A tool for evaluation of the Software Development Process

Peter Henderson (P.Henderson@ecs.soton.ac.uk)
Yvonne Margaret Howard, (Y.M.Howard@ecs.soton.ac.uk)
Robert John Walters, (R.J.Walters@ecs.soton.ac.uk)

Declarative Systems and Software Engineering Group
Department of Electronics and Computer Science
University of Southampton
Southampton, UK,
SO17 1BJ.

Abstract

As part of their effort to reduce development time and improve the quality of their products, Software Developers are looking at the software development *process* they use. They reason that a high quality process is likely to lead to a high quality product. However, few developers receive high ratings on the accepted scales when their process is evaluated.

To achieve the highest ratings for their processes, developers need to measure and manage their process and to do this effectively, they need tool support. However, they also have significant investments in their existing processes and support systems which they cannot afford to abandon. This makes it difficult for them to adopt the process model implicit in most process support tools.

In this paper, we describe our tool, RolEnact which attempts to address the requirements for a tool for the simulation and evaluation of software development processes and illustrate its use with an example.

Keywords: Process Modelling, RolEnact, Simulation, Software Development Process.

1. Introduction

There is continuing pressure on Software Developers to improve both the quality and speed of development of software. One response is to review and to improve the software development *process*. There are accepted schemes [5, 8] which attempt to classify the software development process (not the software produced) according a measure of its "quality" or "maturity". However, few software developers have high ratings on these scales.

One reason is that, despite their modest ratings, software developers have a considerable investment in their existing development process and the infrastructure supporting it. They could not adopt an "ideal" process even if such a process could be identified and defined. Instead, they need to adapt and enhance their existing processes. To achieve the highest ratings, developers need to measure and manage their processes.

We describe RolEnact which attempts to address the requirements for a tool to support the simulation, evaluation and improvement of software development processes.

2. Requirements for an evaluation tool

The tool must be able to work with an understanding of the existing process. Typically this process will be unique to the developer as it will have evolved over a period of time.

The tool needs to be easy to understand as many of the people who will use it will not be familiar with process modelling.

Once a description of the existing process (or part of the process) has been captured, the tool must then be able

to run simulations of the process using appropriate data from the real process (where appropriate).

The tool needs to be able to adapt to the process under consideration as it develops and to allow novel views of those processes as well as providing features to enable analysis and evaluation of the process.

3. Motivation

Most schemes and tools directed at helping developers to monitor and improve their processes have a notion of how the process operates which is essentially fixed. To make these systems work the developers may need to collect data which is not readily available and be prepared to accept analysis results which require further interpretation.

For example, consider an experiment carried out by ICL to improve cost estimation of software projects [1]. The experiment looked at a novel method for monitoring and controlling the progress of a software project. The results show two typical features of this type of exercise:

- The figures collected are analysed manually because either the figures themselves or the analysis required cannot be handled by the existing support system.
- The figures show unexpected features. In this case, the data suggests that almost half of the work of the project was carried out in the penultimate month.

The first of these problems is often addressed by the construction of a custom support tool which is tailored to the particular situation. However, once built and implemented, the new tool can be expected to cause the similar problems to the one it replaced. "Exceptional" behaviour may be dealt with by adjustments to the data to more accurately reflect "what must have happened" or by explaining away the analysis results. Neither of these approaches is really satisfactory.

We believe that a better approach is to modify the analysis of the model and the model itself in the light of the observed behaviour. To do this, we need analysis and simulation tools which are able to adopt and evolve with (understanding of) the process under consideration.

4. Description of RolEnact

We have a modelling and simulation tool called RolEnact [2]. RolEnact views a process as a collection of interacting "Roles". Each Role in a model corresponds to a collection of actions or responsibilities of some form of actor in the process. In some models, it is appropriate for the roles to map directly to the people involved in the process or the jobs that they perform. In others, the mapping is less direct with Roles corresponding to general responsibilities. Some of these may be shared between teams of people (such as a committee) and some individuals may perform more than one Role. The activities of some Roles may also be undertaken by different individuals at different times.

Roles in RolEnact have a named state and change state by taking part in events. There are four types of event:

Action: In an action, a Role makes a unilateral change of state.

Interaction: In an interaction, a Role causes a change of state of a known instance of another type of Role at the same time as it changes its own state. For the interaction to be possible, both the "driving" role and the other Role must be in a specified "before" state.

