile = SHG)

   Every SHG has a unique SHGId.

   Every task has a unique taskid.

   Members are responsible to SHGs.

3. **context** SHG **inv**: responsibility

   SHG → forAll(g:SHG | g.responsbile = SHGC)

   SHGs are responsible to coordinator.

4. **context** Member **inv**: membership

   forAll(m: member, g1,g2: SHG | m.g1.membership = true implies m.g2.membership = false)

   Each member has membership in only one SHG.

5. **context** SHG **inv**:skills

   forAll(g: SHG| g.hasSkill() = true)

   Each SHG has at least one skill. SHG uses this skill(s) for participating in collaborations.

6. **context** SHG **inv**: SHGskills

   forAll(g: SHG | g.skill → size() = select(g.member.skills))

   The union of all skills of members of an SHG forms the skills of that SHG.

7. **context** SHG **inv**:values

   forAll(g: SHG | g.hasTranspVal = true & g.hasCollabRating = true & g.hasDependVal = true)

   Every SHG has a Transparency value, a CollabRating value and a Dependability value.

8. **context** Task **inv**:TaskDistance

   Task → nonEmpty( ) implies

   TaskDistance → size() > 0

   TaskDistance of any uncompleted task is greater than zero.

Now that we have studied the *relationships* between the sub-modules of SOCIALIZE (fig: 6.2), we shall have a closer look into each of the sub-modules.

6.2.2 Transparency

As discussed in the Transparency chapter, each SHG needs to be transparent in its task execution. Also, SHGs with high transparency values are preferred for collaborations. So, the system's objective is to see that a logically correct and reasonable transparency value is ascribed to each SHG in correspondence to its behavior during task execution. The chapter defines that transparency in task execution can be achieved by reporting relevant information whenever there is a task state change or an update is available. Transparency messages (task updates) are also sent periodically as and when alerted by the Message Scheduler. It defines the criteria for judging a received message to be qualified as a transparency message. There are five criteria that every message should possess. They are timeliness, relevance, completeness, consistency and understandability. While checking for validity of received transparency messages, an SHG should also prepare messages in such a way that they will meet that criteria. Messages that do not meet the criteria are not transparent messages, and they cannot be used in transparency computation.

So, the system has three functions - first is to see that an SHG is ascribed a right value for its behavior, second is to see that an SHG uses this transparency value while forming collaborations, and third is to see that an SHG becomes transparent *in* and *through* task execution. Implementation of this transparency concept involves making an SHG transparent, evaluating the transparency of an SHG, and using this transparency value during collaborations. We used transparency messages as the conduits for affecting the transparency of an SHG.

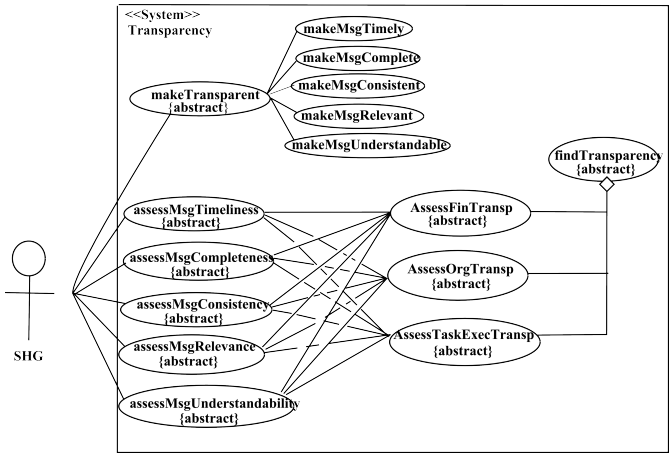


Figure 6.3 Use Case diagram for Transparency

Use Case diagram (fig: 6.3) shows the functionality of the system inside Transpa. The same system is implemented at each SHG. From the figure, it is evident that there are basically two fundamental functions. One is about making an SHG transparent, and the second is about assessing the transparency of a target SHG. The aim of mkTranspa component is to make an SHG transparent. It has five sub-components to achieve this objective. These five subcomponents work on each transparency message that an SHG sends. The roles of the sub-components are intuitive from their names. The role of mkMsgTimely is to see that each transparency message is timely. Therefore, it uses a message scheduler to alert the SHG to send transparency messages in due time. Also, when a message is prepared for dispatch, it will verify the message for timeliness. Suppose a message is to b e sent at the end of a month, but the message is being sent in the middle of the month, then this mkMsgTimely will suggest for consideration about dropping the message. The role of mkMsgComplete is to see that all the required information regarding the task is being added in the message. It will provide a checklist of items which an SHG has to ensure filling information before dispatching a message. The role of mkMsgConsistent is to ensure that a message is logically correct. It will compare the current message with the previous messages. Mostly, logical correctness may be about task progress and financial conditions. The role of mkMsgRelevant is to ensure that a message is related to the stakeholders' interests and matches their subscriptions. Since the stakeholders vary in interests, it will verify that the message matches with the interests of the stakeholder who is the message recipient. The role of makeMsgUnderstandable is to ensure that the message is in readable and comprehensible format. Its functionality includes checking message's format, language, special characters, and also suggesting the SHG to check the logical meaning. It is not possible at the moment to exactly state which things will be incomprehensible. Since the users (SHGs) are from different cultures, geographical areas, businesses and backgrounds, when the system is implemented, in due course of time there will definitely be some specific issues that are not commonly understood by everyone. And, the role of this sub-component is to overcome that incomprehensibility.

The second functionality of Transpa is to assess the transparency of an interacting SHG. Mostly, interacting SHGs would probably be collaborators. When a collaborator sends a transparency messages as specified by the coltsk, the receiving SHG will assess the transparency of the collaborator based on the received messages. Since an SHG can maintain a different level of transparency with different SHGs and for different tasks, the receiver SHG will evaluate the transparency for each message. For each received transparency message, it verifies timeliness, completeness, consistency, relevance, and understandability with the

help of the sub-components of Transpa. A transparency rating is given after verification by the sub-components. There are three more sub-components, viz. AssessFinTransp, AssessOrgTransp and AssessTskExecTransp. They will use each of the five sub-components (assessMsgTimeliness, assessMsgCompleteness, assessMsgConsistency, assessMsgRelevance and assessMsgUnderstandability) to assess the financial, organizational and executional transparency of the target SHG. Based upon these three values, findTransparency will evaluate a final value for the transparency of the target SHG.

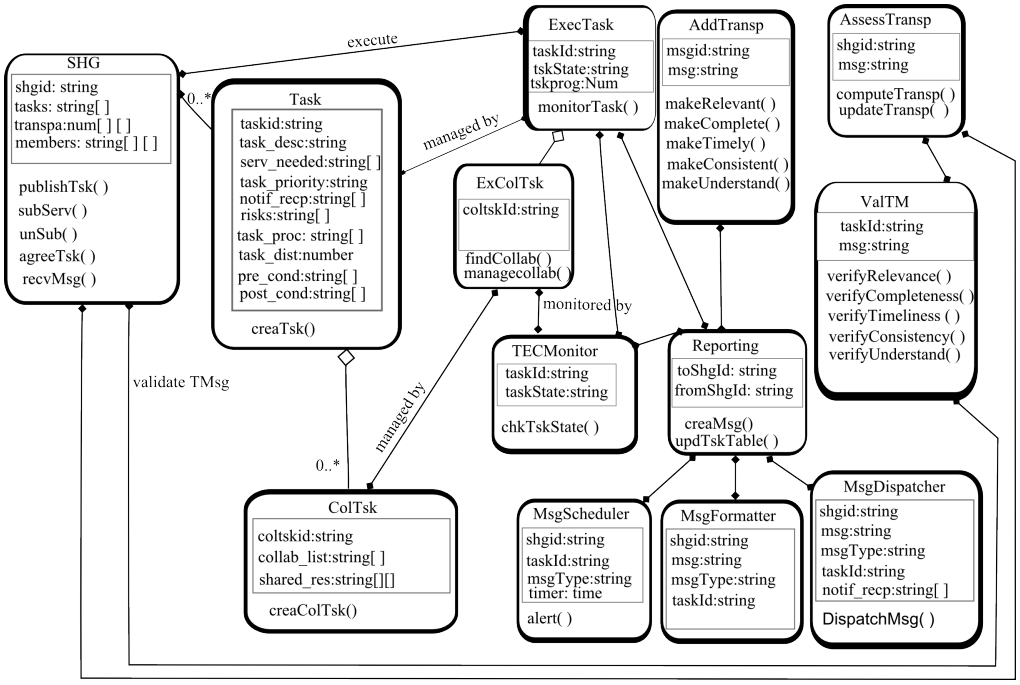


Figure 6.4 Class Diagram for Transparency

The class diagram of transparency (fig: 6.4) depicts classes and their methods involved in implementing the proposed concept. It shows the associations and cardinal relationships among the classes.

Description of Classes

SHG class contains attributes and methods related to transparency of an SHG. Attribute $shgid$ is the unique identification number which identifies an SHG over all the SHG network. The attribute $tasks$ is a list item which holds a list of ask_ids that an SHG is executing. The attribute $transpa$ is a two dimensional array containing the interacting/collaborating SHGs' shgids and the corresponding transparency values assessed for them. These values are supplied by the $ValTM$ class. When an SHG receives a message, it will forward it to the $ValTM$ class which verifies each message for its relevance, completeness, timeliness, consistency and understandability. $ValTM$ forwards its assessment to $AssessTransp$ which computes

a transparency and updates the transparency value for that message sending SHG. The method $publishTsk()$ is used to create and publish a task to the SHG coordinator. To create a task it will use the $Task$ class. Once the task is returned to it, it will send the task and other required identification information to the SHG Coordinator. The method $subServ$ is used for subscribing to the SHG Coordinator with available roles/services. $UnSub$ method is used for canceling the subscribed roles; $AgreeTsk$ method contacts the task sender and informs about the SHG's agreement to execute/collaborate for a task. $RecvMsg$ method is a continuously monitoring method which keeps monitoring for any incoming messages and receives them.

The $Task$ class creates tasks for an SHG. Whenever a new task is to be created, the SHG class will call this $Task$ class which creates a new task in the standard format. CreaTsk method ascribes appropriate values and data to the attributes — taskid, task_desc, serv_needed, task_priority, task_proc, task_dist, notif_recp, risks, pre_cond and post_cond. The $ColTsk$ class inherits all its attributes from $Task$ class and adds a few more attributes related to collaborations. It adds a $coltskid$ which is an identification for a task when executed with collaborations ; $collab\_list$ is a list of collaborators for executing a coltsk; $shared\_res$ is a list of resources that are being shared during collaborations.

$ExTsk$ class has two attributes namely $tskState$ and $tskprog$. The attribute $tskState$ is updated according to the state of a task in execution, and $tskprog$ is a value indicating how much of the task was completed. These two values will be accessed by the $Reporting$ class to notify the stakeholders. $TECMonitor$ is a monitoring class which monitors the task for state changes. Whenever it finds a change in the task state, it will invoke the $Reporting$ class. $TECMonitor$ has one method called $chkTskState$ which keeps checking for changes in task state. $Reporting$ class has two methods $creaMsg$ and $updTskTable$. The $creaMsg$ method will initiate the creation a transparency message. Transparency messages vary in type depending upon the type of message and task state. $Reporting$ class also has $updTskTable$ which updates task's current values in the task table, a step towards making an SHG transparent.

$Reporting$ class has relationship with three other classes namely, $AddTransp$, $MsgFormatter$ and $MsgDispatcher$. $AddTransp$ class will add details to the necessary organizational, financial and executional information to qualify a message as a transparency message. It has methods to check and make a message relevant, complete, consistent, timely and understandable. $MsgFormatter$ class will format a message according to the type of update. This is because, each type of update has a different information and so they have different message structures. $MsgDispatcher$ will send the messages to the notification recipients. $MsgScheduler$ will not wait for task state to change but alerts the $Reporting$ class

at specified intervals to send the transparency messages. The intervals may be given by task sender, coordinator, or specified during collaboration agreement.

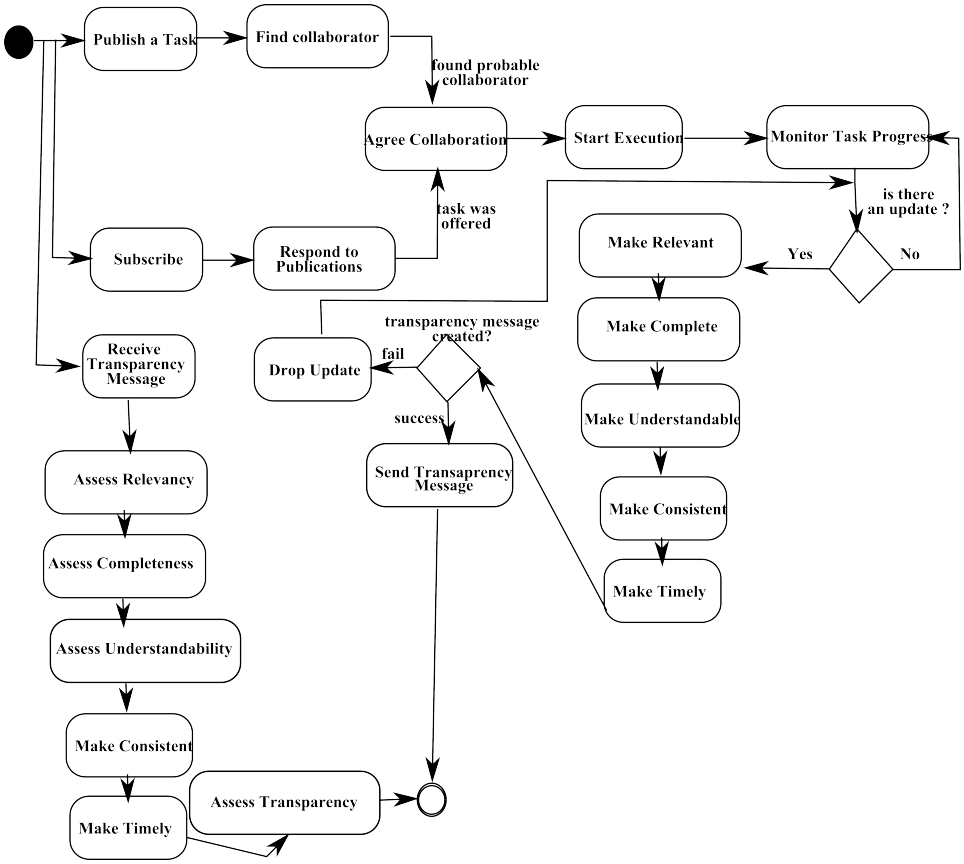


Figure 6.5 Activity diagram for Transparency

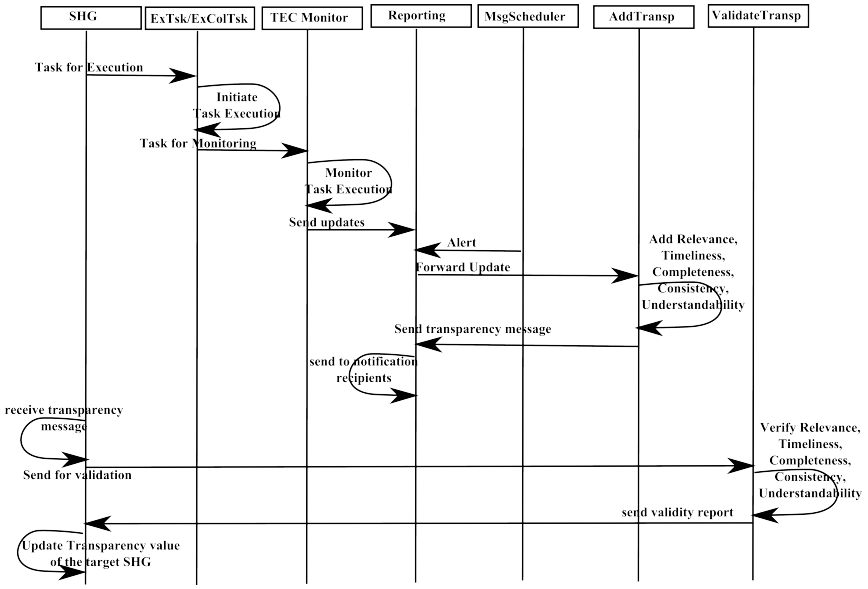


Figure 6.6 Interaction Diagram for Transparency

OCL constraints of Transparency:

1. **context** MsgFormatter **inv**: message

   forAll(msg1, msg2 | msg1.format = msg2.format $\Longrightarrow$ msg1.type =msg2.type)

   Each type of transarpency message has a specific format and they carry varied transparency information.

