An Agent Based Approach for Effort Estimation in Production Support

Ashish Kumar Jha
System Research Laboratory
Tata Research Development and Design Center
Pune, India
Email: ak.j@tcs.com

Abhinay Puvvala
System Research Laboratory
Tata Research Development and Design Center
Pune, India
Email: abhinay.puvvala@tcs.com

Veerendra K rai
System Research Laboratory
Tata Research Development and Design Center
Pune, India
Email: veerendrak.rai@tcs.com

Sanjit Mehta
System Research Laboratory
Tata Research Development and Design Center
Pune, India
Email: sanjit.mehta@tcs.com

Harrick M Vin
System Research Laboratory
Tata Research Development and Design Center
Pune, India
Email: harrick.vin@tcs.com

Abstract

Estimating the effort required in resolving an incident for a production support engagement has been a long standing problem. Although, a lot of work has been done in software engineering field in estimating effort required for software development, there is no adequate body of work on production support effort estimation. Reliable effort estimates for handling each kind of incidents can have the benefits of planning the complete project engagement for many services firm which thrive on production support projects. This paper presents a framework for estimating mean effort required to resolve an incident. It takes agent based modelling approach to capture the attributes and interactions of the agents. The framework has been tested with few policy experiments to assess the amount of effort required to resolve the incidents and its impact on SLA compliance. The model has been tested with industry data and holds promise to be a reliable tool to conduct policy experiments and enhance resource utilization.

Keywords

Software Maintenance, Production Support, Incident management, Effort Estimation, Optimization, Agent Based Modelling

INTRODUCTION

With increasing automation and thrust for use of software, software projects are rarely small and never simple. A lot of inherent complexities are getting built in owing to its increasing ubiquity. Maintenance of such software encompasses handling incidents, which is an event that is not part of standard operation of a service and causes or may cause an interruption to or reduction in quality of that service" (Kapella, 2003 pp.2). Such a comprehensive software maintenance contract is often called as production support or production management. Average maintenance cost of software over its complete life cycle can be as much as 90% of the total cost (Wiederhold, 2006). In such a scenario the importance of production support becomes immense and has to be carefully factored in while budgeting for the complete software life. A lot of work has been done to estimate the effort required for software development using a lot of approaches from fuzzy logic to optimization (Jorgensen & Boehm, 2009), but there is a relative lack of research literature when we come to the dominant cost component i.e. service maintenance or production support.

For the vendor firm, to fulfil the above stated engagement commitments at minimum costs, it is essential that it has the ability to estimate with high accuracy the amount of effort required. A better resource allocation, enabled in parts by better effort estimation, can help firms optimize their resources and in turn their financials as well. This is the problem that the current work aims to solve by proposing a method for estimating the effort for production support with a high degree of reliability.

In our review of the literature we found a glaring absence of frameworks which helps a firm estimate the effort required for handling each incident at production support level. Since incidents are of different types and serviced by various levels of support team, estimating the effort gets all the more tricky as there is no reliable data of effort put in by employees per incident. In our work we have attempted to present a framework for estimating the average effort required per incident, a bugbear of firms for long, by using a novel framework of triangulating it using the more reliable and available data of employee utilization and SLA compliance of the engagement. We validate the method by testing it against the data of a production support engagement managed by one of the largest software service firms of the world. The results of the simulation show that the method can not only be used by firms for effort estimation but also plan for incident assignment rules.

The paper is arranged according to sections. In the section on "*Agent Based Modelling*" we explain the approach adopted in this work providing a brief overview of its literature and the extant of the work being done by the modelling approach. The section on "*Model Simulation*" explains the simulation technique, the agents and their behaviours and the process of simulation amid the environmental constraints. We explain the results as obtained and analyse the implications of such a model and its importance for firms and industry at large in the section *"Model Analysis and Results"*.

LITERATURE REVIEW

ITIL defines service management as "*a set of specialized organizational capabilities for providing value to customers in the form of services*" (Cannon et al., 2007, pp.273). It is also referred to as Production support and project service in various contexts. Production support is aimed at ensuring maximum uptime for serviced software during the tenure of the engagement (Cannon et al. , 2007). It also aims at service quality. Since there is inadequate body of research in the area of managing production support engagements and estimating efforts, we have drawn from a wide array of literature of relevant fields. Some ideas and concepts of managing software have been drawn in from software engineering discipline; methodologically Agent based Modelling has provided the relevant method literature and also allied project management literature has been touched upon.

