# A communication workload estimation model based on relationships among shared works for software development projects

Noriko Hanakawa
*Hannan University*
*5-4-33 Amami Higashi*
*Matsubara, Osaka*
*580-8502 Japan*
*hanakawa@hannan-u.ac.jp*

Ken-ichi Matsumoto
*Nara Institute Science and Technology*
*8916-5 Takayama, Ikoma, Nara*
*630-0101 Japan*
*matumoto@aist-nara.ac.jp*

Koji Torii
*Nara Institute Science and Technology*
*8916-5 Takayama, Ikoma, Nara*
*630-0101 Japan*
*torii@aist-nara.ac.jp*

## Abstract

*Software development project managers have already known that adding manpower to a late software project makes it later. This is a famous sentence in "Mythical man-month" written by Brooks. The managers also recognize that even if two man-month workloads are assigned to two developers, the development period is not one month. However, the managers want to see when the two developers finish the two man-months workload. Of course, the managers don't think that the development period reduces from two months to one month. If the development period in the two developers is shorter than two months, the managers will decide to add the new developer to the work. Therefore, we propose a new workload estimation model based on communications among shared works. The model includes Concept Model of UML of a software development project. The classes and relationships in the Concept Model present the kinds of works and relationships among the works, respectively. Sharing works causes increments of communication workload. The communication workload is derived from the relationships among the classes in the Concept Model. In case studies using the model, we were able to reconfirm quantitatively the famous sentence of "Mythical man-month".*

## 1. Introduction

"Adding manpower to a late software project makes it later". This is a famous sentence of "Mythical man-month" written by Brooks [2]. Everyone in large-scale software development projects has already experienced it many times. That is beginning to be common sense in software development projects. In addition, project managers don't think that two developers finish two man-month workloads within one month. The development period should be more than one month at least; sometimes it will be more than two months.

However, even if the managers recognize that the adding manpower is not an efficient way of reducing the development period, the managers will be pressed for decision of adding the manpower to their project. For example, it assumes that a developer takes two months to complete a work, that is, the work is two man-month workloads. But the developer can not meet the deadline of the work. Because the work must be finished as soon as possible; the deadline of the work cannot be set after two months. A manager wants to reduce the execution period of the work by even a little time. If the period of the work can reduce to less than two months, the manager will decide to add a new developer to the work. In the case, what the manager wants to know most is when the two developers complete the two man-month workload [1]. Of course, the manager never dreams that the two developers will complete the work within one month. He wants to see if the two developers will complete the work within less than two months. If the execution period of the work is over two months, the manager will not add a new developer to the work. If the execution period is less than two months, the manager will decide to add a new developer. The quantitative workload estimation under the sharing work is important for the project manager.

Therefore we propose a new workload estimation model for shared works. The model includes Concept Model of UML (Unified Modeling Language)[10]. In the Concept Model, kinds of works are described as "classes", relationships between the works are described as "associations". In addition, the number of the association between the classes means strength of the relationship between the kinds of the works, a multiplicity of the association means the strength of the relationship among instances of the class. Values of the two strengths are bases of calculation of the communication workloads that are caused by sharing works. Total workload is the sum of the original workload and the communication workload. The development period is derived from the total workload divided by developer's productivity taking the concurrent executable works [5] into account.

In this paper, section2 describes a Relationship Diagram of Works using the Concept Model, section3 shows the new estimation model for the communication workload, section 4 explains a formula of development period taking concurrent executable works into account using case studies. Section 5 shows related works and section 6 shows a summary and future researches.

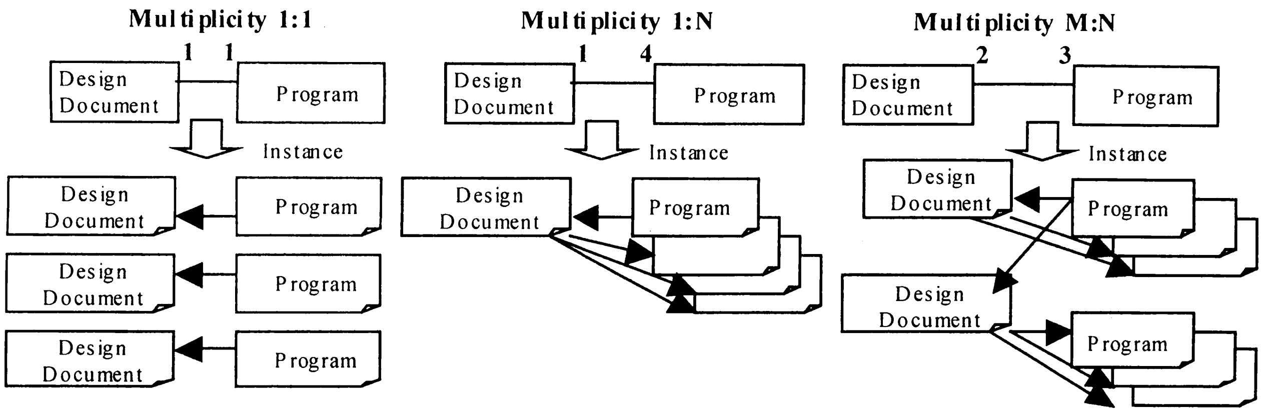

**Figure 2 Outline of the independence of the instances**

## 2. Relationships among works

Sharing a work to two developers causes communications between the developers. The communication workload depends on relationships between the shared works. For example, the relationship between a program designing work and a programming work is very strong. A program design document describes flow-charts, algorithms and interfaces of a program. A programmer should be in much communication with a designer who made the program design document. The programmer must understand what the designer wants to do. If the designing program work and the programming work are assigned to two different developers, the communication workload will be very much. In contrast, the relationship between a making use-case work and a programming work is weak. The developer who makes use-cases does not need to communicate with the programmer. Even if the works are shared to two different developers, the communication workload does not occur.

