Modelling Processes Using RAD and UML Activity Diagrams: an Exploratory Study

M. Odeh, I. Beeson, S. Green and J. Sa
Systems Modelling Research Group
Faculty of Computing, Engineering & Mathematical Sciences
University of the West of England
Coldharbour Lane
BRISTOL BS16 1QY

tel: +44 (0)117 965 6261
fax: +44 (0)117 344 3155
email: mohammed.odeh@uwe.ac.uk (or ian.beeson or stewart.green or jin.sa @uwe.ac.uk)

Abstract

In Software Engineering and Information Systems, increasing attention has been focused lately upon modelling organizational processes - as a starting point for developing computer-based systems to support (or control) such processes. A number of process modelling methods are available, but it is not yet clear what the relative merits of these are, nor whether they might be more or less useful in particular contexts. We have applied two well-known process modelling techniques, Role Activity Diagramming and UML Activity Diagramming, to a particular process in our own organization, that of managing the registration of research students. We developed an RAD first and then translated it into a UML AD, to compare the two techniques and check the feasibility of such translation. We conclude that translation from RAD to UML AD is likely to be feasible in particular cases, but will rely on the ability of the translators to establish and maintain the equivalence between the two (i.e. the equivalence will be partly a matter of local interpretation).

1. Introduction

As IT systems have penetrated further and further into everyday organizational processes, interest has been growing in how we might understand and represent those processes in a sufficiently comprehensive but also lucid manner, to serve as a basis for introducing new technological arrangements which will effectively support, and perhaps in some cases direct, the organizational processes. From a software engineering perspective, the task here is not so much to capture a process in order to automate it, as to comprehend a process in detail in order that the human activities and interactions which constitute the process can be given appropriate technological support or can be in some ways improved or better integrated through the inclusion of computer-based components. Ould has observed that this has involved a switch of emphasis away from information structures and flows as such towards the activities that individuals and groups carry out in their work in organizations:

> "...in our approach to process modelling we concentrate unashamedly on what people *do*, rather than on what people do it with. Once we have chosen an organizational structure and the processes it will operate, then we can decide on the information needed by individuals and groups to perform those processes in that organizational structure."
> (Ould, 1995, p. 16)

In an earlier study (Beeson et al, 2002), we used Role Activity Diagramming to investigate the processes linking strategic decision making with information systems provision in an insurance company. We argued in that paper that the interconnection between the general business and IT/IS units in the company, and between strategic and operational levels, was being achieved more by dynamic and continuous processes of negotiation and communication than by the implementation of a

fixed strategy or architecture. We found that RADs were an effective tool for charting these processes and had some evidence that they also helped organizational members grasp and verify the processes.

The present paper emerged out of our interest in the similarities or differences between role activity diagramming as we had used it in the previous study and activity diagramming in UML. We were aware that the Unified Modelling Language (UML) was becoming increasingly important in software engineering, and that it included Activity Diagrams as one of its techniques. We understood that UML aimed to provide system developers with a coherent set of notations which would help them visualize, specify, and build a software system (Scott, 2001, p.1). Several related notations are included in UML, including Use Case, Class, Interaction, State and Physical Diagrams as well as Activity Diagrams. Because this appears to hold out the promise of an orderly development process from activity diagram to implementation, it seemed to us worthwhile to see whether we could use UML activity diagrams to describe a business process we were interested in. This would be an initial study, not venturing beyond activity diagramming into other UML techniques at this stage.

It should be said that Activity Diagrams are not an original part of UML, but were added later, and draw on a number of earlier techniques for modelling events, states, and workflows (Fowler & Scott, 2000, p. 129). They appear to be an attempt to project or extrapolate an object-oriented approach into the organizational context of the software development process. As such, they may reflect an under-developed conceptual framework for representing the activities of human beings in a business process (as distinct from software objects in a system). The books on UML differ as to what Activity Diagrams are good for: whereas Fowler and Scott call them "a great tool for workflow modelling" (p. 137), Stevens and Pooley, rather more cautiously, remark that "although activity diagrams can be useful for modeling workflow, there is a lot more to business modeling than this." (p. 146). So ADs in UML do not yet present the clear focus on modelling business processes found in Ould's use of RADs.

