A hybrid simulation model of individual and team performance in software project environment

M. Alshammri and S. Qin

College of Science and Engineering, Flinders University, Australia
Email: shaowen.qin@flinders.edu.au

Abstract: Software development is a human- intensive effort, hence software project success heavily depends on performance of the members of its development team. A team member's performance can be influenced by many factors, such as level of technical skill, motivation, stress level as well as team culture and organizational environment. Understanding the impact of these factors on software project outcome can lead to better project management and human resource management decisions, and as a result improve project success rate. However, it is a non-trivial task to evaluate such impact, due to difficulties in quantifying human behaviour in a complex environment such as that of software project, hence little literature exists on this research topic. This study attempts to explore the impact of individual and team performance on software project outcome with a simulation model that introduces human factors to a virtual software project environment. The simulation model includes the interplay of three aspects typical of a software project environment, namely: a) individual behaviour of each developer and communication among team members; b) software development workflow of activities and project management practices; and c) organizational environment including culture and HR practices. A hybrid model that includes the application of agent-based, system dynamics and discrete event approaches has been developed. Agent-based modelling (ABM) is one of the most suitable techniques for investigating the complexity of agents' behaviour, actions, and interactions, thus allows us to observe the emergent behaviour of each individual agent; system dynamics (SD) modelling is suitable for observing, at an aggregated level, the cumulative and dynamic effect of varying conditions that influence the behaviours of model components; and discrete event simulation (DES) is suitable for capturing work flow under resource and/or other constraints. In our hybrid model, the agent-based component is developed as the core to simulate each team member's behaviour and states of each unit of work, the system dynamics component is used to capture the cumulative mood of each developer while working as a member of the project team, and the DES component is used to simulate the integration process. Various experiments ("what if" scenarios) can be conducted by changing model parameter values and observe the impact of such changes on project outcome. A sample application is provided to demonstrate the impact of team performance under different levels of requirement volatility on a relative scale.

This paper presents the design, development as well as a brief discussion on validation of such a hybrid simulation model. The research not only benefits those who are interested in understanding the impact of individual and team performance on software project outcome, but also those who are considering using hybrid simulation modelling approach to study human factors in other complex social-technical systems.

Keywords: *Hybrid simulation model, agent-based simulation, software project, human factor, team performance*

1. INTRODUCTION

Software projects are complex. This is very much due to the fact that software engineering is intrinsically a collaborative as well as individual human endeavour. It is well known that the human factor is at least as important as technology and process in any technology based business operation, especially organisations for software development. It is people who carry out the activities defined in software development processes by applying both their knowledge and skills. The development team's performance depends on the behaviours and abilities of the team members. The developers' behaviours are, in turn, influenced by their skill level, motivation level, stress level as well as interactions with others in the team and the organizational environment. Understanding the impact of human factors on software project outcome can lead to better project management and human resource management decisions, and as a result improve project success rate.

Simulation models help researchers to understand complex systems and to investigate the system's complexity, relationships and interactions (Jun, Jacobson & Swisher, 1999). Adopting a simulation modelling approach often is the only way to demonstrate the behaviour of system as well as its components in a dynamic environment. It allows users to run various experiments ("what if" scenarios) in virtual time and communicate with stakeholders of the pros and cons of potential changes to the system being modelled.

Simulation modelling has been applied to study various aspects of software development, such as project planning and open source software evolution (Joslin & Poole 2005, Smith et al. 2006). However, little work could be found in the literature on simulation modelling of software development process that includes individual and team performance.

This paper present a hybrid simulation model developed to investigate the impact of the human factors on software development team performance and project outcomes. Agent-based modelling (ABM) is one of the most suitable techniques for investigating the complexity of agents' actions and interactions, driven by predefined rules, thus allows the researcher to observe emergent behaviours of agents and the system; system dynamics (SD) modelling is suitable for observing, at an aggregated level, the cumulative and dynamic effect of varying conditions that influence the behaviours of model components; and discrete event simulation (DES) is suitable for capturing work flow under resource and/or other constraints. In our hybrid model, the agent-based component is developed as the core to simulate each team member's behaviour and states of each unit of work, the system dynamics component is used to capture the cumulative mood of each developer while working as a member of the project team, and the DES component is used to simulate the integration process. Various experiments ("what if" scenarios) can be conducted by changing model parameter values and observe the impact of such changes on project outcome. A sample application is provided to demonstrate the impact of team performance under different levels of requirement volatility on a relative scale.