To clarify the relationships among the works, we make a relationship diagram of works using Concept Model of UML (Unified Modeling Language)[10]. Figure 1(refer to the last page) shows a relationship diagram of a software development project in an object-oriented development methodology. The classes of the Concept Model of Figure1 mean the works' documents; use-cases, design documents, programs and specifications of test. The associations among the classes mean the relationships among the works' documents. To be clear the quantities of the communications among the documents, multi-associations are set to two classes. In addition, a multiplicity is set to the association between the classes. If we make the distinction of the multiplicity clear, all multiplicities of Figure 1 will be 1:N or M:N. But the value of the multiplicity is an important factor when the workload is estimated in our proposed model. The values of multiplicities (M:N) are determined by using the experiential values in real projects [4]. The values of the multiplicities of Figure 1 are set the rough values that are derived from the real projects.

## 3. Workload Estimation Model

An estimation model for workload $(INC_{workload})$ using the relationship diagram of works is shown as formula (1).

$$INC_{workload} = \sum_{i=1}^{W_{num}} (COM1_i + COM2_i) \qquad \cdots\cdots\cdots(1)$$

| | |
|---|---|
| $INC_{workload}$ | :Increment workload caused by sharing work |
| $W_{num}$ | :Total number of kinds of works |
| $COM1_i$ | :Communication workload caused by dividing works according to the kind of the ***i-th*** work. (Assigning the developers to the different works.) $COM1 = COM1_{num} \times COM_{time}$ |
| $COM2_i$ | :Communication workload caused by sharing the ***i-th*** work among developers. (Assigning the developers to the same work.) $COM2 = COM2_{num} \times COM_{time}$ |

$COM1_{num}$, $COM2_{num}$ and $COM_{time}$ are shown as formula (2), (3), (4) in the following section.

### 3.1. Two types of works' sharing

There are two types for sharing works in software development projects. The first is that developers are assigned to the different kinds of works. We call it "COM1" type. In this paper, we use "divide" as the first type's verb, "division" as the first type's noun. The second is that the developers are assigned to parts of the same work. We call it "COM2" type. Similarly, we use "share" as the second type's verb, "sharing" as the second type's noun. In the COM1 type, for example, a developer who has sufficient domain knowledge is assigned to the work of making "use-cases", a developer who is a good engineer of "object-oriented methodology" is assigned to the work of making the concept model of UML. The reason of dividing works in COM1 is for rather "right person in right place" than for reducing the execution period of the work. The division of works in COM1 is based on a judgment of whether the developer's ability is sufficient to do it or not. In the relationship diagram of works (Figure1), each developer who has good experience for executing the work of the "class" is assigned to each the "class" of the Concept Model of Figure1. However, important cuttings between the works occur in the COM1's division. The work division based on "the right person in the right place" cuts "associations" between the "classes" that are assigned to two different developers. That is, the developer has to communicate with the other developer about the "associations" that are cut by the work division. The number of times of the communications between the developers depends on the number of the associations that are cut by the work division. The number of times of the communication in COM1 is calculated as follows:

$$COM1_{num} = R_{num} \times I_{num} \qquad \cdots\cdots\cdots\cdots(2)$$

| | |
|---|---|
| $COM1_{num}$ | :Number of times of the communication by work division. (COM1) |
| $R_{num}$ | :Number of the associations that are cut by the work division. |
| $I_{num}$ | :Number of instances that are created from a class. |

In the second type (COM2), a main purpose is that an execution period of the work reduces by sharing too much work with developers. The too much work is a same kind of work. When a manager predicts that the deadline of the work may be not met, the manager will decide to share the too much work. The number of times of the communication in COM2 is calculated as follows:

$$COM2_{num} = H \times \sum_{j=0}^{R_{num}} (Mul_j \times I_{num}(1-1/D_{num})) + C \cdots(3)$$

| | |
|---|---|
| $COM2_{num}$ | :Number of times of the communication by work sharing. (COM2) |
| $I_{num}$ | :Number of instances that are created from a class. |
| $D_{num}$ | :Number of sharing the instances that are created from the class. |
| $R_{num}$ | :Number of the association of the class. |
| $C$ | :Number of times of the communication for standardization of the work |
| $H$ | :Increment ratio of the workload with the sharing work [2]. $D_{num}(D_{num}-1)/2$ |
| $Mul_j$ | :Variable based on the multiplicity of the $j$-th association of the class that is shared. if Multiplicity is 1:1 or M:1 then $Mul_j$=1, else is $Mul_j$=N$\times$M |

In a case of programming work, even a developer who has sufficient knowledge and skill cannot meet the deadline of work because the programming work's quantity is too much. In the case, the developer will share the programming work with another developer, and the new developer will be assigned to a half of the programming work. Let me explain in the relationship diagram of Figure1. When some instances are created from a class (program), a developer is requested to complete all instances by a deadline of the programming work. However, the developer cannot complete all instances of the programming work by the deadline, the instances of the programming should be shared out two or three developers. Each cluster of the shared instances will be assigned to two or three developers, respectively.