Selection: A selection event is like an interaction in that the "driving" Role and another role of a specified type change state simultaneously. A Selection event is distinguished from an Interaction event in that the "driving" Role may interact with *any* instance of the required type of the other Role which is in the required "before" state. On execution of a Selection event, the Roles involved retain a reference to each other so that they may Interact at a later time.

Create: In common with the other types of event, a Role is only able to execute a Create event when it is in the required "before" state. Execution of the event causes the "driving" Role to change state and a new instance of the stated type of Role to be created. As with a Selection, the Roles involved retain references to each other enabling them to interact. This ability for new instances of Roles to be created as part of the execution of a model distinguishes RolEnact from other modelling languages.

A RolEnact model is built by describing the "events" that may happen during execution of the model. Describing events in this way can make understanding the possible actions of a single Role more difficult than with a language like CSP in which communication is achieved using shared events which are described in both communicating processes (because the events of other Roles may cause changes of state in this Role). However, we find that most non-specialist modellers have difficulty constructing models which use the shared event style of communication. They seem to prefer to make a single description of the event in which the transitions of both Roles are described.

RolEnact models may be written using a simple syntax which is described fully elsewhere [7] or constructed using the model generator. Models may be stored as simple text files. Alternatively, they may be stored in a sheet of an

Excel spreadsheet. This latter storage format has been added to enable users of the simulation tool to store their model in the same file as their analysis results. The Simulator generates its results into an Excel spreadsheet because it was felt that the users would wish to perform additional analysis on the output from the Simulator and use the results in a variety of documents.

RolEnact has a collection of tools which support the development and analysis of its models [3, 4, 7]. The complete set comprises a model building/visualising tool, a "stepper" for interactive execution of a model and the Simulator which executes a model automatically.

4.1. The RolEnact Model Generator

For a process model to be useful, it needs to be accessible to everyone involved in the process. Our experience suggests even a simple model description language like that used by RolEnact represents a significant barrier to many people. The generator enables a modeller to build a model without writing code. When using the Generator, the modeller has the opportunity to view the model as text, or as a diagram in the style of a "RAD" [6]. When viewing a model as a diagram, the modeller has the option of viewing the entire model, or just a representation of a single Role. Figure 1 shows an example of a diagram describing a single Role in a model.

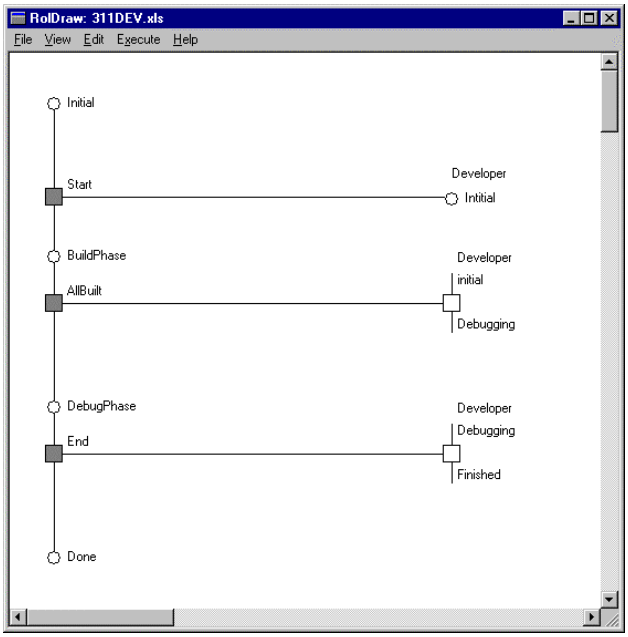

Figure 1: A view of a single Role in the model

A model is constructed using a collection of dialogue boxes and the modeller works by making selections from lists wherever possible.

4.2. The RolEnact Stepper

Using the Stepper, the modeller is able to execute a RolEnact model. If the model is being constructed using the modelling tool, the Stepper can be invoked at any time using a command from the "execute" menu. Once loaded into the Stepper, the modeller is presented with an application in which there is a window for Role in the model. These windows show the identity of each Role (which is derived from its type, suffixing a digit to distinguish each instance), the present state of the Role and a list of events which the Role is able to initiate. Execution proceeds with the modeller selecting an event and "making it happen" by a double click on its name. After each event, the modeller sees the states of the affected Roles change and the lists of available events are re -evaluated for all Roles in the model.

