Modeling of Tasks and Task Execution For Self Help Groups [SHGs] For Computer Science Research
A B Sagar, Hyderabad Central University, Gachibowli, Hyderabad, Andhra Pradesh

Introduction

A Self Help Group [SHG] is a group of about 10 to 20 people who come together to form a small scale business, saving and credit organization. They pool their skills for their business growth personal development. SHGs also federate into larger organizations through clustering. At the cluster level, there are inter-group borrowings, exchange of ideas, sharing of costs and discussion of common interests. SHGs have been very effective in India. There were several thousands of them spread geographically all over the country. SHGs allow members to save money, provide employment, provide skill development. They help individuals to achieve their business and social objectives. They provide access to larger bank loans. 200 million poor women today have access to savings and credit services through 3.36 million SHGs all over the country. Over 30,000 branches of regulated banking structure are involved in this phenomenon and have mobilized loans of over $ 450 million. However, SHG management is not automated and only a few processes are computerized. Since majority of the members are either non-literate or semi-literate, it did not attract the industries to project them. Now, most of the tasks in the SHG management are manual which makes SHG management stood cumbersome. Automation of SHGs will greatly impact the management’s performance and enhance the efficiency of the SHGs. At present, research on SHGs from computer science dimension exists only at piece-meal stage. There is no holistic approach for the SHG management yet. We believe that a holistic approach to the SHG management is very essential. The present idea of modeling the tasks of the member of the SHG and Task Execution Cycle, greatly enhance the development of a holistic approach for the automation of SHG management.

Related Work

My previous paper titled “People, SHGs and Social Values: A Formal Framework” published in the Special Issue of IJCCT, Vol-2 provides the basis for the present paper and described the structure of SHGs, SHG members and the social values that are desired in the SHGs. The present paper attempts to formalize the tasks of the SHGs.
Modeling a Task

Tasks are business functions that are executed by all SHGs. Task execution is the basic functionality of SHGs and they are also the basis for collaborations. Once a task was given to a SHG, it is the responsibility of that SHG to execute it and return the result to the issuing authority. A task, in the present model, is an entity with several attributes. Every task, besides others, has the following key attributes:

**Task_Id:** Identification token of the task by which each task is uniquely identified.

**Parent_Task:** If the task is subtask of a bigger task, then this is the id of the parent task.

**IsPartitionable:** Yes, if the task can be split and distributed; No, if it has to be executed at only one place.

**Risks:** Risks involved with the task.

**Task_Type:** Whether the given task is technical or non-technical.

**Creation_Date:** Date when the task was created.

**Excluded_Executors:** Restricted executors list.

**Task_Stakeholders:** All the stakeholders who are related to the task.

**Task_Status:** Status of the task, indicating how far the task was completed.

**Skills_Needed:** List of skills needed to execute the task.

**Skills_Levels_Needed:** Level of skills needed for executing the task; excellent, fair, medium, poor skill levels.

**Task_Infrastructure:** Infrastructure that would be required for the execution of the task.

**Task_Priority:** Priority of the task; Low, medium, high.

**Start_By_Date:** The date by which one has to start the execution of the task.

**Interdependencies:** Interdependencies of the task.

**Task_Supervisor:** The member / work group in-charge of the task.

**Complete_By_Date:** The date by which one has to complete the execution of the task.

**Notification_Recipients:** Stakeholders that are to be notified about the status of the task.

**Responsibilities:** Responsibilities that are to be undertaken by the task executor.

**Task_Initiator:** Identity of the one who initiated the task.

**Potential_Executors:** Executors for whom the task is meant.

**Cost_of_Task:** Cost of the task estimated using variables like the resources and skills it involves and its significance (priority)
Task Execution Cycle

Each member (and also group’s) responsibility consists of executing tasks. Each task has several properties, phases and associated financial and non-financial information.

The Task Execution Cycle (TEC) is the integral part of ExTsk of MBM. It has all the states that a task can undergo. The different states are described below:

- **Recv**: The task is in Recvd state if it is just received. It will wait here until the member/ work group is ready to consider it for initiation and execution.
- **Initiate**: The task moves to Initiated state if the member/ work group has considered execution.
- **Execute**: The task moves to Executing state when the member/ work group has started execution or is continuing execution after waking up from sleep.
**Assign:** The task moves to assigned state if the task was assigned to a different member/work group.

