1. Introduction

In software engineering, the process of the sequence of changes that occurs during the software systems lifetime comprising both system development and maintenance, was first termed a *software evolution process* by Lehman and Belady in the 1970s [1]. "The successful evolution of software is becoming increasingly critical since the increasing dependence on computers and software at all levels of the society" [2]. Therefore, in order to find ways to manage and control the evolution of software systems, researchers and practitioners have strived to reveal the process by which software systems have evolved [3].

One of these endeavours was Lehman's description of the "*global software process"* [4] as a feedback system of a collection of people and events that control the evolution of software-based systems. Lehman presents this process as being driven by feedback which is made explicit in the 7th law of software evolution that the "E-type evolution processes constitute multi-level, multi-loop, multi-agent feedback systems" [5].

Based on this view of the software evolution process, Wernick and Lehman [6] and Kahen et al. [7] developed several simulation models of software evolution in an attempt to understand and reveal the reasons and the factors behind software evolution. Recently, Wernick et al. [8] have developed a simulation model of the software evolution process by applying social view actor-network theory (ANT) presented by Latour [9]. ANT can be described as a perspective for viewing and describing social and technological situations by considering both human and non-human elements equally as active entities within an interconnection network [10]. Wernick et al. built this model by using a system dynamic (SD) general purpose simulation environment. They justified using this tool for modeling as it provides a usable and simple toolset. However, according to [8], a pure SD simulation environment is not the most appropriate environment for representing some ANT aspects. The SD environment provides no support for the flexibility to represent the complexity of the participants making up the model.

Therefore, Wernick et al. suggested reworking the SD model to a more appropriate, agent-based simulation environment and modifying the new model to a more accurate simulation model, then checking the behaviour of the new model against the existing SD model. Based on this suggestion, the study presented in this paper was undertaken to rework the current SD model to an agent-based simulation environment and to check the behaviour of this new model compared with the current SD model. Therefore, in this work, the Repast simulation toolkit was chosen as an agent based simulation platform that was used to develop the new model based on the specifications of the existing SD model.

Furthermore, this study addresses the issue of calibrating the SD model which was conducted without referring to any real world data due to the lack of available data in representing real world processes of evolution [8]. In other words, the ability of the model to reflect the real world of software evolution processes has not yet been investigated. This study addresses this issue through an investigation conducted to check the ability of the new agent-based model to reflect real-world aspects of software evolution. At this stage of the research, the new model does not take into account individual human aspects; the model is a starting point for future work where this aspect will be further developed.

Consequently, this work aims to address the following research questions:

RQ1 How does the ANT-based model of software evolution built in an SD simulation environment behave if reworked to an agent-based simulation environment, in comparison to the existing SD model?

RQ2 Does the new agent-based simulation model of software evolution processes have the ability to reflect the real-world process of software evolution?

Section two presents the background and literature review of actor-network theory (ANT), an explanation of the ANT-based model of software evolution 'SD model' and a description of the agent-based simulation modeling including its advantages and issues. Section three describes the methodology of conducting this research which includes specifications, design and implementation phases that were undertaken to build the Repast simulation model of software evolution. In section four, the investigation conducted to evaluate the Repast simulation model and the findings of these investigations is presented. This section consists of two parts. The first part presents a comparative evaluation between the new Repast model and the previous SD model, including their results. This part addresses the first research question. In the second part, an investigation of the Repast model to check its ability to reflect the real-world process of software evolution is described, including the calibration of the investigation results. The second research question is addressed in this part. In section five, the discussion of the above findings is presented. Finally, this paper ends with section six which presents evaluation of the research work including its limitations and future work, and the concluding view of the contributions made in this work.

2. Background and Related Work

This section presents a background of ANT theory, an explanation of the SD simulation model of software evolution and a literature review of agent-based simulation modeling.

2.1 Actor Network theory (ANT):

With the aim of explaining complicated interactions in a research setting, Bruno Latour and Michel Callon described, in the early 1980s, the principle of actor-network theory (ANT) as a perspective for viewing complex social situations [11]. Latour [10] claimed that ANT theory differs from the traditional view of social and technological theory. In the traditional view, elements forming the social situations are described as categories such as large, small, human and non-human [12], while ANT theory describes both human and non-human elements equally as an active entity within an interconnection network [10]. According to Wernick et al. [8], the ANT view of the social world can be described as seeing the complexity of social world behaviour and the technical situations within it as a situation caused by the correlations between the elements that form the social world. In the concept of ANT, Latour [10] presented three types of elements: actors, mediators and intermediaries.

Within ANT, the actor was described as an active entity that is "not the source of an action but the moving target of a vast array of entities swarming toward it" [9]. The actor can be a collective of human 'developers, manager', non-human 'system' and even intangible elements such as 'idea, situation' [13]. Latour [9] also stated that using the term actor was not arbitrary. According to the author, it refers to the actors in theatre shows whose acting is constrained by different factors that shape their roles.

Latour described a mediator as an element within ANT that can transform, translate, distort and modify the meaning or elements that they are supposed to carry [9]. This means that even if the mediator looks very simple it may turn out to be complex since it can create unpredictable behaviour which will affect the entire element connected with it in the actor-network. Therefore, mediators can be recognised as the elements that are both written and interpreted by humans and by whom the interpretation of the written elements may differ from one person to another [3]. Latour described possible examples of the mediator as *law, science, religion, economies, psyches, moralities, politics* and *organizations* [9].

An intermediary was described by Latour as an element whose output depends only on its inputs. It is anything that passes information from and stands between one actor and another [13], transparently moving data without affecting its meaning. Therefore, in order to define an intermediary's outputs, it is enough to define its inputs. No matter how complex the intermediaries, they can be ignored in building cause-and-effect models. However, Latour illustrates the changeability and complexity of a description of both intermediary and mediator by example of a simple intermediary such as a computer function that can turn out to be a *horrendously complex* mediator if it breaks down; intermediaries can only be ignored if they continue to make no semantic changes to their inputs over a simulation run.

The essential concept in ANT is the communications channel: networks that link together the actors, mediators and intermediaries. Within this network of communications channels, actors may join or leave a network and are constantly changing the web of relationships. By joining the network, these actors may bring their own network with them; an actor can sometimes be decomposed into one or more sub-networks. Moreover, within ANT, the level of commitment of each actor to the goal of the system is illustrated as a changeable commitment that depends on the actor's own situation and the influences on it from other network elements [9].

2.2 System dynamic (SD) simulation model of software evolution:

The following description of the SD software evolution simulation model is based on Wernick et al.'s paper [8]. The reason of focusing on Wernick et al.'s paper in illustrating SD model is that the concept and the implementation of this model was essentially presented and described through this paper. Furthermore, the new model (Repast model) is the continuing work of Wernick et al.'s paper recommendations.

By using ANT, Wernick et al. developed a software evolution simulation model of a *global software process* based on a system dynamics environment. A global software process includes in one concept the collections of people, things and events that control software-based system evolution [4]. The purpose of developing an ANT-based model of software evolution is to reveal and illustrate better the factors under which a software-based system is evolved. By adopting ANT in the modeling of a software evolution process, it is possible to consider both human and non-human elements within the system as active elements. This enables a wider range of entities to be considered in this model than in previous models such as [7], [6]. It also provides the ability to consider the software system as a participant on its own. Wernick et al. illustrated that the first task in building an ANT model of software evolution is identifying the entities, 'actors, mediators and intermediaries' that make up the social and technical situation within which evolution occurs through the connections between them. Accordingly, this model is structured as 16 entities, comprising 13 actors and 3 mediators, which can be seen in Appendix 1. For the reason noted above, intermediaries such as programming language compilers have not been included as elements in this model.

In [8] these entities were identified by the authors based on their experiences as users and software developers, their previous research [14] and general knowledge of software development. The behaviours of these actors and mediators were quantified in the form of SD equations to provide the ability to quantify the changing degree of support of each of these participants to the evolution process based on the structure of the ANT model. A typical equation for the participants in the model can be seen in the *Health of software evolution process* equation as follows:

Health of system evolution process = (Health of system evolution process × health own weighting) +
(Developers + Immutable tools
+ Mutable tools + Project manager
+ System change input queue + System design/architecture + System development owners)/7× (1 -Health own weighting)

This equation illustrates how each participant re-computes its value based on the average of the values of each of those participants that influences it in each time step. The average of these participants is calculated in the following part of the equation:

Average = (Developers + Immutable tools
+ Mutable tools + Project manager + System change input queue + System design/architecture + System development owners)/7

Then the average value is weighted against the (health of system evolution process) value from the immediate past as following:

Health of system of evolution (current time step) = Health of system evolution process (past time step) × health own weighting + Average x (1- health own weighting)

The 'own weighting' variable above is contained in each participant. It refers to the percentage of the participant's own existing state weighted against the extent to which other participants affect it. The degree of commitment of each participant actor and mediator is represented numerically with a default value of 1, which represents the situation in which this participant maintains a position of neutrality as to whether system evolution is necessary or desirable – irrespective of whether this evolution is objectively necessary or not. A health value greater than 1 represents a positive attitude of the participant, i.e. a desire to evolve the system, while a value of less than 1 represents a definite negative stance against evolving the system.

The equation above was used in SD model to perform the connection and behaviour of that participant (Health of system evolution process in the previous case) with other participants in the network. This is illustrated in Fig 1.

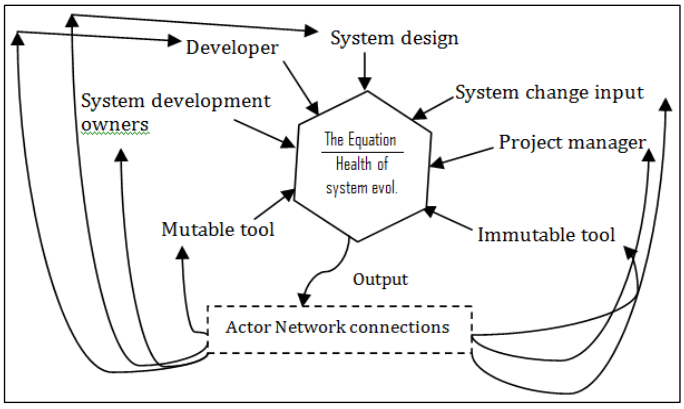


Fig 1: Sample of SD model equations interconnection.

