International Journal of Information Technology and Business Management
29th March 2016. Vol.47 No.1

ISSN 2304-0777 www.jitbm.com

COMMUNICATION MANAGEMENT IN IT PROJECTS USING UML LANGUAGE

Viliam Malcher*, Studeničová Andrea
Faculty of Management, Department of Information Systems, Comenius University,
820 05 Bratislava 25, Odbojárov 10, P.O. Box 95, Slovak Republic, Europe
e-mail: *viliam.malcher@fm.uniba.sk

Abstract

The ability to communicate is an essential life skill and one that be continually developed. Communication in IT projects is a complex process that involve several iterations before understanding is achieved. You can communicate using words, symbols, pictures, messages and other elements. In this paper we have focused our attention on the communication in complex projects by program code based on the Unified Modeling Language (UML) and design patterns.

Keywords: UML Language, Communication Management, IT Projects, Design Patterns.

I. INTRODUCTION

The greatest threat to many projects is a failure to communicate. Effective project communications is paramount to success on all projects and especially on complex IT projects. It has been estimated that 90% of project manager's time is spent in some for of communication.

Every project should include some type of communication management plan, a document that guides project. The communication plan must answer to the following questions: who are stakeholders? What information do the stakeholders need, when do they wait it, and what level of detail do they need, and in what forms? Who on the project team is responsible for collecting of data, creating the reports? We need to design the communication management plan, which has the project name and complies with software development lifecycle, process and the project management process. The plan identifies all activity, which must be performed to communicate for the project information and it also specifies the role and responsibilities of project team members. The communication can include also several other definitions: (1) an exchange of information, (2) a verbal or written message, and (3) techniques for expressing ideas effectively.

The work [1] provides an overview of basic concepts and processes that guide project communication at the department. This is the project communication handbook for the project team. In the manual [2] was developed to guide project managers through corporate project management methology. This manual presents a framework for managing projects using basic tools needed for success.

Today's business environment two factors have became common: change and complexity. We work in an environment of constant change and increasing complexity, and must be competive, productive, customer-focused and profitable.

In this paper we try to clarify a communication infrastructure between objects in a complex IT project. We have proposed a design for a software code using UML language which allows to effective communication and the communication is precisely driven by a code in a project.

II. PROJECT ORGANIZATION AND COMMUNICATION

Communication is the most critical and time consuming activity. Communication includes the exchange of models and documents about the system, representing the status of work products, providing feedback on the quality of work products and communication decisions. To deal with communication issues, project participants have many tools available. The most effective one is conventions when participants agree on notations for representing information on tools for manipulation information, they already have eliminated substational sources of misunderstanding. Examples of notations include UML diagrams, templates for writing documents and meeting minutes, and identification schemes for naming software components.

Notations enable us to articulate complex ideas succinctly and precisely. In projects involving many participants, often of different technical an cultural background. It can be critical as the cost of miscommunication increases rapidly. For a notation to enable accurate communication, it must

International Journal of Information Technology and Business Management
29th March 2016. Vol.47 No.1

ISSN 2304-0777 www.jitbm.com

come with a well-defined semantic, it must be well written for representing a given aspect of a system.

Software engineering is a collaborative activity. The development of software bring together participants form different background. No single participant can understand and control all aspects of the system. Moreover, any change in the system requires participants to update their understanding of the system – these dependencies make critical the share information in an accurate and timely manner.

Communication can take many forms depending on the types of activities. Crises and misunderstanding are handled through spontaneous information exchanges such as telephone calls, messaged, hallway conversations, and ad hoc meetings. As software engineering projects become large, the time each participants must spend in communication increases. The organization of projects into teams and the sharing of information through formal and informal channel is essential.

We first describe the basic concepts associated with the project organization, such as work product and deliverable. We then describe the communication mechanisms available to participants. We describe it from the perspective of a project participants who needs to understand the project organization and communication infrastructure.

Participants need to focus on communication information accurately and efficiently. Techniques and notations based on the modeling by UML language enables project participants to build models of the system and communicate about them [3-6].

From developers perspective a project consists od four elements [7] as it is shown in Figure 1:

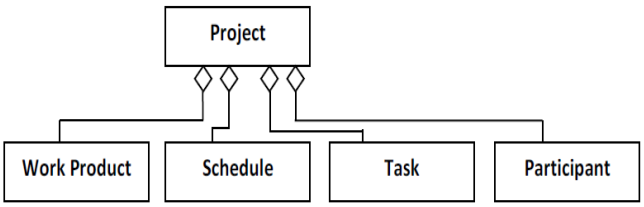

Figure 1: UML class diagram of a project.

Work Product is an item produced by the project such as a piece of code, a model or a document. *Schedule* specifies when work on the project should be accomplished. *Task* is a the work to be performed by a project participant to create a work product. *Participant* is any person participating in a project.