2. MODEL STRUCTURE

2.1. Key model parameters

The focus of our model is team performance, which is characterised by key parameters presented in Figure 1 using system thinking and casual loop diagram. System thinking helps us to understand complex systems' components and the linkages between those components while causal loop diagram is a powerful tool to capture the feedback relationships between system elements. The parameters are described in Table 1.

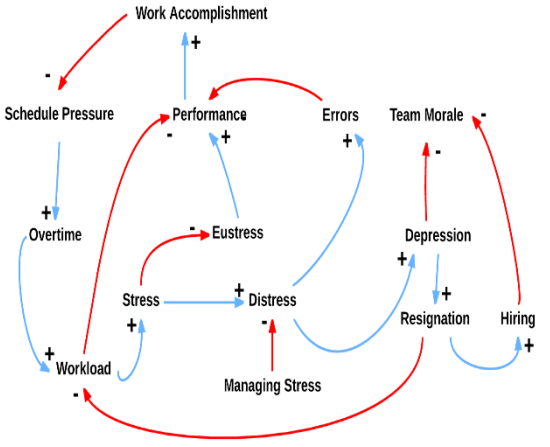

Figure 1. Causal-loop diagram of main aspects influencing team performance.

2.2. The ABM component

We attempted to follow the framework named PARTE (Properties, Actions, Rules, Time, and Environment, Hammond, 2015) for the design of the Agent-based component of our simulation model.

Properties:

A software developer is modelled as an agent with two state diagrams, one represents the agent's physical activity, another the agent's stress. In regards to physical activity, an agent is in one of the three states: available for work, working or in a rehabilitation break (Figure 2). Simultaneously with physical states, a developer agent is in one of the 4 states of stress: normal, eustress, distress and depression (Figure 3). A unit of work is also modelled as an agent. Its state diagram, representing the stages a unit goes through during development, is shown in Figure 4. Other properties of agents were identified as parameters and variables. These are presented in Table 1 due to space constraints.

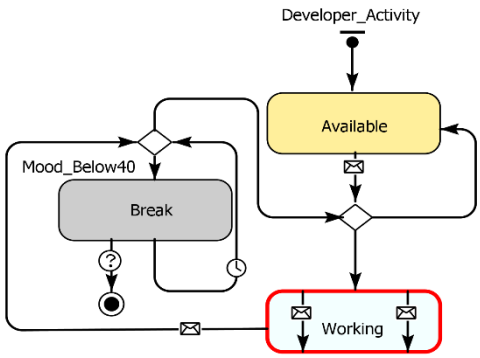

Figure 2. State diagram of a developer – physical activity

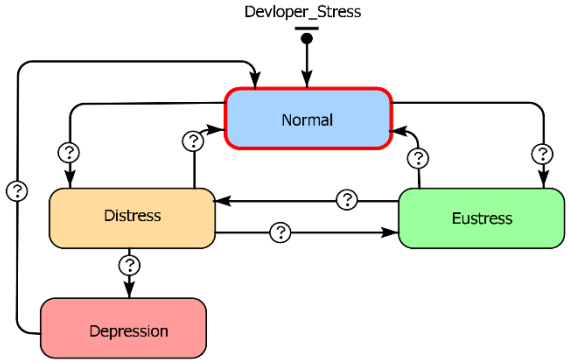

Figure 3. State diagram of a developer – stress level

Actions and rules:

There are mainly two types of actions that agents can perform: self-actions and interactions with the environment (including other agents). Actions of agents are governed by rules. It is therefore important to clearly define the rules in terms of when and how agents will act. We adopt the FIFO (first come, first serve) rule for processing of events and one to one rule for matching developers and units (one developer only works on one unit at any given time and one unit needs only one developer). We assume that there is a correlation between workload and mood which in turn will affect stress level and performance. We also assume that stress has a negative impact on both project environment and work quality. Specifically, a developer's mood has impact on failure rate and other team members' mood. Team interaction was represented in two ways: a) exchange of influences (spread of feelings) takes place each time face to face communication takes place; and b) organisation management practises such as motivation events have an effect on relieving stress. If a developer's mood entered depression state, developer would be given a break for rehabilitation. Variation to a unit's time to completion is affected by individual developer's performance. Daily working hours are defined by a schedule (9am -5pm, weekdays). It is possible to apply up to 3 hours /day overtime as well as adding new members to team if needed. The parameter values shown in Table 1 are picked from the symmetric triangular distribution when applicable.