**Sleep:** The task moves to sleep state if the task was assigned to some other member or SHGs.

**Abort:** If the task was found to be infeasible or if the assignment fails for some other reason, then the task moves to aborted state.

**End:** The task reaches the final state by any means.

The different possible paths and corresponding states of Task’s lifecycle are:

1. Recvd → Initiate → Execute → End
2. Recvd → Initiate → Execute → Assign → Sleep → Execute → End
3. Recvd → Initiate → Execute → Abort → End
4. Recvd → Initiate → Execute → Assign → Sleep → Execute → Abort → End
Goal of a Task:

Each task $T$ has a ‘start state’ and a ‘final state’. Before beginning of execution of a task, the task is at ‘start state’ $S$.

During execution, the task moves slowly towards completion, and finally reaches the final state called the goal state $G$. The distance between the task’s start state and the goal state is $D$.

The subdistances $d_1, d_2, d_3, \ldots, d_n$ are the small distances that a task moves towards $G$ with the completion of each of its subtasks. And,

$$\sum_{i=1}^{i=n} d_i = D$$
From Fig: 4, a simplified formalization of a task would be: Task, \( T = (l_i, d_i, t_m, n) \) where \( l_i \) is the initial location of the task, \( d_i \) is the destination of the task, \( t_m \) is the total subtasks and \( n \) is the number of collaborators involved. If a task has no subtasks or there exists a constraint that it should not be split into subtasks, then \( t = 0 \). And, if a task does not have any collaborators, then \( n = 0 \). A task is called a complex task if \( t > 1 \) and \( n > 1 \).

If the task has several subtasks, then with the completion of each subtask it moves a distance \( dm \). How far a task has progressed after \( k \) subtasks is \( \Sigma d_k - l_i \), and how much more is remaining is the difference between \( d_k \) and \( \Sigma dm \). When \( \Sigma dm = D \), the task execution is finished.

If complexity is the complexity involved in executing the task \( t_i \). That is, complexity is the complexity to move the task a subdistance of \( d_i \). Then total complexity of the task, Complexity \( T \) is the summation of complexities of subtasks /subdistances.

\[
Complexity_T = \sum_{i=1}^{i=m} (d_i \times complexity_i)
\]

Collaborated Distance: The total distance moved by collaborators is called collaborated distance. The collaborated subtasks may be executed sequentially or nonsequentially, however, if there are \( k \) subtasks then collaborated distance =

\[
\sum_{i=1}^{i=k} d_i
\]

The complexity of collaborated tasks =

\[
= \sum_{i=1}^{i=k} (d_i \times complexity_i)
\]
Task-Collaborator Graph: This graph gives the detail of the subtasks and the corresponding collaborators. This also details the concurrent subtasks. The subtasks that are enclosed between the orange colored lines indicate concurrent subtasks. The green edges of the Task-Collaborator graph are weighted. The weights denote the subdistances i.e. the distance a task moves towards the destination when the task is completed. When a task is executed by more than one collaborator, it has arrows pointing to all those collaborators. The circle with darkened center marks the goal of the task.

![Task Collaboration Graph](image)

**Fig 5: Task Collaboration Graph**

Task Classification: Task classification helps in defining the complexity of the task.

![Task Classification](image)

**Fig 6: Task classification**
A task is classified as independent if, for its execution and completion, it does not need any collaborators. If the task needs the collaborators then it is a collaborative task. And, if the task is completely infeasible for the SHG, and needs to be transferred to some other SHG, then it is an assignable task. The complexity of an independent task is zero, complexity of a collaborative task is some arbitrary value calculated depending on the task, complexity of assignable task is infinity. If the collaboration is needed for a short span of time, then it is ‘partial collaboration’, and if the collaboration was needed throughout the task, then it is ‘complete collaboration’. Complexity of partial collaboration is lesser than the complexity of complete collaboration. If the collaborative tasks need both parties [i.e. both the task owner and the collaborator] to involve in the execution, then it is ‘synchronous’, else if part of the task is executed independently by the collaborator, then it is asynchronous. Complexity of the synchronous tasks is greater than the asynchronous tasks. The spatial or geographic dimension is that the collaborators are either in the same place (co-located) or in different places (remote). Complexity in case of remote collaborators is greater compared to co-located collaborators.