A lot of work has been done on estimating efforts for software project implementations over past 30 years. The initial attempts in this domain were based on predicting the estimated amount of code that the software would have in terms of lines of code (LoC) (Finnie et al. , 1997). This approach estimated the effort required in terms of the number of lines required to be written in code. Here the basic assumption was that almost all effort is required in writing code and all lines of code are equivalent. This approach was challenged due to its premise of assuming all lines of a code as similar (Boehm et al., 2000; Li et al., 2009). Later the efforts shifted towards analysing the function of the software in terms of inputs, outputs and the functions that the software is required to perform (Allan & Gaffney, 1983; Finnie et al. , 1997). This approach like the previous one was based on the assumption of equality of efforts over various functions (Briand et al., 1999; Li et al., 2009). Since measuring of effort required based on the functional parameters like Lines of Code or functions did not give very good results, the focus further shifted to use of multiple Project Delivery Rates (PDRs) to derive the effort required (Fujita & Marik, 2009). These were requirement analysis, software design, UI design etc. The use of such metrics mandated the need of experienced professionals to convert likely Use Cases from requirement documents and other such information (Pfleeger et al. , 2005).

All the above mentioned approaches are targeted towards estimating effort required for creating software. We need an approach to estimate effort in incident management environment which is managerially more efficient, in terms that it requires minimum human intervention and interpretation. The approach must be intuitive to project managers of the production support engagements who are the actual decision makers in the production support environment.

A very small strand of literature in software project management also talks about the effort estimation for software maintenance. Various software benchmarking group including IEEE have talked of the activities required for software maintenance. Few models of estimating programming efforts required in software maintenance have been suggested. The major ones of these are Putnam's SLIM, Albrecht's Function Point method of estimation and COCOMO and COCOMO II (Boehm, 1984; Boetticher, 2001). Certain other methods using advanced computational method have also been discussed in literature. Among this analogy based estimation has been one of the most widely used concepts for estimating efforts for software projects. Fuzzy numbers were used to create similarity measure and adaptation technique for analogy based estimation efforts (Azzeh et al., 2010, 2011). A separate strand of literature also utilized evolutionary algorithms for estimating performance of effort estimation at various stages of maintenance (Minku & Yao, 2013). These methods are based out of estimating costs from drivers such as process maturity, reuse etc.

Table 1 summarizes few of the major works in field of software project estimation and software maintenance effort estimation and characterizes their objectives. The last column of the table lists whether the works had scope for estimating the complete production support effort. We found that most of the software maintenance effort estimation only went as far as computing additional programming effort required and were not able to provide a complete estimation for other efforts like response etc. which would be useful for project managers to plan resource requirements. Such analysis can also be used by project managers to design policies for handling of production support engagements.

Table 1: Summary of literature review of effort estimation approaches

Year	**Author(s)**	**Objective**	**Method**	**Production support effort estimation**
1980	Putnam	Software cost estimation	Lines of code based	No
1984	Boehm	Software cost estimation	Lines of Code based	No
1983	Albrecht et al	Programming effort estimation	Function Point based estimation	No
2002	Lucia et al	Software maintenance effort estimation	Analogy based	Only programming effort estimation
2004	Sneed	Software maintenance effort estimation	Fuzzy logic based	Only programming effort estimation
2005	Jorgensen	Software maintenance effort estimation	Fuzzy logic based	Only programming effort estimation
2007	Jorgensen	Effort estimation for product changes	Expert judgement	Only programming effort estimation
2010	Azzeh et al.	Software changes effort estimation	Fuzzy logic based	Only programming effort estimation
2013	Minku & Yao	Performance measures of software effort estimation	Evolutionary algorithm based	Only programming effort estimation

RESEARCH QUESTION

The current work takes a process oriented view of the complete system and there are various steps in the process which are modifiable by the managers. The processes in place are the policies put in place by the management to handle the engagement. So the process oriented view provides better understanding of the impact of change in policy rules on average effort required for the handling each individual issue.