Two kinds of communication occur from the work sharing in COM2. The first kind of communication is for standardization of the shared work, e.g., coding rule, design templates. The standardization is important to create the unified documents of the works. The variable $C$ of the formula (3) means the number of times of communication for the standardization. The value of the $C$ is determined by using experiential values in real projects [4].

The second kind of communication that occurs from the work sharing in COM2 is based on independence among the instances that are created from the class of the Concept Model. An outline of the independence of the instances is shown in Figure2. The independence of the instances is measured by the multiplicities of the associations that are set between classes of the Concept Model. The multiplicity is set to 1:1 or M:1 or 1:N or M:N. If we put the multiplicity in order of high independence, the order of the multiplicity will be 1:1, M:1, 1:N and M:M. If the independence of the instances is high, the number of times of the communication in COM2 is a little. In contrast, the independence is low; the number of times of the communication is a lot.

We explain the independence of each multiplicity; 1:1, 1:N and M:N. It assumes that there are work "A" and work "B". The work "A" and work "B" are identified as class "A" and class "B" of the relationship diagram of works. If the association that is set between class "A" and class "B" has "1:1" multiplicity, the independence of the instances of the class "B" is high. For example, if the class "A" is a program design document and the class "B" is a program, the multiplicity will be "1:1". A program design document indicates algorithms and flow-charts of a program. There are ten program design documents with ten programs. In this case, a developer who makes a program is required to understand only a program design document. He does not need to understand the other program because the program design document includes all information that is needed to make the program.

In contrast, it assumes that the multiplicity between the program design document and the program is 1:N. That is, the program design document indicates algorithms, interfaces and flow-charts of some programs. The developer needs to understand the other programs because the design document shows a common concept and interfaces among the ten programs. And a program's algorithm may be described as a part of an algorithm in the design document, a program's flow-chart may be shown as a section of a flow chart in the design document. In such design document, if the ten programs are shared out the ten developers, the developers will have to understand all programs. The understanding of all programs requires the communication among the developers who are assigned the different programs. In addition, if the multiplicity between the program design document and the program is M:N, the understanding among developers will be more complicated. A developer who makes a program should understand the contents of M deign documents, moreover he should understand the contents of N programs per a design document, that is, he must understand the contents of N $\times$ M programs (See Figure2).

Therefore the developer does not need to communicate with the other developers when the multiplicity between the programming and the designing programs is 1:1. He should communicate with the other developers who are assigned to the other programs when the multiplicity is 1:N. Moreover, the developer must hold developers' meetings many times when the multiplicity is M:N. That is, the multiplicity between the classes of the relationship diagram of work is an important measure for calculating the number of times of the communication among developers in COM2.

## 3.2 Communication time by communication way

The ways of the communications among developers are various. Table1 shows communication ranks that are determined by the communication ways and the communication's efficiency. If the communication ranks is high, it takes little time to communicate with the developers. If the rank is low, it takes much time to

**Table 1 Rank of the communication**

| Rank | Communi-cation type | Communication way | Communication time | Attendant time |
|---|---|---|---|---|
| 1 | Interacti-ve | A meeting with face to face | Execution time of the meeting | Traveling time of developers |
| 2 | | Teleconference | Execution time of the teleconference | Preparing time of the teleconference system |
| 3 | | Telephone | Time of talking by telephone | 0 |
| 4 | Non-interacti-ve | Sending documents by e-mail | Reading the documents | 0 |
| | | Sending documents by mail | Reading the documents | Time of sending document by mail |

communication among the developers. "Standard time" $H_{time}$ (See formula (4)) is a basis of the calculation of the communication time. The standard time means the average of time in once communication. The standard time is derived from the experiential communications of real projects.

The communication ways among the developers are classified into two types. The first communication type is "interactive communication"; a meeting with face to face, a teleconference and talking by telephone. The second communication type is "non-interactive communication" like a notification by documents. Of course, the interactive communication is more efficient than the non-interactive communication. In the interactive communication, if the communication ways are put in order of high efficiency, the order of the interactive communication will be a meeting with face-to-face, teleconference and talking by telephone. The non-interactive communication is basically a communication with documents. There are two ways of sending the document; by e-mail and by mail. Of course the sending by e-mail is more efficient than the sending by mail.

Here, let me explain the "attendant time" $Add_{time}$ of formula (4). The attendant time is important in discussion of the communication time. Although the meeting with face-to-face is most efficient communication way, it takes the traveling time of the developers. If the developers who work on same place have a meeting with face-to-face, the traveling time of the developers will be very a little, that is no problem in the communication time. In contrast, if the developers who work on the distant places have a meeting with face-to-face, the traveling time of the developers will be very large. The efficiency of communication should be calculated in consideration of the attendant time like the traveling time. Similarly, the attendant time of the non-interactive communication are discussed. If the document of the non-interactive communication is sent by e-mail, the attendant time is almost $0$ because the sending time of the document is a few seconds. If the document is sent by mail, the attendant time is one day or two days. The attendant times such as traveling time or sending time are shown in Table1. The once communication time $COM_{time}$ are calculated as follows:

$$COM_{time} = H_{time} \times Rank + Add_{time} \qquad \cdots\cdots\cdots\cdots(4)$$

$COM_{time}$ : Total time of once communication
$H_{time}$ : Average time of once communication
$Rank$ : Efficiency rank of the communication
$Add_{time}$ : Attendant time like the developers' moving or the documents' sending

$H_{time}$ is an average time of once communication in which developers discuss about a problem or information. It will be derived from experiential values of real projects.