Since we already had experience in using RADs, we decided to proceed by first building a RAD of a process we were interested in, and then generating an equivalent UML AD. We might alternatively have generated RADs and ADs independently, and compared them. But our principal interest is not so much in comparing the two techniques as in seeing how one might translate from RAD to AD. If that turns out to be a straightforward process, the benefit might be that we can then proceed more easily into the mainstream of UML techniques. There might be a case, if it turned out to be as easy to produce an AD from scratch as from an intermediate RAD, for adopting UML activity diagramming at the outset, instead of role activity diagramming. The present paper thus explores and reflects on our experience of using these two kinds of diagramming, one after the other, in a particular case. The theoretical or conceptual concerns alluded to in the previous paragraph are not taken further in this paper, but awareness of them will produce a degree of caution in our conclusions.

2. The Case Study

The case study used in the work presented in this paper is part of a real life process in the administration of research degrees in the Faculty of Computing, Engineering and Mathematical Sciences (CEMS) at UWE Bristol. The full research degrees process is a rich one extending from student selection to final examination. In this paper, an early section of the process, covering admission, enrolment and registration of research students is used to illustrate our work. The CEMS research degrees administration process is available on our Faculty website (CEMS, 2002).

The rationale for choosing this particular process are as follows:

- The full description of the research degrees administration process is written in a formal manner. There is a demand from users to have a description which is easier to follow, and which might usefully complement the formal document.
- All authors of this paper are involved in the research degrees administration process and so have a direct interest in understanding the process as well as in modelling it.
- Some of the authors are closely involved with the design and modification of the research degrees administration process. Therefore there is a real interest in process improvement.
- An academic audience will have a good understanding of and interest in the selected process.

3. RAD Description of the Selection, Enrolment and Registration Process

3.1 Role Activity Diagram Notations

We have chosen to use Ould's **R**ole **A**ctivity **D**iagrams (RADs) (Ould, 1995) to model our process initially. The basic concepts of RAD were first introduced by Holt et al (1983), and later enriched by Ould (1995). The models presented in this paper are defined using Ould's variant of RAD, called STRIM (**S**ystematic **T**echnique for **R**ole and **I**nteraction **M**odelling'). Using STRIM, a process is modelled as a number of *roles* that interact with one another. A role can be thought of as a set of related responsibilities that can be carried out by a machine, a person, or a group of people. Within each role there are a number of *activities* that take place in a certain order. The RAD notations used in this paper are explained in Appendix A.

3.2 RAD model of the Selection, Enrolment and Registration Process

The RAD model in figure 1 illustrates the process from admission to registration after a successful interview, where the proposed supervision team would like to offer a place to the candidate student and thereafter the candidate would like to accept this offer.

There are four roles in this process: *Director of Studies* (DoS), *Student*, *Research Admini*strator (RA) and the *Research Degree Board* (RDB). Each role is responsible for a number of activities in this process. The proposed DoS (and at least one other member of staff) interview the candidate (student). Following a successful interview, the proposed DoS needs to make sure that all financial issues are resolved before informing the RA about the potential student by completing and sending an RD0 form. Once an RD0 form is received, the RA sends an official offer letter to the student. The student sends a confirmation of acceptance to the RA. It is assumed that in this process the student will normally accept the offer.) The RA then creates a student record. When it is the starting date, the RA enrols the student on the programme. The actual enrolment date is recorded. The proposed DoS and the proposed supervision team work together with the student to complete the RD1 form in order to make an application for registration. The proposed DoS sends the RD1 form to the RA within 3 months of enrolment. Then, the RA sends RD1 form to the RDB for consideration. When the RDB meets, the RD1 form is evaluated and the outcomes are sent to the RA. The RA records the outcomes and informs the DoS and the student. If the outcome is "not approved", both DoS and student revise the RD1 form together. The proposed DoS sends the revised RD1 form to the RA. Then, the above process will be repeated. If the outcome is "approved", the registration process is completed.