The fact that in any production support engagement there are very few metrics that can be reliably measured and analysed complicates the issue even further as there is no reliable data to compute such factors. Two of the most reliably collected data in any production support engagement are

- Employee Utilization: - It is the percentage of time wherein an employee is working on resolving or responding to an incident of the engagement.
- Service Level Agreement (SLA) Compliance: - The percentage of issues resolved within the stipulated contract time.

With data variables as few as those mentioned above, the aim of the work is to find a method for estimating average effort required in resolving an issue using the above mentioned data. The attempt is to find a metric which can be used easily by middle and higher managers to analyse the production support engagements and not be only programming efforts centric.

AGENT BASED MODELLING

To solve the above mentioned question, we have used a Modelling technique departing from the tradition of neural networks and fuzzy systems used in software engineering literature as this Modelling method makes it possible to take systems or process view. We have used Agent Based Modelling (ABM) because of the fact that we could identify various agents in the incidents and the resources who individually interact with each other under defined processes which are a result of the policies. These interactions give rise to emergent phenomena which needs to be studied and /or analysed.

Agent Based Modelling (ABM) is a relatively new, growing Modelling approach to analyse various phenomena. It is run through simulations of agents. It can be applied to a problem by defining a set of agents with related attributes, behaviours and fitness function; the simulation environment and the overall performance-measuring objectives of the environment. Agents and interactions between them are two most important things in Agent Based Modelling. Every Agent has certain attributes, rules/actions, goals and decisions to make. There can be many different breeds of agents. Each set of agents is generally governed by a fitness function, which again creates heterogeneity by differences in parameters of the fitness function. Sometimes these agents act independent of each other and on other occasions they interact with each other while competing or collaborating towards their individual goals. As a result of countless interactions new behaviour 'emerges' which had not been programmed into the behaviour of the individual agents (Waldrop, 1992). Agent based Modelling has already been extensively used in economics (Agent Based Computational Economics (ACE)). Zaffar et al. (Zaffar et al. , 2008) used it to identify the impact of Variability of Open Source Software (OSS) support costs, length of upgrade cycle and interoperability costs on OSS diffusion.

The implications of ABM results on theory development are also well established. Newell and Simon (1972) established that if a particular instance of Agent based model A gives result R, then the sufficiency of theorem "R if A" has been established for deterministic models. The principle offered by Newell and Simon (1972) works very well for deterministic cases, where the context does not change according to time. For stochastic model, where the dynamism in the environmental context is higher, multiple realizations are necessary (Axtell, 2000). The averages of multiple realizations helps establish theory even in case of dynamic environment by appropriating the results over multiple runs.

RESEARCH MODEL

We would first explain the model of engagement that has been studied in this work. This model of engagement, although based on the engagement studied for validation, is generic enough and can be viewed as a sample model used across the industry. In particular, this model represents the engagement for which the sample data was analysed. This engagement as mentioned earlier was handled by one of the largest software services firm of the world and the client was a leading global investment bank.

A typical production support engagement is aimed at ensuring minimum downtime for the software under service. The typical industry average Service Level Agreements (SLAs) for acceptable uptime ranges between 95% -99% and varies according to the criticality of the application for the firm among other factors.

The various terms and entities in the model are:

- Incident: - International Network Services' Incident and Problem Management Framework (Kapella, 2003, pp. 2) defines incident as "*an event that is not part of standard operation of a service and causes or may cause an interruption to or reduction in quality of that service*". Incidents are the only type of tickets considered for this model. We have observed following 2 types of incidents in the engagement and the same has been considered in the model.
    - Commoditized Incidents: - These are the generic incidents which do not require any special skills to be resolved. They might even be resolved with Standard Operating Procedures (SOP) documentation that firms maintain or may need to maintain if they observe frequent occurrence of these types of incidents
    - Technical incidents: - Technical incidents are resolved at software code or architecture level and ask for specialized skills resources such as application developers and programmers.
- Resources: - Resources are the employees who work on incidents to resolve them. Resources are categorized according to their skill level and skill type. Since the service to the client has to be provided continuously for 24 X 7, there are 3 sets of resources working in 3 shifts of 8 hrs. each. There are three types of resources in this model as explained below:
    - Service Desk Resources: - The service desk resources are only supposed to follow SOPs and respond to the incidents generated by the users. They might be able to resolve some of the incoming commoditized incidents using SOPs but none of the technical incidents.
    - Resolution team level 1 (R1) :- The level 1 employees of the resolution team are more skilled employees than the service desk employees and can solve all commoditized incidents.