The value of $H_{time}$ in ***COM1*** (work division, that is, the works are divided by the kinds of works) is larger than the value of $H_{time}$ in ***COM2*** (work sharing, that is, the same works are shared out developers). If we observe developers' behavior of a software development project, we will be able to measure the communication time that is a basis of $H_{time}$. The communication ***Rank*** of formula(4) and $Add_{time}$ of formula(4) are shown as "***Rank***" and "***Attendant time***" of Table1.

## 4. Case studies

In this section, two case studies are shown using the proposed model. The development periods are calculated as follows:

$$DEV_{time} = \sum_{g=1}^{W2_{num}} (Work_g / D_{numg} + COM2_g / D_{numg}) + \sum_{g=1}^{W1_{num}} COM1_g/2 \qquad \cdots\cdots\cdots\cdots(5)$$

$DEV_{time}$ :Total development period.
$W2_{num}$ :Total number of classes that is shared in COM2
$Work_g$ :The $g$-th work's development period. workload/(productivity × the number of developers)
$D_{numg}$ :Number of sharing the $g$-th work.
$W1_{num}$ :Total number of classes that is divided in COM1.
$COM1_g$ : COM1(see formula(2)) of the $g$-th work.
$COM2_g$ : COM2(see formula(3)) of the $g$-th work.

Case1 shows the workload estimation in a case that developers are assigned to different works (work division in COM1) in the upper process of a software development project. Case2 shows workload estimation in a case that new developers are added to same works (work sharing in COM2) near the close of the project because the deadline of the project may be not met. Table2 shows the common values of the both cases. The "Class" of Table2 means the class of the Concept Model in the object-oriented development project as Figure1. The "Num. of instance" of Table2 is calculated using the multiplicities that are set to the associations between the classes of the Concept Model of Figure1. Assuming that the number of the "domain document" of the Concept Model of Figure1 is set to ***10***, all numbers of the instances that are created from the classes were calculated as Table 2. For example, four associations are set between a "domain document" class and "use-case" class. The associations are named "Extracting use-case", "Extracting actor", "Extracting actor's behavior" and "Extracting response of system". When the number of the use-case is generated from the number of the domain document, the most important association is "Extracting use-case". Therefore, the number of the instance of the class "use-case" is determined as ***100*** using the multiplicity ***1:10*** that is set to the association "Extracting use-case". The value of N(***10***) multiplied by the number(***10***) of the instance of the class "domain document" is ***100***. The ***100*** means the number of the instances of the class "use-case".

In addition, the productivity of the each class was set (See Table 2). The values of the productivities were derived from experiential values of real projects. The value of ***C*** (the number of times of communication for standardization of work sharing) of formula (3) was set to same value of $D_{num}$(the number of sharing the work). The value of $H_{time}$ (average time of once communication) in work division of COM1 was set ***2 hours*** (***0.0125 months***), the value of $H_{time}$ in work sharing of COM2 was set ***20 minutes*** (***0.002085months***). And assuming that developers work on same place, the developers communicate on meetings with face-to-face. The value of $COM_{time}$ of ***COM1*** is ***0.0125***, $COM_{time}$ of ***COM2*** is ***0.002085***. The total workload of Table2 is 24.95 man-months, that is, the development period in working a developer is 24.95 months.

**Table 2 Common values of case studies**

| Class | Num. of instance | Productivity | Class | Num. of instance | Productivity |
|---|---|---|---|---|---|
| Domain document | 10 | - | Message of collaboration | 50 | 100 messages/months |
| Use-case diagram | 100 | 50 diagrams /month | Instance of collaboration | 25 | 100 instances/month |
| Concept model | 1 | - | Class design | 25 | 25 designs/month |
| Class of concept model | 25 | 25 classes/month | Program | 25 | - |
| Association of concept model | 50 | 20 associations /month | Attribute of program | 125 | 250 attributes/month |
| State diagram | 50 | 25 diagrams /month | Methods of program | 250 | 50 methods/month |
| Contract | 25 | 25 contracts/month | Specification of system test | 100 | 100 specifications/month |
| Sequence diagram of system | 1 | - | Specification of combination test | 125 | 100 specifications/month |
| Actor of sequence diagram of system | 5 | 100 actors/month | Specification of unit test | 250 | 125 specifications/month |
| Event of sequence diagram of system | 25 | 100 events/month | Execution of system test | 125 | 250 tests/month |
| Real use case diagram | 25 | 50diagrams /month | Execution of combination test | 250 | 125 tests/month |
| Collaboration diagram | 25 | - | Execution of unit test | 100 | 50 tests/month |

