OPT:
An Approach to Organizational and Process Improvement

Carolyn B. Seaman *
Victor R. Basili
Department of Computer Science
University of Maryland
College Park, MD 20742

Abstract

Software development and maintenance enterprises constitute an extremely complex, varied, and poorly understood class of organizations. This is due in part to the newness of the technology and the dynamic nature of the field, but it also stems from the complexity of human-machine interactions. A major driver of the effectiveness of such an organization is the relationship between the software development process and the organizational structure. Little attention has been paid to this relationship, Scacchi's work [6] being one exception. Our approach addresses this issue in more detail.

This paper describes the OPT method for improving both the organizational structures and processes that constitute software development environments. This method is meant to be part of a continuous improvement program, and is modeled after the Quality Improvement Paradigm [1]. The approach includes mechanisms for modeling the relationship between an organizational structure and a development process, for measuring this relationship quantitatively, and for using this information to plan specific improvements to the environment.

1 Introduction

The role that organizational design plays in the software industry is different from its role in nearly any other business. The processes which constitute software development present an awkward blend of completely automated tasks, steps that require human-machine interaction, and some purely creative activities. The organizational structure that is appropriate for a particular software development effort does not arise straightforwardly from these processes, as is often the case with manufacturing efforts. Nor is it largely irrelevant, as might be the case in strictly artistic endeavors. Clearly, the organizational design of a software development project is a non-trivial task. At the same time, the effect of organizational structure on the proper execution of development processes can be profound. Communication can be facilitated (or hindered). Responsibility can be fairly distributed and well understood (or easily denied and passed on). Software development organizations must be designed to facilitate the smooth execution of development processes, both automated and human.

The OPT approach to improvement of software development environments is based on the combined improvement of organizational structure and development process. The approach is an iterative improvement method based on the Quality Improvement Paradigm [1]. The OPT modeling formalisms and the OPT metrics are central to the approach.

The goal of the OPT modeling formalisms is to create an organizational model. An organizational model completely describes those aspects of the relationship between organization and process that we wish to measure. An organizational model has three parts. First is a process model which describes part of a particular lifecycle development process, an entire lifecycle process, or a group of interrelated processes, depending on the scope of the modeling effort. The second part includes information about the actors, the interactions, and the support processes that are instrumental to the execution of the development process. The third section represents the purely organizational (non-process) relationships in the environment. Information about the process (first part) and the organization (third part) are

*This work is supported in part by IBM Canada Laboratory Centre for Advanced Studies and by NASA.

related by the second part. In other words, the second part of the model describes the organization/process relationship.

We have focused on two major factors which characterize the relationship between an organizational structure and a process. These two characteristics are the distribution of *responsibility* and the *communication* required by the process. The OPT metrics are designed to quantify these characteristics.

Responsibility for process activities is distributed among the members of the organization. The process can be said to impose this responsibility on an organization. Speaking in the abstract, an organization does not take on any responsibility unless it is given some sort of process. In this way, the issue of responsibility captures the effect that the process has on the organization.

The second factor which characterizes the relationship between an organizational structure and a process is the process communication within the organization. An organizational structure can either facilitate or hinder the efficient flow of information between process activities and participants. In this sense, this second factor characterizes the effect of an organization on a process.

The OPT metrics, which are designed to capture the characteristics outlined above, are also used as the building blocks of OPT goals and constraints. Quantitative relationships are defined between the various properties of responsibility and communication. Constraints specify which of these relationships must be preserved. Goals specify which of these relationships must change. How the relationships should change, and by how much, is what defines improvement.

In the sections which follow, the OPT improvement method is first described. Then the OPT modeling formalisms and the OPT metrics are presented. Finally, the way in which the metrics are used to form goals and constraints is illustrated.

2 The OPT Approach

The OPT approach is an iterative improvement method based on the Quality Improvement Paradigm (QIP) [1], an iterative, goal-driven framework for continuous improvement of software development. The QIP is a closed-loop process which includes steps for planning, executing, and evaluating improvements to software development environments, as well as for incorporating experience gained from improvement efforts into future development.