Figure 1 is a sample of how the equation works in each participant. The hexagon shape represents health of system evolution process and its interactions with other participants in the actor network.
The result of each equation will then be going through complex interconnections in the actor network to return eventually to affect its own value; this can be seen in Appendix 1.

This earlier model was developed using an SD simulation environment because using this tool for modeling provides a usable and simple toolset. However, an issue encountered with this simulation environment is that the SD environment is not the most appropriate environment to represent some aspects of ANT. The SD simulation environment provides no straightforward support for flexibility in representing changes in the linkages between participants making up the model. Therefore, it was suggested by Wernick et al. that this SD model be reworked in a more appropriate simulation environment, *an agent-based simulation environment*, to form the basis of a model more representative of the real world.

2.3 Agent-based simulation modeling

Agent-based simulation modeling (ABM) can be defined as a modern computational simulation method that enables researchers to build, analyse and investigate models consisting of autonomous agents that interact with each other within an environment [15], [16], [17]. This new method of modeling has gained increasing importance, growth and popularity during the last 10 years [16], [17]. According to Railsback et al. [15], the growth of ABM is driven essentially by its ability to address the issues and problems that cannot be solved in traditional simulation modeling, such as implementation of social complexity. The meaning of the term *agent* in the context of ABM is controversial between authors. Bonabeau [18] argues that the term *agent* in agent-based simulation modeling refers to the collection of autonomous decision-making entities that are used in simulation modeling. On the other hand, Gilbert [16] states that the agents are "either separate computer programs or, more commonly, distinct parts of a program that are used to represent social actors—individual people, organizations such as firms, or bodies such as nation-states". While, according to Macaland North [17] in the context of ABM, the term *agent* has no accurate definition or clarification and it is the subject of occasional arguments and discussion.

Each agent independently determines its position and situation and makes decisions based on a set of rules and conditions [18]. Helbing and Balietti [19] also argued that agents' behaviour and interaction can be formalised as equations, but they are commonly built through conditions such as the *if-then* kind of logical operation that provides the modeling approach with more flexibility. These agents are typically represented in a programming language implementation as objects that contain their state and behaviour rules. In building each agent, the modeler needs to encode the rules that define its behaviour, how these rules interact and the specification of the agent's activation [20].

According to Helbing and Balietti [19], the advantage of ABM is that it not only represents interactions between agents but also allows the ability to determine the implementation of different assumptions and hypotheses; it also gains from *modularity*, *great flexibility*, *large expressiveness*, and the *possibility to execute agent actions in a parallelized way*.

Bonabeau [18] describes three issues related to the implementation of ABM in different areas such as social, political and economic science. The first issue is common to all simulation modeling techniques in that the model should be built for a specific purpose with an accurate level of description and detail. Accordingly, simulation modeling is considered more as an art than a science. The second issue is related more to social sciences in which the simulating of human agents is difficult to quantify as human behaviour is complex, subjective and potentially *irrational*. The third issue is related to the practical aspect of using ABM. Within the modeling of a system using ABM, beside the aggregate level of describing the system, a description of low-level agents that make up the system is also required. With regard to modeling complex systems, it can be difficult and time-consuming to model such systems in sufficient detail; hence the simulation of a complex system remains a problem.

Based on previous literature, although the use of ABM can provide usability, flexibility and expressiveness for investigation and experiments in different scientific areas such as social science, economics and political science, there are some issues that need to be taken into account in creating ABM simulations, especially in social modeling, which require simulation of human behaviour and need detailed descriptions for low-level agents.

3. Research Methodology

To address the research questions an agent-based simulation platform was chosen and the existing SD model reworked using the latter as a specifications. A comparative evaluation between the new model and the current SD model was conducted to check the behaviour of the new model followed by further tests to check why the new model performed with such behaviour. Finally, an investigation was carried out to check the ability of the new agent-based simulation model to reflect the real-world simulation of software evolution. This included an interview with an expert in software development to help in the quantification of inputs to simulation variables.

3.1 Specification and Design:

3.1.1 The specifications required to develop the new model

The development of the new agent-based simulation model of system evolution process was based on reworking the current SD model.

Therefore, the design specification of this new model was required to meet the specifications of the existing SD model as presented by Wernick et al. [8] as follows:

- Each participant in the ANT network is represented and treated as an autonomous entity that has its own identity, a potentially complex, multi-variable state.
- Represent an arbitrary number of inward and outward links that represent participants' connections with each other in the network.
- Produce and export graphical and numerical outputs that represent the behaviour of the new simulation model.
- Cope with complex multilevel feedback flows between the actors and mediators.
- Provide the ability to control and schedule time steps during the running process.
- The model should have the ability to change and control variable values that represent each participant in the network during the running period in order to monitor the rate of change in the degree of commitment of each of these participants.

Moreover, in order to build a new simulation model that represents more accurately the real-world process of software evolution, Wernick et al. [8] proposed an improvement in the specifications of the new simulation model to support the ability to calculate and store complex data that represents the condition of each participant, actor or mediator, in the actor-network.

3.1.2 Description of the Repast simulation environment

Repast can be defined as a free, open source, agent-based simulation toolkit that was developed by Sallach et al. at the University of Chicago in close collaboration with Argonne National Laboratory [21] , [17]. It is applicable in a pure Java and Microsoft implementation as an agent-based simulation environment [21].

Although the Repast toolkit focuses essentially on simulating social behaviour, it can be used for a different range of applications from social systems, *to evolutionary systems, to market modeling*, *to industrial analysis* [21]. Due to this variety of uses, Repast is represented in two editions, Repast Simphony and Repast for high-performance computing (Repast HPC) [22]. The edition used for the work reported here to implement the new simulation model is Repast Simphony 2.2, released on 26 June 2014.

Repast Simphony is defined as a Java-based modeling system that provides a richly interactive, tightly integrated platform running on Microsoft Windows, Apple Mac OS X and Linux, and supports the ability to develop models of interacting agents with high flexibility [22]. Repast Simphony consists of two basic platforms of programming language: Java and Repast ReLogo which is defined as a workplace tool based on the Groovy programming language. According to [23], Repast ReLogo represents the construction of models in the form of packages, each of which contains a number of default Groovy classes such as *UserGlobalsAndPanelFactory* and *UserObserver*. These classes provide the ability to create, control and perform interactions between the agents in the simulation model. ReLogo also symbolises the agents in the model in terms of *turtles*, which represent a class of code that can be used to state the behaviour of each agent in the model [24].

3.1.3 Reasons behind choosing Repast

The reasons behind choosing the Repast Simphony simulation environment, particularly the ReLogo platform, to rework the current SD model are that this simulation environment provides the following abilities:

- It can create an autonomous entity, agent, in the form of Groovy classes termed turtles [23]. These turtles can be easily controlled to represent the behaviour and characteristics of actors and mediators. This capability is crucial to build the structure of each actor and mediator in the simulation model.
- It provides different platforms of programming languages and tools, including "the ReLogo dialect of Logo, point-and-state charts, Groovy, or Java, all of which can be fluidly interleaved" [22]. It also provides the capability of transferring from one platform to another without rebuilding the model from scratch. These capabilities provide the flexibility to rework the new model into further platforms in future work.
- Repast Simphony also supports a flexible environment to build links between the agents [24]. This provides the ability to develop and represent the linkages between ANT elements 'actors and mediators' and in particular the ability to cope with the frequent dissolution of actors from the network.
- According to [15] Repast Simphony can export output data in files of different formats such as text and Excel worksheet, and also record and schedule actions at predefined times.
- Repast ReLogo is a suitable environment for representing agents in separate pre-built classes and objects [23] that make it easier to reuse these classes to code new agent types without building them from scratch.

3.1.4 Design and structure of the Repast simulation model:

The participants: actors and mediators

Wernick et al.'s SD ANT-based model of software evolution consists of 16 participants: 13 actors and 3 mediators [8]. In order to meet the requirement to represent each of these participants as an autonomous entity, the design of the new model exploits the ability of Repast to create independent agents.

The structure of each agent consists of two parts: a declaration part and a behaviour part. In the declaration part, the local variables and values that represent the characteristic of the actor or mediator are declared, while in the behaviour part, the rules that form the behaviour of each participant are defined using equations and if-then conditions, as proposed by Helbing and Balietti [19] (see Section 2 above). The common structure of each participant is shown in Fig 2. The same shape of box used by Wernick et al. in the diagram of their SD model is used here to illustrate both actor and mediator structure.

The linkages between the participants in the model

In order to meet the specification of representing linkages between actors and mediators, the design of these linkages was performed using global variables. Each of these variables carries the participant's degree

of support and commitment toward the software evolution process and represents the influence of each of these participants in the model. Fig 2 shows the structure of the interaction between the participants in the Repast model. The term "Link variable" in this Figure represents the global variables that perform the links between actors in the model.

