Comparative Analysis of Cost Estimation for Agent Oriented Software & Traditional Software

Sapna Mahar
Computer Science and Engineering, Guru Jambheshwer University of Science & Technology
Hisar, Haryana, INDIA
sapna_mahar@yahoo.com

Dr. Pradeep Kumar Bhatia
Computer Science and Engineering, Guru Jambheshwer University of Science & Technology
Hisar, Haryana, INDIA
pkbhatia.gju@gmail.com

Abstract - Agent Oriented Software Engineering is an emerging programming paradigm which provides for the natural way of tackling structure and behavior of complex system. The proposed paper proves that Agent Oriented software technique minimize the cost of the software system. In this paper we compare the software cost of agent oriented software technique with reference to traditional software technique and prove that the cost of agent software is less as compared to the traditional one.

Keywords- Agent Oriented Software Engineering, COCOMO II, Cost Estimation

I. INTRODUCTION

The new generation of systems must be able to behave in autonomous, dynamic, unpredictable and flexible manner, also capable to behave in social domains. Agent in fact satisfies all of these demands, and that is why they nowadays have become very popular. Agents are like objects, but its characteristics like autonomy, intelligence, adaptiveness, mobility, pro-activeness, reactivity, etc. differentiate it from objects. No doubt agents play an effective role in software engineering in its life cycle, yet there is no confirmed answer whether agent oriented software are efficient in their size and complexity. The proposed paper collects the data from real agent based project, and cost estimation model applied on it [1]. We estimates the cost assuming the project is based on traditional techniques, and then compare the result with that of an agent-oriented approach, which is less as compared to former.

Software cost estimation techniques forecasts the amount of efforts required to develop a software system. The cost estimates are essential throughout the software development life cycle phases to determine the feasibility of software project and to appropriate the allocation and reallocation of the available resources. The accuracy of estimates is highly dependent on the reliable information required to build the project. The accurate cost estimation reduces the cost of project and enhances the efficiency and effectiveness of project. The cost of the project mainly depends on the nature and characteristic of the project. The recent researches demonstrate on the cost estimates of the agent oriented software and found that there is a significant improvement of the estimates over the prior technologies.

There are various cost estimation models like COCOMO, COCOMO II, or Putnam [2]. These models collect the experience from previous developments to predict the cost of similar developments. With this experience software managers can build estimates of cost, taking into account the variables like average expertise of

programmers, complexity of problem, source lines of code and many more. For agent oriented applications similar type of metrics are needed such as source code of agent oriented applications to make measurements and adaptation of software engineering methods. Both of these issues are little bit tough to handle as some of the applications of agents make their code private whereas conventional metrics approaches are somehow different form that of agents. In this paper we address on two issues first, the estimates done via three European research projects which are multi-agent systems of reasonable size, for definition of metrics they consider a list of agent features that were implemented into that project and integrated them into a COCOMO II model [3]. Second, assuming these projects are based on conventional software engineering, the estimations are done manually.

The rest of the paper is structured as follows. Section II introduces the projects chosen, which the base is. Section III gives the basic concepts of cost estimation with reference to COCOMO II model. Section IV provides some agent properties which are considered as the metrics for cost and size estimation. Section V explores the calculation part. Section VI evaluates the experimental results and finally section VII gives conclusions.

II. PROJECT DOMAIN

Eurecom P815. *Communications Management Process Integration Using Software Agents (1999-2000)* [5]. The above mentioned project was implicated with the role of agents in betterment of the workflow management applications. The main emphasis was on the specification and implementation of agents. The system specifies the interface agent interacting with user and the interoperability with an existing system. Two types of agents are implemented:

- Personal Manager Agent (PMA) who assist to the project manager
- Personal Developer Agent (PDA) for developers in the project

The project involves the Belief Desire Intention (BDI) architecture implemented with Java and JESS (Java Expert System Shell). CORBA was used for the communication purposes i.e. with other agent or with already existing system. More detailed information about the project is available at http://www.eurescom.de/public/projects/P800-series/P815 (IN SCE Case Study prototype).