4. UML Model of the Selection, Enrolment and Registration Process

Although activity diagrams are used to model the dynamic aspects of software-intensive systems, we use activity diagrams in this research to model collaboration in a real business process, namely the process of selection, enrolment, and registration of postgraduate research students.

We use activity diagrams to mainly model flow of work between activities. It is worth mentioning that there is no distinction in the notation between action and activity states in UML. Action states are executable atomic computations such as the execution of an expression. But, an activity state may represent or be further decomposed into further action and activity states. In UML action or activity states are represented using a lozenge shape with horizontal top and bottom and convex sides. The *"Interview Candidate"* in figure 2 represents an activity state.

When an action or activity state has finished executing, control is passed to the next action or activity state, which is modelled as a transition. A transition is modelled as a directed arrow from the source activity to the target activity. For example, the directed arrow representing the transition from the *"Interview Candidate"* (source state or activity) to the *"Resolve Financial Issues"* (target state or activity). Though these transitions are trigger-less, they can have guard conditions which should be met before the target state is entered.

Branches from a transition may happen based on a certain condition (logical expression) that has been satisfied. A branch is a represented as a diamond that has one incoming transition and one or more outgoing transitions based on guard conditions to be met. For example, after the *"Check RDB Outcomes"* is executed by the DoS, a check is made whether the RD1 form has been approved or rejected (guard conditions *"RDB Outcomes=Rejected/Approved"*) with resulting respective transitions as shown in DoS processing part in figure 2.