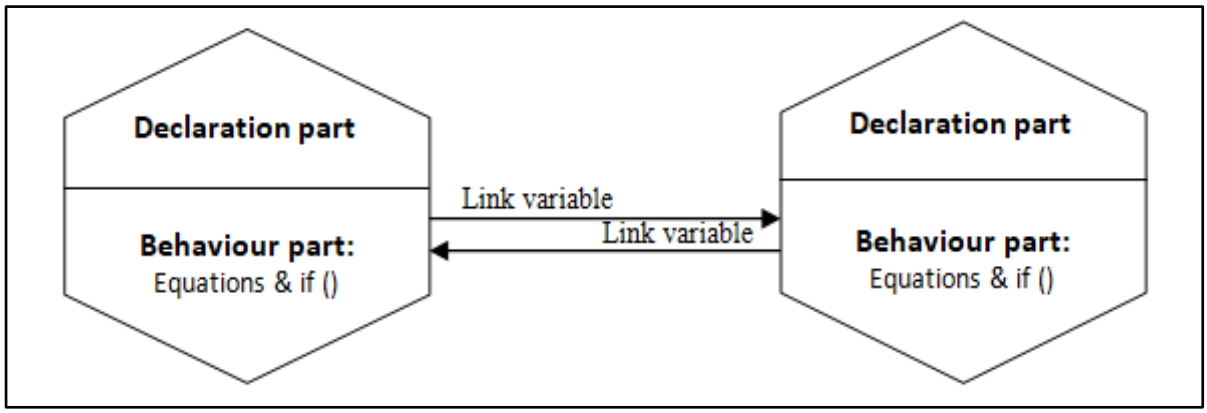


Fig 2: Structure of the participants in the Repast model and the linkages between them (Edited Source: [8]).

3.1.5 Constraints and limitations of the design

In the design of the Repast model it has been noticed that there are limitations in the interface design. The current interface design is built upon the execution screen provided by the Repast simulation toolkit which is not flexible enough to control all the output tools accurately. Therefore, it is recommended for future work to design an appropriate interface that provides more flexibility to control the Repast simulation model.

3.2 Implementation:

This subsection presents the implementation work undertaken to develop the Repast simulation model of software evolution.

3.2.1 Participants: actors and mediators

In order to meet the specification requirement of representing each actor-network participant as an autonomous entity, these participants were implemented by exploiting the ability of the Repast ReLogo toolkit to represent each participant as an independent agent. Such agents are represented in Repast ReLogo as *turtles.* To build the model, the 16 participants (developers, project manager, mutable and immutable tools, etc.) were implemented as 16 independent turtles. Fig 3 shows a sample of these turtles.

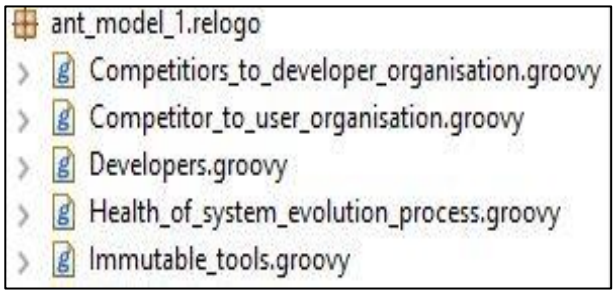


Fig 3: Sample of implementing participants in the model as turtles.

Within each of these turtles, the participant's structure is partitioned into two parts as stated previously, representing data declaration and behaviour respectively.

In the declaration part, the value of 'own health weighting' for each actor is defined as a*float variable* named *health_weighting*. The own health weighting represents "the effect of potential opinion-forming inputs on a key individual depending on how receptive that individual is to ideas from and the opinions of others" [8]. The reason for defining the *health_weighting* as a float variable is to provide the ability to set the weighting value of each participant's own health to a more accurate value than would be the case were an integer type to be used. In the behaviour part, the characteristic that controls the behaviour of each participant in the model is defined using an *if-then* type of condition code and equations.

3.2.2 Linkages between the participants

As mentioned in the specification and design subsection, the *link variables* which represent the linkages between participants are developed using *global variables* to meet the specification requirement of building the interactions between the participants. The declarations of the global variables are implemented using the Repast method *addGlobal (variable name, value)*. This method is used to declare the linkages between the participants in the model and also to set the initial value of each of these links that represent the commitment of each these participants. It is used to build 16 links, each of which represents the link of one participant in the model. Fig 4 shows a sample of the declaration of these variables within the Repast ReLogo workplace and the values which were set for each variable.

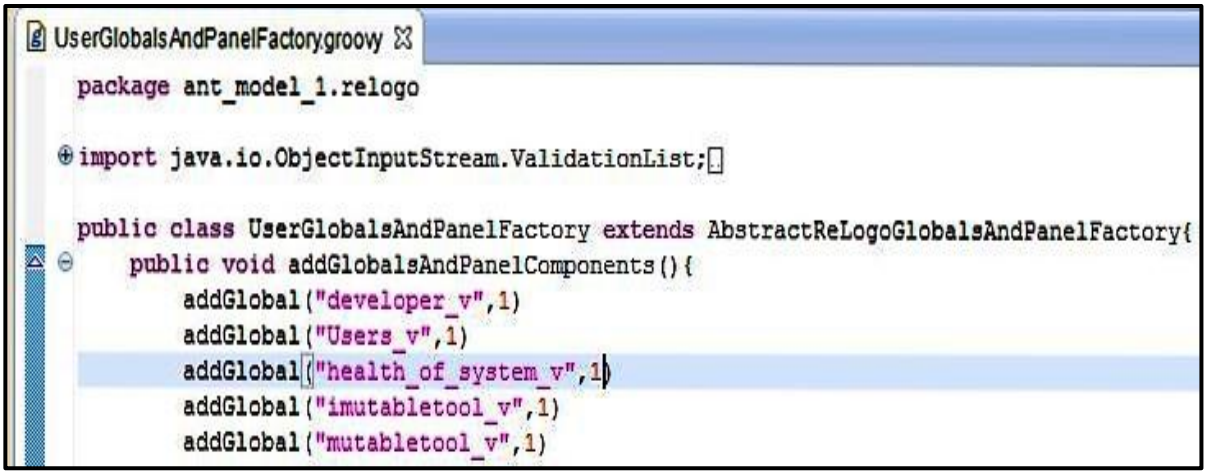


Fig 4: The declaration of the linkages between participants.

The default value for each of these variables was set to a value of 1 to represent the case in which there is no positive or negative attitude from the participants toward the evolution process of the system.

3.2.3 The construction of a running platform

In the Repast ReLogo simulation toolkit, a description of how the program runs must be written in the *UserObserver.groovy* class [23]. According to Kielbasa [23], the *UserObserver.groovy* consists of two functions that always exist, *setup( )* and *go( )*. The *Setup()* function creates instance types of agent-named instantiation and indicates their initial state, while the *go( )* function specifies the order and modality of agent behaviour. Accordingly, the running platform of the Repast simulation model was built using the *UserObserver.groovy* class. In the *setup( )* function, an instance of each participant in the model was created using the method *createTurtle_name(Turtle_number)*. While in the *go( )* function, the behaviour of each of these participants is implemented by calling each turtle using the method *ask (turtles()){method()}*. Within this method the behaviour part built in each participant is called. By calling the behaviour part, each participant will behave according to the conditions and equation in the behaviour part that was explained previously; see Fig 5 shows samples of creating and calling the instance turtles.

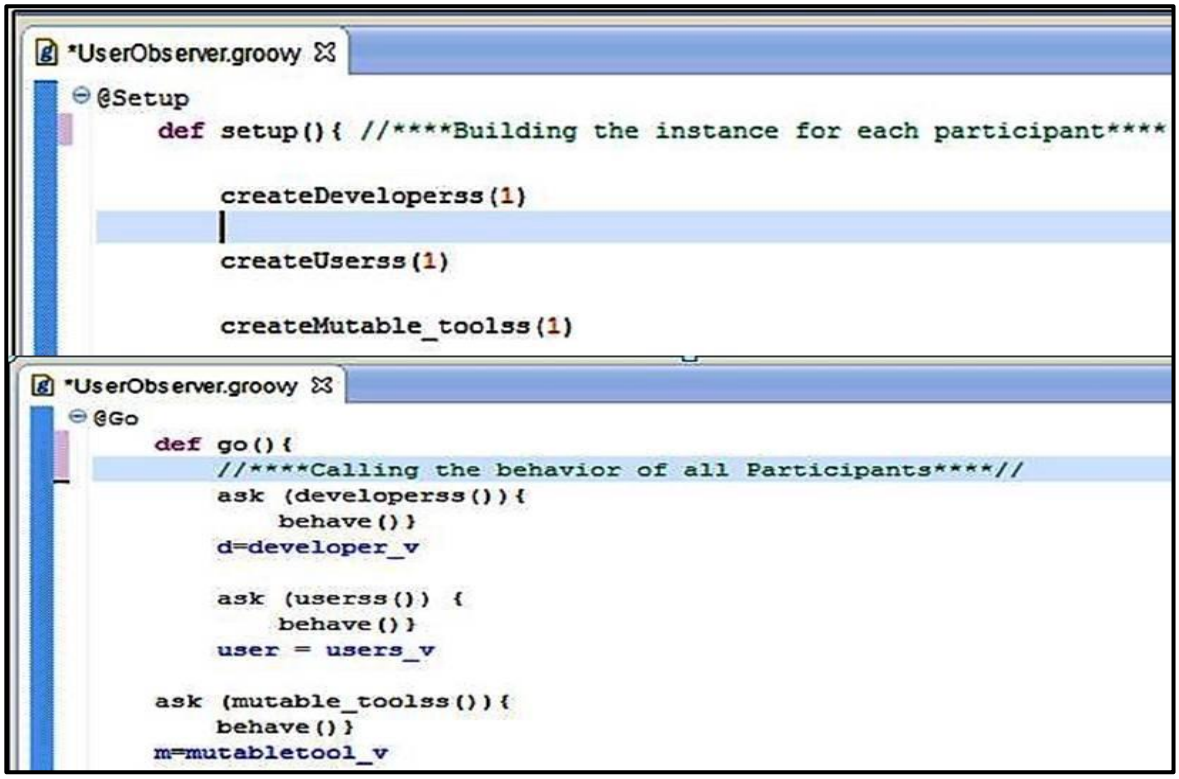


Fig 5: Samples of creating the instance for each participant and calling its behaviour.