The OPT approach to improvement of software development environments relies on the combined improvement of organizational structure and development process. Like the QIP, it is a closed-loop improvement cycle. The steps are outlined below:

1. Model the initial relationship between the organization and the process
2. Set high-level project goals, as well as specific organizational and process improvement goals
3. Define constraints which represent management policies and which limit the possible changes
4. Using the model, measure various properties of the organization/process relationship
5. Based on these measurements, select candidate changes to the organization and process which will contribute to the satisfaction of the stated goals
6. Simulate these changes by applying them to the model, or experiment by applying the changes to some subset of the organization and/or process
7. Re-measure and evaluate the results of the simulation or experimentation
8. Either iterate back to step 2 to find more candidate changes, or institutionalize the changes in the actual environment.

The OPT approach considers an organizational structure and a software development process together as a single system. The purpose of the steps outlined above is to identify the most appropriate changes to be made to the organizational structure and the development process in order to satisfy the general project goals.

The result of step 1 is a baseline organizational model, described in the next section, which captures the relevant properties of the organization and process. This model is designed to facilitate the evaluation of the OPT metrics, also described in a later section. The goal-setting step (number 2) in OPT results in two sets of goals. The first set are high-level project goals that concern such issues as quality and productivity. Goals in the second group are defined in terms of OPT metrics. They reference specific attributes of the relationship between organization and process, such as responsibility and communication. The OPT goals must be chosen so that their satisfaction contributes to the satisfaction of the project goals. Constraints (defined in step 3) are also expressed in terms of OPT metrics. The OPT metrics are also used in step 4 to quantify relevant attributes of the relationship between the organization and process. The attributes are quantified in such a way that they highlight anomalies, or parts of the environment that are out of the ordinary. This facilitates the selection of candidate changes in step 5. Steps 6 and 7 allow the evaluation of changes before they are implemented. The measurement results of step 7 are used to modify goals, constraints, and candidate changes in subsequent iterations of the OPT cycle.

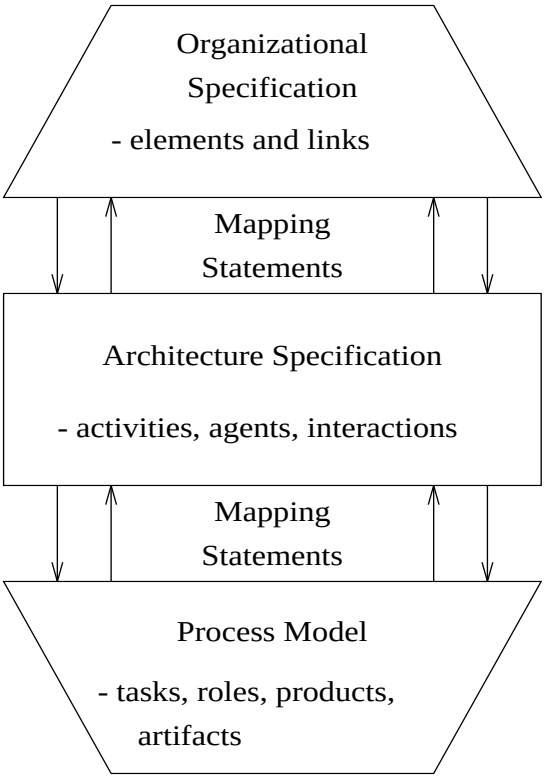

Figure 1: An organizational model.

3 Organizational Models

Organizational models have three parts:

- an organizational specification, which describes the organizational structure
- a process model, which describes the development process, and
- an architecture specification, which describes the relationship between organization and process

This section describes the notations used for representing organizational models. Figure 1 shows the different parts of an organizational model. Forming the base is a model of a particular development process. The architecture specification, the middle layer, describes the relationship between the process and the organizational structure charged with executing the process. Information in the architecture specification is present in, but obscured by, the process model. It represents those aspects of the process that impact, or are impacted by, the structure of the organization. The third part, the organizational specification, represents the purely organizational (non-process) aspects of the software development environment. An organizational model also includes explicit mappings between corresponding objects in these three parts.

An *organizational specification* represents an organizational structure very simply as a set of nodes and links. It describes the non-process relationships, or links, between different elements of the organization. Such relationships might include "manages", "reports to", "funds", "sits near", and "evaluates performance of". Link types are all user-defined. In most organizations, the relationship of greatest interest is the one that defines the management structure. However, other relationships exist in organizations, and thus should be studied as well. A single organizational specification can contain any number of different types of links. The elements of the specification usually represent people, but could represent other organizational components, such as teams, divisions, and locations.