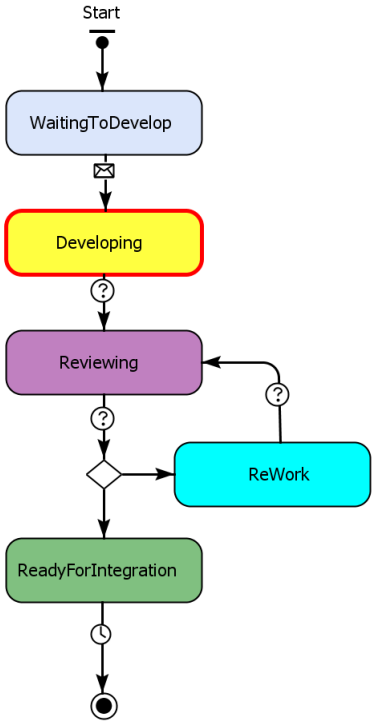

Figure 4. State diagram of a unit

Table 1. Description of model parameters and variables

Name	Description	Typical Value
Team Size	Number of developers in the team.	3~10 developers
Experience Level	Experience level of a developer.	70~90 %
Total Units	Total number of units that need to be developed.	100~1000 units

Unit size	Time required for one developer to develop one unit.	7~10 hours
Employee pay rate	Hourly cost of a developers' time.	40 ~60 dollars
Review time	Time required to review one worked unit.	0.5~1.5 hours
Rework time	Time required to rework one worked unit.	2~5 hours
Integration size	Number of units to be integrated into one component.	3~7 units
Integration time	Time required for integration of one component.	1~3 hours
Failure rate normal state	Likelihood of failure of a unit at review.	0.01~.0.05 %
Failure rate under stress	Likelihood of failure of a unit completed by a stressed agent at review.	0.1~ 0.25%
Extra workload impact	Percentage mood decrease caused by extra workload every hour.	0.01~0.09% / hour
Communication frequency	Average number of communications between employees per day.	1~ 4 times
Environment influence	Same mood majority threshold that triggers environment change	60% of developers
Break duration and limit	The duration and maximum number of rehabilitation break per developer.	2~9 days , 3 times max
Developer' s stress state	Correlations between a developer's stress state and mood indicator (m)	m >90 Eustress 60 < m <=90 normal 40 < m <=60 distress m < =40 depression
Adding new member	The assimilation time for a new member to become fully productive.	8 ~ 12 days

Time:

Choice of time unit is essential in capturing the dynamics of simulated system behaviours at the right level of granularity without wasting computing time. We chose to use hour as the unit of time in our simulation. It can be easily set as ¼ or ½ hour but it is certainly not necessary to use minute for the purpose of such study.

Environment:

Our model environment includes three dimensions, namely: a) organizational process: this dimension covers the workflow of activities and management practices; b) team interaction: this dimension covers the relationship and communication among developers; and c) individual behaviour: this dimension covers the individual behaviour of each developer.

An environment is subjective to the viewing perspective. For example, the environment for a project as a whole could be different from that for an individual developer, or that for a unit. The actions an agent takes, defined by the rules, represent the agent's response to its environment. Therefore design of environment is inseparable from that of actions and rules.

2.3. The SD component

System dynamics (SD) modelling is suitable for capturing the resultant dynamic behaviour from causal relationships among parameters identified as important to characterise a complex system. We introduce a SD component to measure and monitor a developer's mood. Figure 5 shows the stock and flow of the SD model component. It shows that a developer's mood, the stock named Mood Indicator, is affected by three factors (modelled as flows) in a cumulative manner: a) the extra workload; b) positive team/organisation events; and c) interaction with peers at work.