From the above figure, we can classify a task as:
1. independent task, collaborative task, assignable task
2. partial collaborative task, complete collaborative task
3. synchronous partial collaborative task, synchronous complete collaborative task, asynchronous partial collaborative task, asynchronous complete collaboration task
4. co-located synchronous partial collaborative task, co-located synchronous complete collaborative task, co-located asynchronous partial collaborative task, co-located asynchronous complete collaboration task, re-mote synchronous partial collaborative task, remote synchronous complete collaborative task, remote asyn-chronous partial collaborative task, remote asynchronous complete collaborative task.

Every task has an attribute called ‘IsPartitionable’ which specifies whether the task can be partitioned and executed as subtasks at different places, or it cannot be split and has to be executed entirely at one place. This attribute affects the type of collaboration a SHG would choose. If the task is partitionable, then it splits the task into subtasks and assigns to different collaborators. But if the task is not partitionable, then it looks for a single collaborator who can execute the whole task.
REPORTING MESSAGES DURING TASK EXECUTION

Reporting task status information can be done using reporting messages associated with the task. These reporting messages are application specific. However, some of the following reporting messages are mandatory.

Ack  
RecvdTask  
InitiatedTask  
ExecutingTask  
AssignedTask  
AbortedTask  
EndedTask  
RecvdRequest  
RecvdReport

Fig 6: Task’s Collaborated Execution
DESCRIPTION OF MESSAGES:

<Ack> - Ack to be sent to the report sender to acknowledge the receipt of the report.
<RecvdTask> - Message to be sent to the task sender acknowledging that the task was received.
<InitiatedTask> - Message to be sent to the notification recipients that the task execution is initiated.
<ExecutingTask> - Message to be sent to the notification recipients that the task is executing and its status.
<AssignedTask> - Message to be sent to the notification recipients that the given task was assigned to another member/ SHG.
<AbortedTask> - Message to be sent to the notification recipients that the task execution was Aborted.
<EndedTask> - Message to be sent to the notification recipients that the given task execution has ended.
<RecvdRequest> - Message to be sent to the requester that the request was received
<RecvdReport> - Ack sent to the report sender that the report was received successfully.

FORMAT OF REPORTING MESSAGES:

<RecvdTask> - <RecvdTask> <taskId> from <task assigner Id> on <date> <time> <\RecvdTask>
<InitiatedTask> - <InitiatedTask> <taskId> on <date> <time> with <responsibilities> and <interdependencies> <\InitiatedTask>
<ExecutingTask> - <ExecutingTask> <taskId> <date> <time> <status> <problems> <\ExecutingTask>
<AssignedTask> - <AssignedTask> <taskId> <assignedto> on <date> <time> <\AssignedTask>
<SleepTask> - <SleepTask> <taskID> <wake up on> <event> <\SleepTask>
<AbortedTask> - <AbortedTask> <task Id> aborted due to <reason> on <date> <time> <\AbortedTask>
<EndedTask> - <EndedTask> <taskId> on <date> <time> <ended with success/failure/abort> <feedback> <performance understanding> <\EndedTask>
<Ack> - <Ack> <ackId> on <date> <time> for <taskId/ requestId/reportId> from <memberId> <shgId> <\Ack>
<RecvdRequest> - <RecvdRequest> from <member Id> <shg Id> regarding <task Id> on <date> <time> <to be responded by date & time > <\RecvdRequest>
During the execution of the task, task status is to be notified to the notification recipients and requesters. Also, if a log or report is received, it is stored in the repository.

**Performance Index Computation**

Performance index depends on the amount of task completed. If there is progress in the task, then there will be a message to be sent to the stakeholders. If there is no progress in the work, then there is no message. So, this idea can be used to calculate the performance index of the SHG. The basic idea is that if a task is being executed properly, then there will be messages to be sent, and depending on the messages sent, we can compute the message flowrate, and depending in the flowrate we can compute the performance index of the SHG.

\[ \text{Flow}(\text{info}, \text{SHG}, S) \]

is a function to indicate the information of task (‘info’) flowing from SHG to Stakeholders (‘S’). In a half-duplex communication system, the info flow is only unidirectional i.e. SHG to S. So, \( \text{flow}(\text{info}, \text{SHG}, S) \) holds true. But the converse, \( \text{flow}(\text{info}, S, \text{SHG}) \), is not true. That is, flow of info from S to SHG does not happen. There is no possibility for the stakeholders to request and obtain information of their choice in half-duplex mode. They only get what the SHG chooses to reveal.