Eurecom P907. Methodology for Engineering Systems of Software Agents (2000-2002) [6]. This project defines the MESSAGE/UML methodology with the aid of a Travel Assistant Service which includes:

- Personal Assistant Agent that represents as a user in the system
- Information Finder Agent who is responsible for finding information source related with airports
- InfoAENA Agent that encapsulate the airport information source

Service implemented was the notification of flights incidences. User-personal Assistant interaction was web based and implemented with servlets. Servlets communicated user's orders to the personal Assistant through JAVA RMI. Inter-agent Communication was implemented with JADE. The behaviour of the Personal Agent was BDI based, whereas the other agents were reactive.

PS13. Personalized Service Integration Using Software Agents (2001-2003) [7]. This was an IST project where agent based collaborative information filtering system was implemented. The system tried to form virtual communities of users where every registered user received only interesting information. Each user was represented in the system by an agent that knew user's current interests and learned new ones. These agents were named as Personal Agents. They got together into one or many virtual communities, represented with a Community Agent. The challenge here was to evolve communities' topics, managed by Community Agents, as user interests were shifting, and how Personal Agents could learn new topics from the user. Also, whether Community Agents and Personal Agents together could prevent unwanted behaviours lie information spam. This system is more complex than previous. It was made with an undetermined number of agents (the system was tested with up to three hundred heavy agents) and was rather scalable (new computers could be added to increase the number of agents registered). Agents were implemented with CORBA communications and JESS control. This time it was not a BDI control, but a more conventional session oriented control.

These three projects make an excellent data repository to extract conclusions upon. Average time per project was more than a year involving at least two developers at the same time. More information about these projects is available at http://grasia.fdi.ucm.es .

III. COST ESTIMATION MODEL COCOMOII

A series of mathematical formulae are used to predict effort based on software size and other factors [2]. The COCOMO model uses exponential component in estimation because, normally efforts do not increase linearly with size. As the size of software increases, the development team size increases, increasing system development overheads. The latest version of COCOMO model is COCOMO II. COCOMO II has three estimation models to estimate effort and cost. The models are Application Suite, Early Design and Final Design Architectural Model. In this paper we are using Early Design model for estimating the effort and cost of agent oriented software. The Early Design stage involves exploration of alternative software architectures and concepts of operation. At this stage, not enough is generally known to support fine-grain cost estimation. The corresponding COCOMO II capability involves the use of function points and a set of 5 scale factors and 7 effort multipliers.

A. Cost Factors: Sizing

The cost estimation of any project is usually measured in terms of size, effort and time. With reference to the estimation model COCOMO II [2], the size of the project can be expressed in the following ways:

- Source Lines of Code (SLOC):

A line of code refers to the logical source statement i.e. a smallest code a programmer can write. It is measured as thousand lines of code (KLOC). It is difficult to define a line of code due to conceptual differences involved in accounting for executable statements and data declarations in different languages. To belittle this difficulty, the Software Engineering Institute (SEI) definition checklist for a logical source statement is used in defining the line of code measure.

- Unadjusted Function Points (UFP):

The function point cost estimation approach is based on the amount of functionality in a software project and a set of individual project factors. Function points are useful estimators since they are based on information that is available early in the project life cycle. Usually UFP are translated into SLOC. COCOMO II counts the ways an application communicates with the user or environment by categorizing into: External input (EI), External Output (EO), Internal Logical File (ILF), External Interfaces File (EIF) or External Inquiry (EQ). Each instance of these function types is classified by complexity level. The complexity levels determine a set of weights, which are applied to their corresponding function counts to determine the Unadjusted Function Points quality. This is the Function Point sizing metric used by COCMO II.

B. Scaling Drivers

COCOMO II Model use the base equation

$$PM = A \times [Size \qquad (1)$$

Where, PM stands for Person Months, A is the constant representing nominal productivity and B is the scaling factor. Size is in KLOC. The Early Design model uses KSLOC for size. Unadjusted function points are converted to the equivalent SLOC and then to KSLOC. The emphasis in the COCOMO II model is on scaling factors, which together give rise to 'B'. The model uses five factors for arriving at economies/ diseconomies in scale, namely Precedentness (PREC), Development Flexibility (FLEX), Risk Resolution (RESL), Team Cohesion (TEAM), and Organisation Process Maturity (PMAT), values for these factors can be look at from the tables presented in COCOMO II software modelling manual[9] . The selection of scale drivers is based on the rationale that they are a significant source of exponential variation on a project's effort or productivity variation. Each scale driver has a range of rating levels, from Low to Extra High. Each rating level has a weight, W, and the specific value of the weight is the scale factor as mentioned above, hence the exponent B used in equation 1 can be determined via the following equation:

$$B = 1.01 + 0.01 \times \sum_{i=1}^{5} I \qquad (2)$$

C. Cost Derivers

In addition to the above mentioned factors there are some more relevant factors that affect the development efforts; these are known as the cost drivers. The Early Design cost drivers are obtained by combining the various Post-architecture model cost drivers. These are:

1. Product Reliability and Complexity (RCPX) is the combination of Software Reliability (RELY), Database Size (DATA), Software Complexity (CPLX) and Documentation (DOCU).