- Resolution team level 2 (R2): - The level 2 employees of the resolution team are the highest skilled employees of the incident management team and they can resolve all type of incidents.

Using the above entities as defined we have created an Agent Based Model with following agents, interactions and the attributes.

Agents

- Resources: - Resources have been treated as an agent in the model with following attributes
  - Resource ID:- A unique identifier for each resource
  - Resource Type:- Denotes the type of the resource i.e. Service desk or Resolution team
  - Level of Resource: - Denotes the level to which the resource belongs i.e. level 1 (R1) or level 2 (R2) as explained in section above.
  - Incident ID: - ID of the incident to which the resource is currently assigned.
  - Shift of the resource: - Denoted by 1, 2 and 3 for resources working in 0000 hrs. to 0800 hrs.,0800hrs. to 1600 hrs., 1600 hrs. to 0000 hrs. respectively.
- Incidents: - Incidents are the central agent in the model. Table 2 describes all the attributes and the description for the incident agent

Table 2: Attributes of incident agent and its description

Attribute	Description
Incident ID	Unique identifier for each Incident
Priority	Priority of the Incident. It can range be one of Critical, High, Medium and Low
Incident Type	Type of the Incident i.e. Commoditized or Technical
Effort Required	It is the maximum effort required to resolve the Incident
Effort Required SD	It is the maximum effort required at service desk to either resolve the Incident using SOPs or log and respond to the Incident
SLA Time	Time within which the Incident needs to be resolved to fulfil SLA obligation
SLA SD time	Time within which the Incident needs to be responded by service desk
Generated Time	The time at which the Incident was generated by the client
Assigned Time SD	Time when the Incident was picked up at service desk
Logged Time	Time when the Incident was logged for resolution to wither R1 or R2 resource
Assigned Time	Time at which the Incident was assigned to the last resource working on the Incident
Time remaining	The amount of effort required to be put in on the Incident for it to be resolved
Resolved Time	Time at which the Incident was fully resolved
Resolution Time	Time difference between Incident generation and Incident resolution
Respond ID	ID of the service desk resource who last worked on/responded to the Incident
Resolved ID	ID of the resolution team resource who last worked on/responded to the Incident
SLA compliance	Whether the resolution time was lower than SLA time or not
SLA SD compliance	Whether the responded time was lower than SLA SD time or not

Policies for Interaction of Agents

The agents mentioned above interact with each other based on the following definitions and rules

- Generation of incidents: - Based on our analysis of various engagements data we found that incidents are generated according to a Poisson distribution of a given mean which varies from one engagement to other.

The incidents are distributed between the various priority levels and also between commoditized and technical.

- Service Desk actions: Service desk resources are the first point of contact for the incidents raised. These resources allot the tickets to the resolution team or it attempts to resolve the incident, if possible, by use of Standard Operating Procedures (SOPs). The two possible paths are shown in figure 1 with black arrow markings.
- Resolution of Incident - An Incident can be resolved by a Service Desk resource with a given probability if it is of type commoditized. Commoditized Incidents not resolved at service desk are allotted to the resolution team following the incident assignment policies which state that a commoditized incident can be resolved by both R1 and R2 resource but should be allotted to R2 resource only if R1 is not free while a technical incident can be resolved by only R2 resource.
- Interruption: - Interruption meant an incident's resolution could be stopped midway and the resource could move to resolving any other incident based on some pre-specified interruption condition. A section in this paper is dedicated to investigate the impact of different interruption conditions on effort to resolve and consequent SLA compliance.
- Handover Policy: - In case of interruption an handover policy was implemented which implied that if an interrupted incident was later picked up by the same resource which was earlier working on it, then the time required for the resource to resolve the incident would be the time remaining from last attempt. However, if the interrupted incident goes to a different resource then the new resource would take the time remaining to solve the incident plus handover time. The handover time has been introduced based on our first hand analysis of incident handling process where the new resource has to go through the work already done on the incident, understand the approach and then restart. Handover time would also be implemented when incidents are handed-over at the turn of the shift.