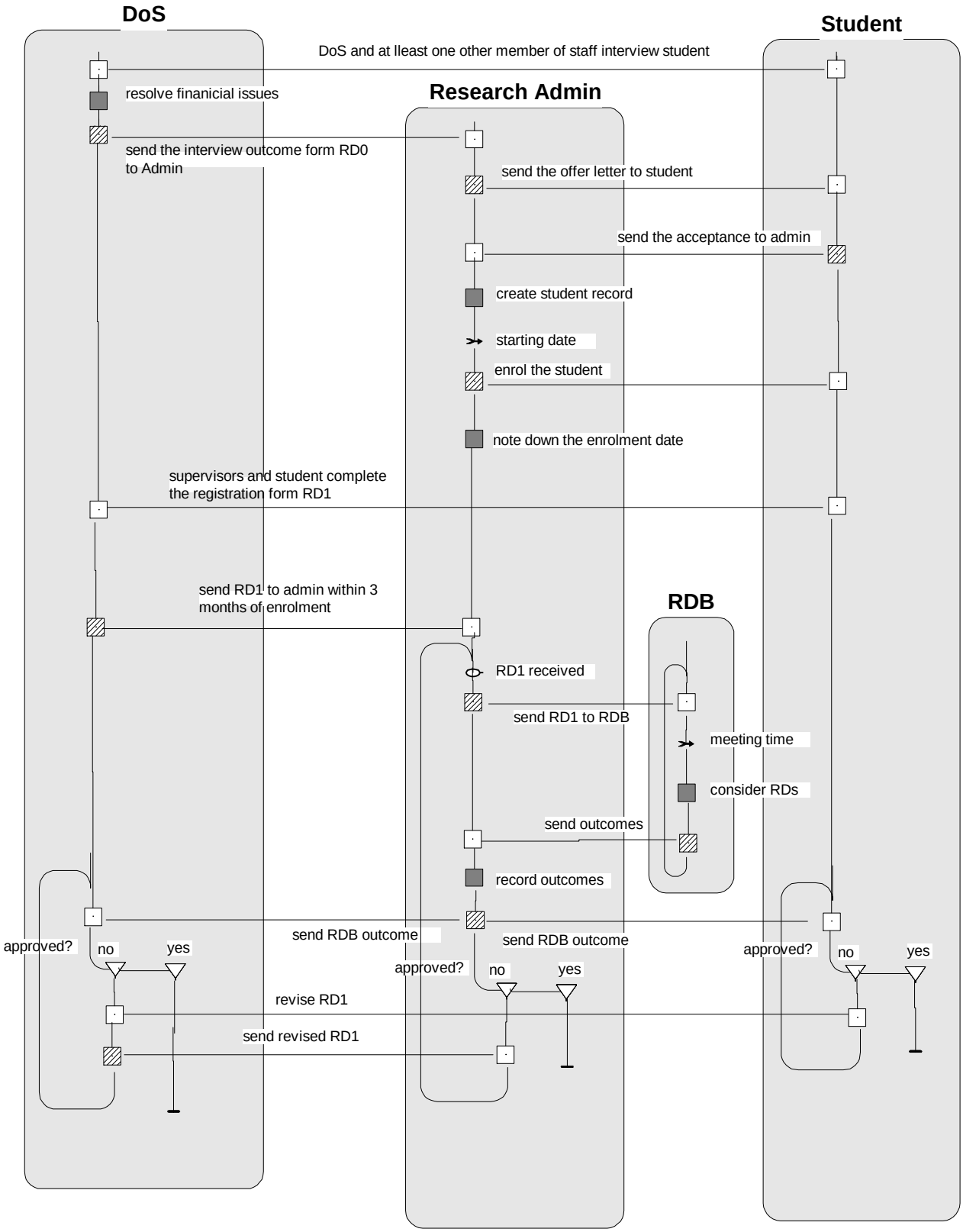

Figure 1: RAD for the Admission, Enrolment and Registration process.

In addition, it is possible to model concurrent workflows of business processes using fork and join in activity diagrams. Synchronization bar (a dark thick one) is used in UML to specify the forking and joining of concurrent workflows. For example when the *"Send RDB Outcomes on RD1"* is executed, transitions to two activities are forked, namely *"Check RDB Outcomes"* activities by both the DoS and Student. Then, a join bar is used to specify the synchronization after executing these two activities to lead to the "*Send Revised RD1 Form"* activity.

It is often the case that when modelling the workflows of a certain business process to partition the responsibilities/roles performed by individual actors in the problem domain. As a result this partitioning would result in explicit allocation of activities to individual actors. We use the concept of Swimlane in activity diagrams to represent these roles in RAD. Swimlanes are used to partition the PAEARP activity diagram in figure 2 into four partitions: Director of Studies, Research Administrator, Research Degree Board, and Student. Furthermore, one may represent this partitioning as four packages in UML, where each package is decomposed into a set of activity states and there respective transitions. Finally, it is worth mentioning that further work is being carried out to specify a process on how to transform a RAD Process Model into UML Models.