2. **context** ValTM **inv**:coltskstates

   forAll(msg: message, TMsg: Transparency message | msg is TMsg $\Longrightarrow$

   msg.relevance = true,

   msg.timely = true ,

   msg.complete = true ,

   msg.consistent = true, and

   msg.understandable = true)

   A message can be called a transparent message if its information is relevant, timely, complete, consistent and understandable.

3. **context** MsgScheduler **inv**:msgsched

   forAll(alert, msg | alert is true $\Longrightarrow$ sendMsg(notifrecp, msg) =true)

   A transparency message is sent whenever there is an alert from MessageScheduler

4. **context** AssessTransp::Transparency : transpmetric

   derive:

   let table : Set(Tuple(range : Sequence(Real), inference : transpmetric)) =

   Set{

   Tuplerange={0 .. 0.1} , inference=transpmetric::opaque,

   Tuplerange={0.2 .. 0.7} , inference=transpmetric::translucent,

   Tuplerange={0.8 .. 1} , inference=transpmetric::transparent, }

   in table→any(range→includes(SHG→transparency())).transpmetric

5. **context**: AssessTransp **inv**:valrange

   forAll(g1| 0 <= g1.service.transp→ value <= 1)

   The range of transparency value for any service of an SHG is 0 and 1.

6.2.3 Collaborations

SHG collaborations are discussed in the Collaboration chapter. Collaborations submodule is added to SOCIALIZE module to facilitate meaningful collaborations in SHGs. Usually SHGs collaborate to achieve higher objectives or execute complex tasks. Self Help Groups, by definition, ought to be collaborative, resulting in combined groups empowered to execute tasks which otherwise are not feasible for execution at individual level. That is, collaborations in SHGs occur over complex tasks which cannot be completed by a single SHG. In such a case, an SHG will publish a task specifying the services required. Subscriber SHGs will respond to the publications. When both subscriber and publisher make an agreement, then collaboration will begin and they will start executing the task. So, implementation of collaboration involves implementation of publishing, subscribing (Publish Subscribe model) and agreement functionality. This functionality is implemented by the three components — Publish, Subscribe, and CAgree. Also, for each published task, there may be several subscribers willing to collaborate. In such a case, an SHG has to choose one (or few) from a collection of SHGs to be the collaborator(s). The criteria used for filtration are locality, transaction costs, services offered, and speed. So, for implementation of the above concept, we have a component called ChooseCollab which helps in picking the best SHG(s) as collaborator(s). ChooseCollab uses four other components — ChkLocality, ChkTransCost, ChkServ and ChkSpeed — which check the locality, transaction costs, services offered and speed of the SHGs. During and after the completion of collaboration, both the parties will rate each other. The rating is based on timeliness, productivity, communication and commitment. This concept of collaborator rating is implemented using the CollabRating component. CollabRating uses four other components (ChkTime, ChkProd, ChkCommn, and ChkCommit) which will be continuously monitoring the timeliness, productivity, communication and commitment of each collaborator during a collaboration. At the same time, while monitoring the CollabRating of the collaborators, an SHG has to improve its own CollabRating. Hence, an SHG has to improve its timeliness, productivity, communication and commitment. Timeliness, productivity and commitment are improved through physical endeavor. Communication can be improved through sending transparency messages. We implement this idea using a mkCollabRating component which sends timely and informative transparency messages (updates), improving the communication of the SHG.

Figure (fig: 6.8) is the class diagram used for implementing collaborations. SHG class has several methods to facilitate collaborations. CreatTsk method will use the Task class to create a

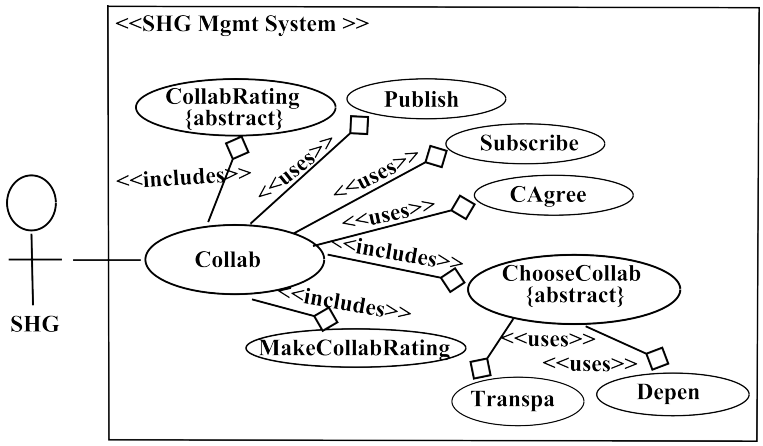


Figure 6.7 Use Case diagram for Collaborations

new task. Publish method will send a task to the coordinator for publication. Subscribe method will send available services/resources to the coordinator for subscription. UnSubscribe method is to cancel the subscribed services/resources at the coordinator. Task class creates a new task with all required specifications. If a task is to be executed using collaborations, it is passed to ColTsk class which adds additional collaboration information to a task, making it a coltsk.

ExTsk class has getData. It obtains all data of a task in execution. TECMonitor observes for changes of variables during task execution. Whenever it finds a change in task status or a considerable progress in task execution such that it needs to be reported, then it will call the Reporting class which sends the report. ExColTsk is derived from ExTsk. This class is to get data during execution of coltsks. It has agreeTsk method which helps in making a collaboration agreement with the collaborator. MonitorColTsk method observes for changes in the coltsk execution. Whenever it finds a change in coltsk's status or a considerable progress in coltsk execution such that it needs to be reported, then it calls Reporting class which sends a report to the notification recipients.

ExColTsk uses FindCollab class to search for probable collaborators. FindCollab has three methods. GetMatchSubs method will search the entire list of subscribers and select all the subscribers whose services match with the required services. SortCollabs method will sort the retrieved list according to SHGs' performances, and also keeps the SHGs with high transparency and dependability values in the top. DisplayDetail method will display the list to the user.

ChooseCollab class is used for making fine grained decisions regarding selection of collaborators from the sorted list. It has seven methods to help in this process. LocalityChk method will verify the distance of the collaborator. Different tasks will have different requirements about collaborator's location. Some tasks require the collaborator to be present in the geographical neighborhood, while for some tasks it does not matter. Some technical

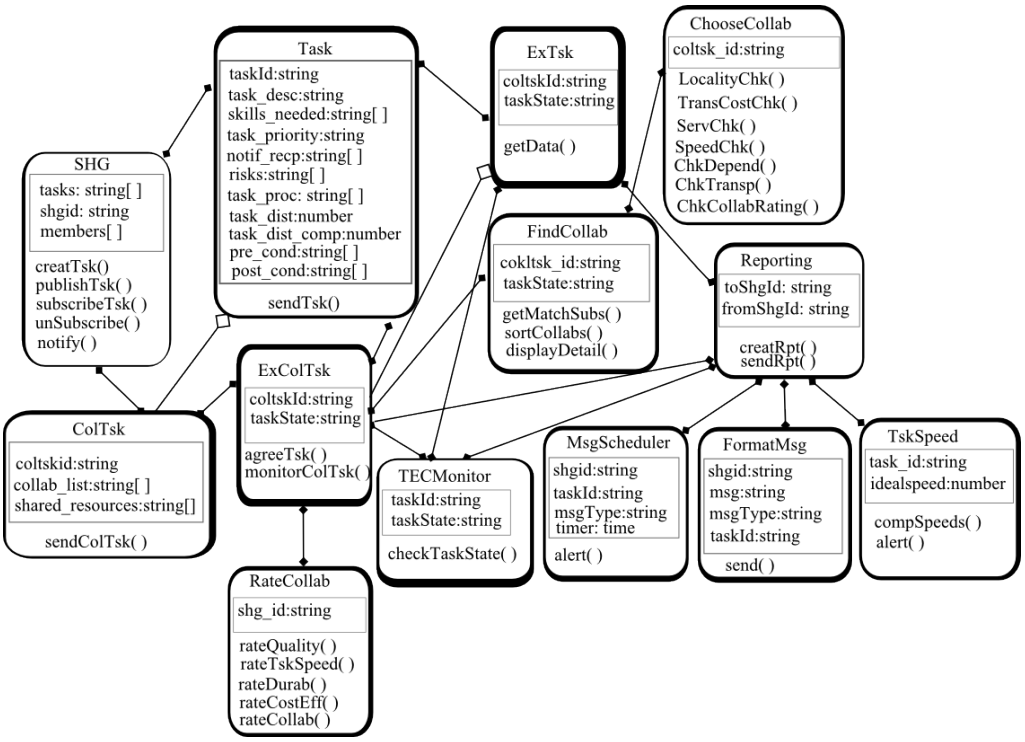


Figure 6.8 Class Diagram for Collaborations

tasks require the collaborator to be located in certain technologically advanced locality. So, this method will verify the requirement of the locality and will prioritizes those SHGs that meet the requirement. TransCostChk method will check the transaction costs and will prioritize the lesser transaction costs collaborators. ServChk method will search for matching services and gives the collaborators preference based on their accuracy of matching. Reporting class has creatRpt method which will create a report about the current status of collaboration and coltsk execution.

MsgScheduler class alerts Reporting class to send reports at specified intervals agreed during coltsk collaboration agreement. MsgFormatter class helps in formatting the messages according to the type of report. TskSpeed class has compSpeeds method to compute the speed of the current task/coltsk and alerts a collaborator if its task execution is lagging. RateCollab class monitors the collaborators' performances and gives the rating accordingly. It has five methods to rate each characteristic of the collaborator. RateQuality method observes the quality of services supplied by the collaborator and rates a collaborator accordingly. RateTskSpeed method measures the speed of task execution of a collaborator. RateDurab method measures the durability of the services supplied by a collaborator. RateCostEff method measures the cost effectiveness of a collaborator. RateCollab method will give a collaboration rating to the collaborator based on the values computed by these four methods.

OCL constraints of Collaborations:

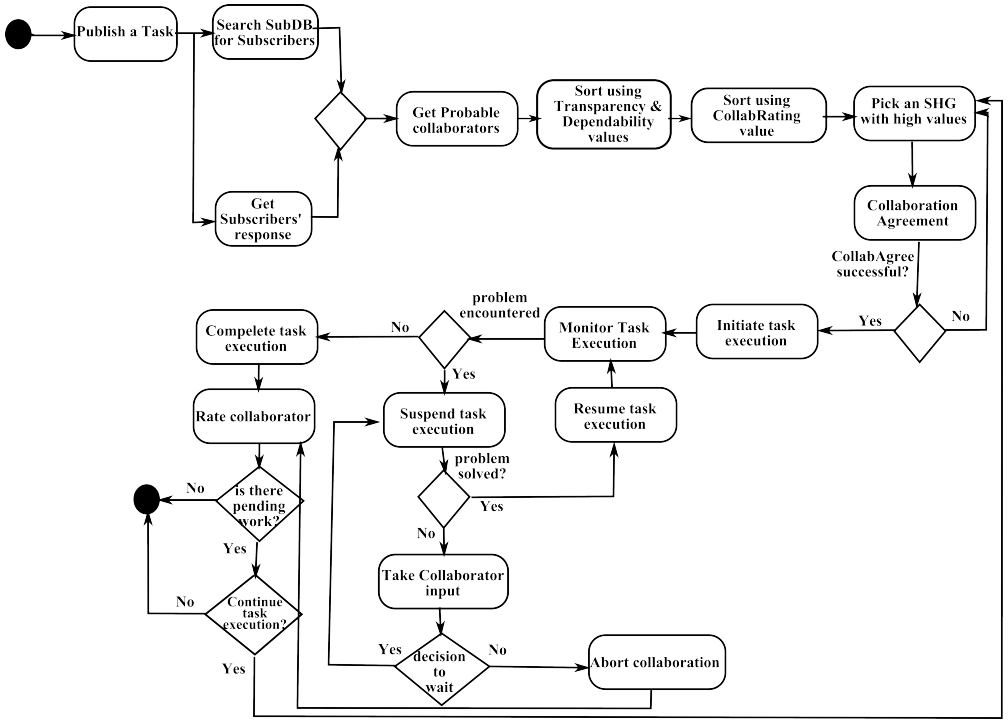


Figure 6.9 Activity diagram for Collaborations

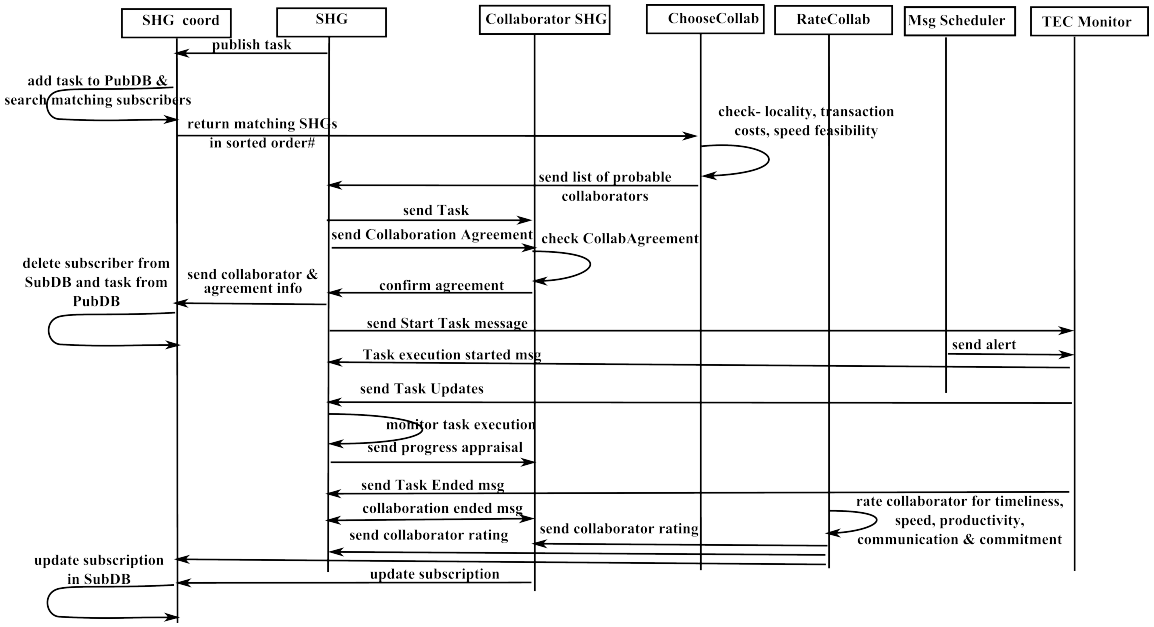


Figure 6.10 Interaction Diagram for Collaborations

1. **context** SHG::publish(task: string):void **inv**:publish

   post: PubDB.pubtable.publications = PubDB.publications@pre + task

   Making a publication adds a task to PubDB.