2. Reusability (RUSE)

3. Platform Difficulty (PDIF) is the combination of Time Constraint (TIME), Main Storage Constraint (STOR) and Platform Volatility (PVOL).

4. Personnel Capability (PERS) is Analyst Capability (ACAP), Programmer Capability (PCAP) and Personnel Continuity (PCON).

5. Personnel Experience (PREX) combines Analyst Experience (AEXP), Programmer Experience (PEXP) and Language and Tools Experience (LTEX).

6. Facilities (FCIL) combines Uses of Software Tools (TOOL) and Site Environment (SITE)

7. Required Development Schedule (SCED)

Values for these factors can be look at from the tables presented in COCOMO II software modelling manual [9]. Now the complete formula for estimating the effort in terms of person month is as follows:

$$PM = A \times [size]^{B} \times \prod_{i=1}^{7} EM \qquad \textbf{(3)}$$

Where, EM_i is the effort multiplier or cost factors.

IV. AGENT

In COCOMO II, the final cost is affected by the estimated size of the software, and this size is expressed in SLOC or UPF. There may be some more elements for agent specific indicators, but only relevant for the development are mentioned here [1]:

- A BDI control. This approach includes some goals and these goals are satisfied by tasks.
- Session based control. There should be some protocols to handle conversation with other agents or human users.
- Perception of the environment. Agent sensors are used to determine the system for designing the system.
- Abstract Communications. To abstract from the different technologies, agents were built over a resource layer, which was responsible of implementing technology specific communication artefacts, and offering upwards a homogeneous interface.

  To gather information about these aspects, we have defined several variables:

- *Sociability.* Means how social an agent is. This aspect is measured as the count of interaction specific elements such as:
  - *Interactions*. Being one interaction, for instance, the capability of an agent to engage into one conversation with another agent. Usually, the number of interactions is identified with the number of protocols that an agent is able to understand and follow.
  - *Messages*. This gives an idea of the complexity of the conversations used by the agent. Message means a prototype of information that it is intended to be exchanged along a conversation. Messages sent within loops, for instance, would not count as many messages, since each message would have the same format with subtly different data.
- *Behaviour.* Taking into account that agents can contain either reactive or deliberative behaviours, elements are selected that would be shared by most implementations:

- *Task***.** This element informs of the capabilities of an agent. If, in the agent architecture, the task is a method or a procedure, then the number of such methods is counted. In general, it is more probable to find tasks identified with isolated structures, such as modules, scripts, or classes.
- *Rules***.** Most agents implement their control with rules. General rules are counted directly, without taking care of the purpose of the rule.
- *Goal Management rules***.** These would be rules whose responsibility is to control the lifecycle of goals in the agent. These rules tell what to do when the goal has been achieved or when it has failed. They also may tell which goal to focus on next. Of course, they make sense when the control of the agent is goal oriented.
- *State Machines***.** It is another classic way of implementing agent control. By counting the number of different state machines and the states they consider, we have an initial idea of the complexity of the behaviour of the agent.
- *Information.* Signify how the agent perceives the world, and how he manages to take decisions.
- *Mental entities***.** This counter is related with cognitive agents having a mental state. By counting mental entities, an integrated numeric representation of how the agent represents its environment is obtained and how precise is the control of the agent. This serves also as a generalization of the counting of events and goals, which are more specific measures that may not be applied in concrete representations.
- *Events***.** It refers to the perception of the agent. Whatever information that is perceived from the environment takes the form of events, once the agent start processing it. Here, depending on the paradigm, events are implemented as predicates in a knowledge base, specialised classes, or, perhaps, only as strings.
- *Goals***.** Goals are special mental entities that are dedicated to control purposes in most cognitive agents. Counting them gives an estimation of how complex is the control of a cognitive agent.