An important part of any project organization is to define the relations among participants. In a team-base organization consists of organization unit as it is shown in Figure 2:

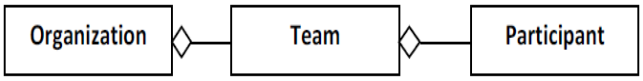

Figure 2: UML class diagram of a team-based organization.

The participants are grouped into teams, where a team is a small group of participants working on the same task. Into the team project participants interact with each other. The three major types of interaction are: reporting, decision, and communication. Communication is the type of interaction where the informations are exchanged needed for decision. In complex software projects many decision need to made locally by the developers, but depend on information from developers in other teams. The solution to this problem is to exchange information via an additional communication structure that allows participants to communicate directly with each other and in ways different from the reporting structure.

It is important also to define a role the set tasks that are expected from a participant on a team. In a team-based organization, we assign tasks to a person or a team via a role. The specification of work to be accomplished in completing a task is described in a work package. A work package includes many elements as are packages, activities, tasks, role and work products. A schedule is the mapping of tasks onto time: each task is assigned start and end items. This allows us to plan the deadlines for individual deliverables [7].

We have talked about the organization of a project. We now describe communication in a project and we are going to do some designs for it. There exist two type of communications − planned communication and unplanned communication. We focus our attention on planned communication. The basis are the teams. Each team is made up of several participants who work on a common task. Participants need to communicate. We do draft a communication, which will be managed program code according plan (a schedule plan). Here we use UML class diagrams. We propose the simplified design of the communication in the team is as follows:

International Journal of Information Technology and Business Management
29th March 2016. Vol.47 No.1

ISSN 2304-0777 www.jitbm.com

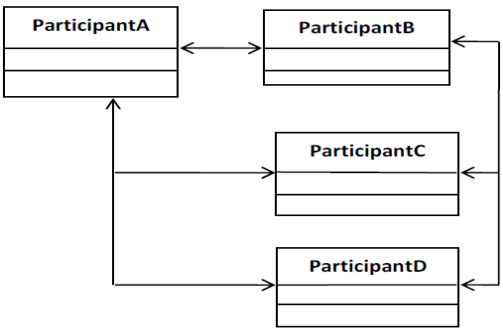

Figure 3: UML class diagram with bidirectional association.

For simplicity we consider four team members only. The communication will be mainly carried out according to a defined schedule. Relationships between classes could be reversible. We use association as a relation between classes. Associations are UML concepts that denote collections of bidirectional links between two or more classes. For two participants in the team we have

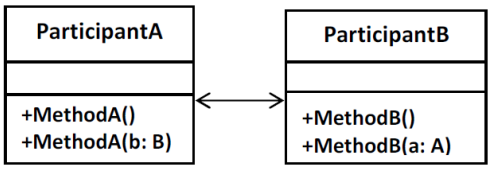

Figure 4: UML class diagram of an bidirectional association.

The software code is as follows:

```
<%@ Page Language="C#" Debug="true" %>
<%@ import Namespace="System.Data" %>
<%@ import Namespace="System.IO" %>
<html><body>
  <form id="Form1" runat="server">
    <asp:button id="Button1" Text="Participant Files"
        onclick="OpenFile" runat="server"/>
  </form>
 </body></html>
 <script runat="server">
 class A {
  public void MethodA() {
    System.Diagnostics.Process.Start(
                    @"c:\abox\ParticipantA.txt");
 }
 public void MethodA(B b) {
   b.MethodB();
  }
}
 class B {
  public void MethodB()  {
    System.Diagnostics.Process.Start(
                    @"c:\abox\ParticipantB.txt");
  }
  public void MethodB(A a) {
    a.MethodA();
  }
 }
 void OpenFile(Object s, EventArgs e)  {
    A a = new A();
    a.MethodA(new B());
  }
</script>
```

Listing 1: The program code for the communication between participant.

The flat file of the communication is as follows:

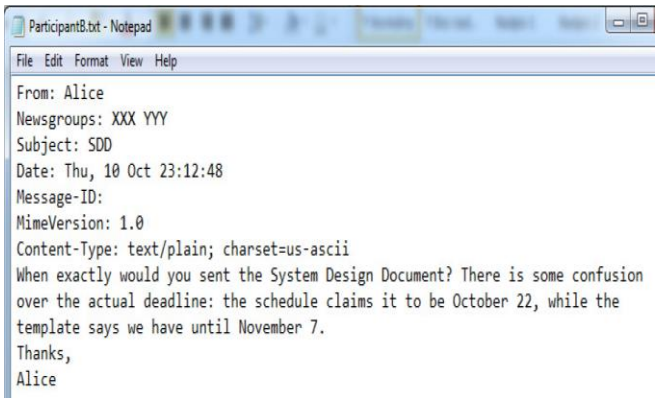

Figure 5: The communication flat file of the participantB.