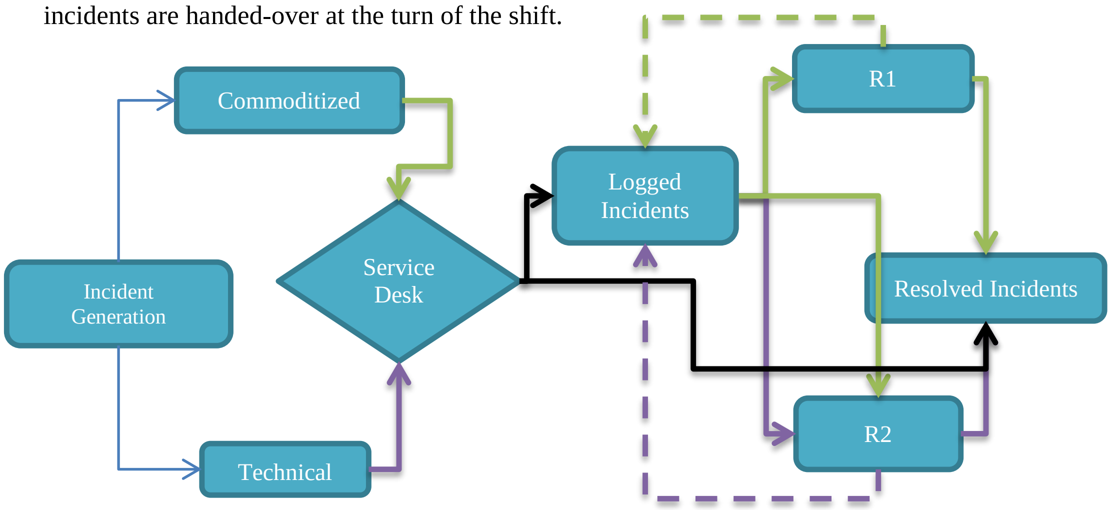

Fig 1: Model of the Incident flow handling

MODEL SIMULATION

The model has been built and simulated using Scilab® software. Scilab® is an open source and free to use software with powerful optimizing toolboxes as well as a large usage. Scilab® was chosen for its ease of use, free availability and compatibility with MATLAB®, a widely used tool.

Model parameters simulation

We have run the model based on the data of a production support engagement of the firms as mentioned above. The simulation was run for a period of 60 days each day having 3 shifts for employees. Details for simulation parameters for base case are mentioned in Appendix 1. We simulated the model for different managerial policies to check for the levers that can be most useful to managers while designing a production support engagement. A **total of four engagement** policies were simulated and the optimized mean effort was computed

Estimated effort: After analysing a lot of effort and utilization data from few engagement projects we found that the effort per incident broadly follows a power law function and is true to the statement that "Most of the incidents can be resolved in least time". On an average 80-90% of the incidents can be solved in 15-20 minutes of the time, but some technical and complex incidents may take inordinately long time leading to a long tail for the function. Figure 2 shows a sample incident distribution and time that incidents take to be resolved. Based on this analysis we estimated effort time using power law function.