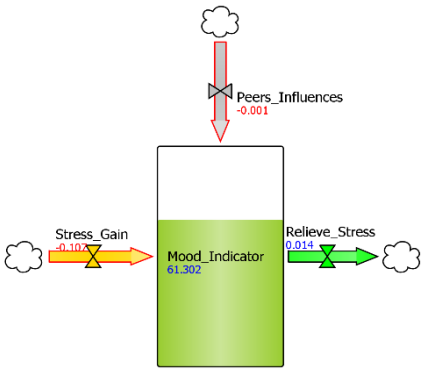

Figure 5. System dynamics model of a developer's mood

2.4. The DES component

Units passed developing states will be going through an integration process to form components. The DES approach is used to simulate the integration process (Figure 6), where a batch of units will be converted into one component. The availability of a developer and the batch size are the main constrains imposed on this process. A component may need to go through re-integration based on a given probability.

2.5. Platform

The first step of implementing our model is to choose the most appropriate platform for simulation modelling. We have chosen *Anylogic* (see http://www.anylogic.com) for two major reasons. First, *Anylogic* supports building hybrid models, that is, it allows the adoption of any combination of System Dynamics, Discrete Events, and Agent-Based modelling approaches in one simulation project. Second, *Anylogic* provides a simple yet powerful animated graphical interface that helps the user to visualise model behaviours at any level of details modelled (e.g., observing the change of a specific agent's mood). Run time change of control parameters are also provided here. A screenshot of the main interface during run time is shown in Figure 7.

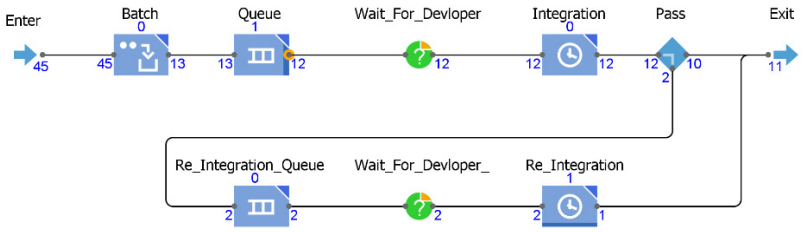

Figure 6. A DES model for component integration

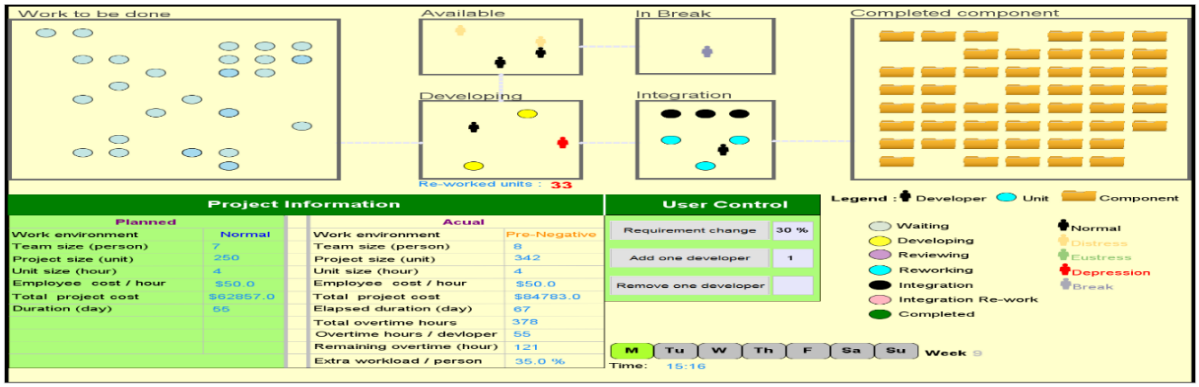

Figure 7. A screenshot of model's main interface during run time.

2.6. Initialization

We provided an initialization window that allows the user to set the initial values for parameters. It is important to keep in mind that the purpose of this simulation model is to compare the relative pros and cons of different scenarios in a generic manner, rather than to get absolute results under a specific setting for a specific project. Figure 8 shows the initialisation interface with a set of reasonable default values. Users can easily reset the parameter values to what they think are more suitable for their project environment.