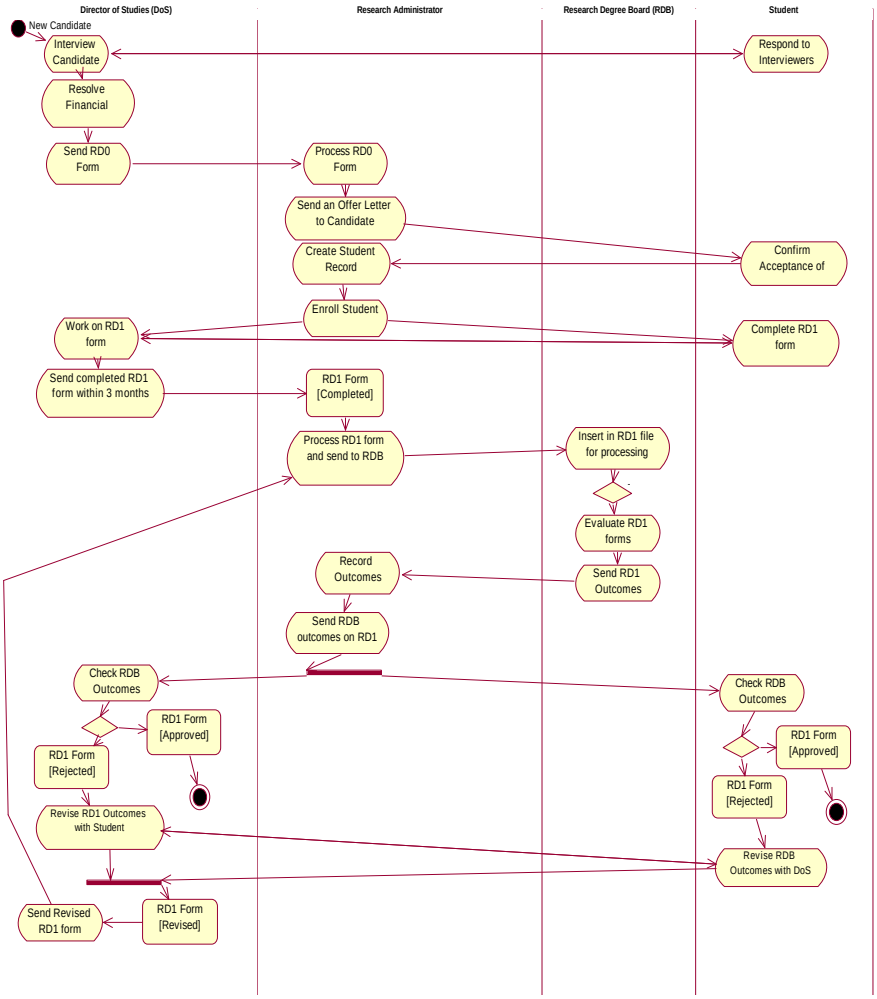

Figure 2: UML AD of the Admission, Enrolment, and Registration process.

Table 1: A Comparative Analysis of RAD and UML Activity Diagrams

Process concept	RAD notation	RAD semantics	AD notation	AD semantics	Notes
Activity	A black box labelled alongside by the activity name.	An activity expressed by the verb and noun in the activity name	A rectangle with rounded ends with the activity name inside.	An activity expressed by the verb and noun in the activity name.	The AD activity comprises one or more actions: entry, lifetime, event or exit actions.
State	A line connecting activities.	Denotes a state or sub state of a role.	A rectangle with rounded corners.	Denotes a state.	
Transitions	A line into a black or transparent box and a line out of the same box.	The activity changes the precondition state to the post condition state.	An arrow from state to state, activity to activity, state to activity, or activity to state.	A state may change to another. An activity may end and lead to another activity. An activity may end and lead to a state. A state may change and lead to an activity.	The semantics of the arrow in ADs is potentially confusing. See the discussion below.
Goal	A magnifying glass across a "state" line annotated with a state name	Denotes that this state includes the named component.	Not supported explicitly.	Not applicable.	In RADS this is often used at the end of a process to indicate a goal satisfied.
Start state	Not supported explicitly.	Not applicable.	A big, black dot	Denotes the initial state of a process	The RAD magnifying glass could be used on a RAD to indicate a "start" state.
Stop state	Not supported explicitly.	Not applicable.	A big black dot with a circle around it.	Denotes an end state of a process.	The RAD magnifying glass could be used on a RAD to indicate a "stop" state.
External event	A small arrow	Denotes an external event	Control icon	Denotes an external event	
Sequencing activities	A black box A1 connected by a line to another black box A2	Denotes that activity A1 will be followed by activity A2	An open box with rounded ends A1 followed by an arrow to another A2	Denotes that activity A1 is followed by activity A2	
Looping	A line that loops upwards to join itself higher up.	Iteration of activities.	An arrow that loops back to an earlier state or activity.	Iteration of activities and/or states.	
Role	A shade rectangle with rounded corners containing activities, interactions, states, etc.	An organising unit for bounding activities (and states) that are strongly related, e.g. usually performed by the same agent or agency.	Swim lanes containing activities, states, transitions, etc.	An organising unit for bounding activities (and states) that are strongly related, e.g. usually performed by the same agent or agency.	
Process activation	A small, transparent box with lines form corner to corner	Denotes that a new role will be instantiated.	Not explicitly supported.	Not applicable.	
State test	A question testing the state, and upside down triangles with possible answers on the line below.	Depending on the answer to the question only one out of more than one thread will be followed.	1.An arrow into a diamond with up to three arrows coming out, each labelled with a possible state. 2. Multiple arrows from either state or an activity where each arrow is labelled.	Depending on which state is true only one out of more than one thread will be followed. Depending on which label is valid only one thread will be followed.	