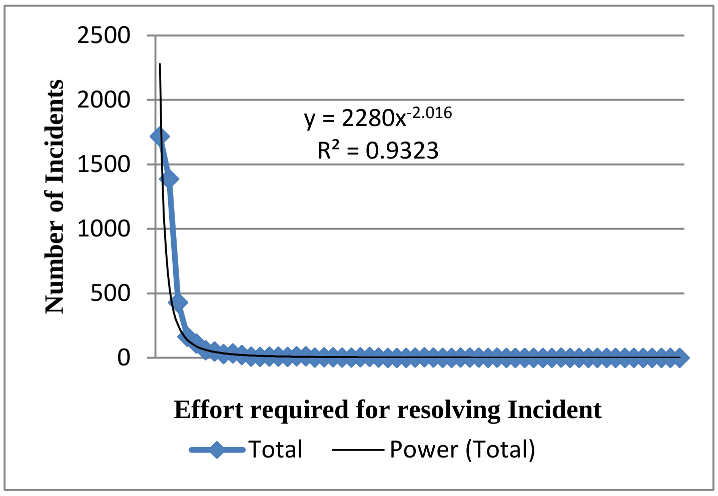

Figure 2: Distribution of incidents and the time required to solve them

Data

The data for analysis was drawn from an engagement between one of the largest banks and one of the largest IT services providers of the world. The data is a log of the incidents resolved by the vendor, IT services provider, for the client, the large bank. We have individual incident ID and the following timestamps a) time of incident creation by client's users b) Time incident was logged and allotted to resolution team by Service Desk c) time when the incident was interrupted (if any) d) time when the incident was resolved. The total resolution time is taken as the difference between the time incident was created by the user and the time incident was actually resolved. This way we also take into account the parking time wherein the ticket is registered by the client but not yet acknowledged by the vendor. The SLA compliance of each incident is also recorded in the data in terms of whether the incident was resolved within the specified SLA requirements. The resource data has records of all resources by their resource IDs, the incident IDs which the resource was working on by each timestamp. Utilization of each resource is computed based on the percentage of time wherein the resource was engaged with an incident.

Optimization

The aim of the optimization routine was to get the optimum values of mean effort required in resolving incidents. Optimization was performed in Scilab® using Limited Memory-BFGS algorithm which is a limited memory approximation of Broyden-Fletcher-Goldfarb-Shanno algorithm of quasi Newtonian family of algorithms (Nocedal, 1980). The method uses iterative runs of the function to find the optimum value. Byrd et al (1994) have mentioned it as an algorithm of choice while modelling complex functions with limited memory.

In our implementation we have attempted to find the optimum value of the mean effort required by triangulating it with the recorded reliable values of SLA compliance and utilization of resources. The aim was to bring the simulated values of utilization and SLA compliance closer to the actual values.

$$sse = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(\left(U_{ij} - \bar{U}_{ij}\right)^2\right) + \sum_{i=1}^{n}\sum_{k=1}^{l}\left(\left(SLA_{ij} - \overline{SLA}_{ij}\right)^2\right), \qquad \forall\, i, j, k$$

Where,

U_{ij}= Simulated average utilization of all employees of type j on day i

$\bar{U}_{ij}$= Actual average utilization of all employees of type j on day i

SLA_{ij} = Simulated SLA compliance of all incidents resolved of priority k on day i

$\overline{SLA}_{ij}$ = Actual SLA compliance of all incidents resolved of priority k on day i

MODEL ANALYSIS AND RESULTS

The assignment rules for the simulation are explained below and the results of their simulation based on simulation settings and parameters given in appendix 1 are tabulated in Table 3

Resolving incidents without interruption

This assignment rule indicates that once a resource has been assigned on an incident, it would continue to work on that incident till it is resolved and then only move to the next incident. As shown in figure 1, the dotted links from resources R1 and R2 to Logged Incidents would not exist in this case. This assignment rule was very prevalent in early days of the service industry and is till sometimes used in situations where the engagement is small and expected number of incidents are small.

Priority based interruption

This assignment rule attempts to solve incidents of higher priority first. This assignment rule is very commonly followed in situations where penalty for not meeting SLA compliance of more critical incidents is higher than lesser critical incidents. In such a scenario, if an incident with higher priority is waiting for lack of resources and a resource is resolving a low priority incident then that resolution is interrupted and handover rules explained in section 4 follow. One disadvantage of such an interruption scheme is that there might be situation when resources are scarce and incident volume moderate to high then lower priority incidents would more often than not starve in absence of available resources.

Interruption based on Time to SLA expiry

The disadvantage of priority based interrupt is that it might lead to starvation of low priority incidents. To deal with such situations a different assignment policy is sometime followed which is based on the time to expiry of SLA. In this policy, an incident being resolved by a resource is interrupted if an incident already exists in logged incidents queue whose time to SLA expiry is very near. This policy ensures that incidents of all priority are attempted to be resolved within SLA and does not cause incidents to inordinately starve.

Table 3: Results of the simulation

Data points	Resolving incidents without interruption	Priority based interruption	Interruption based on time to SLA expiry	Actual Values from data
Mean effort for response time of commoditized incident	12.213 min.	10.048 min.	10.718 min.	N/A
Mean effort for response time of technical incident	9.5423 min.	8.9795 min.	9.213 min.	N/A
Mean effort for resolution time of commoditized incident	15.9722 min.	11.981 min.	13.841 min.	N/A
Mean effort for resolution time of technical incident	18.312 min.	14.91 min.	15.323 min.	N/A
Average resolution SLA compliance for priority Critical	94.35%	100%	99.5%	99.6%
Average resolution SLA compliance for priority High	95.69%	99.74%	98.1%	98.4%
Average resolution SLA compliance for priority Medium	90.46%	97.01%	97.8%	97.2%
Average resolution SLA compliance for priority Low	94.24%	94.56%	97%	96.4%
Resource utilization at Service Desk	59.06%	63.43%	66.26%	65.4%
Resource utilization R1 resources at resolution desk	90.03%	87.12%	89.32%	87.4%
Resource utilization R2 resource at resolution desk	89.32%	90.4%	92.87%	90.1%