If we have a manager in the team he can use the Observer pattern in the system. The Observer is a design pattern for notification to other participants [8, 9]. The Observer pattern facilitates communication between a parent class and any dependent child classes, allowing changes to the state of the parent class to be sent to the dependent child classes. This pattern to allow the state changes in a class to be sent to all its dependent classes. The class relationship is one-to-many between the class and all its dependents. The pattern generally consists of two base classes. The first is called the Subject class, and this class acts as the notification engine. The Observer classes act as receivers of the subject notifications:

International Journal of Information Technology and Business Management
29th March 2016. Vol.47 No.1

ISSN 2304-0777 www.jitbm.com

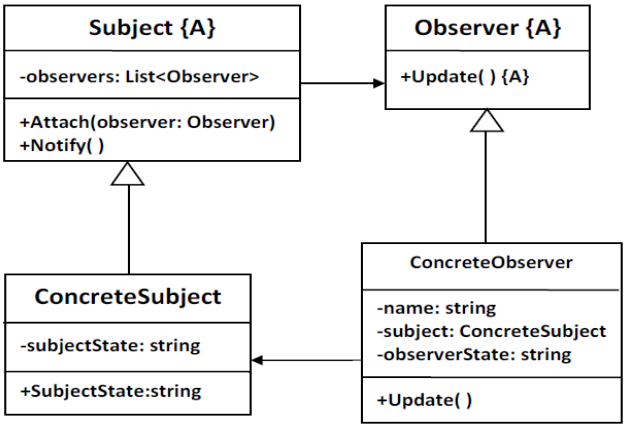

Figure 6: UML class diagram for the Observer pattern.

The symbol {A} is a note for an abstract class or a method. The program code reads:

```
<%@ Page Language="C#" Debug="true" %>
<%@ import Namespace="System.Data" %>
<%@ import Namespace="System.IO" %>
<%@ import Namespace ="System.Collections.Generic" %>

<html><body>
  <form id="Form1" runat="server">
    <asp:button id="Button1" Text="Run"  onclick="DoNotify"
runat="server"/>
  </form>
 </body></html>
 <script runat="server">
  abstract class Observer  {
    public abstract void Update();
  }
  abstract class Subject  {
    private List<Observer> observers = new List<Observer>();

    public void Attach(Observer observer) {
      observers.Add(observer);
   }
    public void Notify()    {
     foreach (Observer o in observers)
     {
      o.Update();
     }
   }
  }
  class ConcreteSubject : Subject {
   private string subjectState;

   public string SubjectState  {
    get { return subjectState; }
    set { subjectState = value; }
   }
  }
  class ConcreteObserver: Observer  {
   private string name;
   private string observerState;
   private ConcreteSubject subject;

   public ConcreteObserver(
        ConcreteSubject subject, string name) {
    this.subject = subject;
    this.name = name;
   }
   public override void Update()  {
    observerState = subject.SubjectState;
    System.Diagnostics.Process.Start(
          @"c:\abox\Manager.txt");
   }
   public ConcreteSubject Subject  {
    get { return subject; }
    set { subject = value; }
   }
  }
   public void DoNotify(Object o, EventArgs e)
   {
    ConcreteSubject ss = new ConcreteSubject();
    ss.Attach(new ConcreteObserver(ss, "X"));
    ss.Attach(new ConcreteObserver(ss, "Y"));
    ss.Attach(new ConcreteObserver(ss, "P"));
    ss.SubjectState = "New file is released";
    ss.Notify();
   }
</script>
```

Listing 2: The program code for the Observer pattern.

III. DISCUSSION AND CONCLUSIONS

There are many ways of communication throughout a project. The project manager will define the technology required for all project communications. Different communications will require different technologies and it is the project manager job to define the communication requirements.

We have proposed the design of communication by program code where the informations between participants is exchanged by flat files. All informations are saved in documents and managers can view a time history of communications in the project. The communication is not lost if we can all informations between participants recover from the files.

Effective communication involves both sending and receiving the message. Communication is the way exchange of information between entities. Communication during projects can be of many different types such as oral, written an non-verbal. We have focused our attention on the written communication and we have proposed mapping communication by program code. The written

ISSN 2304-0777 www.jitbm.com

communication has advantage, it is more precise a it means more effective than other types of communications.

REFERENCES

[1] Baars W., *Project Management Handbook*, 2006.

[2] *Project Communication Handbook*, 2nd edition, September, 2007.

[3] Gomaa H., *Software Modeling and Design: UML, Uses Cases, Pattern, and Software Architectures*, Cambridge University Press, 2011.

[4] Erikson H. E., Penker M., Lyons B., Fado D., *UML 2 Toolkit*, Wiley Publishing, Inc., 2004.

[5] Miles R., Hamilton K., *Learning UML 2.0*, O'Reilly, 2006.

[6] Pilone B., Pitman N., *UML 2.0 in Nutshell*, O'Reilly, 2005.

[7] Bruegge B., and Dutoit A. H., *Object-Oriented Software Engineering: using UML, Pattern, Pattern, and Java*, third edition, Pearson Education, Inc., 2010.

[8] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[9] Lasater Ch., *Design Patterns*, Wordqare Publishing, Inc. 2007.