3.2.4 Output tools

The output tools of the new Repast simulation model of software evolution are implemented by exploiting the ability of the Repast

ReLogo simulation environment toolkit. According to [23] The ReLogo workplace provides the ability to display each agent's 'participant' outcome through a *Time chart* screen that shows the behaviour for that agent in each time step. Accordingly, the output tools of the new Repast simulation model are implemented by building two observation windows of time series chart: the e*volution health observer* chart and the *participant tracer* chart**.**

1) **Evolution health observer**: this chart displays the output of the participant 'Health of the system evolution' at each time step which represents the evolution health of the system.
2) **Participants' tracer**: this chart displays the output of all 13 actors and 3 mediators in the Repast model. By displaying the output of these participants, the model provides the ability to trace each of these participants and their behaviour in response to the change of influence of other connected participants.

These two output tools can be seen more clearly in the Results section below.

4. Findings and Results

4.1 Comparative evaluation:

The aim of conducting a comparative evaluation method in this study was to explore how the new Repast simulation model of software evolution process behaved in comparison with the existing SD model, and to reveal and understand the reasons behind this behaviour. Therefore, the Repast simulation model of software evolution was tested using the same conditions that were used in the calibration of the SD model in order to conduct a comparative evaluation between the two. According to Wernick et al. [8] in the calibration of the SD model, the default values of the participants in the model were represented by a value of 1. This value represents the behaviour of each participant in the SD simulation model that has no positive or negative effect on the system evolution process. A value of >1 in the model represents a positive feeling and attitude toward the system and the process of software evolution, while a value <1 represents negative feelings and support toward the system [8]. In addition, the inputs of the computation of each participant were given an equal weighting percentage, 50%, to enable the model output to be computed. According to Wernick et al. [8], the first calibration was carried out by running the SD model for 100 time steps without any change in the actors' attitudes, while in the second calibration, the system sponsor attitude was reduced to -0.4 for one time step at time step 45.

Accordingly, similar to the calibrations of the SD model, in this study the first test was conducted to check the behaviour of the Repast simulation model by setting the nominal 'default' value of the model participants to 1, and equal weighting percentages of 50%. Following this the model was executed for 100 ticks. For the second test, the value of the sponsor's support was reduced to -0.4 at tick 45 to represent a temporary loss of enthusiasm towards system evolution by this actor. In the Repast model, this change was conducted by using a condition code of, *if (tick_count ==45) {Sponsor=-0.4}* in the behaviour part of the Sponsor's agent *turtles*. Note that the reason for using the reduction of to -0.4 in testing the Repast simulation model is to apply the same conditions that were used by wernick to calibrate the SD model, consequently to perform the comparison evaluation precisely.

Following this, the comparison evaluation was conducted based on Vartiainen's proposal. According to Vartiainen [25], in order to conduct an effective comparative evaluation between two similar cases, the comparison should be conducted by finding out the differences between them rather than the similarities. Vartianinen illustrated the relationship between the methods of comparison and similarities of the cases in Fig 6.

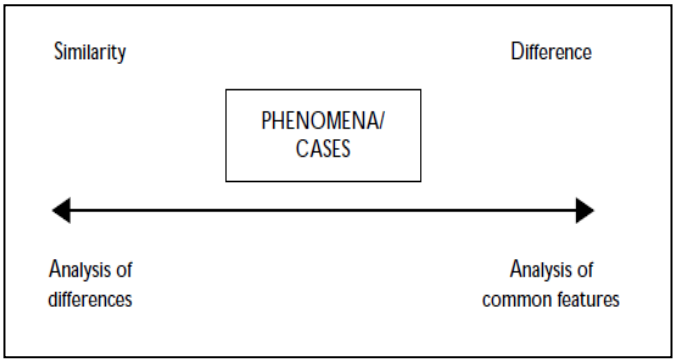


Fig 6: Similarity versus difference of the case compared **[25]**.

Therefore, the results of the first and second test of Repast simulation model were compared with the results of the SD model calibration presented by Wernick et al. [8]. This was undertaken by checking the differences between each case. By using the *evolution health observer* output tool and applying the first test to the Repast simulation model in which there are no changes in actors' attitudes; the result illustrated in Fig 7 shows that the health of the system evolution process denotes a stable behaviour fixed on a value of 1 for 100 *ticks* 'time step'.

.

While the result of the second test shows that when the sponsor's attitude is reduced to -0.4 for one time tick in tick 45, the health of the evolution process dropped to 0.993 at tick 45, to 0.987 at tick 46, and to 0.982 at tick 47 as illustrated in Fig 8.

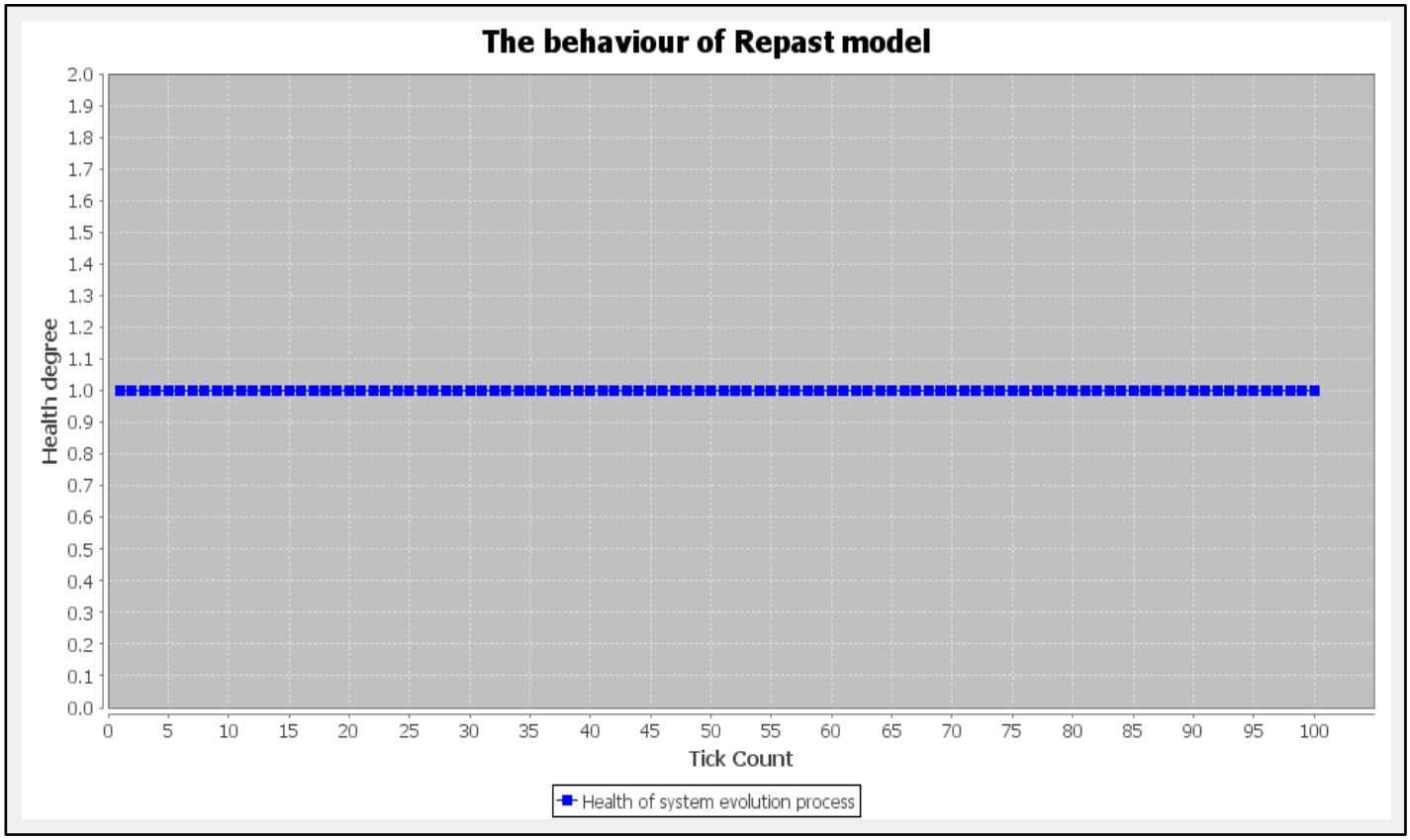


Fig 7: Health of system evolution process: ANT equal weighting.