As we can observe from table 3, the assignment rule of interruption based on time to SLA expiry ensures that all priority of incidents are as close to SLA compliance levels as possible but this entails too many hand-overs of the incidents due to frequent interruptions. In this rule, all incidents are treated as equal and those incidents whose time to SLA expiry are resolved first. Since time itself is a dynamic element, so an incident which can be interrupted now comes closer to SLA expiry after some time. Thereby creating a situation where almost all priority incidents are treated fairly and equally and all priorities are as close to SLA levels as possible. But the catch is that multiple hand-overs lead to extra effort in resolving incidents due to hand-over time taken, as explained above. This in turn leads to higher effort time required by the resources.

The assignment policy of priority based interruption, on the other hand, gives definite priorities to the incoming incidents. So, a high priority incident cannot be interrupted by a lower priority incident. This ensures that number of hand-overs required is lower than what it was in the other case. Since, critical incidents are more important to business processes by definition themselves, so resolving these incidents faster has higher incentive for all the parties. This interruption rule ensures lower average effort due to lesser number of hand-overs but has a disproportionate SLA compliance for more critical incidents as compared to lesser critical incidents.

CONCLUSION

The results obtained in this paper indicate that the framework for estimating mean effort time using the very easily available utilization and SLA compliance data with the services firms gives a more composite estimate of mean of the effort time required per incident. Also we found that some assignment policies are more efficient than others and having a method to calculate the mean effort time beforehand can provide the managers of such engagements to better utilize the resources available and design better assignment policies. In this sample engagement studied we find that interruption based on priority of the logged incident gives the best SLA compliance. The work presented in this paper is an effort in a direction to stimulate research on estimating production support efforts. This is very desirable for managers of services firms of a 100 billion dollar global industry to better manage resources. The work currently is limited by the small number of assignment policies analysed and also on few small datasets of some engagements. The work can be further bolstered by analysing larger engagements and more assignment policies of the services firms.

REFERENCES

Albrecht, A. J., & Gaffney, J. E. 1983. “Software function, source lines of code, and development effort prediction: a software science validation”. *IEEE Transactions on Software Engineering,* (9:6),November, pp.639-648.

Axtell, R. 2000. “Why agents?: on the varied motivations for agent computing in the social sciences”, *Washington: Center for Social and Economic Dynamics*, Working paper Series. (17), November

Azzeh, M., Neagu, D., & Cowling, P. I. 2010. “Fuzzy grey relational analysis for software effort estimation”. *Empirical Software Engineering*, (15:1), pp.60-90.

Azzeh, M., Neagu, D., & Cowling, P. I. 2011. “Analogy-based software effort estimation using Fuzzy numbers”, *Journal of Systems and Software*, (84:2), February, pp. 270-284.

Boehm, B. W. 1984. “Software Engineering Economics:, *IEEE Transactions on Software Engineering*, (10:1), January, pp.4-21.

Boehm, B., Abts, C., & Chulani, S. 2000. “Software development cost estimation approaches—A survey”. *Annals of Software Engineering*, (10:1-4), pp. 177-205.

Boetticher, G. D. 2001. “An Assesment of Metric Contribution in the construction of a Neural Network Based Effort Estimator”, *Second International Workshop on Soft computing applied to Software Engineering*, Enschade, NL.

Briand, L. C., El Emam, K., Surmann, D., Wieczorek, I., & Maxwell, K. D. 1999. An assessment and comparison of common software cost estimation modeling techniques. In Proceedings of the 21st international conference on Software engineering .pp. 313-322. ACM.

Byrd, R. H., Nocedal, J., & Schnabel, R. B. 1994. “Representations of quasi-Newton matrices and their use in limited memory methods”, *Mathematical Programming*, (63:1-3), January, pp.129-156.

Cannon, D., Wheeldon, D., & Sharon, T. 2007. ITIL.[4]. *Service operation*. TSO (The Stationery Office).