2. **context** SHG::subscribe(service: string):void **inv**:subscribe

   post: SubDB.Subtable.subscriptions = SubDB.subscriptions@pre + service

   Making a subscription adds a service to SubDB

3. **context** FindCollab :: getMatchSHG(serv): Boolean
   pre: forAll(t: task | (t.req_serv = SubDB.avail_serv))
   For findCollab method to return true for a task, the required services of the task must match with the available services in the SubDB.

4. **context** RateCollab **inv**:CollabRating
   enum metric{low, normal, high}
   m : metric, g: SHG
   g.isGoodCollaborator = true $\Longrightarrow$ g.TranspVal $\rightarrow$ value() = m.high & g.CollabRating = m.high & g.DependVal = m.high
   A good collaborator an SHG has high quality, speed of execution, durability, and cost effectiveness.

5. **context** ColTsk **inv**:coltskstates
   forAll(tsk, coltsk | coltsk.states > tsk.states)
   ColTsk has more states than a normal task.

6. **context** SHGCoord::collab(tsk: string, serv: string):void **inv**: collabsuccess forAll(tsk, serv | collab(tsk, serv) = true $\Longrightarrow$
   (PubDB.publications = PubDB.publications@pre - task) and (SubDB.subscriptions = SubDB.subscriptions@pre - service)
   On successful collaboration over a published task, the published task is removed from the PubDB and the corresponding subscribed service is removed from the SubDB.

7. **context** CollabRating::Collaboration : CollabRating
   derive:
   let table : Set(Tuple(range : Sequence(Real), inference : CollabRating)) =
   Set{
   Tuplerange={-8..0} , inference=collabrating::badcollaborator,
   Tuplerange={1.. 4} , inference=collabrating::goodcollaborator,
   Tuplerange={4..8} , inference=collabrating::bestcollaborator, }
   in table $\rightarrow$any(range$\rightarrow$includes(SHG$\rightarrow$Collaboration())).CollabRating

8. **context**: CollabRating **inv**:valrange

   forAll(g1| -8 <= g1.service.CollabRating→ value <= 8 ;

   -2 <= g1.service.timeliness→ value <= 2 ;

   -2 <= g1.service.productivity→ value <= 2 ;

   -2 <= g1.service.communication→ value <= 2 ;

   -2 <= g1.service.commitment→ value <= 2)

   The above are the ranges of values for attributes of CollabRating.

6.2.4 Dependability

Dependability is discussed in the Dependability chapter. For collaborations to be successful, we need to know dependability values of SHGs before they become collaborators. Dependability is computed from three basic attributes — trust, competency and integrity. Trust is of three types — direct, indirect, and recommender trusts. Direct trust is a resultant of computation based on interactions and experiences with a target SHG. Indirect trust is computed from either feedback, associations, or social impact. Recommender trust is computed using recommendations of experts. Trust is a final result of summation of these three trusts. Competency is of two types — social competency and technical competency. Technical competency gives the technical capability of an SHG to perform a task. Social competency gives the details of an SHG's social behavior and attitude. Training, skill, experience, awareness, resources & equipment, etc. are classified under technical competency. Social relationships/collaborations, social responsibility, social impact, etc. are classified under social competency. Competency is a summation of these technical and social competencies. Integrity is the agreement of the actions and stated values to the governing authority and abiding by them.

In implementation (fig: 6.11), values for the three trust types are computed by three components — CalTrust, CalComp, and CalInt. CalTrust has two sub-components — DirTrust and InDirTrust. DirTrust and InDirTrust calculate the direct and indirect trusts of a target SHG respectively. DirTrust uses four sub-components — CalQual, CalSpeed, CalCostEff, and CalDur — which calculate the quality, execution speed, cost effectiveness and durability values of a target SHG respectively. InDirTrust uses three sub-components — CalFB, CalAssn, and CalSID — to calculate the feedback, association, and social intimacy degree values of a target SHG respectively. CalComp computes the competency of a target SHG for a specific task, as competency values vary from task to task. CalComp has two sub-components — CalTechComp and CalSocComp. CalTechComp computes the technical competency based on training, skill,

experience, awareness, resources & equipment, etc. of a target SHG. CalSocComp computes the social competency of a target SHG. It uses social impact or social intimacy degree to measure the social competency. CalInt computes the integrity of a target SHG. It checks the conformation of an SHG to the rules and regulations framed by the coordinator for the SHGs. It checks for organizational, financial, task and political integrity of a target SHG. CalDepend computes the final dependability value of an SHG.

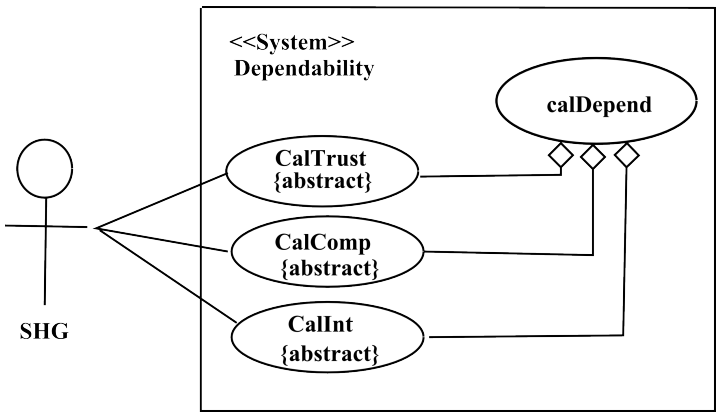


Figure 6.11 Use Case diagram for Dependability

Dependability class diagram (fig: 6.12) explains the classes used in the process of dependability computation in detail. SHG class has three basic methods. The role of giveFB() method is to return a feedback values on the attributes such as cost, quality, transparency, timeliness, response time, trust, etc. of an SHG. The updInfo() method will update the competency and integrity related information. Coordinator class has a method called creatRecom() which will create set of recommenders for each type of services of SHGs. The Recom class organizes the recommenders/experts for a specific service as requested by an SHG. It has one method called recom_shg() which returns a recommendation value for an SHG. AssessDepend class computes dependability of a target SHG with the help of three other classes — AssessTrust, AssessComp and AssessInt. It has one method called compDepend() which aggregates the values provided by those three classes and computes a final dependability value for a target SHG. AssessTrust has three methods that compute values that are required to compute trust of a target SHG. They are — assessDirTrust(), getInDirTrust() and getRecTrust(). AssessDirTust() method analyzes all the previous interactions with a target SHG and provides a value as result. GetInDirTrust() method uses methods such as getting feedback, analyzing associations or social intimacy degree of a target SHG and provides a value as result. GetRecTrust() method accesses Recom class and obtains a recommendation value for a target SHG. AssessTrust also has updTrust() method. This method will track the

interactions that have happened with other SHGs and adds/updates the direct trust value for that SHG. AssessComp class will compute technical and social competencies of a target SHG with the help of two methods — calTechComp() and calSocComp(). AssessInt class computes the integrity of a target SHG. There are four main methods — assessOrgInt(), assessFinInt(), assessTskExecInt() and assessPolInt, which compute organizational, financial, executional and political integrity values of a target SHG.

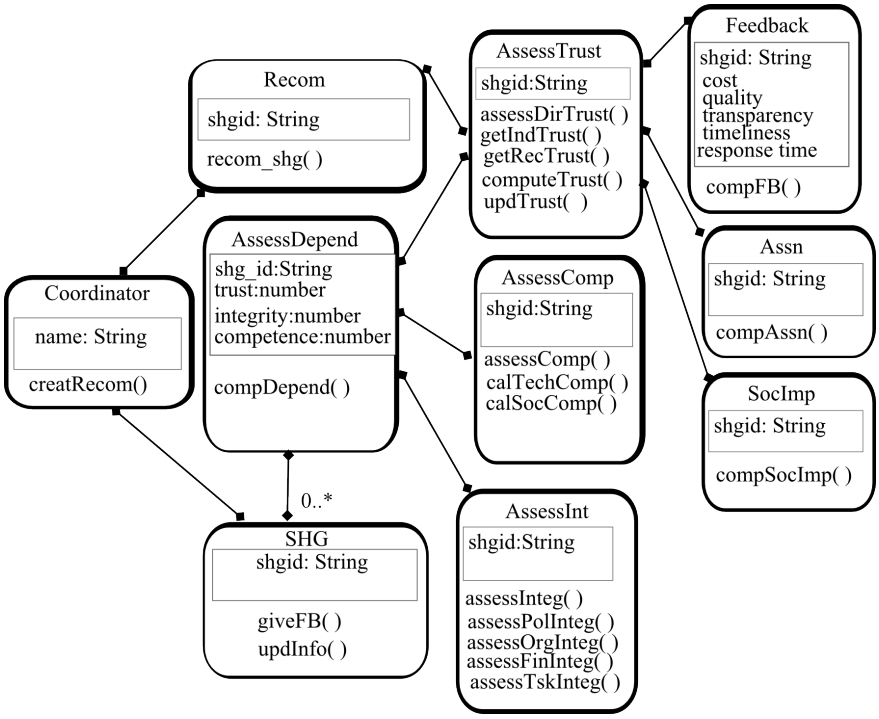


Figure 6.12 Class Diagram for Dependability

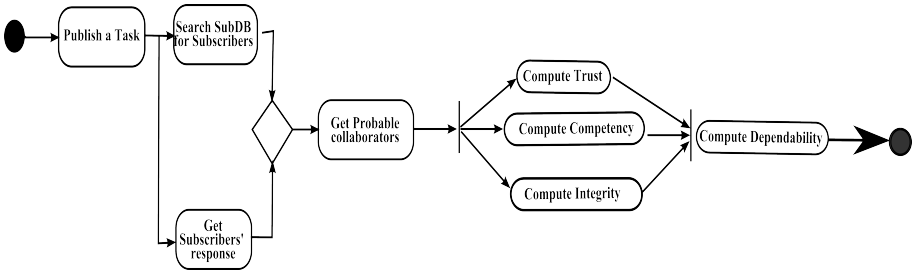


Figure 6.13 Activity diagram for Dependability

OCL constraints for Dependability:

1. **context**: AssessDepend **def**

   let t : Real = service.trust → value()

   let c : Real = service.competency → value()

   let i : Real = service.integrity → value()

   let d : Real = service.dependability → value()

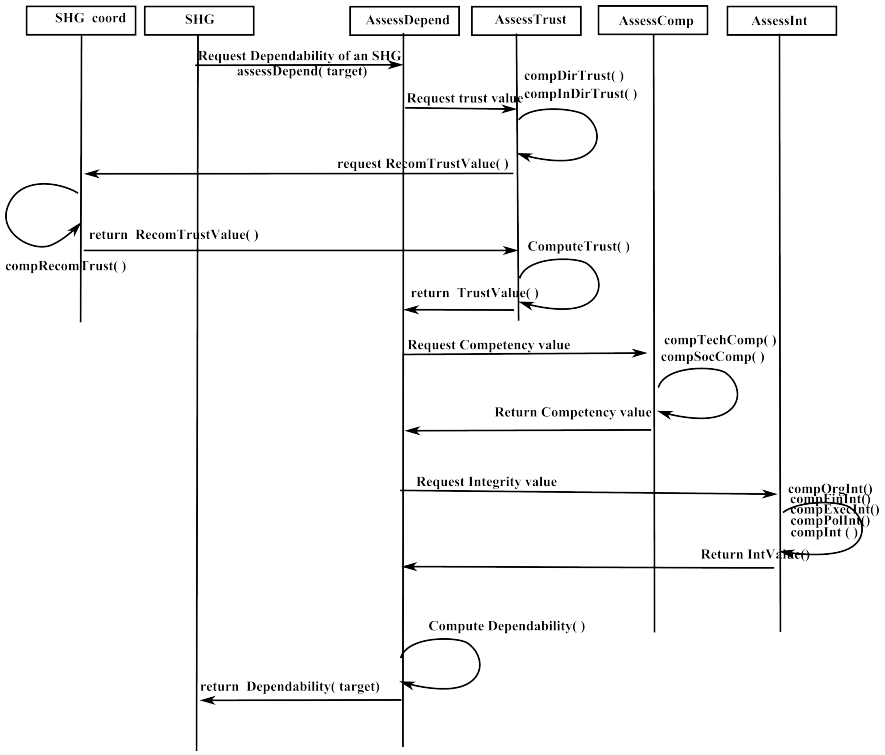


Figure 6.14 Interaction Diagram for Dependability

d = sum(t,c,i)

Dependability is a summation of trust, competency and integrity.

2. **context** AssessDepend::Dependability : DependRating

   derive:

   let table : Set(Tuple(range : Sequence(Real), inference : DependRating)) =

   Set{

   Tuplerange={0 .. 6} , inference=DependRating::notdependable,

   Tuplerange={7.. 12} , inference=DependRating::moderatelydependable,

   Tuplerange={13..16} , inference=DependRating::highlydependable, }

   in table → any(range→includes(SHG→Dependability())).DependRating

3. **AssessInt**: SHG **def**

   let finint : Real = self.financialIntegrity → value()

   let orgint : Real = self.organizationalIntegrity→ value()

   let execint : Real = self.executionalIntegrity → value()

   let polint : Real = self.politicalIntegrity → value()

   let integ : Real = self.integrity → value()

   integ = sum(finint,orgint,execint,polint)

   Integrity of an SHG depends on financial , organizational, executional and political

conformation to federation's/coordinator's standards.

4. **AssessComp**: SHG **def**

   let tc : Real = self.technicalcompetency → value()

   let sc : Real = self.socialcompetency → value()

   let comp : Real = service.competency → value()

   comp = sum(tc,sc)

   Competency of an SHG is a resultant of its technical and social competencies.