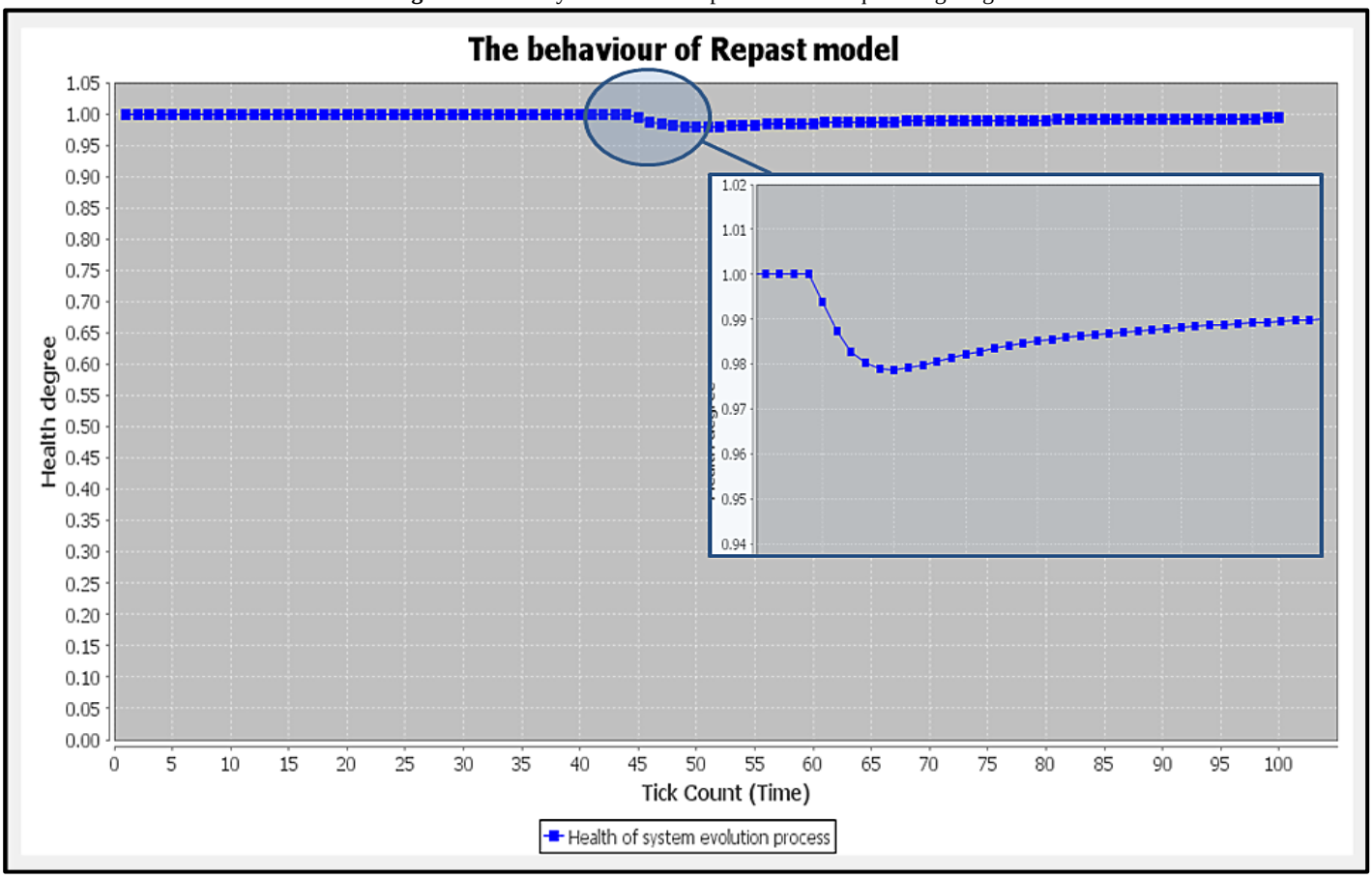


Fig 80: Health of system evolution process with negative support.

The result in Fig 8 shows that the health of the evolution process continues to decline until tick 50 when the health of evolution is equal to 0.978. According to this result, after tick 50 the evolution health starts to increase again to become 0.979 at tick 51 and 0.980 at tick 52. The system evolution health continues to increase to 0.980 at tick 53 and to 0.981 at tick 54. However, the software evolution health does not return to its stable health at the value of 1, even at tick 100. To indicate the tick step in which the health of the software evolution process returns to its previous stability at a value of 1 before applying the pulse, the model is re-run for 200 ticks. The numerical result shows that the software evolution process will not return to its stable health '0.999 ~1' until tick 152 as shown in Table 1.

To evaluate the results above, a comparison was conducted by checking the difference between these results and the results of the calibrations of the SD model proposed by Wernick et al. [8]. According to Wernick et al. [8], the results of the first calibration of the SD model show a pattern of stable behaviour illustrated in Fig 9. This can be explicitly seen in the *ANT equal weighting without pulse*. While in the second calibration Wernick et al. presented the SD model "shows a pattern of increasing oscillations. ...due to a single stimulus" [8]. This can be seen in Fig 9, see *ANT equal weightings with pulse*.

Table 1:
The tick in which system health return to stability

Health of system evolution	Tick 'time step'
0.998	**151**
0.999	**152**
0.999	**153**
0.999	**154**
0.999	**155**

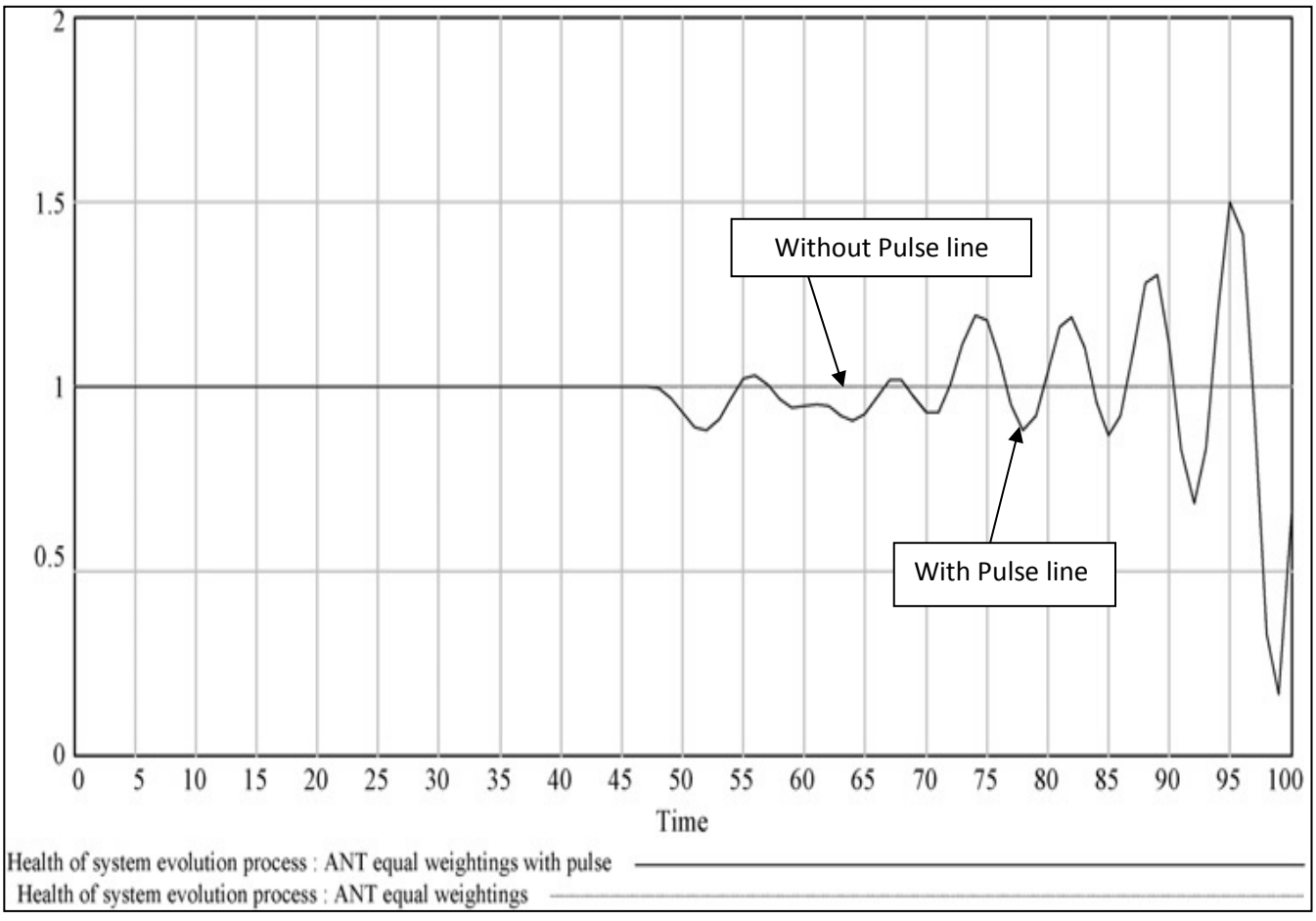


Fig 9: The behaviour of the existing SD model [8].

In comparison, the result of the first test of the Repast simulation model in Fig 7 shows no differences in behaviour, compared to the result of the first calibration of the SD model. This shows stable behaviour fixed on a value of 1 for 100 time steps, when there is no change in the

attitudes of the participants in the model, whereas the results of the second test show that when the attitude of participant 'Sponsor' is reduced to -0.4 in tick 45, the health of the system evolution in the Repast simulation model reduces rapidly to 0.978 in 5 tick times and then gradually increases to its stable trend in 135 tick times. This shows a pattern of stability behaviour which differs from the pattern of increasing oscillations in the behaviour of the SD model in the result of the second calibration [8]. While on the other hand, the Repast model shows similar behaviour to the SD model, showing a pattern of decrease in the evolution health when affected by negative pulse of a participant.

Further observation:

In order to gain a better understanding of possible real-world reasons underlying the Repast model results, the behaviour of all participants in the model was traced using the participants' tracer output tool. This tool was developed especially to provide the ability to observe the behaviour of each actor and mediator in the Repast model, as mentioned in section three. The results of this show that all actors and mediators in the model are affected by the change of sponsor's support; in particular, system development owners, users and sales people, as shown in Fig 10. The results indicate that the support value of the system development owner 'in blue' was reduced to its lowest value of 0.899 at tick 46. The results also show that the lowest point of support value of the users 'in pink' was 0.905 at tick 47 and the value of sales peoples' support 'in light blue' was reduced to 0.910 at tick 65 as shown in Fig 10.

It can be seen from Fig 10 that the Sponsor's health fell to 0.3 at time step 45 instead of 0.4 this is because the effect of feedback in the ANT network, which made the sponsor's reduction in commitment return to affect itself.

Based on these observations, the results show that all the participants in the model are affected by this temporary negative support of the Sponsor. These participants in turn pass on this change, which eventually shapes the behaviour of the system evolution health. This shows a multilevel and complex feedback of processes between the participants in the actor-network which is compatible with Lehman's 7th law of software evolution in which he states that, "E-type evolution processes constitute multi-level, multi-loop, multi-agent feedback systems" [5].

As presented by [8], the SD model was calibrated without reference to any data or estimations that reflect real-world aspects of system evolution. Therefore, in order to check the ability of the Repast model to reflect a real-world system evolution environment, the next investigation was conducted.