The purpose of the *architecture specification* is to describe the organizational properties of a software development process. The agents that execute the process are described in terms of the specific activities that they perform and the interactions between them. In other words, those parts of the process that are relevant here are those that can affect, or be affected by, the organizational structure. This information is an abstraction of information assumed to be available in the process model. Such an abstraction is necessary because most process models obscure this type of organizational information among other details, making it scattered and often implicit. In other words, the information that is most important to the modeling of an organization is difficult to conceptualize in a process model and must be abstracted out to be seen clearly. An architecture specification provides this abstraction and thus is more useful than a process model for organizational analysis. It also provides the bridge between a development process and an organization which allows the two to be considered together as a system.

The objects of an architecture specification (activities, agents, and interactions) correspond to objects that appear in most process models (see [5], for example). Activities refer to individual tasks or process steps in a process model. Most process modeling languages allow the specification of who or what is responsible for executing each task. This type of object is usually called a role, and corresponds to an architectural agent. Interactions describe an instance of communication that is required by the process. They are sometimes difficult to map to specific objects in a process model. They correspond roughly to artifacts, or products, or documents, which serve as input or output to a task.

Agents are defined, in part, by the activities they perform. Also associated with each activity is information about the priority of that activity, and the formality of the responsibility associated with that activity.

```
agentDeveloper is
    activity Develop_Component;
      priority = (1,1);
      formality = written;
    activity Request_Component;
```

```
    priority = (2,2);
    formality = spoken;
end Developer;
```

Agents are also defined by interactions between them. Each interaction definition describes one part of the development process which requires some type of information to pass from one person or group to another person or group. An interaction, then, defines an instance of process communication. A simple example of an interaction definition is shown below.

```
interaction Component_Request is
    purpose:      directional;
    participants: Component_Library;
                  Project1_Developer;
    unidirectional  from Project1_Developer
                    to Component_Library;
    dependent activities:
          Component_Library.Answer_Request;
          Project1_Developer.Request_Component;
end Component_Request;
```

The notation used to write architecture specifications has been designed specifically for that purpose, and is described in detail in [7] and [8].

A *process model* representing the development process of interest forms the third part of an organizational model (the bottom part in Figure 1). A discussion of process modeling and the various formalisms that are available can be found in a number of references, including [2], [4], and [5]. An organizational model can incorporate a process model in any representation that provides certain required capabilities. These capabilities are:

- representation of roles, or objects that are the performers of process steps
- description of products, or inputs and outputs of process steps
- functional decomposition of process steps
- representation of resource allocation information (effort or cost information)

We require this information from a process model for two reasons. First, it allows mappings to be defined between the process model and the architecture specification. Also, this information is explicitly used in the calculation of OPT metrics, described in the next section. Many process modeling formalisms are available that provide at least the first three of these requirements. In some cases, the resource allocation information must be obtained separately.

An organizational model also requires four types of mapping statements, to describe four types of mapping relationships. These four types describe mappings between:

- architectural agents and roles in a process model
- architectural activities and tasks in a process model
- architectural interactions and products or artifacts in a process model
- architectural agents and elements of an organizational specification

4 Organizational and Process Attributes

A software development environment is an information-feedback system, which is defined as a system in which

> ...the environment leads to a decision that results in action which affects the environment and thereby influences future decisions. [3]

This applies to the actions and decisions that take place in a software development environment. They have an effect not only on the software product, but also on the environment itself, and how software is produced in the future. This attribute of software development environments influences how we measure them. The metrics designed for development environments must examine the internal workings of the system and how the different parts of the system interact with and affect each other. Measuring the end product, or even the intermediate products (documentation, specifications, designs, etc.) is not sufficient.

The overall goal of the OPT method is to determine, given a software development process and the organizational structure meant to perform that process, how well suited these two elements are to each other. That is, two questions must be asked:

> How good is this process for this organization?
> How good is this organization for performing this process?

Because it is often possible and desirable to modify both the organizational structure and the process to achieve improvement, these two questions must be considered simultaneously.