With the above mentioned data, we can apply the COCOMO II model as section V indicates in order to obtain the manpower and the expected time for a given project.

V. CALCULATIONS

A. Statistical Data

The various data used in the projects is considered in this section. The differences between conventional statistics i.e. object oriented implementation and agent statistics related with the agent oriented implementation is also given. This data will be used we evaluate COCOMO II models and to find their results.

Data given in “Table 1” were used to adjust different variables and make them fit with the actual costs of the project. On applying (3) on the SLOC Logical Lines, we get the result presented in “table 7”.

B. Applying Estimation Models

The Early Design Development Model of COCOMO II is configured with the following considerations:

- The project was structured as a single model which represents the whole system, size is taken as SLOC determined by logical SLOC in “table 1”.
- Time for the project was strictly limited by the time assigned in the respective project plan.
- To weight the effort, the man power committed is adjusted in order to provide a realistic view of what effort is required.
- Maintenance effort was not considered since project prototype need not to be maintained. Project was assumed to be based on Waterfall development process.
- The scale factors applied to the project is as shown in “Table 4”. Values for the scale factors are extracted from the table given in COCOMO II software modelling manual [9].

- The Cost factors applied to the projects are as shown in "Table 5". Values for the cost drivers are obtained by adding the values of at least two or more other factors given in post architectural model as discussed in section III.

From this information, we can have an estimation of an agent based project by knowing its initial specification in terms of events, tasks, goals, state machines and expected rules. There size can be converted into SLOC for the estimations.

When we are not having accurate evaluation technique for estimation size, proxy can used for an estimation technique for predicting the result. A proxy is a value that is much easier to estimate or predict than the output value.

The events, rules, goal, task and state machines count can be such proxies for the size estimation. Total SLOC for these elements can be calculated by equation,

$$\textbf{Total SLOC} = A_E \times E + A_R \times R + A_G \times G + A_T \times T + A_S \times S \quad (4)$$

Where A_E is average event size in SLOC and E is the number of events, A_R is average rule size in SLOC and R is the number of rules, A_G is average goal size in SLOC and G is the number of goals, A_T is average task size in SLOC and T is the number of tasks and A_S is average state machine size in SLOC and S is the number of state machines.

On applying (3) on each project individually we find the result as mentioned in "table 7" and we also find that the estimated cost for agent oriented system is much less than as compared to the traditional system.

Table 1 Statistical Data About The Implementation Of Project P815, P907 & Ps13

Element	**P815**	**P907**	**PS13**
Number of Classes	482	172	130
Number of Packages	45	31	23
Average methods per class	5.17	4.09	5.3
SLOC Logical Lines	**15843**	**5393**	**9862**
SLOC Physical Lines	20009	7007	13102

Table 2 Data Refer To Elements Commonly Associated With Agents Whose Control Is Expressed With Rules.

Element	**P815**	**P907**	**PS13**
Total number of Interactions with other agents	3	5	4
Total number of Messages interchanged	15	19	11
Total number of events considered	61	10	10
Total number of rules	198	48	39
Total number of tasks	71	9	39
Total number of state machine applied	5	10	8
Total number of states in every machine	13	46	37

Table 3 Statistical Data About BDI Behaviour

Element	P815	P907
Total number of Events(E)	61	10
Total number of Facts(F)	8	3
Total number of Goals(G)	135	29
Total number of types of mental entities (E+F+G)	204	42
Rules dedicated to management of metal entities	190	46

Table 4 Scale Factors Applied To The Projects

Scale Factors	P815	P907	PS13
Procedentness	Nominal	High	High
Development Flexibility	High	High	Nominal
Architecture/ Risk Resolution	Extra High	Extra High	Nominal
Team Cohesion	High	Very High	Very High
Process Maturity	Low	Nominal	Nominal

Table 5 Cost Factors Applied To The Projects

Cost Drivers	P815	P907	PS13
Product Reliability and Complexity	High	Nominal	Very High
Reusability	Very High	Very High	Nominal
Platform difficulty	High	Very High	High
Personnel Capability	Nominal	Nominal	Nominal
Personnel Experience	Low	Nominal	High
Facilities	Nominal	High	High
Required Development Schedule	Nominal	Nominal	Nominal

Table 6 Equivalence Of Each Elements Into Sloc

Element	P815	P907	PS13
Event	443	86	172
Rule	2130	923	1047
Goal	1581	110	
Task	793	520	303