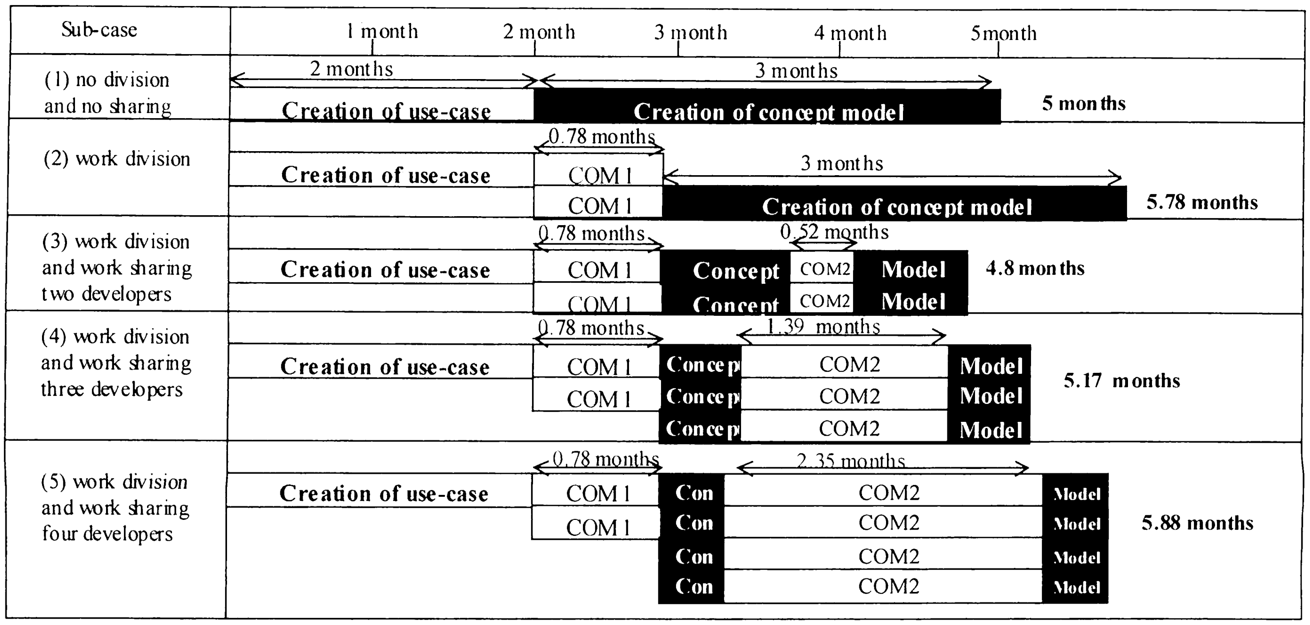

**Figure 3 Project plans of case1**

## 4.1 Case 1

We estimated the workload of the upper process that is from creating use-cases to creating a concept model (See Figure1). In sub-case (1), a developer is assigned to all works from the creation of the use-cases to the creation of the concept model. In sub-case (2), a developer is assigned to the creation of the use-cases, the other developer is assigned to the creation of the concept model, that is the work division in COM1. In sub-case (3), two developers are assigned to the creation of the concept model. In sub-case (4), three developers are assigned to the creation of the concept model, in sub-case (5) four developers are assigned to the creation of the concept model. In sub-case (3), (4) and (5), the uses-cases are created by a developer. That is, work division between the creation of the use-cases and the creation of the concept model occurs, moreover, the work sharing of the creation of the concept also occurs in sub-case (3), (4) and (5). The workload estimations using the proposed model are shown in the Table3.

## 4.2 Discussion of the Case 1

Bar charts that are based on the results of case1 in Table3 are shown in Figure3. We thought that the order of the works is important to create the bar charts. The creation of the use-case is finished before the starting of the creation of the concept model.

In sub-case (1), the development period is *5* months because there is no communication workload. In sub-case (2), it is an example of work division in COM1; dividing into kinds of works. The communications occur when the development process shifts from the creation of the use-cases to the creation of the concept model. Total communication workload is *1.5625* man-months. The communication period is *0.78* months because the *1.5625* man-months is divided by *2* (two kinds of works). Total development period is *5.78* months because the *0.78* months is added to the *5* months. In sub-case(3), it is an example of sharing same work in COM2 and dividing works in COM1. The creation time of the concept model is *1.5* months because *3* man-months are divided *2* (two developers). However, communication workload between the developers occurs during the creation of the concept model. The communication workload (COM 1 + COM2) is about *2.6* man-months using the proposed model. The development period is about 4.8 months. The formula is as follows:

**Table 3 Results of case 1**

| Case | $Inc_{workload}$ (man-month) | $COM1_{num}$ (num. of times) | $COM2_{num}$ (num. of times) | Total workload (man-month) |
|---|---|---|---|---|
| (1) a developer | 0 | 0 | 0 | 5 |
| (2) a developer create use-cases<br>a developer create a concept model | 1.5625 | 125 | 0 | 5.7813 |
| (3) two developers create a concept model | 2.6092 | 125 | 502 | 7.6092 |
| (4) three developers create a concept model | 5.7388 | 125 | 2003 | 10.7388 |
| (5) four developers create a concept model | 9.3825 | 125 | 4504 | 14.3825 |