5. **AssessComp**: SHG **def**

   let tc : Real = self.technicalcompetency → value()

   let sk : Real = self.skillcompetency → value()

   let tr : Real = self.trainingcompetency → value()

   let exp : Real = self.experiencecompetency → value()

   let res : Real = self.resourcescompetency → value()

   tc = sum(sk,tr,exp,res)

   Technical competency is a resultant of its skills, training, experiernce, resources and equipment.

6. **context**: AssessComp **def**

   let sc : Real = self.socialcompetency → value()

   let sr : Real = self.socialresponsibilitycompetency → value()

   let si : Real = self.socialimpactcompetency → value()

   sc = sum(sr,si)

   Social competency is a resultant of its social responsibility, social impact, etc.

7. **context**: AssessDepend **inv**:valrange

   forAll(g1| 0 <= g1.service.dependability → value <= 15 ;

   The range of dependability value for any service of an SHG is 0 and 16.

   The ranges of other values is as follows:

Now that transparency and dependability processes are represented, we now represent the process of decision making regarding choosing of a collaborator. The collaboration process starts with a publisher publishing a task. Based on the task published and its required services, subscription table is searched and a list of SHGs matching the services are returned. These returned SHGs are sorted based on their $CollabRating$, $Transparency$ and $Dependability$.

From the returned results, the SHG chooses some of them and corresponds with the probable collaborators. If it finds a suitable SHG, it proceeds to make a collaboration agreement. And, if the collaboration agreement is successful, then they become collaborators for the task. If it is not satisfied with any of the resultant SHGs, it will withdraw its task.

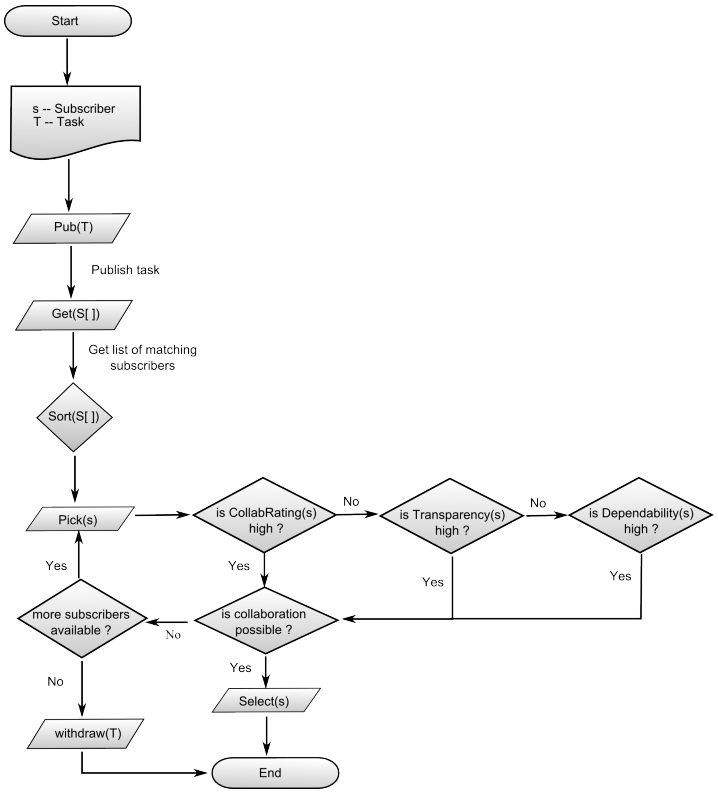


Figure 6.15 Decision about Choosing a Collaborator

Figure 6.16 gives UML Communication diagram of the system. It starts with publication of a task and ends with rating the collaboration and updating the transparency and dependability values of a collaborator SHG. The sequence numbers indicate the order of messages within an interaction.

The class diagram for the complete system prototype is given in fig: 6.17 showing the classes for collaborations, transparency and dependability of the system.

0 <= g1.service.trust→ value <= 6
0 <= g1.service.dirtrust→ value <= 4
0 <= g1.service.indirtrust→ value <= 1
0 <= g1.service.rectrust→ value <= 1
0 <= g1.service.competency→ value <= 5
0 <= g1.service.socrep→ value <= 1
0 <= g1.service.socrep→ value <= 1
0 <= g1.service.execInt→ value <= 1
0 <= g1.service.integrity→ value <= 4
0 <= g1.service.training→ value <= 1
0 <= g1.service.skills→ value <= 1
0 <= g1.service.experience→ value <= 1
0 <= g1.service.awareness→ value <= 1
0 <= g1.service.reseqip→ value <= 1
0 <= g1.service.finIn → value <= 1
0 <= g1.service.orgInt→ value <= 1
0 <= g1.service.polInt→ value <= 1

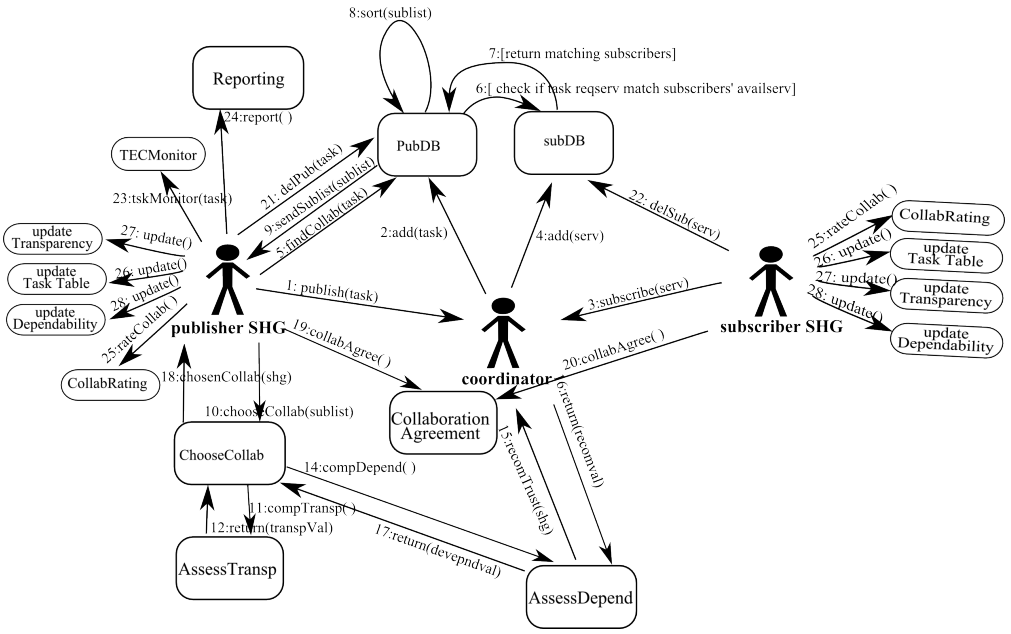


Figure 6.16 Communication Diagram

6.3 System Development

The prototype system was developed on a desktop computer and programming was done in J2EE, using Java Servlets and Java Server Pages; JavaScript is also used, but not extensively. NetBeans is used for programming IDE, while database is supported by MySQL. We also used MySQL Query Browser, a graphical tool provided by MySQL AB for creating, executing, and optimizing queries in a graphical environment.

User Interfaces and Data Sections are presented in the next section. User Interface (UI) section shows how a user can interact with the system through various screens of the system. Data section provides information about the various tables used and their descriptions.

6.3.1 User Interface (UI)

The first screen that a user encounters while accessing the system is the login page. This page checks the user type and determines the information and functionality access. It redirects the users to further screens and content based on the type and identity of the user. If the user is of the type coordinator, funding agency, government, etc., the screen redirects to management pages. Else if the user type is SHG, then it redirects to tasks such as — publishing a task, subscribing to a task, finding collaborators, sending transparency messages, viewing publications, subscriptions, executing tasks, etc.

After SHG login, a user has three options to manage the publications — publish a new task, update an existing task or delete a published task.

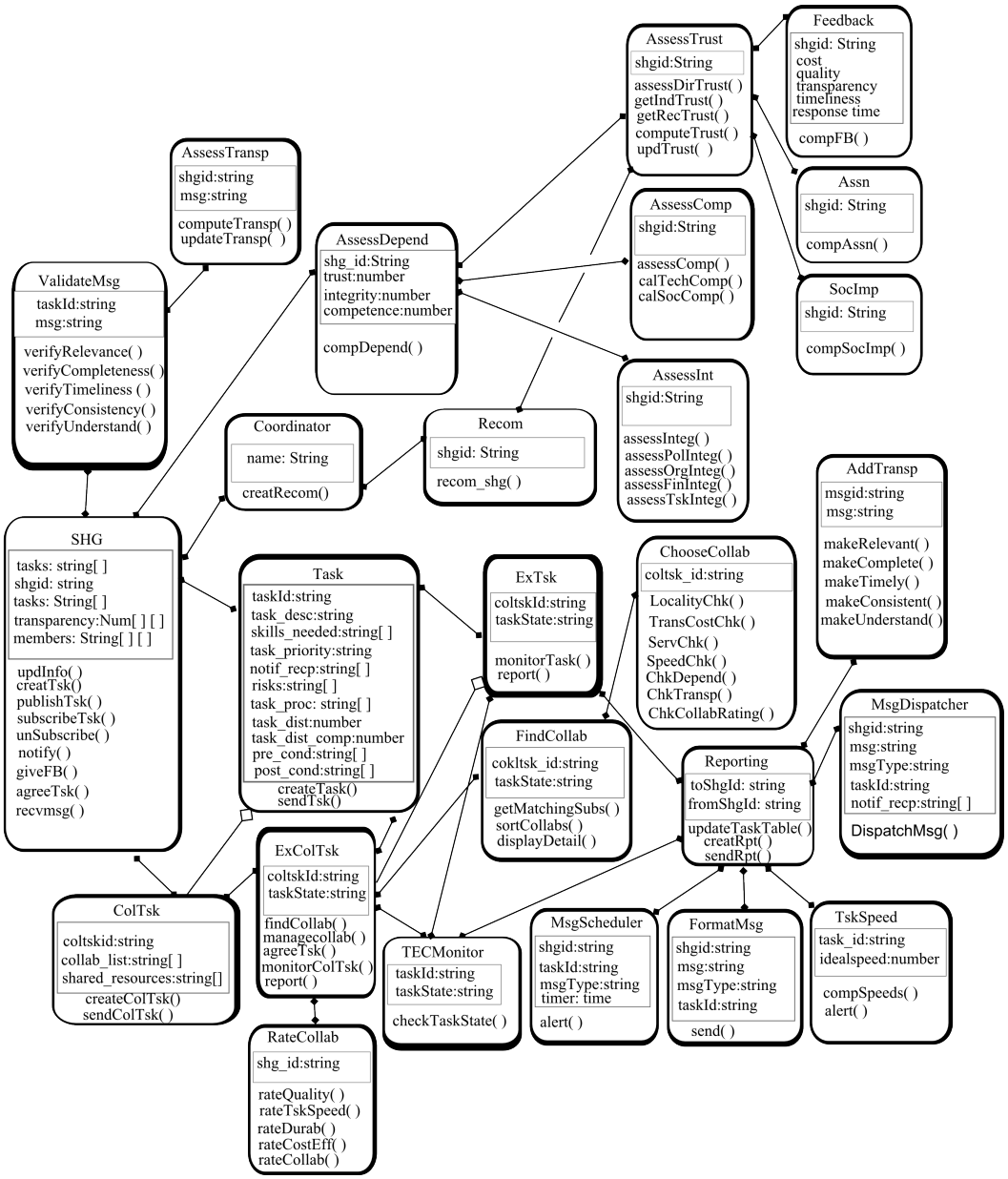


Figure 6.17 Class Diagram of System

If the user chooses to publish a new task, then a new page is opened with a set of empty fields. And the 'Add task' button will add a new task to the PubDB, there by publishing it to the system.

If ever there was a change to the published task, like extension of closing date, change of conditions, change of services needed, etc., update task option will provide for updating an existing task.

Delete task option will provide field for entering task id of an existing task which is to be deleted. When the form is submitted, it deletes the task from the publications table. Deleting of a task is usually done when collaborators are found for a published task, or when an SHG

Figure 6.18 Login Screen

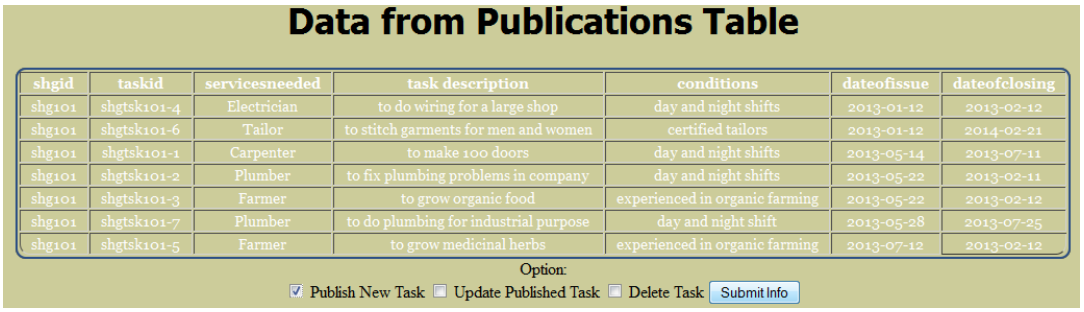


Figure 6.19 Task options for an SHG

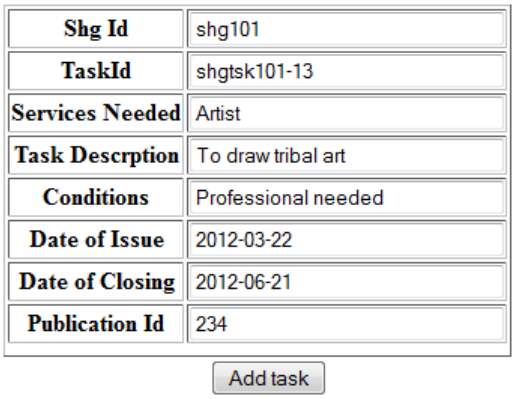


Figure 6.20 Add Task (publication)

withdraws its task.

To manage subscriptions, a user has three options — add subscription, update subscription and delete subscription. Add subscription is for an SHG to add its services that are available for sharing or collaboration. Whenever an SHG subscribes with a service, a new row is added to the Subscriptions table corresponding to the SHG's id.