Table 1: Cont'd, A Comparative Analysis of RAD and UML Activity Diagrams

Process concept	RAD notation	RAD semantics	AD notation	AD semantics	Notes
Concurrent threads	A series of triangles on a "state line"; each starts a new "sub state line".	Denotes a series of parallel threads of activities. Activities may be interleaved or performed simultaneously.	Two or more arrows issuing from a synchronisation bar.	Denotes a series of parallel threads of activities and states. Activities may be interleaved or performed simultaneously.	
Replication of threads	A triangle and asterisk on a line with a name.	The activities below the triangle will be performed in parallel a number of times.	An asterisk beside an arrow labelled with a repetition statement.	The activities below will performed in parallel a number of times.	
Synchronise threads	Lines from multiple "threads" merge into one line.	The threads represented by the lines are synchronised.	Multiple arrows arrive at a synchronisation bar.	The threads represented by the arrows are synchronised.	
Data flows	Named grams	Information flowing between activities.	Not explicitly supported.	Not applicable.	In ADs data flows may be indicated in activity, event, or condition names.
Interaction between roles	1.A line between two or more boxes in different roles. The line is labelled with an interaction description. 2. A line between two or more boxes one with hatched lines. The line is labelled with an interaction description.	Denotes synchronous interaction between activities in two different roles. Such interaction cannot occur until the activities in both roles are ready. 2. Similar but the hatched lines box denotes the interaction initiator.	Not explicitly supported.	Not applicable.	Interaction can be modelled in ADs using a synchronisation bar. E.g. an activity in one swim lane says "Send x", and activity in another lane says "Receive x" and arrows from both activities end in the same synchronisation bar labelled, x sent and x received.
Pre-existing role	A "role" with a tick on top.	Indicates that an instantiation of this role always exists.	Not explicitly supported.	Not applicable.	
Replicated pre-existing role	A "role" with a number, n, and a tick on top.	Indicates that n instantiations of the role always exist.	Not explicitly supported	Not applicable.	

5. RAD and UML Activity Diagrams: A Comparative Analysis

The previous two sections of this paper have documented the results of modelling the Selection, Enrolment, and Registration process using Role Activity Diagram (RAD) notation and UML's Activity Diagram notation. This section describes the extent to which important process concepts like activity, state, transition, concurrency, etc. are supported by each of the two notations, RADs and UML ADs.

This is done in order both to see how well process concepts are supported in general by two prominent process modelling languages, and to compare the process modelling expressiveness of each notation.

Table 1 lists process concepts in column one, e.g. Activity, State, Transitions, etc. Columns two and three first describe the RAD syntax for each process concept and then give the associated RAD semantics. For example, for the process concept "state", the RAD syntax is "A line connecting activities"; while the associated RAD semantics is "Denotes a state or substate of a role". Columns four and five similarly provide the syntax and semantics of AD notation that supports process concepts. The final column, "Notes", either provides additional relevant information, or indicates how a process concept may be supported using the available notation (syntax and semantics) in the absence of dedicated notation. For example, the RAD notation does not explicitly support the "Start state" and "Stop state". However, these can be supported using the RAD notation for supporting "Goals", i.e. by using the magnifying glass symbol and its associated semantics.