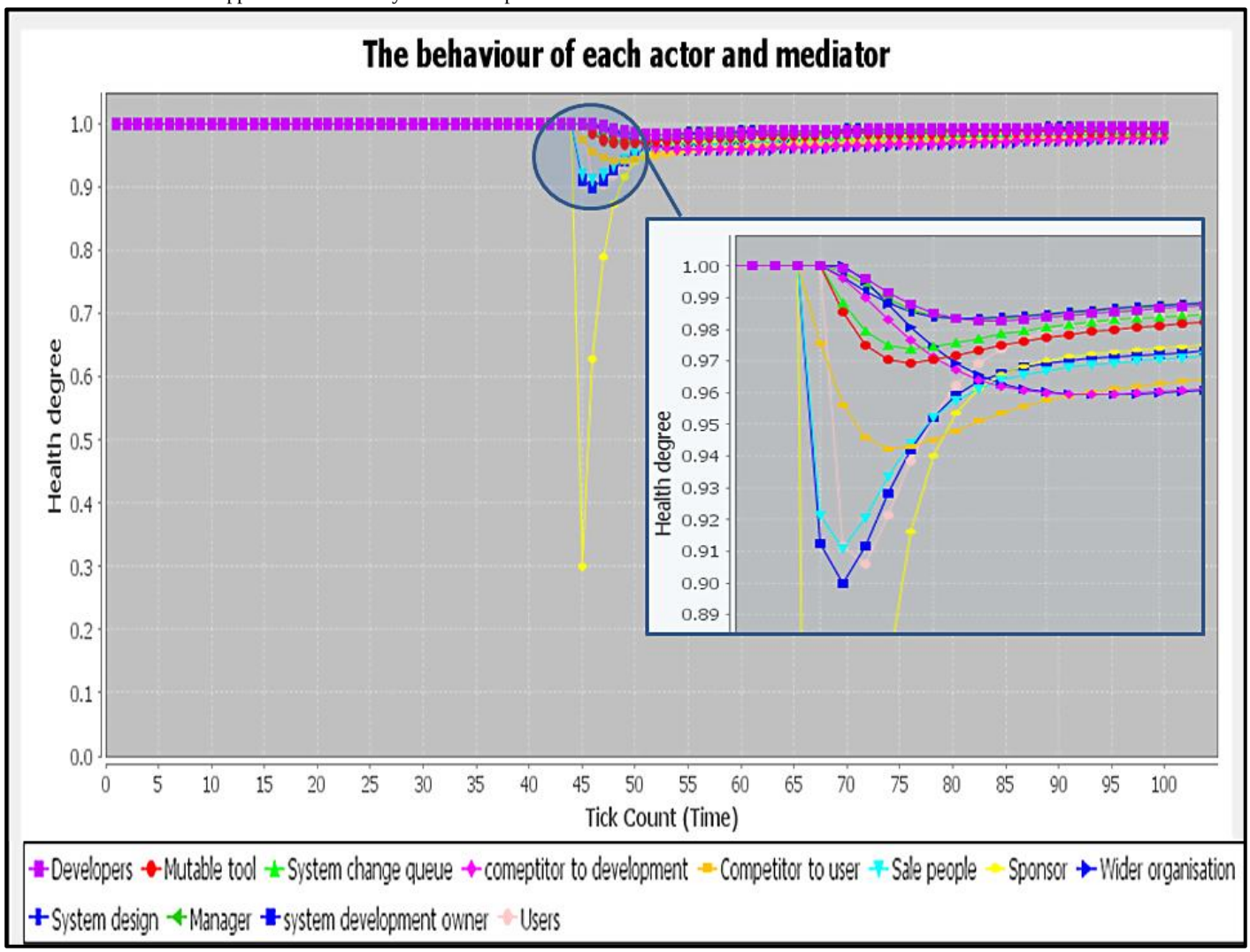


Fig 10: The response of each actor and mediator to the change in sponsor's attitude.

4.2 Investigation using the Repast model:

To investigate the ability of the Repast simulation model to reflect real-world process of software evolution, Wernick et al. [8] proposed checking the behaviour of this simulation model by using real-world data. Wernick et al. also noted that this sort of quantitative data are not readily available but can be approximated through experts' views. Based on these suggestions, an interview was conducted with one of our authors (Wakelam), a researcher at the University of Hertfordshire and an expert with 40 years' experience in managing software development [26], to investigate the Repast model. According to Wakelam [27], in the software development industries, the attitude and influence of human factors (particularly the project manager), have more effect on software evolution health and software project success or failure than the tools used in the development process.

With regard to this, Charette [28] stated that the, "Bad decisions by project managers are probably the single greatest cause of software failures today. Poor technical management, by contrast, can lead to technical errors, but those can generally be isolated and fixed". Similarly, according to Gulla [29], the documented causes of software project failure show that the majority - 54% - are associated with project management, while technical and tools were the least likely factors at 3%.

Based on the above it was concluded that the criterion used for investigating the behaviour of the Repast model is that the behaviour of the Repast simulation model should test whether the health of the software evolution process is affected significantly by the level of support from the project manager.

From our discussions, estimates were made of the effect that participants, in particular sponsor, project manager, developer, user and mutable tools; have on the health of system evolution. On this basis, the most significant factors in the software development process were arranged in the following orders and percentages:

1) Project Manager 70%
2) Sponsor 65%
3) Developers 60 %
4) Mutable Tools 25 %

Each of these percentages was taken individually to represent the degree of the effect of each of these four factors on software development in industries without intending to be added up to 100 %. In other words, these percentages refer to, for example, the project manager's relative impact on the success of the software development worth 70 points.

Although these estimates of the participants' relative influence are to some extent subjective and depend strongly on the particular situation, they do form a first step in representing these real-world impacts in the calibration of ANT-based software evolution modeling. As presented in section three, the *own health weighting* for each participant in the Repast simulation model refers to the percentage of these participants' effect on the health of system evolution. Therefore, in the Repast model, the assumptions above are used to reset the own health weighting value for each of the participants 'project manager, sponsor, developers and mutable tools'. Accordingly, instead of the arbitrary value of 50 %, the own health weighting value is reset to 70 % for project manager, 65 % for sponsor, 60% for developers and 25% for mutable tools.

Following this change to the model, a test was conducted to check and measure the behaviour of the simulation by measuring the health of the system evolution when it is affected by change in the level of support of one of these participants. This test was conducted by reducing the degree of support of the project manager arbitrary to -0.4, while the other participants in the model retain their initial degree of support at a value of 1. "In the real-world, such temporary reductions in an individual's support could be due to causes such as financial or political pressures" [8]. This was represented in a simulation run of the Repast model by applying the condition of reducing the value of support of project manager in the test, to -0.4 for one time tick at tick 45, when the Repast model is run for 100 ticks.

4.2.1 Testing Project Manager's Influence

By conducting this test and reducing the support of the project manager for the evolution to -0.4 at tick 45, the result shows that the health of the system evolution declined to 0.921 at tick 45, as shown in Fig 11.

The result also shows that the health of the system evolution continues to decrease rapidly to 0.863 at tick 46, 0.834 at tick 47 and to its lowest value of 0.825 at tick 48. Then it gradually increases to 0.827 at tick 49, 0.836 at tick 50 and to 0.848 at tick 51, toward its stable trend. This behaviour of the health of system evolution shows that negative support and attitude of the project manager toward the development goal for just one tick time will reduce the health of the system to 0.825 in 3 ticks and will not return to its normal trend until tick 260 when the tick value is 0.999 ~ 1 as shown in Table 2. This means that the system will remain affected by this negative pulse for 215 ticks. The model therefore suggests that a short-term change in the behaviour of an important person in the process has resulted in a long-term perturbation to the process behaviour.

To further examine the behaviour of the Repast model, three additional tests were conducted by reducing the support of the participants of "developer, sponsor, and mutable tool" to -0.4 for one time tick at the tick 45 separately for each test. These tests were conducted similarly to the project manager's test by applying the same conditions to each of these other agents. The results of these tests show that the Repast model behaviour responses vary according to each change in the support of each of these participants in a manner equivalent to the results obtained from the project manager test.

Table 2:

The tick in which system health returns to stability

Health of system evolution	Tick ‘time step’
0.998	259
0.999	260
0.999	261
0.999	262
0.999	263