State Machines	142	691	1048
Total SLOC	**5089**	**2330**	**2570**

Table 7 Values Evaluated For Traditional And Agent Software

Project	Traditional(in PM)	Agent(in PM)
P815	196	54
P907	31	12
PS13	65	17

VI. EXPERIMENTAL EVALUATIONS

To evaluate the cost for agent software, we selected the set of concepts introduced in section IV and measured their implementation cost in SLOC, as shown in table 6. We calculated the cost using COCMO II for all the projects and the results are shown in table 7. With the values we have obtained, we can clearly see that the cost of agent software is much less as compared to the traditional one.

The pictorial representation of "table 7" is as shown as in "Fig 1".

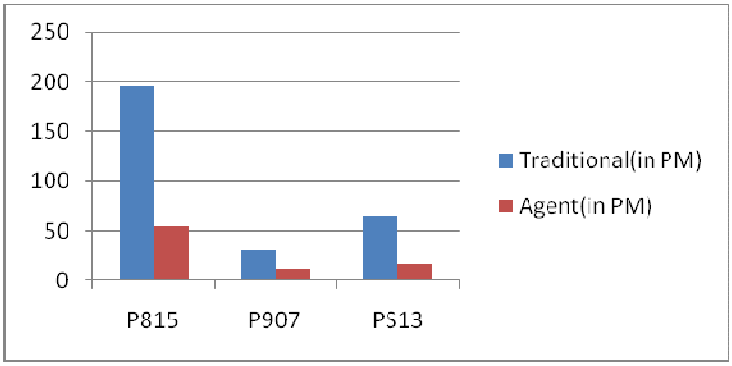

Figure 1. Comparison of cost estimation (in Person Month) between Agent Software and Traditional Software.

VII. CONCLUSION

Estimation of the cost of software is not an easy task and it become more tedious when we talk about agent oriented software because yet a very few researches have been done in this context. The goal of this paper is to set an alternative for traditional software engineering practices. The experiment includes the statistical data of three projects, and the estimation of SLOC for each element of these projects and inclusion into a COCOMO II model. We found that the cost of agent Oriented Software is less as compared to Traditional Software. We need some more data and further experimentation with COCMO II, this paper is just the first step. The data collected is not sufficient for the accurate estimations. We still have to collect more data form more projects so that we can have database of cost associated with each projects and find more accurate measures. Hence, if more data about projects exist, the better we can foresee the cost of new multi-agent systems.

REFERENCES

[1] Jorge J. Gomez-Sanz, Juan Pavon, Francisco Garijo, "Estimating Costs for Agent Oriented Software", Agent Oriented Software Engineering VI, 218-230(2006).

[2] Boehm, B.W., Sullivan, K.J., "Software Economics: a roadmap", Proceedings of the Conference on the future of Software Engineering. ACM Press 319-343(2000).

[3] Boehm, B.W., Sullivan, K.J., " Software Cost Estimation with COCOMO II". Prentice Hall (2000).

[4] Gomez-Sanz, Pavon J., Garijo F., "Intelligent Interface agents Behaviour Modelling". MICAI 2000: advances in Artificial Intelligence. Lecture Notes in Computer Science, Vol. 1793, Springer Verlag (2001) 598-609.

[5] Caire G., Evans R. Massonet P., Coulier W., garijo F. J., Gomez J., Pavon J., Leal F.,Chainho P., Kearney P.E., Stark J., "Agent Oriented Analysis using MESSAGE/UML", In The second International Workshop on Agent-Oriented Software Engineering(AOSE 2001). Lecture Notes in Computer Science, Vol. 2222, Springer-VErlag 119-135(2002).

[6] Gomez-Sanz, Pavon J., Daiz Carrasco, "The PSI3 Agent Recommender system", International Conference on Web Engineering (ICWE 2003), Lecture Notes in Computer Science, Vol. 1793, Springer Verlag 30-39(2003).

[7] W.S. Hum,W.S. Humphrey, "A Discipline for Software Engineering", Addison-Wesley Publishing Company, Reading, MA, 1995.

[8] USC COCOMO II application v2000, Available only from the support CDROM of 1. V 1999 available from http://sunset.usc.edu/available_tools/index.html.

[9] CodeCount.http://csse.usc.edu/research/CODECOUNT