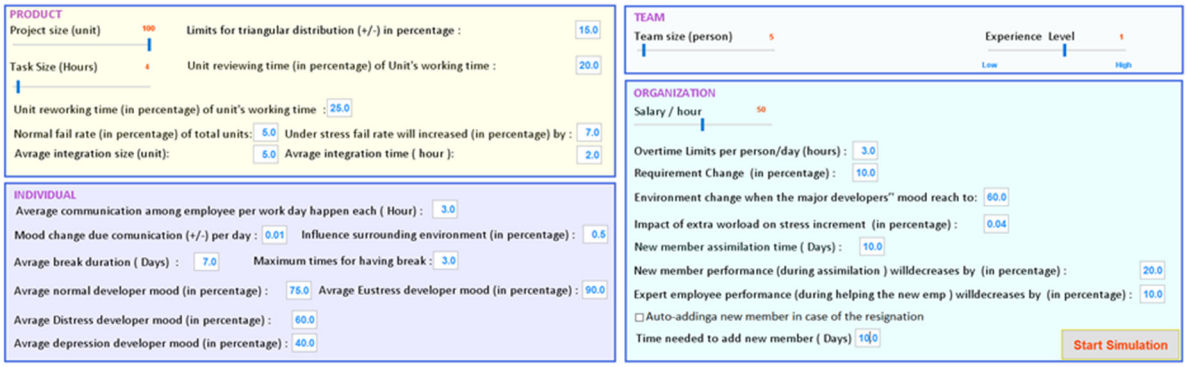

Figure 8. Initialization window used to set the experiment's parameter values.

2.7. Iterative and incremental model development

Similar to developing software, we followed an iterative and incremental development approach. Each iteration includes planning, analysis, design, implementation, and testing activities. Such a development approach

allows us the opportunity to gradually build the simulation model through growing stages of complexity. The focus of the first stage was to build a simple working model in a short time, which allows us to demonstrate and learn the capabilities of *Anylogic* as well as building a robust base version of our model. Additions to the earlier versions allows us to observe the impact of newly added features on model behaviour. It also puts us in a better position if a change of direction is needed. Finally, Incremental model development provides the benefits of enhanced confidence in the model result.

3. MODEL VALIDATION

Validation of our simulation model are discussed from conceptual and operation points of view as follows.

Conceptual validity: Conceptual validity checks that the choices of model components, their relationships, as well as the assumed rules and actions are reasonable. If the direction of the effect is known, then the input and output of the simulation model should not violate this qualitative knowledge, otherwise the model needs to be questioned (Kleijnen, 1999). Our choices and decisions in regards to the aspects and their relationships are based on our general knowledge of software development process as well as team work experiences (both authors worked in software development organisations), in addition to review of relevant literatures (Athavale & Balaraman, 2013; Athavale & Singh, 2014). Our model output are consistent with qualitative knowledges.

Operational validity: Developing a simulation model is very much similar to developing software. As such, software testing techniques are applied throughout the model development process to ensure our model's operational validity. These include a) animation: visual representation was watched to track the agents' actions and the model's outputs over time which helps us to make a preliminary judgement on model behaviours; b) extreme conditions test: the model was test to be reasonable and credible for any extreme combination of factors (Sargent, 2013); c) predictive validation: the forecast model behaviour was compared to the actual and made sure that they were consistent (de Franca & Travassos, 2012); and d) structured walkthrough test: the technical aspects of model were thoroughly reviewed, including line by line inspection of code for any potential defects (Sargent, 2013).

Validation of a simulation model often involves using real data collected from the system being modelled, which allows modeller greater opportunity to calibrate the model parameters for the targeted specific system being modelled, resulting in more confidence in the model. The validation of an agent-based simulation of open source software evolution represents such an example (Smith et al. 2006). However, it is important to bear in mind that such validation only contributes to tuning of model parameters so that the model output could match other real data collected from an instance of the type of system being modelled. It does not change the characteristics of model that is dictated by its structure design. For example, a non-linear relationship between an input parameter and a model output will remain non-linear whether real data based validation is performed or not. We argue that simulation models can still be used to explore hence gain insights into generic system behaviours as long as conceptual and operational correctness are validated. In fact, it is the comparison of simulation results from different scenarios on a relative scale that offers most value in helping people to understand and optimise a complex system. When models try to study human interaction the expected outcome should provide insight for debate, not decision automation (Pidd, 1997).

Furthermore, while widely admitting the benefits of ABM, some researchers have classified ABM as a non-testable program (Zhou et al., 2004) since it has no "oracle" (Pullum & Cui, 2012) for testers to decide if the system's behaviour or output is correct or not. This is certainly the case when counter-intuitive emergent system behaviour is observed with simulation.