De Lucia, A., Pompella, E., & Stefanucci, S. 2002. “Effort estimation for corrective software maintenance”, *Proceedings of the 14th international conference on Software engineering and knowledge engineering,* ACM. pp. 409-416.

Finnie, G. R., Wittig, G. E., & Desharnais, J.-M. 1997. “A Comparison of software effort estimation techniques: Using function points with neural networks, case based reasoning and regression models”, *Journal of systems and software*, (39:3), December, pp. 281-289.

Fujita, H., & Marik, V. 2009. “The role of benchmarking data in the software development and enhancement projects effort planning. New Trends in Software Methodologies, Tools and Techniques”, *Proceedings of the Eighth SoMeT_09*. pp. 106-107.

Jorgensen, G., & Boehm, B. 2009. “Software Development Effort Estimation: Formal Model or Expert Judgement?”, *IEEE Software*, (26:2), March, pp. 14-19.

Jørgensen, M., & Sjøberg, D. I. 2002. “Learning from experience in a software maintenance environment”, *Journal of Software Maintenance*, (14), pp.123-146.

Jørgensen, M. 2007. "Forecasting of software development work effort: Evidence on expert judgement and formal models". *International Journal of Forecasting*, (23:3), pp.449-462.

Kapella, V. 2003. *A framework for incident and problem management*. International Network Services.

Li, Y. F., Xie, M., & Goh, T. N. "A study of project selection and feature weighting for analogy based software cost estimation." *Journal of Systems and Software*, (82:2), pp.241-252.

Minku, L. L., & Yao, X. 2013. An analysis of multi-objective evolutionary algorithms for training ensemble models based on different performance measures in software effort estimation. In Proceedings of the 9th International Conference on Predictive Models in Software Engineering (p. 8). ACM.

Newell, A., & Simon, A. H. 1972. *Human problem solving*. NJ: Englewood Cliffs.

Nocedal, J. 1980. "Updating quasi-Newton matrices with limited storage", *Mathematics of computation*, (35:151), July, pp. 773-782.

Pfleeger, S. L., Wu, F., & Lewis, R. 2005. *Software cost estimation and sizing methods: issues, and guidelines*. (269). Rand Corporation.

Putnam, L. H. 1977. "The software life cycle: practical application to estimating cost, schedule and providing". *IEEE Computer Society's First International Computer Software & Applications Conference*, Chicago. pp. 39.

Sneed, H. m. 2004. "A cost model for software maintenance & evolution. Proceedings". *20th IEEE International Conference on Software Maintenance*, IEEE. pp. 264-273.

Waldrop, M. M. (1992). *Complexity: The Emerging Science at the Edge of Order and Chaos*. New York, NY: Touchstone.

Wiederhold, G. 2006. "What is your software worth?" *Communications of the ACM*, (49:9), September, pp. 65-75.

Zaffar, M. A., Kumar, R. L., & and Zhao, K. 2008. "Diffusion Dynamics of Open-Source Software in the Presence of Upgrades: An Agent-Based Computational Economics (ACE) Approach", *Twenty Ninth International Conference on Information Systems (ICIS 2008)* Paris: Association for Information Systems. Paper.55.

APPENDIX 1: THE SIMULATION PARAMETERS

- Incident Generation:- Incidents per hour were drawn from a random Poisson distribution of mean 40
- Number of resources:- 2 in each shift at Service desk and 1 in each shift at Level 1 of resolution team (R1) and 2 in each shift at Level 2 of resolution team (R2)
- Handover time: 20% time of the time remaining on the incident was the handover time
- Resolution at service desk : 10%
- Distribution of incidents: Commoditized and Technical 50% each. According to priority, the distribution was Critical- 15%, High- 25%, Medium- 40%, Low- 20%.
- SLA Time duration

Priority	Resolution SLA time duration (in Mins.)	Response SLA Time duration (in Mins.)
Critical	40	2
High	60	5
Medium	90	10
Low	150	20

Optimization Settings for Scilab®

- Stopping criteria:- No. of iterations- 80, no. of Function calls- 3
- Initial Seed value of mean effort required : -
  - Response Desk: Commoditized- 7 mins. Technical – 8 mins.
  - Resolution Desk: Commoditized- 11 mins. Technical- 14 mins.

COPYRIGHT