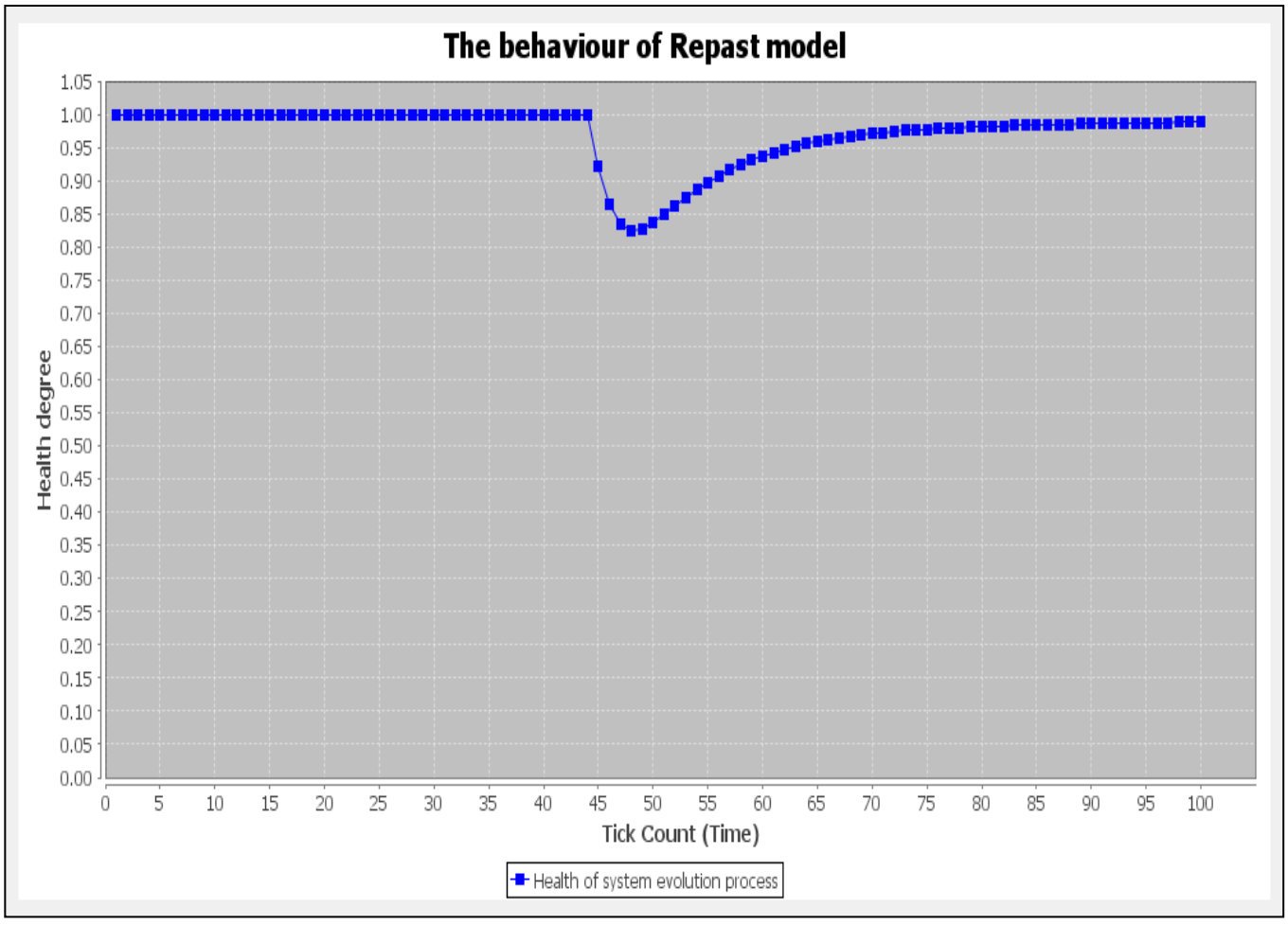


Fig 11: The behaviour of the Repast model in response to the test.

4.3 Results analysis:

The results of the test above show that the health of the system evolution process declined significantly as a result of reducing the support of the project manager, as illustrated in Table 3. It shows the participant for which the reduction of its attitude has the highest effect on the health of the software evolution. Moreover, the table shows the time tick in which the health of the system evolution is reduced to its minimum value and also the period needed for the evolution health to return to its stability for each test case.

Table 3:
Repast model responses in the test

Effect on the behaviour / Participants	Minimum health of system evolution	In tick (time step)	Period needed to return to stability
Project Manager (in influence test)	**0.825**	**48**	**215 ticks**
Sponsor (in comparative test)	**0.978**	**50**	**102 ticks**

5. Discussion

This section considers the results obtained from the modelling activities in the context of the Research Questions set out in Section 1 above.

RQ1: ***How does the ANT-based model of software evolution built in an SD simulation environment behave if reworked to an agent-based simulation environment, in comparison to the existing SD model?***

The results of the comparative evaluation presented in Section 4 above show that by re-implementing the ANT-based model of software evolution as a Repast agent-based simulation model, the new simulation model in comparison with the behaviour of the current SD model has the following characteristics:

- Similar behaviour to the SD model when there are no changes in the attitude of the participating actors and mediators.
- Similarly to the SD model, the Repast model also shows decreasing behaviour of evolution health when affected by a negative pulse applied to a participant.
- Greater stability in simulation runs; a return to stability after temporary changes in parameter values compared with the increasing oscillation behaviour of the SD simulation model.

By having the ability to measure the lowest point that the health of the software evolution reaches and being able to measure the time in which health returned to its stability in the Repast model as shown previously in Fig 8, the Repast simulation model shows that ANT-based model behaviour representing the health of software evolution can be calibrated quantitatively, *at least in theory*. This result can be considered as additional support to the conclusions drawn from the SD model calibration that, "the ANT-based model can be calibrated quantitatively, at least in theory" [8].

In addition, the Repast simulation model provides the ability to trace each participant in the model by using the participants' tracer tool which was developed specially for this purpose. Such functionality was not available in the SD model. Through this ability, the Repast model shows that all the participants in the actor-network are affected by a single pulse from an actor for one tick time generating complex multi-level feedback behaviour between these participants: see Fig 10. This behaviour supports the conclusion previously drawn by Wernick et al. [8] that the ANT-based model supports Lehman's 7th Law that "E-type evolution processes constitute multi-level, multi-loop, multi-agent feedback systems" [5].

This behaviour of the Repast model was also tested to reveal the factors that drive its stability trend. The results show that the stability trend of the Repast model behaviour is driven by its initial stability value before the negative pulse. This result requires more calibration in future work to check this deduction.

However, the comparative evaluation that was conducted to answer this question has the same weakness and limitation as the calibration of the SD model that was performed by Wernick et al. [8], since it was undertaken without reference to any data or assumptions that reflect the real-world aspect of software evolution. The value of all participants and their weighting factors are arbitrary. This means that it cannot yet be calibrated against real-world software evolution processes. Hence to address this limitation, an investigation was undertaken to answer the second research question:

RQ2*: Does the new agent-based software evolution process simulation model have the ability to reflect real-world software evolution?*

In order to answer this question a test was conducted by modifying the 'own weighting' values for developer, project manager, sponsors and mutable tools, based on values obtained through an expert interview.

This test was performed by resetting the weighting of own health of project manager to 70%. Then in this test, the support of the "project manager" was reduced to -0.4 in the tick 45 in order to measure its effect on the health of system evolution.

By calibrating the results of these tests against real-world factors of the above participants on project failures in industries, it been concluded

that these results are compatible with expected real-world software evolution process behaviour.

This compatibility gives confidence that the Repast simulation model of software evolution has the ability to reflect real-world software evolution if sufficiently accurate participant 'own weighting' and other parameter values can be obtained.

According to Wernick et al. [8]:

> "We intend to develop our current model into a more detailed simulation [an agent-based simulation model], and expect that this simulation, when calibrated to values representing real-world activities and actions, will be able to replicate behaviours observed in real-world software evolution processes".

Wernick et al. also illustrate the potential contribution of this step when they suggest that "Such a calibrated model would undoubtedly assist in improving the understanding of the global software process and its behaviours" [8]. However, the investigation conducted to answer this research question has some limitations that need to be addressed in order to reflect more accurately real-world software evolution process calibration in future work. These limitations and suggested future works are presented in section 6.

6. Conclusions and Future Work

6.1 Conclusion:

The outcome of this study is the development of a new agent-based 'Repast' simulation model of system evolution process based on an existing SD simulation model. This study has shown how the Repast simulation model behaves in comparison with the previous SD model. This work further demonstrates the ability of the Repast model to reflect the real-world process of software evolution by conducting an investigation built upon expert views of real-world software development, which forms a first step in representing real-world assumptions in the calibration of ANT-based software evolution modeling. Another outcome of this study is an observation tool, the 'Participants' tracer', to trace the behaviour of all participants in the model, something which was not possible in the previous simulation model.

6.2 Threats to validity:

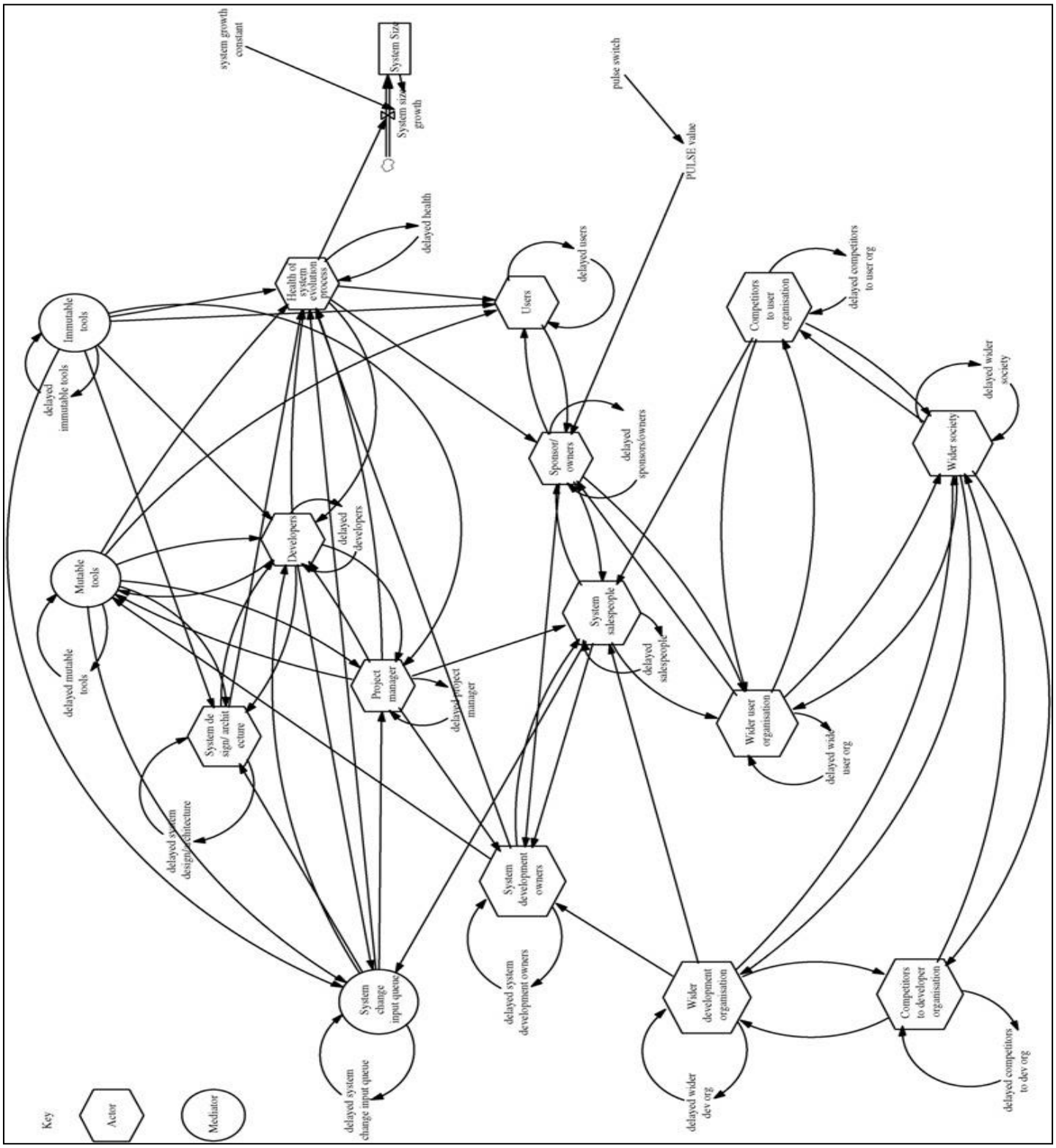
As presented previously, the assumptions in this work were based on an interview with an expert instead of real-world data. The reason of this was that the sort of soft data needed over a long-term real-world project will be difficult to capture and it might not be kept, or only be revealed in confidential or commercially sensitive information such as progress meeting minutes and opinions recorded in one actor's archives on another actor's apparent stance. However, this might cause a potential threat to validity. Therefore, in order to mitigate this threat, these assumptions were calibrated against real-world studies of the factors that impact on failures and success of software development in industry. With regard to this matter, [28] stated that "Bad decisions by project managers are probably the single greatest cause of software failures today. Poor technical management, by contrast, can lead to technical errors, but those can generally be isolated and fixed". Similarly, according to [29], the documented causes of software project failure show that the majority - 54% - are associated with project management, while technical and tools were the least likely factors at 3%. [30] Also argued that large software projects fail because of people, particularly the executive sponsor, rather than tools and technology used in the development of software. In addition, according to the [31], the most important person in the project is the sponsor. Accordingly, the above studies show that the assumptions made in this work are compatible with those of the real-world. In addition, another potential threat to the validity of this work was the methodoloy of comparative evaluation conducted to test the differences and similarities between the SD model and the new REPAST simulation model. Hence ,in order to prevent this threat, the compartive evaluation was carried out based on Vartiainen methodology as presented in section 3.