'Update subscription' and 'delete subscription' are intuitive. 'Update subscription'

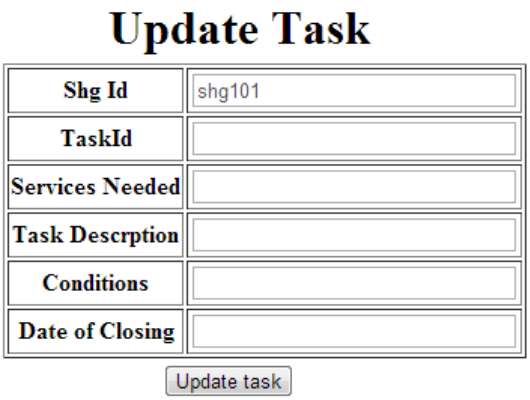


Figure 6.21 Update Task

Figure 6.22 Delete Task from the Publications Table

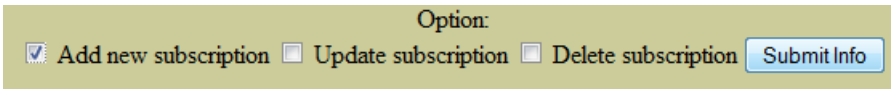


Figure 6.23 Subscription options for SHG

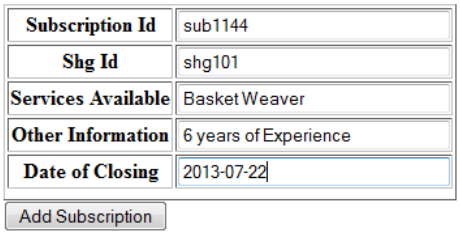


Figure 6.24 Adding Subscription to Subscriptions Table

Figure 6.25 Subscription Added

will update an existing subscription, and 'delete subscription' will delete a subscription corresponding to the $subid$ that is provided.

To execute a $coltsk$, an SHG needs to find other SHGs that are willing to collaborate. These

collaborating SHGs are the ones that are found in the subscriptions table. A $coltsk$ is same as general task except that it is executed with the help of collaborators. Now to find collaborators, a user can use the option 'Get Collaborators For Task'. When this option is taken, it redirects to a new page which takes the $tskid$ and searches the task for the 'services needed' field. It then searches the Subscriptions table for matching subscriptions. If there are matches for the services, it implies that there are some SHGs available for collaboration. It displays a list of probable collaborators. The list does not carry complete information about the resultant SHGs, hence another window is provided to give complete details. Upon clicking the 'Get more info' button (which is specific to each resulting SHG), more information regarding Transparency, Dependability and CollabRating of a specific SHG are displayed. This method of presenting the results allows us for future modeling of our model when more attributes such as serviceability, maintainability, security, safety, etc. are added to the system.

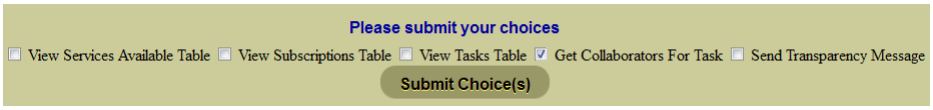


Figure 6.26 Find Collaborators For Task

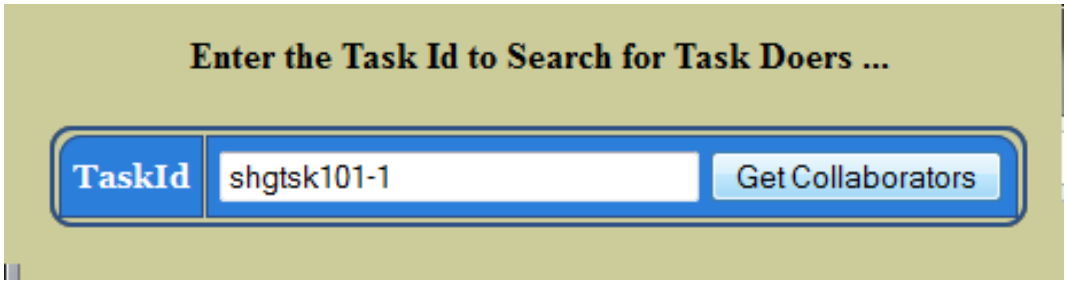
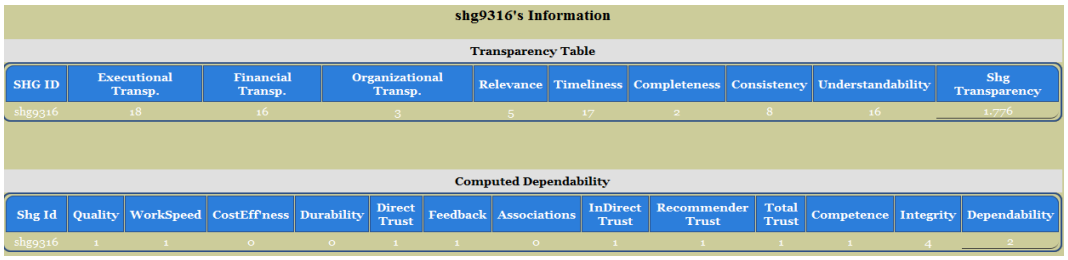


Figure 6.27 Enter Task Id to get Matching Collaborators

Shg Id	Sub Id	Services Available	Closing Date	Other Info	Collab-Index	Depend-Index	Transparency	Get more info
shg9316	sub11042	Carpenter	2014-10-29	14 years of experience	8	11	4.5	shg9316
shg7960	sub15810	Carpenter	2014-11-13	3 years of experience	8	11	3.9	shg7960
shg329	sub8990	Carpenter	2014-4-26	6 years of experience	8	13	3.8	shg329
shg8131	sub5524	Carpenter	2014-6-27	10 years of experience	8	7	2.3	shg8131
shg3016	sub18492	Carpenter	2014-9-6	5 years of experience	7	6	8.7	shg3016

Figure 6.28 List of Matching Collaborators - Result

Transparency is improved by sending transparency messages. Here we have provided a few transparency message formats. The formats of messages are not rigid. We only provided a basic format, and the subsequent screens to the end show how transparency is affected by transparency messages.



shg9316's Information

Transparency Table

SHG ID	Executional Transp.	Financial Transp.	Organizational Transp.	Relevance	Timeliness	Completeness	Consistency	Understandability	Shg Transparency
shg9316	18	16	3	5	17	2	8	16	1.776

Computed Dependability

Shg Id	Quality	WorkSpeed	CostEff'ness	Durability	Direct Trust	Feedback	Associations	InDirect Trust	Recommender Trust	Total Trust	Competence	Integrity	Dependability
shg9316	1	1	0	0	1	1	0	1	1	1	1	4	2

Figure 6.29 Further Details of Collaborator - Result

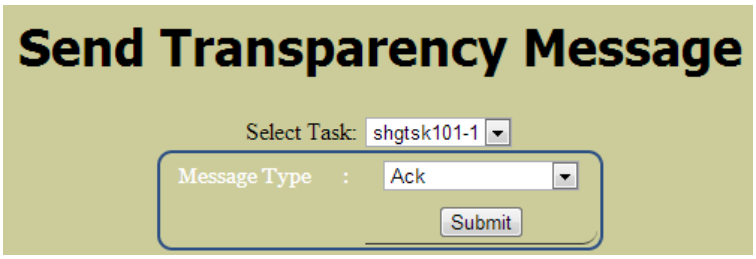


Figure 6.30 Transparency screen 1

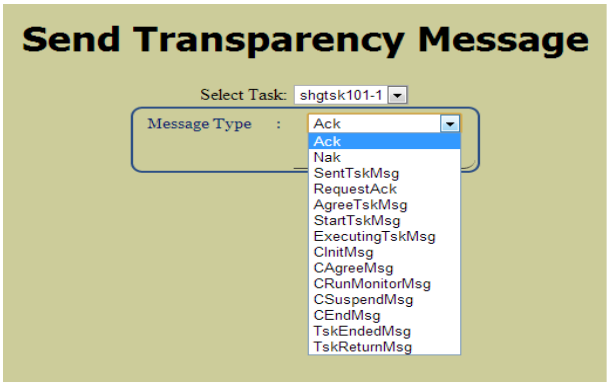


Figure 6.31 Transparency screen 2

Figure 6.32 Transparency screen 3

Figure 6.33 Transparency screen 4

Figure 6.34 Transparency screen 5

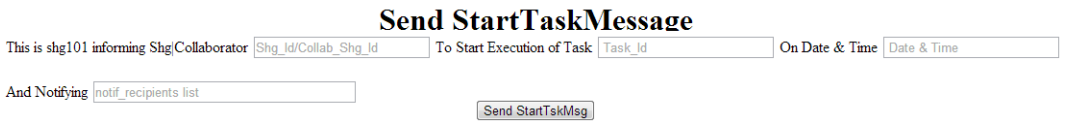


Figure 6.35 Transparency screen 6

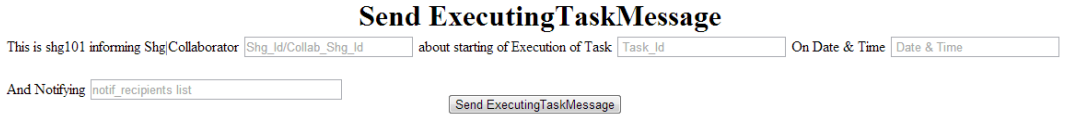


Figure 6.36 Transparency screen 7

Figure 6.37 Transparency screen 8

Figure 6.38 Transparency screen 9

Figure 6.39 Transparency screen 10

Figure 6.40 Transparency screen 11

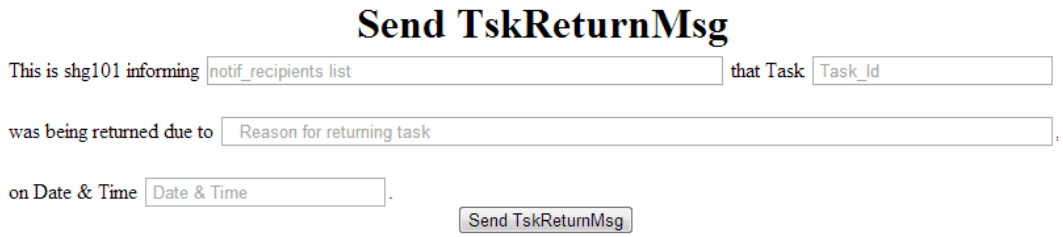


Figure 6.41 Transparency screen 12

Taskid	Shgid	Task Status	Total msgs to send	Total valid msgs	Task Transparency
shgtsk101-2	shg101	CollabInit	100	40	4.0

Figure 6.42 Transparency before sending a transparency message

Transparency Message Sent: New Task Transparency

Taskid	Shgid	Task Status	Total msgs to send	Total valid msgs	Task Transparency
shgtsk101-2	shg101	CollabInit	100	41	4.1

Figure 6.43 Transparency after sending a transparency message

6.3.2 Data Section

This section provides some basic details of data storage used in various tables in the present work. Below (Figure: 6.44) is the Publications table.

shgid	taskid	servicesneeded	conditions	dateofissue	dateofclosing	taskdesc
shg101	shgtsk101-1	Carpenter	Day and Night Shifts	2013-07-24	2013-12-25	Creative work of Garments
shg4716	shgtsk4716-6	Designer	Day and Night shifts	2014-09-15	2014-09-15	Good quality garments with new designs.
shg863	shgtsk863-12	Designer	Day and Night shifts	2014-10-12	2014-10-12	Good quality garments with new designs.
shg9325	shgtsk9325-0	Designer	Day and Night shifts	2014-06-22	2014-06-22	Good quality garments with new designs.
shg2103	shgtsk2103-12	Designer	Day and Night shifts	2014-03-18	2014-03-18	Good quality garments with new designs.
shg9367	shgtsk9367-14	Designer	Day and Night shifts	2014-07-10	2014-07-10	Good quality garments with new designs.
shg5220	shgtsk5220-8	Designer	Day and Night shifts	2014-06-12	2014-06-12	Good quality garments with new designs.
shg4826	shgtsk4826-12	Carpenter	Day and Night shifts	2014-10-29	2014-10-29	Good quality garments with new designs.
shg8822	shgtsk8822-9	Designer	Day and Night shifts	2014-11-21	2014-11-21	Good quality garments with new designs.
shg9573	shgtsk9573-0	Designer	Day and Night shifts	2014-02-26	2014-02-26	Good quality garments with new designs.
shg4441	shgtsk4441-0	Designer	Day and Night shifts	2014-07-26	2014-07-26	Good quality garments with new designs.
shg9668	shgtsk9668-5	Designer	Day and Night shifts	2014-11-29	2014-11-29	Good quality garments with new designs.
shg3404	shgtsk3404-5	Designer	Day and Night shifts	2014-09-11	2014-09-11	Good quality garments with new designs.
shg1716	shgtsk1716-3	Designer	Day and Night shifts	2014-02-26	2014-02-26	Good quality garments with new designs.
shg7592	shgtsk7592-0	Designer	Day and Night shifts	2014-10-16	2014-10-16	Good quality garments with new designs.
shg2741	shgtsk2741-4	Designer	Day and Night shifts	2014-04-21	2014-04-21	Good quality garments with new designs.
shg2745	shgtsk2745-13	Designer	Day and Night shifts	2013-12-11	2013-12-11	Good quality garments with new designs.
shg9482	shgtsk9482-13	Designer	Day and Night shifts	2014-04-13	2014-04-13	Good quality garments with new designs.
shg9806	shgtsk9806-12	Designer	Day and Night shifts	2014-06-02	2014-06-02	Good quality garments with new designs.
shg6825	shgtsk6825-4	Designer	Day and Night shifts	2014-01-31	2014-01-31	Good quality garments with new designs.
shg4209	shgtsk4209-1	Designer	Day and Night shifts	2014-02-22	2014-02-22	Good quality garments with new designs.
shg6846	shgtsk6846-14	Designer	Day and Night shifts	2014-07-01	2014-07-01	Good quality garments with new designs.
shg7774	shgtsk7774-10	Designer	Day and Night shifts	2014-03-04	2014-03-04	Good quality garments with new designs.
shg7908	shgtsk7908-9	Designer	Day and Night shifts	2014-02-18	2014-02-18	Good quality garments with new designs.
shg7052	shgtsk7052-3	Designer	Day and Night shifts	2013-12-02	2013-12-02	Good quality garments with new designs.
shg1389	shgtsk1389-7	Designer	Day and Night shifts	2014-10-15	2014-10-15	Good quality garments with new designs.
shg239	shgtsk239-2	Designer	Day and Night shifts	2014-08-07	2014-08-07	Good quality garments with new designs.
shg6741	shgtsk6741-9	Designer	Day and Night shifts	2014-04-13	2014-04-13	Good quality garments with new designs.
shg6834	shgtsk6834-9	Designer	Day and Night shifts	2014-04-23	2014-04-23	Good quality garments with new designs.
shg3572	shgtsk3572-11	Designer	Day and Night shifts	2014-05-13	2014-05-13	Good quality garments with new designs.
shg4784	shgtsk4784-8	Designer	Day and Night shifts	2014-01-25	2014-01-25	Good quality garments with new designs.
shg82	shgtsk82-6	Designer	Day and Night shifts	2014-10-26	2014-10-26	Good quality garments with new designs.
shg23	shgtsk23-11	Designer	Day and Night shifts	2014-02-24	2014-02-24	Good quality garments with new designs.