4. MODEL APPLICATION

In this section, we present a sample application of our model to a case where different levels of requirement volatility are observed. Unplanned overtime is assumed to be the team's response to complete the extra work. Three scenarios were run using the simulation model: scenario 1 represented the base case where no requirement change was imposed; scenario 2 represented 10% requirement increase at an early stage of the project (1st 1/3 of planned project duration); and scenario 3 represented 30 % increase distributed over the early, middle and late stages of the project. All three scenarios were subjected to an identical environment setting.

Due to the stochastic nature of simulation, each scenario was replicated 10 times and the average was calculated. The simulation results were presented in Table 2. It is seen that, despite the overtime team members put in, project schedule still suffered. Other project outcomes such as cost and quality (units' failure rate) also suffered. Actual average workload increased more than the corresponding requirement increase percentage,

and the average mood and performance of individual decreased significantly. Team members' stress level also unsurprisingly increased.

Table 2. Summary of project and team performance results (average of 10 replications)

Project and team performance	Scenario 1	Scenario 2	Scenario 3
Actual cost	$75714	$84784	$105987
Actual duration(day)	53	57	68
Extra workload	0%	12%	35%
Extra stress	0%	10%	19%
Average mood	75%	64%	53%
Average performance	80%	69%	62%
Units' fail rate	7%	9%	19%

5. CONCLUSION AND FUTURE WORK

Simulation of human factors in software development process is highly challenging. We have presented the design, development and sample application of a hybrid simulation model for exploring the impact of individual and team performance on software project outcomes. We are still in the early stage of this work and far from mastering the balance between achieving model objectives and managing model granularities. Other future work includes making the simulation more robust and extensible for ease of application, and exploring different scenarios such as the effects of adding new employees to team or adopting process changes. We trust that this study will not only benefits those who are interested in understanding the impact of the human factors on software project outcomes, but also those who are considering using hybrid simulation modelling approach to study human factors in complex systems.

REFERENCES

Athavale, S. and Balaraman, V. (2013). Human Behavioral Modeling for Enhanced Software Project Management. In *Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering*, pp.15 – 17.

Athavale, S. and Singh, M. (2014). Modeling work-ethics spread in software organizations. In *Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering*, pp. 2-7.

Chen, T. Y., Huang, D. H., Tse, T. H., and Zhou, Z. Q. (2004). Case studies on the selection of useful relations in metamorphic testing. In Proceedings of the 4th Ibero-American Symposium on Software Engineering and Knowledge Engineering (JIISIC 2004), pp. 569-583.

De Franca, B.B.N. and Travassos, G.H. (2012). Reporting guidelines for simulation-based studies in software engineering. In *Proc. 16th International Conference on Evaluation & Assessment in Software Engineering*, pp. 156 – 160.

Hammond, R.A. (2015). Considerations and best practices in agent-based modeling to inform policy. In *Assessment of agent-based models to inform tobacco policy. Washington (DC): Institute of Medicine, National Academy of Sciences Press*, Appendix A, pp. 161–193.

Joslin, D. and Poole, W. (2005) "Agent-based simulation for software project planning," Proceedings of the Winter Simulation Conference, 2005, Orlando, FL, 2005, pp. 1509-1066

Jun, J. B., Jacobson, S. H., and Swisher, J. R. (1999). Application of discrete-event simulation in health care clinics: A survey. *Journal of the operational research society*, pp. 109– 123.

Kleijnen, J. P. (1999). Validation of models: statistical techniques and data availability. In *Proceedings of the 31st conference on winter simulation*, 1, pp. 647-654.

Pidd, M. (2009). Tools for Thinking: Modelling in Management Science. Hoboken, N.J.: Wiley.

Pullum, L. L., & Cui, X. (2012). Techniques and issues in agent-based modeling validation. Technical report, *Oak Ridge National Laboratory*.

Sargent, R. G. (2013). Verification and validation of simulation models. *Journal of simulation*, 7(1), pp. 12–24.

Smith N., Capiluppi A., Fernández-Ramil J. (2006). Users and Developers: An Agent-Based Simulation of Open Source Software Evolution. In: Wang Q., Pfahl D., Raffo D.M., Wernick P. (eds) Software Process Change. SPW 2006. Lecture Notes in Computer Science, vol 3966. Springer, Berlin, Heidelberg

Zhou, Z.Q., Huang, D., Tse, T., Yang, Z., Huang, H. and Chen, T. (2004). Metamorphic testing and its applications, in: Proceedings of the 8th International Symposium on Future Software Technology (ISFST 2004), 346-351.