From the table it can be seen that both RADs and ADs support the most important process concepts. They either do so with dedicated symbols for activity, state, transitions, concurrent threads, etc., or are capable of doing so using available notation. For example, the start and stop process concepts may be indicated in RADs with the RAD goal symbol (magnifying glass); interaction between roles in an AD may be supported using the synchronisation bar notation. But, the modelling of concurrent execution of activities/tasks in UML ADs is explicit - using the fork and join synchronization bars - which is not the case in the RAD notation.

In addition, here are one or two process concepts that are explicitly supported in RADs but not by ADs. For example, RADs include notation to support the notion of creating a new role instance during a process; this concept is not supported in ADs. RADs also explicitly support the indication of a specific number of pre-existing role instances; this is not supported by ADs.

Although for RADs and ADs the power to express important process concepts is similar, it is worth noting one significant difference. A RAD comprises activities and states such that every activity is preceded by one state and results in one state. However, an AD may take one of three basic forms. It may feature just activities, in which case arrows show the movement from one activity to another; it may feature just states, in which case arrows show transitions from one state to another; or it may feature both activities and states, in which case arrows show states resulting from activities and states may lead to activities. On the ones hand RADs may be viewed as being conceptually simpler than ADs and as a consequence may be viewed as being easier to employ. On the other hand, ADs can be used to further translate into or complement other UML analysis and design models to visualize the specification of a software system which implements the process being modelled using ADs.

6. Conclusion

As well as providing a comparison of the notations used for RADs and UML ADs, we have shown it is possible to make a translation from RAD to AD in the test case. We would say that the RAD and the AD for the research degrees (sub-) process are substantially equivalent, though it would be fair to say we are relying to some extent on our ability to interpret the diagrams and our knowledge of the case process. Both diagrams serve to depict quite precisely a flow of activities within a closely defined process. A developer could look forward from either diagram towards some of the interactions and outputs which will need to be built into a system which could support this process.

We were able to represent in the AD, more or less satisfactorily, everything we had in the RAD. Although the translation is straightforward enough in the specific case, we are not in a position to state a general translation process from RAD to AD. While we were able to map RAD notions of interaction and iteration into the UML AD, questions remain to be resolved around the treatment of the direction, timing, and synchronization of interactions in the two techniques. More fundamentally, the mapping of RAD *role* into AD *swimlane*, while natural in this case, cannot be generalized until the basic semantics of the two techniques have been explored further, particularly with regard to the relationships between role, actor, and activity.

In sum, we would suggest that translation from RAD to UML AD is likely to be feasible in particular cases, but will rely on the ability of the translators to establish and maintain the equivalence between the two (ie, the equivalence will be partly a matter of local interpretation). A basic mapping across at

the notational level is given in the paper. Further work on the conceptual frameworks and semantics underlying the two techniques must be done before a general equivalence can be asserted.

7. References

Beeson I, Green S, Sa J and Sully A, 'Linking Business Processes and Information Systems Provision in a Dynamic Environment', Information Systems Frontiers 4-3, 2002, pp 319-331.

CEMS research degrees administrative procedures, 2002: http://www.csm.uwe.ac.uk/~jsa/GraduateSchool/graduate_school.htm#cems_procedures

Holt A W, Ramsey H R and Grimes J D, 'Coordination System Technology as the basis for a programming environment', Electrical Communication 57-4, 1983, pp 308-314.

Fowler M with Scott K, *UML Distilled*, Addison-Wesley/Pearson Education, Upper Saddle River NJ, 2nd ed., 2000.

Ould M A, *Business Processes - modelling and analysis for re-engineering and improvement*, John Wiley & Sons, Chichester, 1995.

Scott K, *UML Explained*, Addison-Wesley/Pearson Education, Upper Saddle River NJ, 2001.

Stevens P with Pooley R, *Using UML*, Addison-Wesley/Pearson Education, Harlow, updated ed., 2000.

Acknowledgements. The authors would like to acknowledge the contributions of other members of the Systems Modelling Research Group at UWE, namely Richard Kamm and Tony Solomonides, to the discussions of the material and ideas in this paper.

Appendix A: Summary of RAD Notation