For start up, we can begin with computation of performance index of two SHGs doing similar businesses. Let us define flowrate as the rate of flow of information from SHG to S. Since this is half duplex mode of communication, the flowrate from S to SHG is zero. Flowrate has two important dimensions namely \textit{Quantity of information and Quality of information}.

For example, if a task made progress and there were 12 messages to be sent, but it was reported only 6 times, then the flowrate is:  \( \frac{\text{number of messages reported}}{\text{total number of actual messages}} \).

Therefore, \( \text{flowrate} = \frac{6}{12} \) which is equal to 0.5.
If the task was reported \( p_i \) times, with respect to \( N_i \) total occurrences, then its flowrate is \( f_i \).

\[
flowrate, \ f_i = \frac{p_i}{N_i}
\]

The Qualitative dimension of flowrate associates a 'v' which is 'value'(or rank/priority) to each business function. This value will be assigned by the task initiators (funding / governing agency).

Some business functions get a higher 'v' while other business functions get a lower 'v' based on their importance in that project.

Suppose a SHG has 'n' business functions \( P_1, P_2, P_3, \ldots P_n \) which are disclosed \( p_1, p_2, p_3, \ldots p_n \) respectively, and values \( v_1, v_2, v_3, \ldots v_n \) respectively, then the Static Transparency of that SHG can be derived as follows:

For example, let us consider two SHGs, G1 and G2, each having five business functions \( P_1, P_2, P_3, P_4 \) and \( P_5 \). Let the total occurrences of each business function be 12.

\[
\text{static} \ T_{\text{SHG}} = \sum_{i=1}^{n} v_i \cdot \frac{p_i}{N_i}
\]

we have flowrate, \( f_j = \frac{p_j}{N_j} \)

\[
\therefore \ \text{static} \ T_{\text{SHG}} = \sum_{i=1}^{n} v_i \cdot f_i
\]

\[
\begin{align*}
\text{static} \ T_{\text{SHG}} &= \sum_{i=1}^{5} v_i \cdot f_i \\
&= (20 \times 0.5) + (20 \times 0.3) + (20 \times 0.25) \\
&\quad + (20 \times 0.5) + (20 \times 0.16) \\
&= 10 + 6 + 5 + 10 + 3.2 \\
&= 34.2
\end{align*}
\]
Suppose that G1 discloses all the messages. So G1’s messages will get the following disclosure values. \( p_1 = 12, p_2 = 12, p_3 = 12, p_4 = 12 \) and \( p_5 = 12 \). Let values \((v_1 , v_2 , v_3 , v_4 , v_5 )\) of messages be 20,20,20,20,20 respectively.

\[
\text{Now, flowrate } f_1 = \frac{p_1}{(Total \text{Occurrences})} \\
= \frac{12}{12} \\
= 1
\]

\[
\text{similarly, } f_2 = \frac{12}{12} = 1 \\
f_3 = \frac{12}{12} = 1 \\
f_4 = \frac{12}{12} = 1 \\
f_5 = \frac{12}{12} = 1
\]

Since the value is 100, G1 is said to have good performance index.

Now G2:

Suppose that G2 does not disclose all the messages. So G2’s messages will get the following values. \( p_1 = 6, p_2 = 4, p_3 = 3, p_4 = 6 \) and \( p_5 = 2 \). Let values \((v_1 , v_2 , v_3 , v_4 , v_5 )\) of messages be 20,20,20,20,20.

\[
\text{Now, flowrate of } P1, \\
f_1 = \frac{p_1}{(Total \text{Occurrences})} \\
= \frac{6}{12} = 0.5
\]

\[
\text{similarly, } f_2 = \frac{4}{12} = 0.33 \\
f_3 = \frac{3}{12} = 0.25 \\
f_4 = \frac{6}{12} = 0.5 \\
f_5 = \frac{2}{12} = 0.16
\]

Since the value is 34.6, G2 is said to have less performance index. Thus we are able to compute performances indices.
Conclusion

The present paper has successfully modeled a task and task execution of a work group. The execution cycle of task execution is also generalized. This paper also calculates performance index of a task executor which can be either a member or a SHG.

Acknowledgments

I would like to thank Prof. Hrushikesha Mohanty, DCIS, HCU for his valuable guidance and thoughts for his guidance in my research work. This work is funded by Rajiv Gandhi National Fellowship.

References

2. APMAS, SHG Federations in India: A Status Report , 2007
6. APMAS, Optimizing SHGs, October 2005