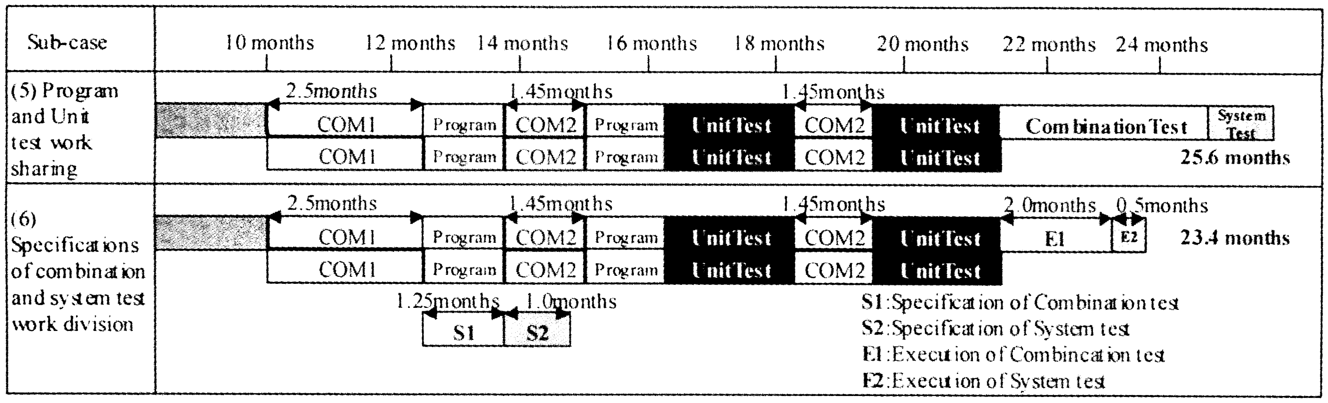
Sub-case
10 months
12 months
14 months
16 months
18 months
20 months
22 months
24 months
(5) Program and Unit test work sharing
2.5months
1.45months
1.45months
COM1
Program
COM2
UnitTest
CombinationTest
System Test
25.6 months
(6) Specifications of combination and system test work division
2.0months
0.5months
E1
E2
23.4 months
1.25months
1.0months
S1
S2
S1:Specification of Combination test
S2:Specification of System test
E1:Execution of Combincation test
E2:Execution of System test

**Figure 4 Project plans of sub-case (5), (6) in case2**

$2^{*1}+1.56^{*2}/2^{*3}+1.04^{*4}/2^{*5}+1.5^{*6}= 4.8$ ***months***

$^{*1}$ :*2* is the development period of use-cases
$^{*2}$ :***1.56*** is communication workload of COM1
$^{*3}$ :*2* means two kinds of works; the creation of the use-case and the creation of the concept model
$^{*4}$ :***1.04*** means communication workload of COM2
$^{*5}$ :*2* means two developers who shared the creation of the concept model.
$^{*6}$ :***1.5*** means ***3*** man-months workload (the creation of the concept model) is divided by *2* (two developers)

Similarly, in sub-case (4) the development period is about ***5.2*** months, in sub-case (5) the development period is ***5.9*** months. Therefore, if a manager want to reduce even slight development period, the manager will adapt the plan of sub-case (3). However, the manager must not add more than two developers. The development periods in three or four developers are more than a development period that a developer executes all works. Adding three and more developers to the creation of the concept model makes it later because the total communication workload in the three and more developers increases drastically. The experiential rule in "Mythical man month" is reconfirmed quantitatively using the results of the case 1.

## 4.3 Case 2

We discuss what's happen in adding new developers near the close of a project. In case 2, it assumes that a manager recognizes a delay of his project near the close of his project. The manager wants to add new developers to the remainders of works because he wants to recover from the delay of the project. The remainders of works are programming (creation of attributes, creation of methods), unit test (specification of test and execution of test), combination test (specification of test and execution of test) and system test (specification of test and execution of test). In sub-case (1), a developer is assigned to all works, in short, no developer is added to the project. In sub-case (2), two developers are assigned to all works, in short, all remainders of the works are shared out two developers. In sub-case (3), three developers share all remaining works in the same way of sub-case (2). In sub-case (4), a new developer is assigned to only the combination test and the system test, in other words, the works of the combination test and the system test are shared to two developers. In sub-case (5), a new developer is assigned to the works in which the remaining quantity of the work is very much such as programming (creation of attributes, creation of methods) and unit test (specification of test, execution of test). The remaining quantity of the work means the numbers of the instances in Table 2. Table 4 shows results of estimation of the case2 using the proposed model.

## 4.4 Discussion of the case 2

**Table 4 Results of case 2**

| Sub-case | $INC_{workloade}$ (man month) | $COM1_{num}$ (num. of times) | $COM2_{num}$ (num. of times) | Total workload (man month) | Percentage of communication in total workload | $Dev_{time}$+10.7 (month) |
|---|---|---|---|---|---|---|
| (1) | 0 | 0 | 0 | 14.25 | 0% | 24.95 |
| (2) | 14.73 | 625 | 3320 | 28.98 | 50.8% | 25.19 |
| (3) | 35.51 | 625 | 13286 | 49.76 | 71.4% | 28.59 |
| (4) | 6.67 | 450 | 502 | 20.92 | 31.9% | 25.91 |
| (5) | 10.88 | 400 | 2818 | 25.13 | 43.3% | 25.64 |
| (6) | 10.88 | 400 | 2818 | 25.13 | 43.3% | 23.39 |

In the case, the project has already consumed ***10.7*** months because the upper process works (creation of use-case, creation of concept model and so on) have been finished. If a developer executes all remaining works in sub-case (1), the development period will be ***14.25*** months because of the ***14.25*** workloads divided by ***1***(one developer). However, if the deadline of the project can not be set after ***14.25*** months, a manager of the project will fall into the temptation to add new developers to his project. However, in all results of the estimation from sub-case (2) to sub-case (5), the development periods with adding new developers are longer than the development period without adding new developers. That is, we have reconfirmed that the addition of new developers near the close of the project is meaningless for recovering from the delay of the project. Especially, in sub-case (3), the percentage of the communication workload in total workload is more ***70 %.*** The high percentage of the communication workload makes the project complicated. The developers of the project consume much time for the communication among the developers. Because the consuming time for the communication is more than the consuming time for executing the works, the developers feel inefficient and boring in such projects.

A manager often adapts a plan such as sub-case (5) to the late project. The manager tends to add new developers to the works that are not finished, because the manager feels that the work sharing is the only way of reducing the development period. However, even if a new developer shared the remaining works with the original developer, the new developer is not useful for recovering from the delay of the project. The results of the case2 in Table4 indicate the miss judgments of the manager under the delay of the project.