A particular feature of the Stepper is that events may be "undone". This greatly assists in the manual exploration of the behaviour of a model since, when faced with a situation where there is a choice of actions, it is possible for the modeller to return to the point in execution where a choice was made and continue execution using a different option. Being able to step backwards through an execution of the model is also very useful when looking for the cause of unexpected or incorrect behaviour in a model: our experience using executable models also suggests that in these situations, execution usually proceeds for while after the problem (or the event that leads to it) has occurred before it is noticed. Usually this means the modeller has to restart the model and retrace the execution. Being able to step back makes this type of task much easier in RolEnact.

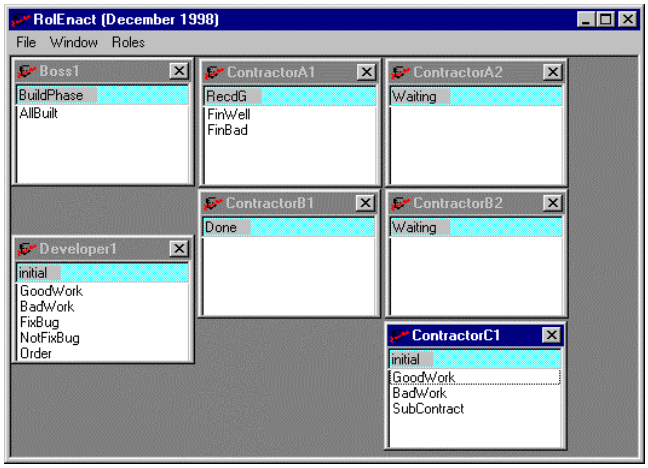

Figure 2: RolEnact running a model of a development project

4.3. The RolEnact Simulator

The RolEnact Simulator takes a completed RolEnact

model and runs it automatically. The model runs in exactly the same manner as in the Stepper: the system evaluates which events each of the Roles in the model is able to initiate, selects one and "actions" it, the affected Roles change state, the list of available events are recalculated and the cycle is repeated.

Although execution of a model using the Simulator is similar to execution using the Stepper unlike the Stepper, where the modeller selects the sequence of events, at each execution cycle, the Simulator has to select an event itself. The Simulator also records the sequence of events into an Excel spreadsheet for later evaluation. Excel was chosen for the features it has which a modeller might wish to use in analysing a model and the ease with which Excel spreadsheets can be incorporated into other documents.

The Simulator application is shown in Figure 3. In addition to buttons for starting a simulation, continuing from the present state, and stopping, the application shows a list of the Roles which presently exist in the model (and how many more interactions of the model must occur before each is able to initiate another event), a list of the most recent events in the simulation and the number of iterations that have taken place so far. The application also shows the total of the "likelihood" values for all of the presently available events. The modeller may wish to know this number since, when this figure is below 100, the model proceeds as if these figures were the probability of an event being chosen when it is available as a percentage. If no event is chosen, "--" is placed in the list of events and the selection process is repeated. If the total exceeds 100, it is certain that an event will be selected in the interaction, but the probability of any particular event occurring in any particular iteration will depend on the numbers attached to the other available events. If the model reaches a state where there are no possible events, the application stops.

Other features of the application permit the modeller to select a worksheet from the model workbook into which to write the results from this simulation, set a maximum number of iterations to complete before stopping, and either run the simulation at a speed intended for the modeller to watch the events as they occur (slow) or as fast as possible so that the results can be examined later (fast).

In building the Simulator, we found that one of the more difficult problems to address is that of selecting which of the available events to select at each interaction of the model. We experimented with policies which were intended to select events according to some approximation to "fairness" such as making a (quasi-)random selection from the list of available events. However, we found in general these are unsatisfactory. Events which caused particular problems included:

- Create events. Events which Create new instances of Roles typically occur rarely during execution of a model. Permitting them and other events to be chosen equally often leads to executions of models where Roles proliferate, but each Role fails to make progress.

- "Exceptional" events. Many models contain events whose nature is that, in a realistic execution of the model they would only ever be selected when no other event is possible. For example; in a model of the interactions between a Client and Server processes over a network the Client Role (or process) might have an option to "timeout" in the event of failure in the Server, but this event would only be invoked when the Server has failed.

- "Usual" events. The opposite of "Exceptional" events. We observe that Roles typically have a subset of events which are nearly always selected when they are available.