We have chosen two major attributes which characterize the relationship between an organizational structure and a process. These were chosen to be broad and thus to encompass a number of smaller issues within their scope. They were also chosen to represent the complementary impact between organization and process. The first characterizes the effect that the process has on the organization, while the second deals with the effect of the organization on the process.

4.1 Responsibility

The first characteristic is the distribution of *responsibility* for process activities among the members of the organization. A process' activities can be said to impose certain amounts of responsibility on the members of an organization. In this way, the issue of responsibility captures the effect that the process has on the organization.

Responsibility for a particular process activity reflects the extent to which the success or failure of the activity affects the professional success or failure of a participant in the activity. Responsibility has a number of properties that are of interest and that can be measured. One of these is *type*. Responsibility can be shared or not. Management responsibility is an additional type.

Another aspect of responsibility is its effect on an organization member's level of motivation. *Commitment* is the amount of benefit or recognition that someone derives from participating in an activity and performing well. *Obligation*, on the other hand, measures the negative impact of not participating in an activity, or performing poorly in the activity. The OPT metrics that quantify commitment and obligation combine the values given for priorities and formalities of each activity in the organizational model.

Another question, which concerns a particular member's level of responsibility over all his or her activities, is how diverse that member's responsibilities are. *Diversity* of responsibility, then, is simply the number of different process activities for which a person holds some type of responsibility.

Finally, *proportional importance* addresses the question of how a member's responsibility for an activity affects his or her time. Proportional importance measures the proportion of a person's time that is spent participating in each activity.

All of these responsibility measures can be evaluated by direct inspection of an organizational model. An architectural agent definition includes all the activities for which the agent holds some type of responsibility. These activities are mapped to process steps or tasks in the process model. The mapping from agents to elements of the organizational specification then allows us to associate members of the organization with process tasks.

4.2 Communication

The second factor which characterizes the relationship between an organizational structure and a process is the *process communication* within the organization. An organizational structure can either facilitate or hinder the efficient flow of information between process activities and participants. In this sense, this second factor characterizes the effect of an organization on a process.

To measure communication, the organizational model is utilized by analyzing the interactions in the architectural specification. Each of these interactions describes some instance of communication that is required by the process. The participants in the interactions are members of the organization. The interactions, then, completely describe the parts played by both the process and the organization in each instance of communication.

The central question that must be raised with respect to process communication has to do with the amount of effort that the process requires for effective communication. In other words, how much effort is required to ensure that all process participants have the information they need, when they need it? This is a complex question whose answer relies on several pieces of information. We operationalize the concept of *interaction effort* as the product of three properties: the *medium* employed, the *purpose*, and the *organizational distance* between the participants of each interaction.

Information about the medium used for a particular interaction is provided by the organizational model through the mappings between interactions and objects in the underlying process model. For example, some types of interactions could be documents, memos, meetings, verbal agreements, etc. Each of these types represents a communication *medium*. The *purpose* of an interaction is provided in the interaction definition and reflects the criticality of understanding in an interaction. For example, some interactions are simply for the purpose of keeping some part of the organization informed about the status of another part. Other interactions involve information that will directly affect decisions made by the recipient of the information. The relative position of interaction participants, or the *organizational distance* between them, is another property of process communication. How far apart, organizationally, interaction participants are reflects not only how difficult it is for them to communicate, but also how difficult it is for them to understand each other. Members which are organizationally close together not only communicate with less effort, but also operate in similar contexts, and thus can understand each other more easily. Organizational distance can be impacted by the hierarchical position of the participants, their physical proximity, and their involvement in cross-organization groupings.

5 Formulating Goals and Constraints

The metrics described above are used to characterize a software development environment. Another important use, however, is the building of OPT goals and constraints. OPT goals are set during the OPT improvement cycle in order to guide the selection of candidate changes to the organization and process. OPT constraints are also formulated during the OPT cycle,

as part of the characterization of the system. They are used to constrain the possible choices of candidate changes.

Goals and constraints are formulated using the metrics defined in the last section. Relationships are defined between the various basic properties of responsibility and communication. Constraints specify which of these relationships must be preserved. On the other hand, goals specify which of these relationships must change. How the relationships change, and by how much, is what defines improvement.