Here, we discuss sub-case (5) further. The program work and the unit test work were shared out two developers because the remaining quantities of the works were a lot. The combination test and the system test were not shared because the remaining quantities of the tests were not so much. But if you watch carefully the associations among the remaining works in Figure 1, you will be able to see that there is no association between the specification of the combination test and the unit test (the specification and the execution of the unit test). Similarly, there is no association between the specification of the system test and the unit test. No association between two works means that the works can execute concurrently without the communication workload.

In sub-case (5), if a new developer is assigned to the specification work of the combination test and the specification work of the system test, the new developer can execute his/her works concurrently with the program work and the unit test work that are executed by the other developers. The result of this case is shown as sub-case (6) in Table4. In sub-case (6), three developers are assigned to the works (See Figure 4). The original developer is assigned to a *half* of the program work and a half of the unit test work and the execution of the combination test and the execution of the system test. The new developer is assigned to a half of the program work and a half of the unit test work. The second new developer is assigned to the specification of the combination test and the specification of the system test. The three developers can execute their works concurrently around the 13months or 14months in the Figure4.

Moreover, in sub-case (6), the new communication workload does not occur by the new division of the works; the specification of the combination test and the specification of the system test. It is because that the new work divisions cut no association among the classes of the Concept Model of Figure1. Therefore, the work division in sub-case (6) can reduce the development period from ***25.64*** months to ***23.39*** months without the addition of the new communication workload. The ***23.39*** months is shorter than the ***24.95*** months in which a developer is assigned to all remaining works. Only sub-case (6) is success to recover from the delay of the project.

In the same way of sub-case (6), if a manager tries to simulate in the proposed model many times taking the work division (COM1), the work sharing (COM2) and the executable concurrency among the works into account, the manager will be able to make the most efficient project plan.

## 5. Related works

Many useful models and methods for a software development project management have been proposed.

Komiya et al. proposed a method and facilities to project risks in a software project through Kepner-Tregoe problem, and proposed schedule re-planning by using generic algorithm for avoiding the projected risks [8]. In the method and facilities, users can recognize their project's risks such as a mistake of assigning workers and a mistake of arranging tools for the project, using KT's program. Because workers can be assigned to activities that are defined in a meta-model of work structure, the divisions of the works are clarified, the execution time for each worker is clarified. However, the communication workload is not mentioned when the work is divided to some the workers.

Onishi et al. proposed a communication model for software requirement specification [11]. *In the model*, the communication means that a symbol is transferred from a sender to a receiver. However, it is difficult to communication in three cases. The first case is that the receiver does not know the symbol, the second case is that the symbol whose expression are difficult between the sender and the receiver, the third case is that the symbol whose expression has difficult meanings between the sender and the receiver. In addition, the authors developed a co-operative virtual requirements definition method via network based on the communication model. The model and method are useful to lead incorrect communications to successful communication. In the real project, this is a valuable tool for communications among developers. However, the model focuses on only a way of the communication between developers. The communication workload is not mentioned.

Kusumoto et al. proposed a new model for describing software processes and an estimation method for the quality, cost and delivery date of a software projects [9]. The project model focuses on three key component; activity, product and developer. The process model consists of Activity models using Generalized Stochastic Petri-Net. In the Activity model, the workload can reflect the communication among developers. The communication rate is derived from a function whose parameters are the number of developer, the sum of each developer's experience and the completion rate of input document. Therefore, the increment of the communication workload that occurs by work division is calculated in the proposed model and method. However, they focus on only the work division that is the division of kinds of works (COM1). The distinction between the work division (COM1) and the work sharing (COM1) are not clear.

## 6. Summary

We propose a new workload estimation model that can indicate quantitatively the common sense in "Mythical man month" written by Brooks. Using the proposed model, a project manager can see if a new developer is added to the remaining works of the project. In addition, he/she can see what works the new developer should be assigned to, what works can be in efficient concurrent executions. The manager can make the most efficient plan even if the manager realizes a delay of his/her project near the close of the project.

In future, various important values of the model will be derived from real projects, or experiments. The multiplicities of the relation diagram of works will be determined by counting the relationships of the contents in the documents. $\boldsymbol{H_{time}}$ (average time of once communication) will be determined based on the observation of developers' behaviors in real project or experiments. In addition, the model will be added important human factors. Learning time [3][4][7] that is caused by the work division and the work sharing will be added to the proposed model.

## 7. References

[1] T.K. Abdel-Hamid, "The Dynamic of Software Project Staffing", *Transaction of IEEE, Software engineering,* Vol.15, No.2, 1989, pp. 109-119.

[2] Brooks, F.P, Jr., *The Mythical Man-Month*, Addision Wesley, MA, 1975.

[3] N. Hanakawa, K. Matsumoto, K. Torii, "A Knowledge-Based Software Process Simulation Model", *International Journal of Annals of Software Engineering*, Vol.14, Oct. 2002, pp.383-406.

[4] N. Hanakawa, S. Morisaki, K. Matsumoto, "A Learning Curve Based Simulation Model for Software Development", *Proceedings of the 20th International Conference on Software Engineering*, IEEE Computer Society Press, Japan, April 1998, pp. 350-359.

[5] . .. Hanakawa, H. Iida, K. Matsumoto, K. Torii, "Generation of Object-Oriented Software Process using Milestones," *International Journal of Software Engineering and Knowledge Engineering, Vol.9, No.4,* World Scientific Publishing, 1996, pp.445-466.