In order to enable the modeller to achieve "reasonable" executions of models in the Simulator, the modeller is able to add figures to the RolEnact model. These figures influence the selection of events by the Simulator. Two numbers may be specified for each event in the model. The first is the probability of an event occurring at each iteration as a percentage, *given that the event is possible*. If during execution, the model reaches a state where the sum of the probabilities for the available events exceeds 1, one of the events will always be selected and the selection is based upon the relative probabilities of the available events. The other number specifies how much time (in iterations of the model) must pass before the Role that initiated the event may initiate another.

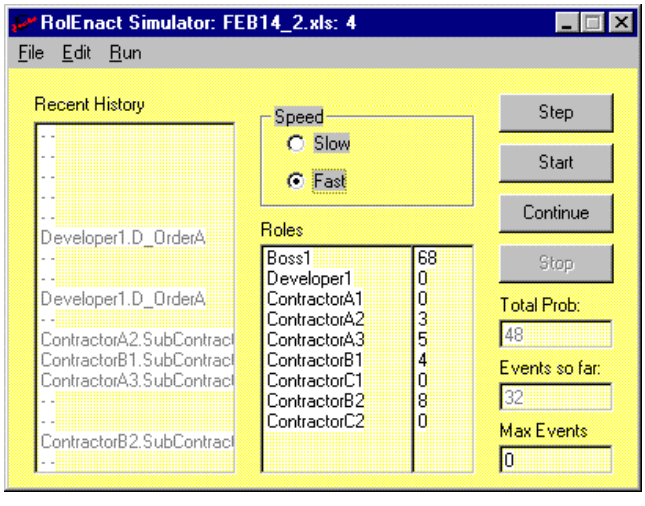

Figure 3: The RolEnact Simulator

In the present implementation of the Simulator, the

complete description of the model (including the figures which guide the selection of events by the Simulator) is stored in the first page of an Excel workbook. The Simulator stores the results of automated executions of the model into subsequent worksheets of the same workbook.

5. An Example process modelled using RolEnact

As our example, we consider a development process in which a product is to be constructed by a developer who divides the work into a number of pieces. Each of these pieces is either build "in-house" by the developer or the subject of an order from a contractor. Contractors are permitted to use the services of (sub-)contractors. In addition to this, it is assumed that the Developer will expend effort on correcting errors.

The Role "Boss" represents the customer, the "Developer" role represents the prime contractor. The "Developer" can either build (parts of) the product itself or place work with one or more contractors (who may also choose to sub-contract work). At each stage the work of each of the Roles can be either "good" or "bad". The relative numbers of these types of event is used in an analysis of each run of the model as a measure of the quality of the product. The Developer also has the ability devote effort to attempting to correct problems.

As an example of the type of analysis that might be performed, if the developer were to be more inclined to pass work to contractors, then it seems reasonable to expect that a poor sub-contractor will have a detrimental effect on the quality final product. At the same time, using more contractors might lead to an improvement because of the extra time and effort the developer is able to devote to correcting problems. The number of interactions between players in the process and the extent of their interdependence makes it hard to understand the effect of such changes. However, a complete understanding is not required. In this context, the information which is really needed is: if *this developer* were to change their process in this way, what would the effect be on the quality of the product? Using the RolEnact Simulator and figures from the real process, it is possible to predict the effect of a change in the developer's policy regarding sub-contracting.

6. Conclusion

Today we use huge software systems which are both highly featured and reliable and it would be easy, based on the size, reliability and performance of these systems, to assume that the problems of building them have been solved. However, the process for creating software systems is much less mature than the products we see and this is evidenced by the low ratings of most software developers in evaluations like CMM and SPICE [5, 8].

Software developers have major investments in their existing processes and support systems which they cannot afford to abandon so, if they are to achieve a high rating on these scales, they need to evaluate and improve their existing processes. To do this successfully, they need a new breed of flexible process modelling and analysis tools. RolEnact is offered here as an example of the sort of tool which supports process improvement.