6.3 Project evaluation and Future work:

As stated in Section 4 above, the investigation conducted to check the Repast model's ability to reflect real-world software evolution has two main limitations which need to be addressed to support more results. The first limitation is that the tests conducted in this work were undertaken by changing the 'own weighting' factor of a limited number of participants, while the 'own weighting' of other participants in the network was still equally and arbitrarily weighted by 50%. The 'own weighting' for all participants needs to be calibrated based on research and interviews to reflect real-world behaviours. Following this it is recommended that further investigations be conducted by applying realistic temporary changes for the value for each actor and the results compared with real-world software evolution trajectories.

The second limitation is that the attitude value of each participant in the model was based on a generalisation from expert opinion, since the investigation was conducted without refereeing a particular system development project. This can be addressed by recalibrating the model, referring to a particular system development project in the real world.

Appendices



Appendix 1: The complete structure of the ANT-based model of the global software process which was used to develop the SD model [8].

Appendix 2: The declaration of the switch variables that control the links between the participants.

UserGlobalsAndPanelFactory.groovy

```
//"Representing the links for each Actor and switch it On and Off"//
addGlobal("developer_1",1)
addGlobal("user_1",1)
addGlobal("health_of_system_1",1)
addGlobal("imutabletool_1",1)
addGlobal("mutabletool_1",1)
addGlobal("project_Manager_1",1)
addGlobal("sponsors_value_1",1)
addGlobal("system_change_1",1)
addGlobal("system_design_1",1)
addGlobal("system_developer_owner_1",1)
addGlobal("salespepole_1",1)
addGlobal("competitors_to_developer_1",1)
addGlobal("competitor_to_user_1",1)
addGlobal("ownhealth_dev",1)
addGlobal("wider_development_1",1)
addGlobal("wider_society_1",1)
addGlobal("wider_organisation_1",1)
addSliderWL("switch_competitor_development", "switch", 0, 1, 1, 1)
// Additional development
addGlobal("motivation",1)
addGlobal("overtimes",1)
addGlobal("work_place_environment",1)
addGlobal("siz1",1)
```

References

[1] M. Lehman and L. Belady, "An Introduction to Program Growth Dynamics," *Statistical Computer Performance Evaluation*, pp. 503-511, 1972.

[2] O. Okwu, "Software Evolution: Past, Present and Future," *American Journal of Engineering Research (AJER)*, vol. 03, no. 05, pp. 21-28, 2014. [Online]. http://www.ajer.org/papers/v3(5)/C0352128.pdf

[3] P. Wernick, T. Hall, and C. L. Nehaniv, "Software Evolutionary Dynamics Modelled as the Activity of an Actor-Network," IEEE, 2006. [Online]. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4595502

[4] M. M. Lehman, G. Kahen, and J. F. Ramil, "Empirical studies of the global software process – the impact of feedback," 1999. [Online]. http://www.eis.mdx.ac.uk/staffpages/mml/feast2/papers/pdf/622.pdf

[5] M. M. Lehman, J. F. Ramil, and P. Wernick, "Metrics and Laws of Software Evolution - The Nineties View," in *Software Metrics Symposium, 1997. Proceedings., Fourth International*, Albuquerque, NM, 1997, pp. 20 - 32. [Online]. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=637156&tag=1

[6] P. Wernick and M. M. Lehman, "Software process dynamic modelling for FEAST/1," *Journal of Systems and Software*, vol. 46, pp. 193-201, 1999. [Online]. http://uhra.herts.ac.uk/handle/2299/637

[7] G. Kahen, M. M. Lehman, J. F. Ramil, and P. Wernick, "System dynamics modelling of software evolution processes for policy investigation: Approach and example," *Journal of Systems and Software*, vol. 59, no. 3, pp. 271–281, 2001.

[8] P. Wernick, T. Hall, and C. L. Nehaniv, "Software evolutionary dynamics modelled as the activity of an actor-network," *IET Software*, vol. 2, no. 4, pp. 321–336, 2008. [Online]. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4595502

[9] B. Latour, *Reassembling the Social.*: Oxford, 2005.

[10] B. Latour, "On Actor-Network Theory: A Few Clarifications," in *Soziale Welt*, 1996, pp. 369-381.

[11] M. Callon and B. Latour, *Unscrewing the Big Leviathan: How Actors Macro-Structure Reality and How Sociologists Help Them Do So*. Boston: New Genetics and Society, 1981. [Online]. http://www.tandfonline.com/doi/abs/10.1080/14636770320001472225

[12] C. Darryl, *A Brief Overview of Actor-Network Theory: Punctualization, Heterogeneous.*, 2009.

[13] R. Heeks, "Development Studies Research and Actor-Network Theory," Institute for Development Policy and Management, 2013.

[14] M. Loomes and Ch. L. Nehaniv, "Fact and artifact: reification and drift in the history and growth of interactive software systems," in *CT '01 Proceedings of the 4th International Conference on Cognitive Technology: Instruments of Mind*, 2001, pp. 25–39.

[15] S. F. Railsback, S. L. Lytinen, and S. K. Jackson, "Agent-based Simulation Platforms:Review and Development Recommendations," *Society for Modeling and Simulation International*, vol. 82, no. 9, pp. 609-623, 2006.

[16] N. Gilbert, *AGENT-BASED MODELS.*: sage, 2008.

[17] Ch. M. Macal and M. J. North, "AGENT-BASED MODELING AND SIMULATION," *IEEE*, 2009.

[18] E. Bonabeau, "Agent-based modeling: Methods and techniques for simulating human systems," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, pp. 7280–7287, 2002. [Online]. http://www.pnas.org/content/99/suppl_3/7280.full

[19] D. Helbing and S. Balietti, "Agent-Based Modeling," in *How to Do Agent-Based Simulations in the Future: From Modeling Social Mechanisms to Emergent Phenomena and Interactive Systems Design.*: Springer, 2012, pp. 25-70.

[20] S. E. Page, *Agent Based Models*. New York: The New Palgrave Dictionary of Economics, 2005.

[21] Mi.l J. North, N. T. Collier, and J. Vos, "Experiences Creating Three Implementations of the Repast Agent Modeling Toolkit," *ACM Transactions on Modeling and Computer Simulation*, vol. 16, no. 1, pp. 1–25, 2006. [Online]. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.331.2313&rep=rep1&type=pdf

[22] Repast Simphony. (2013) Repast. [Online]. http://repast.sourceforge.net/repast_simphony.php

[23] J. Kielbasa, "An Introduction to RePast using ReLogo with Groovy," 2013. [Online]. http://www2.econ.iastate.edu/tesfatsi/RepastSIntroUsingGroovey.JakubKielbasa.2Sept2013.pdf

[24] J. Ozik, "RELOGO GETTING STARTED GUIDE," 2014. [Online]. http://repast.sourceforge.net/docs/ReLogoGettingStarted.pdf

[25] P. Vartiainen, "On the Principles of Comparative evaluation," *SAGE Publications*, vol. 8, no. 3, pp. 359–371, 2002. [Online]. http://evi.sagepub.com/content/8/3/359.full.pdf

[26] Linkedin. (2014) linkedin. [Online]. https://www.linkedin.com/pub/ed-wakelam/1/152/aa9?trk=seokp_posts_secondary_cluster_res_author_name

[27] E. Wakelam, "The factors on software evolution in software inudstries," march 25, 2015.

[28] R. N. Charette, "Why software fails," *Spectrum, IEEE*, vol. 42, no. 9, pp. 42 - 49, 2005. [Online]. http://spectrum.ieee.org/computing/software/why-software-fails

[29] J. Gulla, "Seven Reasons IT Projects Fail," 2012. [Online]. http://www.ibmsystemsmag.com/power/Systems-Management/Workload-Management/project_pitfalls/?page=1

[30] D. Smith, "Why do most IT projects fail? It's not because of technology," 2008. [Online]. http://www.bizjournals.com/portland/stories/2008/10/20/smallb4.html?page=all

[31] C. H. A. O. S. Manifesto, "Think Big, Act Small," The Standish Group International Inc., 2013.