Figure 6.44 Publications Table

Publications table is managed by SHG coordinator. It holds the information of all the published tasks. Each row in the table represents a unique publication. An SHG can have several publications, which means a unique row for each publication, and, perhaps, several rows for an SHG. Each task has a specific $services\_needed$ field which is the basis for collaborations. There is a $pubid$ (publication identification number) which is unique to each publication. A new row is added whenever the coordinator receives a new publication, and the corresponding row is deleted whenever a collaboration agreement succeeds for that publication or a publisher withdraws its publication.

shgid	servicesavailable	closingdate	otherinfo	subid	collabind	dependind	transparency
shg7824	Farmer	2014-1-14	Trained and 7 years of experience.	sub10063	6	15	7.06445
shg3281	Farmer	2014-5-28	Trained and 5 years of experience.	sub10079	-1	13	4.18481
shg6343	Tailor	2014-0-24	Trained, and has 8 years of experience.	sub10107	0	6	9.73069
shg9314	Builder	2014-11-19	Certified by Govt.6 years of experience.	sub10119	-7	5	6.09904
shg695	Painter	2014-4-25	Has 8 years of experience.	sub1012	8	7	1.30314
shg7364	Electrician	2014-7-27	Certified. 12 years of experience.	sub10169	1	3	8.21856
shg2346	Cook	2014-5-12	Trained, and has 7 years of experience.	sub10210	7	9	7.18338
shg6046	Electrician	2014-2-6	Certified. 7 years of experience.	sub10244	7	5	1.26122
shg9771	Painter	2014-2-12	Has 6 years of experience.	sub10341	6	15	4.75598
shg7326	Farmer	2014-10-9	Trained and 1 years of experience.	sub10385	-1	11	9.99624
shg2783	Builder	2014-1-6	Certified by Govt. 4 years of experience.	sub10393	2	11	5.71323
shg1593	Carpenter	2014-5-0	14 years of experience	sub10414	3	7	8.57746
shg1326	Plumber	2014-7-23	0 years of experience	sub10452	0	15	5.74759
shg162	Plumber	2014-5-1	5 years of experience	sub10495	0	8	3.00558
shg6041	Carpenter	2014-6-7	8 years of experience	sub10533	6	9	7.78514
shg5133	Cook	2014-8-16	Trained, and has 3 years of experience.	sub10538	7	5	9.90894
shg2584	Painter	2014-10-4	Has 5 years of experience.	sub10614	6	15	6.18927
shg6734	Plumber	2014-3-3	2 years of experience	sub10689	2	15	1.21953
shg1067	Builder	2014-9-21	Certified by Govt.9 years of experience.	sub10748	-2	11	7.52987
shg2751	Tailor	2014-5-29	Trained, and has 12 years of experience.	sub10865	-6	13	3.99074
shg4744	Carpenter	2014-1-3	9 years of experience	sub10968	0	13	7.3641
shg7526	Tailor	2014-9-17	Trained, and has 13 years of experience.	sub10987	8	13	4.84827
shg2007	Cook	2014-1-9	Trained, and has 3 years of experience.	sub11	0	9	2.14907
shg9316	Carpenter	2014-10-29	14 years of experience	sub11042	8	11	4.55542
shg8163	Electrician	2014-4-16	Certified. 5 years of experience.	sub11045	1	11	4.17554
shg8706	Farmer	2014-2-20	Trained and 10 years of experience.	sub11071	-1	8	3.53044
shg6270	Electrician	2014-11-13	Certified. 3 years of experience.	sub11080	-2	10	6.01798
shg2357	Plumber	2014-11-5	0 years of experience	sub11090	0	10	9.49858
shg846	Painter	2014-11-24	Has 11 years of experience.	sub11124	-2	5	9.43896
shg101	Basket Weaver	2013-05-05	5 years of Experience	sub11134	-5	10	8.69906
shg7462	Painter	2014-0-16	Has 2 years of experience.	sub11145	8	4	5.1784
shg2351	Tailor	2014-7-19	Trained, and has 4 years of experience.	sub11148	-7	3	3.43896
shg8047	Carpenter	2014-11-22	2 years of experience	sub11179	1	9	7.81029
shg9630	Farmer	2014-0-23	Trained and 8 years of experience.	sub11213	3	6	8.73459
shg8668	Embroider	2014-11-4	Trained, and has 1 years of experience.	sub1123	4	4	0.242093

Figure 6.45 Subscriptions Table

taskid	shgid	taskstatus	collab	sharedservices	supervisors	notifrecp
shgtsk101-1	shg101	CRunMonitor	shg105, shg221, shg129, shg240, shg321	Carpenter	shg109	coord, shg112,shg431, shg323, shg332, sh...
shgtsk101-2	shg101	CollabInit	shg209, shg221, shg129, shg240, shg321	Plumber	shg554	coord,shg222, shg665, shg778, shg662,sh...
shgtsk101-3	shg101	CSuspend	shg221, shg129, shg240, shg321	Electrician	shg221, shg129, shg240, shg321	shg221, shg129, shg240, shg321,shg554, ...
shgtsk101-4	shg101	CollabInit	shg231, shg429, shg140, shg221	Cook	shg344, shg343, shg232, shg332	shg344, shg343, shg232, shg332,shg221, ...
shgtsk101-5	shg101	CAgree	shg621, shg149, shg280, shg301	Farmer	shg321, shg146, shg432, shg332	shg321, shg146, shg432, shg332,shg344, ...
shgtsk101-6	shg101	CSuspend	shg121, shg629, shg220, shg121	Weaver	shg243, shg432, shg21`, shg267	shg243, shg432, shg21`, shg267,shg321, ...
shgtsk101-7	shg101	CollabInit	shg721, shg629, shg440, shg331	Tailor	shg431, shg323, shg332, shg325	shg431, shg323, shg332, shg325,shg243, ...
shgtsk101-8	shg101	CEnd	shg541, shg143, shg120, shg561	Gardener	shg222, shg665, shg778, shg662	shg222, shg665, shg778, shg662,shg431, ...
shgtsk101-9	shg101	CollabInit	shg112, shg324, shg254, shg553	Painter	shg554, shg314, shg531, shg326	shg554, shg314, shg531, shg326,shg222, ...
shgtsk102-1	shg102	CSuspend	shg559	Plumber	shg554	coord
shgtsk102-2	shg102	CEnd	shg559	Farmer	shg554	coord
shgtsk102-3	shg102	CollabInit	shg559	Weaver	shg554	coord
shgtsk103-1	shg103	CSuspend	shg209	Electrician	shg554	coord
shgtsk103-2	shg103	CEnd	shg559	Farmer	shg554	coord

Figure 6.46 Task Table

Subscription table (Figure: 6.45) is managed by SHG coordinator. It contains the information of all subscriptions made by subscribing SHGs. Each row in the table represents a

unique subscription of an SHG. An SHG may subscribe with many services. But each service type has only one row, which means a subscriber can have multiple rows if it subscribes with multiple service types. Each subscription has a unique id ($subid$). A new row is added every time the coordinator receives a new subscription, and the corresponding row is deleted once a collaboration agreement succeeds for that subscription or SHG unsubscribes.

The task table (Figure: 6.46) contains complete information about an SHG's tasks. It is maintained at the coordinator. It specifies task state, collaborators, notification recipients, and other information of tasks. Each row in the task table uniquely corresponds to a task. Each task is identified by a unique id. Also, respective task tables are managed by SHGs locally. Each SHG can modify its own task table and also update status of tasks from time to time, and the same is reflected at the coordinator's table.

shgid	quality	workspeed	costeff	dura	direct	feedback	assoc	indirect	recom	trust	comp	integ	depend
shg101	1	1	0	1	1	1	1	1	1	1.0	2	3	2.0
shg1025	0	0	1	0	0	0	0	0	0	0.0	5	3	2.66667
shg1058	0	0	1	0	0	0	0	0	0	0.0	2	1	1.0
shg1067	0	0	1	0	0	0	0	0	0	0.0	4	2	2.0
shg1078	1	1	0	1	1	1	1	1	1	1.0	3	2	2.0
shg109	1	1	0	0	1	1	0	1	1	1.0	3	1	1.66667
shg1091	1	1	0	0	1	1	0	1	1	1.0	1	2	1.33333
shg1092	1	1	0	1	1	1	1	1	1	1.0	4	3	2.66667
shg111	0	0	1	0	0	0	0	0	0	0.0	5	2	2.33333
shg1128	1	1	0	0	1	1	0	1	1	1.0	2	2	1.66667
shg113	1	1	0	1	1	1	1	1	1	1.0	1	1	1.0
shg1147	1	1	0	1	1	1	1	1	1	1.0	5	2	2.66667
shg1152	0	0	1	0	0	0	0	0	0	0.0	3	4	2.33333
shg1175	0	0	1	0	0	0	0	0	0	0.0	1	4	1.66667
shg1181	0	0	1	0	0	0	0	0	0	0.0	6	2	2.66667
shg1217	1	1	0	0	1	1	0	1	1	1.0	3	4	2.66667
shg1238	0	0	1	0	0	0	0	0	0	0.0	2	3	1.66667
shg1239	1	1	0	1	1	1	1	1	1	1.0	3	3	2.33333
shg1286	0	0	1	0	0	0	0	0	0	0.0	6	3	3.0
shg1304	0	0	0	0	0	1	0	1	1	0.666...	4	3	2.55556

Figure 6.47 Dependability Table

shgid	exectransp	fintransp	orgtransp	relevance	timeliness	completeness	consistency	understandability	shgtransparency
shg101	22.0	30.0	2.0	20.0	7.0	13.0	4.0	5.0	2.646
shg1025	23.0	6.0	5.0	14.0	5.0	16.0	15.0	12.0	2.108
shg1058	18.0	28.0	19.0	7.0	3.0	3.0	4.0	4.0	1.365
shg1067	23.0	3.0	18.0	16.0	20.0	3.0	12.0	4.0	2.42
shg1078	29.0	20.0	5.0	16.0	8.0	9.0	10.0	6.0	2.646
shg109	14.0	3.0	1.0	14.0	3.0	13.0	11.0	1.0	0.756
shg1091	16.0	12.0	20.0	1.0	9.0	17.0	7.0	3.0	1.776
shg1092	5.0	20.0	5.0	2.0	14.0	6.0	19.0	14.0	1.65
shg111	7.0	5.0	24.0	5.0	2.0	19.0	15.0	18.0	2.124
shg1128	19.0	22.0	14.0	19.0	8.0	17.0	17.0	8.0	3.795
shg113	14.0	6.0	29.0	20.0	14.0	7.0	18.0	6.0	3.185
shg1147	13.0	23.0	13.0	4.0	5.0	4.0	20.0	4.0	1.813
shg1152	20.0	5.0	9.0	20.0	5.0	20.0	5.0	4.0	1.836
shg1175	3.0	16.0	3.0	5.0	7.0	6.0	3.0	6.0	0.594
shg1181	13.0	7.0	19.0	7.0	18.0	11.0	20.0	19.0	2.925
shg1217	24.0	14.0	26.0	18.0	12.0	18.0	9.0	15.0	4.608
shg1238	24.0	21.0	13.0	10.0	4.0	13.0	5.0	16.0	2.784
shg1239	15.0	2.0	17.0	13.0	2.0	14.0	19.0	18.0	2.244
shg1286	5.0	6.0	14.0	18.0	1.0	8.0	18.0	18.0	1.575
shg1304	9.0	21.0	17.0	8.0	16.0	16.0	12.0	18.0	3.29
shg1326	29.0	30.0	13.0	6.0	15.0	17.0	6.0	14.0	4.176
shg1334	26.0	25.0	13.0	4.0	8.0	17.0	13.0	14.0	3.584
shg1371	12.0	4.0	27.0	2.0	13.0	13.0	7.0	10.0	1.935
shg1381	14.0	4.0	4.0	20.0	2.0	12.0	16.0	6.0	1.232
shg1424	4.0	7.0	30.0	12.0	8.0	1.0	16.0	20.0	2.337
shg1436	8.0	21.0	19.0	20.0	17.0	7.0	15.0	2.0	2.928
shg1437	25.0	26.0	3.0	3.0	19.0	12.0	4.0	8.0	2.484
shg145	12.0	6.0	19.0	16.0	2.0	20.0	17.0	13.0	2.516
shg1468	14.0	12.0	24.0	9.0	14.0	2.0	11.0	19.0	2.75
shg1486	5.0	12.0	4.0	16.0	4.0	9.0	5.0	18.0	1.092
shg1502	10.0	21.0	8.0	14.0	15.0	20.0	13.0	13.0	2.925
shg1551	5.0	9.0	27.0	19.0	5.0	10.0	6.0	11.0	2.091
shg1553	27.0	13.0	22.0	15.0	17.0	12.0	13.0	15.0	4.464
shg1563	28.0	8.0	26.0	18.0	9.0	7.0	8.0	2.0	2.728
shg1576	29.0	1.0	5.0	5.0	16.0	19.0	18.0	3.0	2.135

Figure 6.48 Transparency Table

Dependability table (Figure: 6.47) is managed by SHGs individually. An SHG computes dependability value for various other SHGs which interact with it. Dependability value for a target SHG is not instantly available since recommendation value is to be obtained from the recommenders and indirect trust value is to be obtained from the associated SHGs of a target SHG. An SHG updates direct trust, competence and integrity values of a target SHG during its interactions. And when final dependability value is to be computed, it obtains the recommender trust and indirect trust values and then computes the final dependability value.

Transparency table (Figure: 6.48) is also managed by SHGs individually. Transparency value is sensitive to transparency messages. Whenever an SHG receives a transparency message, it checks the relevance, completeness, consistency, timeliness and understandability of the message, and updates the task table accordingly. Based on the transparency message, it will also update the executional, financial and organizational transparency values accordingly. Using all the values, it will compute the final transparency value for a target SHG. Each row in the table corresponds to one interacting/ interacted SHG.

Conclusion

This chapter discussed implementation details of the proposed system. A prototype system was developed and the Class, Use Case, Activity and Interaction diagrams of the proposed system are detailed. A Communication diagram is also provided showing the communication pattern for basic components and activities of the system, like, collaboration, transparency and dependability. The constraints of the system are given using OCL. User interface section detailed the UI by providing the screens and Data section provided various tables used by the SHGs and coordinator.

Chapter 7

Conclusions, Summary and Future Research

This dissertation set out to develop a framework for Self Help Groups [SHGs]. It identified and studied some fundamental entities that affected the sustainability of SHGs. This final chapter reviews the research contributions of this dissertation, and discusses directions for future research.

This thesis identified three features — collaboration, dependability and transparency, as the basis for sustainability of SHGs and delved into details on their computability and demonstrated their implementation in SHG management systems.

The first chapter outlined the significance of SHGs, their prevalence and organizational structure. It cited statistics to establish the facts. It is understood that government is interested in investing into SHGs and wants to make them sustainable.

The second chapter presented a mention on work available on making of SHG system socially concerned. Lack of guidance from the literature on SHGs in Computer Science posed a challenge and it required us to create a framework from root level. Due to the same reason, literature survey could not detail much about previous frameworks, and so it included similar works carried out in other domains like multi-agent systems, distributed systems, and e-commerce systems.