[6] N. Hanakawa, K. Matsumoto, K. Torii, "Application of learning curve based simulation model for software development to industry," *Proceedings of the 11th International Conference on Software Engineering and Knowledge,* World Scientific Publishing, Germany, 1999, pp. 283-289.

[7] N. Hanakawa, H. Nogi, "Human Factor-Based Quality Control with Technical Reviews," *Proceedings of the second International Conference on Software Quality*, Union of Japanese Scientists and Engineers, Japan, 2000, pp.563-568.

[8] S. Komiya, A. Hazeyama, "Projecting Risks in a Software Project through Kepner-Tregoe Program and Schedule Re-Planning for Avoiding the Risks", *Transaction on The Institute of Electronics, Information and Communication Enginneers*, Vol.E83-D, No.4, April 2000, pp. 627-639.

[9] S. Kusumoto, O. Mizuno, T. Kikuno, "Software Project Simulator for Effective Process Improvement", *Journal of information Processing Society of Japan*, Vol.42, No.3, Mar. 2001, pp. 396-408.

[10] Larman, C., *Applying UML and Patterns: an introduction to object-oriented analysis and design*, Prentice-Hall, 1998.

[11] A. Onishi, "Visual Software Requirements Specification Technique Based on Communication Model", *Transaction on The Institute of Electronics, Information and Communication Engineers*, Vol.E85-D, No.4, April 2002, pp. 615-622.

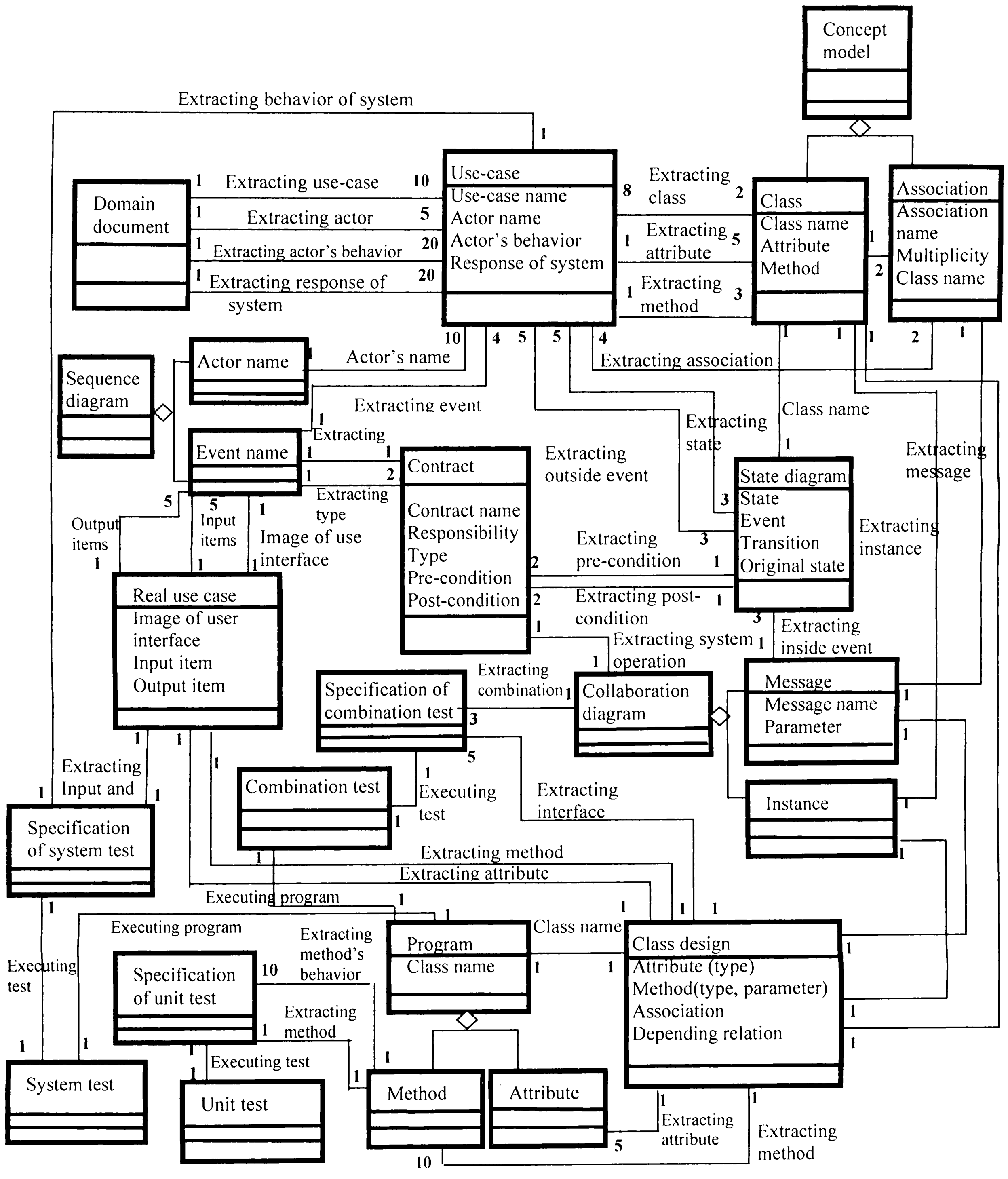

**Figure1 A relationship diagram of works in software development project using an object-oriented**