Often the relationships defined in goals and constraints involve other factors that are not any of the properties defined above. Relationships often involve such properties as the organizational level of members, the importance or criticality of process activities, the estimated effort of process activities, etc. Metrics must be defined for any of these properties that are used in goals or constraints.

For example, suppose that a particular software development environment identifies a problem with slipping deadlines. A decision is made to address this problem by identifying and focusing on those process activities on the critical path. The Quality Improvement Paradigm is used to plan and implement improvements in the environment. As well, OPT is used to plan changes to the organizational structure and the development process. The project goal chosen is to increase the participation of higher-level members of the organization in critical-path process activities. This project goal would translate to the following OPT goals, stated in terms of the metrics defined previously:

- Increase the average organizational level of those who hold some *type of responsibility* for activities on the critical path
- Increase the average *commitment* and *obligation* to critical-path activities for members of the organization with a level more than x

At the same time, assume that there is a concern about overloading middle managers. A general policy is drafted stating that the number of different activities that a manager has responsibility for is bounded, but increases with the manager's level in the organization. This translates to the following constraint:

- Any member's *diversity of responsibility* is bounded by y times the member's organizational level.

This simple example illustrates the way in which OPT metrics are combined with other measurable properties to formulate goals and constraints. Depending on the types of goals and constraints that are chosen, it is sometimes possible to express the relationships as systems of linear inequalities that can be solved with linear programming methods. The solution to such a system gives a set of values which satisfy the goal by maximizing (or minimizing) some property while satisfying all of the constraints. Further research is planned into other mathematical methods that can be applied to systems of OPT goals and constraints.

6 Conclusions

This paper describes the OPT method for organizational and process improvement of software development environments. This method is meant to be part of a continuous improvement program, and is modeled after the Quality Improvement Paradigm. The approach includes mechanisms for modeling the relationship between an organizational structure and a development process, for measuring this relationship quantitatively, and for using this information to plan specific improvements to the environment.

The OPT method requires the support of a tool both to aid in the development of models and to calculate the OPT metrics from an organizational model. Work on this tool is planned in the near term. At the same time, process and organizational information is being gathered from software development environments to build case studies with which the approach will be validated and improved. A parallel activity is the investigation of organization theory literature and practice. The OPT approach will benefit from the incorporation of well-founded results and principles from this discipline. One important part of our future work is the use of the simulation capabilities of many process modeling tools. When an organizational model includes a simulatable process model, the OPT metrics will then reflect dynamic, rather than static, properties of organization and process.

This paper describes very early results in organizational modeling and analysis. This work is part of a general broadening of focus in software engineering research from strictly technical improvement approaches to the measurement and improvement of non-technical factors. This trend in the field reflects a maturity in software engineering that will bring further advances in software quality and productivity.

Acknowledgments

The authors would like to acknowledge the contributions to this work of Gianluigi Caldiera.

References

[1] V.R. Basili, "Software Development: A Paradigm for the Future" (Keynote Address), *Proceedings, COMPSAC '89*, Orlando, FL, September 1989, pp. 471-485.

[2] Bill Curtis, Marc Kellner, and Jim Over, "Process Modeling", *Communications of the ACM*, September

1992, Vol. 35, No. 9, pp. 75-90.

[3] Jay W. Forrester, *Industrial Dynamics*, The M.I.T. Press, 1961.

[4] Nazim H. Madhavji, Kamel Toubache, and Ed Lynch, "The IBM-McGill Project on Software Process", IBM Technical Report 74-077, IBM Canada Laboratory, October 1991.

[5] H. Dieter Rombach, "MVP-L: A Language for Process Modeling In-The-Large", *Computer Science Technical Report Series*, No. CS-TR-2709, University of Maryland, College Park, MD, June 1991.

[6] Walt Scacchi, "Managing Software Engineering Projects: A Social Analysis", *IEEE Transactions on Software Engineering*, 10:1, January 1984.

[7] Carolyn B. Seaman, "AAA: A Modeling Language for Software Production Environments", *Proceedings of CASCON'92*, IBM Canada Ltd. Laboratory Centre for Advanced Studies, November 1992.

[8] Carolyn B. Seaman, "OPT: Organization and Process Together", *Proceedings of CASCON'93*, IBM Canada Ltd. Laboratory Centre for Advanced Studies, October 1993.