It was identified that forming collaborations is the key to sustainability, making SHGs less dependent on funding agencies for funds. It was also observed that with the help of collaborations, SHGs can create business by networking with buyers and sellers. Collaborations enable SHGs to execute larger projects efficiently which, otherwise, are impossible at individual SHG level. A collaboration process is recursive and develops in stages. It enables sharing of

skills and knowledge, and strengthens an SHG. Also, collaboration can be a make or break component of a business. SHGs that work together in collaboration must be compatible with each other, and be willing to accept each other while working in unity as a single group. Differences among groups could make it difficult for collaboration to work. An SHG which prioritizes transparency in its behavior cannot work with another SHG which is not transparent or does not prioritize transparency. Also, an SHG could not rely on another SHG which is not dependable in its performance in business. If features such as transparency and dependability are found in a group of SHGs working together, collaboration could become an ideal concept. Collaboration provides for economic development by making task execution faster and cost effective. When SHGs collaborate, they learn new skills from each other. Collaboration also enhances brain storming by facilitating more thinking, channeling better ideas and concepts about quality and speed of task completion.

The third chapter discussed collaboration, defining it as a match between expertise of an SHG to service/skill required to perform a task. For matching or initiation of collaboration, the use of publish-subscribe model was proposed and the functionality of the model is discussed. For facilitating a match, both specifications of tasks and SHGs are worked. Factors that influence selection of collaborators are identified. The structures of collaborations based on the types of compositions (like parallel and sequential) subtasks of the task are also studied. A representation for collaborations is also proposed. First a basic representation was created, and limitations were studied. Then an enhanced model to represent all the features of collaborations is designed. It was found that the representation is feasible and is able to contain all the details of proposed collaboration specification. In addition to the matching of skills, it was found that there are a few other factors which influence the choice of a collaborator. They are — locality, transaction costs, execution speed, etc. Also, detail of the execution of collaborated task and all its states was given. It was found that almost all states are listed, and that it suffices for most scenarios. Also some of the risks that are involved in collaborations are also identified. However, this study about risks is not exhaustive. It can be studied in-depth when the system is used real life and then more rules may be formulated. Finally, a metric to measure the $CollabRating$ of a collaborator was devised. $CollabRating$ is measured using factors such as timeliness and speed, productivity, communication, and commitment. In future, it might also be needed to check in real life if there will be some more factors that affect $CollabRating$. It was found that after resolving to collaborate, an SHG needs to find a transparent collaborator and that for collaborations to be successful, transparency is a very essential component. An SHG prefers to partner with another SHG which is transparent.

The fourth chapter discussed several aspects of transparency in SHGs. In the proposed model, transparency was achieved through transparency messages. This chapter defined various transparency messages for each state of a task. It also defined the various states of a task which are generic and common to all the tasks. Transparency was defined for three aspects of SHG's behavior — executional, financial and organizational. Transparency messages were defined as validated update messages. Each transparency message was validated based on five properties — relevance, timeliness, completeness, consistency and understandability. Task Execution Cycle [TEC] was defined to identify every state of a task in execution. The SHG Behavioral Model [SBM] defined various processes of an SHG in its task execution and made it easier to define transparency. Transparency was defined for each and every process of the SBM. Also task specification was given in BNF. Tasks have a predefined '*state:context:action*' message condition. So that when a *state* is reached during task execution, and a *context* of the state is encountered, then the system should generate a specific transparency message, thus ensuring the transparency in a task execution. Also the process of reporting of transparency messages detailing the different processes needed was described. An algorithm was designed based on the process of reporting, also a clear example demonstrating how transparency can be computed was detailed. After identifying 'transparent' SHGs, the next most important question that we encountered was, *'which SHGs are dependable for a collaboration'?* Though $CollabRating$ in the Collaborations chapter provided a value for the performance of an SHG as a collaborator, it could not effectively explain the present capability of that SHG for an imminent collaboration. $CollabRating$ only showed a target SHG's behavior during collaborations.

The fifth chapter discussed dependability of SHGs. It was reasoned that an SHG can be counted as dependable for collaborations if it can be 'trusted' in its present position along with 'competence' for the task and 'integrity' in overall performance and behavior. Having agreed that, each of the components were defined and described. Trust comprised of direct trust (which is a value computed based on interactions with the target SHG), indirect trust (which is computed based on other SHGs which are associated with the target SHG) and recommender trust (which is a value given by a body of experts of the SHG system). Each type of trust was explained with a simple example. The components of each type of trust were also explained. Direct trust was computed from attributes such as quality, speed of completion, cost effectiveness and durability. Indirect trust was computed using one of the behavioral attributes such as feedback, associations, or social intimacy degree. Recommender trust was computed using recommendations of experts of federation. Competency and integrity were defined and processes of quantifying them were provided. Competency was defined as a sum of

two types of competencies: technical and social. Technical competency implied the competency acquired by an SHG based on skills, training, experience, resources & equipment, etc. Social competence was computed from social impact of an SHG. Integrity assessment was done using four aspects: political, financial, organizational, and executional. Also computation details for each of the attributes of direct trust, indirect trust, recommender trust, competency and integrity were provided. Then, a dependability metric was proposed which measured the dependability of an SHG on a scale of 0 to 15. SHGs with a rating of 0 - 5 were declared not dependable, while 5-10 were moderately dependable and 10 - 15 were highly dependable. Thus, it can be understood that attainment of sustainability is SHG system can be made possible through collaborations supported by transparency and dependability attributes. There could be several other factors which could directly or indirectly affect the sustainability of SHGs, and these factors can be easily incorporated into the framework.

Sixth chapter discussed implementation details of the prototype system. Application logic of the system was described using Class, Use Case, Activity, and Interaction diagrams of various modules. Also a Communication diagram was given specifying how a collaboration is started and established and the progress of task in execution is monitored. The chapter also detailed some of the constraints of the proposed model using OCL.

The entities discussed in this dissertation provide a natural guide to future research. In this research only collaborations, transparency and dependability were considered. Further, there could be more entities that could affect the sustainability of SHGs. Much research also remains to be done on features that may have high impact on the sustainability of SHGs in real life. A practical study and extensive survey needs to be done to evaluate other important factors of SHG sustainability.

Bibliography

[1] K. Karmakar *et al.*, "Nabard report 2011 - 2012: Progress of shg-bank linkage in india and status of microfinance in india," 2012.

[2] "Indian census," 2011.

[3] IndiaSpend, "India's villages grow larger, smaller ones disappear." `AccessURL:http://www.indiaspend.com/investigations/indias-villages-grow-larger-smaller-ones-disappear`, Oct 2011.

[4] M. Ravallion, "On the urbanization of poverty," *Journal of Development Economics*, vol. 68, no. 2, pp. 435–442, 2002.

[5] B. Madan, "Poverty in india-diagnosis and treatment," *Economic Development Planning and Policy in India: Poverty and development*, vol. 5, p. 169, 1989.

[6] VOICE, "A report on the success and failure of self help groups in india - impediments and paradigm of success," *Planning Commission , Government of India , New Delhi*, 2008.

[7] U. Narang, "Impact of self-help groups bank linkage programme in india," *Trade & Commerce*, 2012.

[8] A. Tankha, "Self-help groups as financial intermediaries in india: Cost of promotion, sustainability and impact," *New Delhi: Sa-Dhan, study prepared for ICCO and Cordaid, The Netherlands*, 2002.

[9] J. V. Moholkar, "Growth and development of shg-bank linkage programme in india," *International Journal of Advance ISSN 2347 7075 and Applied Research (IJAAR) Vol. 1 No.2*, 2013.

[10] S. K. Das and A. Bhowal, "Quality assessment parameters of self help groupâĂŹs: a psychometrics analysis on stakeholdersâĂŹ perception," *Journal of Finance and Economics*, vol. 1, no. 4, pp. 69–83, 2013.

[11] K. Venkatalakshmi and N. Ambujam, "Information and communication technology use frameworks among self-help group women," *European Journal of Social Sciences*, vol. 27, no. 2, pp. 206–212, 2012.

[12] F. Sinha *et al.*, "Self help groups in india: A study of the lights and shades," *Noida and Hyderabad: APMAS and EDA Rural Systems*, 2006.

[13] S. K. Mehta, H. G. Mishra, and M. A. Singh, "Role of self help groups in socio-economic change of vulnerable poor of jammu region," *International Proceedings of Economics Development and Research*, vol. 4, pp. 519–523, 2011.

[14] B. E. Coleman, "The impact of group lending in northeast thailand," *Journal of Development Economics*, vol. 60, no. 1, pp. 105–141, 1999.

[15] M. O. Jackson, "An overview of social networks and economic applications," *The handbook of social economics*, pp. 511–585, 2010.

[16] K. Deininger and Y. Liu, "Economic and social impacts of self-help groups in india," *World Bank Policy Research Working Paper Series, Vol*, 2009.

[17] J.-M. Baland, R. Somanathan, and L. Vandewalle, "Microfinance lifespans: A study of attrition and exclusion in self-help groups in india," *India Policy Forum*, vol. 4, no. 1, pp. 159–210, 2008.

[18] B. Feigenberg, E. M. Field, and R. Pande, "Building social capital through microfinance," tech. rep., National Bureau of Economic Research, 2010.

[19] D. Kinny and M. Georgeff, "Modelling and design of multi-agent systems," in *Intelligent Agents III Agent Theories, Architectures, and Languages*, pp. 1–20, Springer, 1997.

[20] S. Green, L. Hurst, B. Nangle, and P. Cunningham, "Software agents: A review," *Knowledge engineering review*, 1997.

[21] N. R. Jennings, "Coordination techniques for distributed artificial intelligence," *Foundations of distributed artificial intelligence*, pp. 187–210, 1996.

[22] J. Lian, S. M. Shatz, and X. He, "Flexible coordinator design for modeling resource sharing in multi-agent systems," *Journal of Systems and Software*, vol. 82, no. 10, pp. 1709–1729, 2009.

[23] H. Xu and S. M. Shatz, "Adk: An agent development kit based on a formal design model for multi-agent systems," *Automated Software Engineering*, vol. 10, no. 4, pp. 337–365, 2003.

[24] A. L. Symeonidis, D. D. Kehagias, and P. A. Mitkas, "Intelligent policy recommendations on enterprise resource planning by the use of agent technology and data mining techniques," *Expert Systems with Applications*, vol. 25, no. 4, pp. 589–602, 2003.

[25] H. S. Yim, H. J. Ahn, J. W. Kim, and S. J. Park, "Agent-based adaptive travel planning system in peak seasons," *Expert Systems with Applications*, vol. 27, no. 2, pp. 211–222, 2004.

[26] F. Hattori, T. Ohguro, M. Yokoo, S. Matsubara, and S. Yoshida, "Supporting network communities with multiagent systems," in *Community Computing and Support Systems*, pp. 330–341, Springer, 1998.

[27] F. Hattori, T. Ohguro, M. Yokoo, S. Matsubara, and S. Yoshida, "Socialware: Multiagent systems for supporting network communities," *Communications of the ACM*, vol. 42, no. 3, pp. 55–ff, 1999.

[28] E.-I. Osawa, "A scheme for agent collaboration in open multiagent environments," *IJCAI*, vol. 93, pp. 352–359, 1993.

[29] M. Ahmed, M. Ahmad, and M. Yusoff, "A collaborative framework for multiagent systems," *Intelligent Information and Database Systems*, vol. 5990, pp. 329–338, 2010.

[30] A.-M. Khamees Itaiwi, M. S. Ahmad, A. Hamid, N. Hamimah, N. H. Jaafar, and M. A. Mahmoud, "A multi-agent framework for dynamic task assignment and delegation in task distribution," *Computer & Information Science (ICCIS), 2012 International Conference on*, vol. 1, pp. 318–323, 2012.

[31] X. Bai, G. Dai, D. Xu, and W.-T. Tsai, "A multi-agent based framework for collaborative testing on web services," in *Software Technologies for Future Embedded and Ubiquitous Systems, 2006 and the 2006 Second International Workshop on Collaborative Computing, Integration, and Assurance. SEUS 2006/WCCIA 2006. The Fourth IEEE Workshop on*, pp. 6 pp.–, April 2006.

[32] Collins, *Collins English Dictionary Cameron - Complete & Unabridged 10th Edition*. Dictionary.com: HarperCollins, 2011. Collaboration.

[33] S. Helper, J. P. MacDuffie, and C. Sabel, "Pragmatic collaborations: advancing knowledge while controlling opportunism," *Industrial and corporate change*, vol. 9, no. 3, pp. 443–488, 2000.

[34] M. Wooldridge, N. R. Jennings, *et al.*, "Intelligent agents: Theory and practice," *Knowledge engineering review*, vol. 10, no. 2, pp. 115–152, 1995.

[35] X. Luo, C. Zhang, and H.-f. Leung, "Information sharing between heterogeneous uncertain reasoning models in a multi-agent environment: a case study," *International journal of approximate reasoning*, vol. 27, no. 1, pp. 27–59, 2001.

[36] J. A. Sillince and M. H. Saeedi, "Computer-mediated communication: problems and potentials of argumentation support systems," *Decision Support Systems*, vol. 26, no. 4, pp. 287–306, 1999.

[37] M. U. Spence, "Graphic design: Collaborative processes = understanding self and others." University Lecture, 2006.

[38] H. Parker, "Interfirm collaboration and the new product development process," *Industrial Management & Data Systems*, vol. 100, no. 6, pp. 255–260, 2000.

[39] M. Armstrong and R. H. Porter, *Handbook of industrial organization*, vol. 3. Access Online via Elsevier, 2007.

[40] F. J. Contractor and P. Lorange, "Why should firms cooperate? the strategy and economics basis for cooperative ventures," *Cooperative strategies in international business*, pp. 3–30, 1988.

[41] J. Hagedoorn, "Understanding the rationale of strategic technology partnering: Interorganizational modes of cooperation and sectoral differences," *Strategic management journal*, vol. 14, no. 5, pp. 371–385, 1993.

[42] B. Kogut, "Research notes and communications a note on global strategies," *Strategic Management Journal*, vol. 10, no. 4, pp. 383–389, 1989.

[43] J. Hagedoorn, "Understanding the cross-level embeddedness of interfirm partnership formation," *Academy of management review*, vol. 31, no. 3, pp. 670–680, 2006.

[44] B. M. Gilroy, *"Networking in multinational enterprises: The importance of strategic alliances"*. Univ of South Carolina Press, 1993.

[45] F. J. Contractor and P. Lorange, *"Cooperative strategies and alliances"*. Elsevier Science Boston, MA, 2002.

[46] N. M. Kay, *"The Boundaries of the Firm: critiques, strategies and policies"*. Macmillan, 1999.

[47] J. Chen and H.-T. Shih, *"High Tech Industries in China"*. Edward Elgar Publishing, 2005.

[48] B. Gray, *Collaborating: Finding common ground for multiparty problems*, vol. 329. Jossey-Bass San Francisco, 1989.

[49] U. Cantner, "New explorations in the economics of technological change," *Journal of Evolutionary Economics*, vol. 1, no. 2, pp. 169–172, 1991.