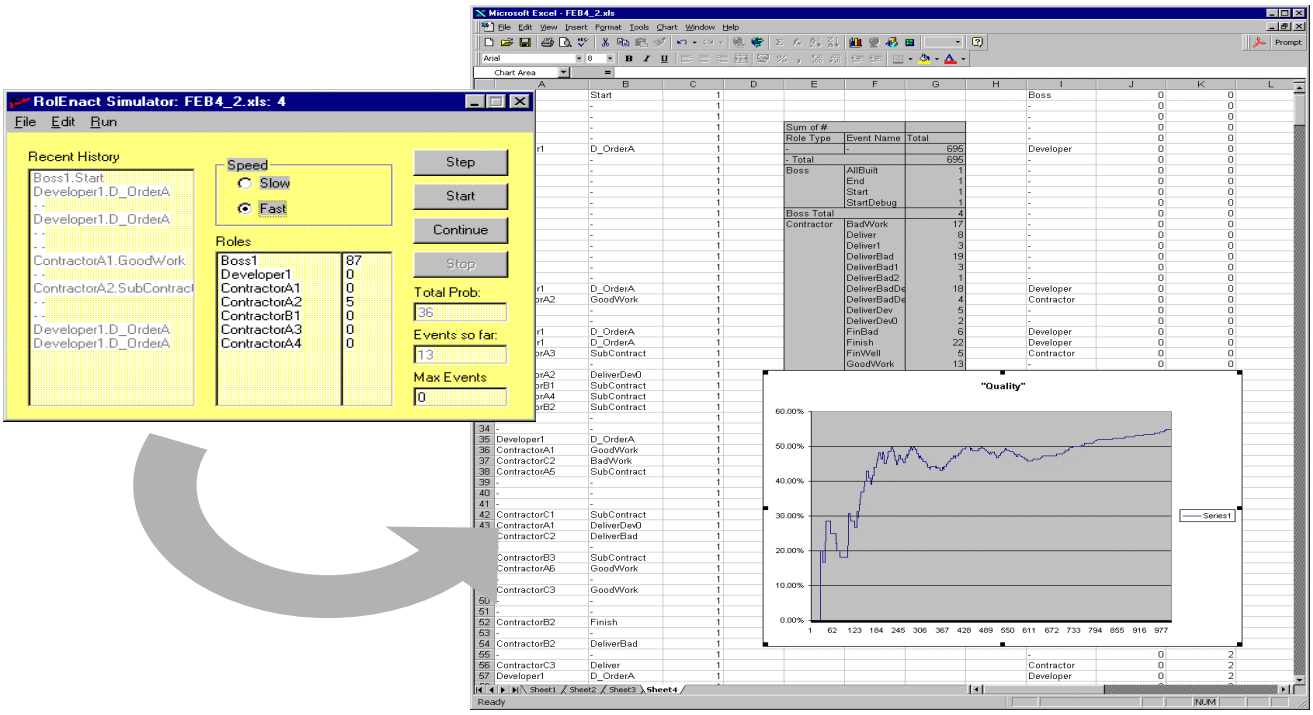

Figure 4: RolEnact Simulator paused whilst running a simulation of the project modelled in Figure 2 and showing a chart created in Excel of the "quality" of the product as execution of the model proceeds.

7. References

[1] B. Chatters and P. Henderson, "An Experiment to Improve Cost Estimation and Project Tracking for Software and Systems Integration Projects," *EuroMicro99*, Milan, 1999.

[2] P. Henderson and R.J. Walters, "Behavioural Analysis of Component-Based Systems," to appear, available from http://www.ecs.soton.ac.uk/~ph/papers/bacbs.pdf

[3] P. Henderson and R.J. Walters, "Component Based systems as an Aid to Design Validation," *14th IEEE International Conference on Automated Software Engineering (ASE99)*, Cocoa Beach, Florida, 1999, pp. 303-306.

[4] P. Henderson and R.J. Walters, "System Design Validation Using Formal Methods," *Tenth IEEE International Workshop on Rapid System Prototyping (RSP99)*, Clearwater, Florida, 1999, pp. 10-14.

[5] J. Herbsleb, D. Zubrow, D. Goldenson, W. Hayes, and M. Paulk, "Software quality and the Capability Maturity Model," *Communications of the Acm*, vol. 40, pp. 30-40, 1997.

[6] M.A. Ould, *Business Processes - Modelling and Analysis for Re-engineering and Improvement*: John Wiley and Sons, 1995.

[7] K.T. Phalp, P. Henderson, G. Abeysinghe, and R.J. Walters, "RolEnact - Role Based Enactable Models of Business Processes," *Information And Software Technology*, vol. 40, pp. 123-133, 1998.

[8] J.M. Simon, "SPICE: Overview for software process improvement," *Journal of Systems Architecture*, vol. 42, pp. 633-641, 1996.