[50] Y. Zhang, S.-B. Guo, J. Hu, and A. Hodgkinson, "Choosing business collaborators using computing intelligence methods," in *Advanced Intelligent Computing Theories and Applications. With Aspects of Theoretical and Methodological Issues*, pp. 529–535, Springer, 2008.

[51] C. Burnett, T. J. Norman, and K. Sycara, "Trust decision-making in multi-agent systems," in *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume One*, pp. 115–120, AAAI Press, 2011.

[52] T. J. Norman and C. Reed, "A model of delegation for multi-agent systems," in *Foundations and Applications of Multi-Agent Systems*, pp. 185–204, Springer, 2002.

[53] J. K. Butler, "Trust expectations, information sharing, climate of trust, and negotiation effectiveness and efficiency," *Group & Organization Management*, vol. 24, no. 2, pp. 217–238, 1999.

[54] D. J. Teece, "Profiting from technological innovation: Implications for integration, collaboration, licensing and public policy," *Research policy*, vol. 15, no. 6, pp. 285–305, 1986.

[55] S. V. Berg, J. Duncan, and P. Friedman, *Joint venture strategies and corporate innovation*. Oelgeschlager, Gunn & Hain Cambridge, 1982.

[56] W. P. Burgers, C. W. Hill, and W. C. Kim, "A theory of global strategic alliances: the case of the global auto industry," *Strategic Management Journal*, vol. 14, no. 6, pp. 419–432, 1993.

[57] P. Ghemawat, M. E. Porter, and R. A. Rawlinson, "Patterns of international coalition activity," *Competition in global industries*, pp. 345–366, 1986.

[58] A. D. Chandler Jr, *Strategy and structure: chapters in the history of the industrial enterprise*. MIT press, 1990.

[59] M. E. Porter, "From competitive strategy to cooperative strategy," *Harvard Business Review*, vol. 65, no. 3, pp. 43–59, 1987.

[60] R. P. Rumelt, *Strategy, structure, and economic performance*. Division of Research, Graduate School of Business Administration, Harvard University Boston, MA, 1974.

[61] Y. Zhang and C. Harvie, "Size still matters when firms choose business collaborators," *The 7th SMEs in a Global Economy Conference 2010 CHALLENGES AND PROSPECTS (pp. 1-11)*, 2010.

[62] J. K. Hemphill and C. M. Westie, "The measurement of group dimensions," *The Journal of Psychology*, vol. 29, no. 2, pp. 325–342, 1950.

[63] C. Gencel and O. Demirors, "Functional size measurement revisited," *ACM Trans. Softw. Eng. Methodol.*, vol. 17, pp. 15:1–15:36, June 2008.

[64] T. Hastings and A. Sajeev, "A vector-based approach to software size measurement and effort estimation," *Software Engineering, IEEE Transactions on*, vol. 27, no. 4, pp. 337–350, 2001.

[65] A. Allaby and M. Allaby, *A dictionary of earth sciences*. Oxford University Press, 2008.

[66] T. Honderich and I. P. Masters, *The Oxford companion to philosophy*. Cambridge Univ Press, 2005.

[67] J. Conaghan and P. Cane, *The new Oxford companion to law*. Oxford university press, 2008.

[68] P. Moles, *Handbook of international financial terms*. Oxford University Press, 1997.

[69] R. Oliver and R. W. Oliver, *What is transparency?* McGraw Hill Professional, 2004.

[70] K. Bickerstaff, R. Tolley, and G. Walker, "Transport planning and participation: the rhetoric and realities of public involvement," *Journal of transport geography*, vol. 10, no. 1, pp. 61–73, 2002.

[71] A. Vaccaro and P. Madsen, "Transparency in business and society: introduction to the special issue," *Ethics and information technology*, vol. 11, no. 2, pp. 101–103, 2009.

[72] D. Ince, *Dictionary of the Internet: Book and CD-ROM with Cdrom*. Oxford University Press, Inc., 2001.

[73] R. Islam, "Do more transparent governments govern better?," *World Bank Policy Research Working Paper*, 2003.

[74] C. Hood and D. Heald, *Transparency: The key to better governance?* Oxford University Press, 2006.

[75] J. Black, N. Hashimzade, and G. Myles, *A dictionary of economics*. Oxford University Press, 2012.

[76] J. S. Hwang, C. S. Leem, and H. J. Moon, "A study on relationships among accounting transparency, accounting information transparency, and xbrl," in *Convergence and Hybrid Information Technology, 2008. ICCIT'08. Third International Conference on*, vol. 1, pp. 502–509, IEEE, 2008.

[77] H. J. Park, H. H. Shin, and D. G. Kang, "Corporate transparency," in *Korean Strategic Management Society, KSMS Summer Conference, 2004*, KSMS, 2004.

[78] J. S. Hwang, C. S. Leem, and H. J. Moon, "A study on relationships among accounting transparency, accounting information transparency, and xbrl," in *Convergence and Hybrid Information Technology, 2008. ICCIT'08. Third International Conference on*, vol. 1, pp. 502–509, IEEE, 2008.

[79] J. H. Han and J. S. Bae, "An empirical study on the factors affecting the transparency of venture businesses," in *Korean Strategic Management Society*, pp. 81–105, Spring Conference, 2001.

[80] S. P. Williams, P. A. Scifleet, and C. A. Hardy, "Online business reporting: An information management perspective," *International Journal of Information Management*, vol. 26, no. 2, pp. 91–101, 2006.

[81] R. Bushman and A. Smith, "Transparency, financial accounting information, and corporate governance," *Financial Accounting Information, and Corporate Governance. Economic Policy Review*, vol. 9, no. 1, 2003.

[82] K. Zhu, "Information transparency of business-to-business electronic markets: A game-theoretic analysis," *Management Science*, vol. 50, no. 5, pp. 670–685, 2004.

[83] A. Schilder, "Accounting standards, transparency and supervision," *Accounting Task Force of the Basel Committee on Banking Supervision*, 2002.

[84] A. M. Elorrieta, "Disclosure and transparency: Accounting and auditing," *The Third Meeting of the Latin American Corporate Governance Roundtable*, 2002.

[85] A. Damodaran, "Information transparency and valuation: can you value what you cannot see?," *Managerial Finance*, vol. 33, no. 11, pp. 877–892, 2007.

[86] K. Piętak, A. Woś, A. Byrski, and M. Kisiel-Dorohinicki, "Functional integrity of multi-agent computational system supported by component-based implementation," in *Holonic and Multi-Agent Systems for Manufacturing*, pp. 82–91, Springer, 2009.

[87] D. H. McKnight and N. L. Chervany, "What trust means in e-commerce customer relationships: an interdisciplinary conceptual typology," *International journal of electronic commerce*, vol. 6, pp. 35–60, 2002.

[88] J. Sabater and C. Sierra, "Reputation and social network analysis in multi-agent systems," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pp. 475–482, ACM, 2002.

[89] V. Basili, P. Donzelli, and S. Asgari, "A unified model of dependability: capturing dependability in context," *Software, IEEE*, vol. 21, pp. 19–25, Nov 2004.

[90] G. Dewsbury, I. Sommerville, K. Clarke, and M. Rouncefield, "A dependability model for domestic systems," in *Computer Safety, Reliability, and Security* (S. Anderson, M. Felici, and B. Littlewood, eds.), vol. 2788 of *Lecture Notes in Computer Science*, pp. 103–115, Springer Berlin Heidelberg, 2003.

[91] A. Bondavalli, I. Majzik, and I. Mura, "Automatic dependability analysis for supporting design decisions in uml," in *High-Assurance Systems Engineering, 1999. Proceedings. 4th IEEE International Symposium on*, pp. 64–71, IEEE, 1999.

[92] J. D. Lewis and A. Weigert, "Trust as a social reality," *Social forces*, vol. 63, no. 4, pp. 967–985, 1985.

[93] R. Perlman, "An overview of pki trust models," *Network, IEEE*, vol. 13, no. 6, pp. 38–43, 1999.

[94] L. Xiong and L. Liu, "A reputation-based trust model for peer-to-peer e-commerce communities," in *E-Commerce, 2003. CEC 2003. IEEE International Conference on*, pp. 275–284, IEEE, 2003.

[95] P. Michiardi and R. Molva, "Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks," in *Advanced Communications and Multimedia Security*, pp. 107–121, Springer, 2002.

[96] C. T. Nguyen, O. Camp, and S. Loiseau, "A bayesian network based trust model for improving collaboration in mobile ad hoc networks," in *Research, Innovation and Vision for the Future, 2007 IEEE International Conference on*, pp. 144–151, IEEE, 2007.

[97] Y. Wang and J. Vassileva, "Bayesian network trust model in peer-to-peer networks," in *Agents and Peer-to-Peer Computing*, pp. 23–34, Springer, 2005.

[98] G. Theodorakopoulos and J. S. Baras, "Trust evaluation in ad-hoc networks," in *Proceedings of the 3rd ACM workshop on Wireless security*, pp. 1–10, ACM, 2004.

[99] D. Gambetta, "Can we trust trust," *Trust: Making and breaking cooperative relations*, vol. 2000, pp. 213–237, 2000.

[100] R. C. Mayer, J. H. Davis, and F. D. Schoorman, "An integrative model of organizational trust," *Academy of management review*, vol. 20, no. 3, pp. 709–734, 1995.

[101] J. C. Trinder, "Competency standards–a measure of the quality of a workforce," in *ISPRS XXI International Congress, Beijing, China*, 2008.

[102] T. J. Norman and C. Reed, "A model of delegation for multi-agent systems," in *Foundations and Applications of Multi-Agent Systems*, pp. 185–204, Springer, 2002.

[103] W. W. Vasconcelos, M. J. Kollingbaum, and T. J. Norman, "Normative conflict resolution in multi-agent systems," *Autonomous Agents and Multi-Agent Systems*, vol. 19, no. 2, pp. 124–152, 2009.

[104] S. Modgil, N. Faci, F. Meneguzzi, N. Oren, S. Miles, and M. Luck, "A framework for monitoring agent-based normative systems," in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 153–160, International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[105] K. Cetnarowicz, G. Dobrowolski, M. Kisiel-Dorohinicki, and E. Nawarecki, "Functional integrity of mas through the dynamics of the agents' population," in *Multi Agent Systems, 1998. Proceedings. International Conference on*, pp. 405–406, IEEE, 1998.

[106] Y. Wang, H. Wang, Y. Wang, and X. Gu, "Study of social integrity behavioral mode based on multi-agent system," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 11, no. 6, 2013.

[107] http://www. businessdictionary.com/definition/transaction cost.html, *Business dictionary*. 2012. Transaction cost.

[108] J. C. Bertot, P. T. Jaeger, and J. M. Grimes, "Using icts to create a culture of transparency: E-government and social media as openness and anti-corruption tools for societies," *Government Information Quarterly*, vol. 27, no. 3, pp. 264–271, 2010.

[109] J. Guida and M. Crow, "E-government and e-governance," *ICT4D: Information and Communication Technology for Development*, p. 283, 2009.

[110] C. R. Baker and P. Wallage, "The future of financial reporting in europe: its role in corporate governance," *The International Journal of Accounting*, vol. 35, no. 2, pp. 173–187, 2000.

[111] A. Hyytinen and T. Takalo, "Enhancing bank transparency: A re-assessment," *European Finance Review*, vol. 6, no. 3, pp. 429–445, 2002.

[112] R. Lambert, C. Leuz, and R. E. Verrecchia, "Accounting information, disclosure, and the cost of capital," *Journal of Accounting Research*, vol. 45, no. 2, pp. 385–420, 2007.

[113] C. Ji-fu, "Information disclosure of internal control of listed company: An empirical study [j]," *Economy & Audit Study*, vol. 2, p. 022, 2005.

[114] M. C. Torri, "Community gender entrepreneurship and self-help groups: a way forward to foster social capital and truly effective forms of participation among rural poor women?," *Community Development Journal*, vol. 47, no. 1, pp. 58–76, 2012.

[115] J. Xin and C. Xiao, "The relationship between information transparency of family listed companies and their performance based on perspective of corporate governance," in *Computational Sciences and Optimization (CSO), 2011 Fourth International Joint Conference on*, pp. 414–417, IEEE, 2011.

[116] Y. Zhan and F. Xiong, "Studying trust in virtual teams," in *Future Generation Communication and Networking Symposia, 2008. FGCNS'08. Second International Conference on*, vol. 1, pp. 36–40, IEEE, 2008.

[117] P. S. Greenberg, R. H. Greenberg, and Y. L. Antonucci, "Creating and sustaining trust in virtual teams," *Business Horizons*, vol. 50, no. 4, pp. 325–333, 2007.

[118] M. Cucciniello, G. Nasi, and G. Valotti, "Assessing transparency in government: rhetoric, reality and desire," in *System Science (HICSS), 2012 45th Hawaii International Conference on*, pp. 2451–2461, IEEE, 2012.

[119] P. Iskanius, H. Helaakoski, A. Alaruikka, and J. Kipina, "Transparent information flow in business networks by using agents," in *Engineering Management Conference, 2004. Proceedings. 2004 IEEE International*, vol. 3, pp. 1342–1346, IEEE, 2004.

[120] R. Abassi and S. G. El Fatmi, "Towards a generic trust management model," in *Telecommunications (ICT), 2012 19th International Conference on*, pp. 1–6, IEEE, 2012.

[121] L. Wen, P. Lingdi, L. Kuijun, and C. Xiaoping, "Trust model of users' behavior in trustworthy internet," in *Information Engineering, 2009. ICIE'09. WASE International Conference on*, vol. 1, pp. 403–406, IEEE, 2009.

[122] F. Bao and R. Chen, "Trust management for the internet of things and its application to service composition," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, pp. 1–6, IEEE, 2012.

[123] H. Mohanty, K. Prasad, and R. Shyamasundar, "Trust assessment in web services: an extension to juddi," in *e-Business Engineering, 2007. ICEBE 2007. IEEE International Conference on*, pp. 759–762, IEEE, 2007.

[124] B. Killinger, *Integrity: Doing the right thing for the right reason*. McGill-Queen's Press-MQUP, 2010.

[125] D. L. Duarte and N. T. Snyder, *Mastering virtual teams: Strategies, tools, and techniques that succeed*. Wiley. com, 2011.

[126] T. L. Guillot, *Team Building in a Virtual Environment*, vol. 205. ASTD, 2002.

[127] C. F. Lange, M. R. Chaudron, and J. Muskens, "In practice: Uml software architecture and design description," *Software, IEEE*, vol. 23, no. 2, pp. 40–46, 2006.

[128] A. Evans, S. Kent, and B. Selic, *UML 2000-The Unified Modeling Language. Advancing the Standard: Third International Conference York, UK, October 2-6, 2000 Proceedings*, vol. 1939. Springer, 2000.

[129] A. Souri, M. Sharifloo, and M. Norouzi, "Formalizing class diagram in uml," in *Software Engineering and Service Science (ICSESS), 2011 IEEE 2nd International Conference on*, pp. 524–527, IEEE, 2011.