Open access • Journal Article • DOI:10.1016/S0950-5849(03)00133-2

Bridging the gap between business models and system models. — Source link

Mohammed Odeh, Richard Kamm

Institutions: University of the West of England, University of Bath

Published on: 01 Dec 2003 - Information & Software Technology (Elsevier)

Topics: Applications of UML, Use Case Points, Business process modeling, Use case and Software requirements specification

Bridging the Gap between Business Models and System Models

Mohammed Odeh* and Richard Kamm§
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

§ School of Management
University of Bath
Claverton Down
BATH
BA2 7AY
tel: +44 (0)1225 386909
fax: +44 (0)1225 386473
email: R.M.Kamm@bath.ac.uk

Abstract

This paper discusses links that may be made between process models and UML software specification techniques, working from an argument that the whole complexity of organisational activity cannot be captured by UML alone. The approach taken is to develop a set of use cases which would be capable of providing information support to a pre-defined organisational process. The nature of the thinking which is necessary to derive the use cases is outlined, using the pre-defined process as a case study. The grouping of transactions and state changes into Use Cases is shown to require design choices which may vary between particular organisational contexts. Conclusions are drawn about the direction of further investigation of links between process modelling and UML.

Keywords: Process Modelling; UML; Use Cases; Business Modelling.

1. Introduction

Is it feasible to make a productive and intelligible bridge between a model of a business and a model of an IT system meant in some way to mirror, support, or perhaps automate the business? The challenge is at least in part to find methods for modelling businesses on the one hand and IT systems on the other, between which a regular form of translation or conversion is possible. In recent years, the debate round these issues has received a significant stimulus from the achievements and claims of proponents of the Unified Modelling Language (UML), and of a unified software development process - particularly in the shape of the Rational Unified Process (RUP) [5, 8, 11, 12].

An earlier study by members of this research group used Role Activity Diagramming to investigate the processes linking strategic decision making with information systems provision in an insurance company [1]. The argument in that paper was 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 any straightforward realization, or downward translation, of an overarching business model, strategy or architecture. The impression gained from that study was that business models - which may only be partly articulated and partially unified in a particular company - cannot be routinely translated into system models, but can be expressed to good effect in

negotiations about projects, priorities, and resource allocation. This leads to the further supposition that, although business modelling tools can be useful representational and analytical devices, and although it may well be possible to achieve some degree of translation between a model of part of the business and a model of a related IT system, the activities of representing, analyzing and translating are themselves conducted against an organizational background characterized by shifting interests, interpretations, and power relations. We look therefore, for a way of understanding modelling, and the contribution that modelling can make to the effective design of systems, within a wider framework of organizational meaning and decision making.

Given the wide range of interests and expertise in any sizeable or complex company, and given that IT, however important it is for an organization's success, is not the only thing that matters in an organization, it seems important to achieve an *autonomy* between business modelling and system modelling activities. The proponents of UML and RUP, however, do not recommend such autonomy, but assert rather that the concepts and frameworks underlying the software development process will serve adequately for representing business processes, and recommend therefore that those concepts and frameworks be extended to cover the business as a whole. Kruchten and Ericsson, for example, in their chapter on the business modelling workflow in RUP, assert categorically that 'software engineering techniques can be translated and used for business modelling' [11, p.154]. In a more extended treatment, Eriksson and Penker, in their proposals for 'business extensions' to UML, describe a set of 13 diagrams and models for business modelling, but aim expressly to represent as many as these as possible using one of the 9 standard UML diagrams, and manage to achieve that for all but two of their own set [5]. One can see the logic in their taking a parsimonious approach here, to avoid a proliferation of diagram types, but must question the wholesale importation into business modelling of the ontology and epistemology which underlie UML. One senses a missionary quality in the UML/OMG movement, which ought to be tempered by independent consideration of good methods for business modelling - given that a business is not a software process.

In the remainder of the present paper we focus on, and give an example of, translation from a business model to a system model. From the above discussion, we adopt a position which expects that such translation is possible and worthwhile, while recognising at the same time that it takes place in an organizational context from which it must derive its significance and validation. Also arising from the above discussion, we look for autonomy between the business modelling and system modelling sides of the translation process. We choose UML on the system side, but Ould's RAD method on the business side - rather than UML again - on the grounds that we see RAD as being an authentic attempt to model the business *per se* rather than to superimpose a software engineering framework on to the business. This paper extends previous work we have done on this topic [13], and relates it to the argument of the earlier paper discussed above.

2. Process Modelling and UML: distinctions and points of contact between Use Cases and RADs

It is widely accepted that UML is a collection of techniques, with no in-built rules laying out an order in which they are to be used. [6, 8] However, the links between

Use Cases and process models are particularly worth exploring because of the importance commonly given to the former in the UML literature as a scoping device, the process of developing them being important in establishing the overall functionality of an application. The idea of use-case driven modelling is one which has been applied to give some pattern or regularity to object-oriented development [9]. More significantly for the purposes of this paper, the efforts to link UML directly into process transformation have hinged on that particular modelling technique. Jacobson et. al. have developed a form of "business engineering" in which the starting point is the modelling of a business through the identification of processes and their mapping directly on to use cases: one process per use case [10].

In principle, this is appropriate to a process view of organisations in that it concentrates on the work that is done, the production of value for a customer, rather than the structures and hierarchies within which tasks are divided. The processes represent activity and the actors will represent the beneficiaries or customers. However, this representation of processes will omit elements of a process which have been identified as essential by Ould and others [15], notably the interactions between the participants in the activities that comprise the process. More generally, the idea of a process as a flow of activities performed upon a unit of work does not translate well to a use case model in which the use cases are distinct from each other and the main linkage is the interaction with the external actor.

The reason for this is Jacobson scales up a modelling approach that is conventionally used to depict the usage of an information system that may support a process, with the user being depicted as the actor who is directly interacts with the interface of the application. The difference between this and a whole process is that the customer will often interact with one or more human participants in the process, so that it could not continue without communications between those participants. Jacobson appears to recognise this by not including participants as actors on his business use cases. He suggests instead that they play the same role as interface objects within an information system [10], but this removes the sense of their being human operatives who will may take unstructured decisions or communicate with each other in informal as well as formal ways: a concept for a programmable aspect of an organisational system which may never need to be programmed. Just as evident is that, even if the same technique is used for documenting business processes and the supporting information system, the problem of extracting requirements for the latter from the former remains difficult. [3]

Use cases, then, are more suitable for modelling the information systems that will be used to support human activity within processes than they are for modelling entire processes in themselves. The essence of the communications and tasks within a process can be developed through a form of process modelling which does take into account those aspects of communication and collaboration between humans which use cases in themselves do not address. We take here the Role Activity Diagram, developed by Ould, as a representative example of such a technique. This would then leave open the possibility of employing UML methods to develop the software support which will enable the process to be improved, but would require some point of contact between the process modelling and the software definition to be identified. The approach adopted here is to partition the process up into areas for which the

required support can be easily identified and deriving use cases for each of these, and the main outcome of this paper is a commentary on that activity.

However, there are difficulties in making a direct transfer from RAD to Use Cases. Process models of the RAD type concentrate on a dynamic depiction of activities, decisions and their outputs. They illustrate an organisational process as a dynamic set of activities involving interactions and decisions. Use Cases, while also regarded as illustrating the dynamic or "behavioural" [1] aspects of an information system, compared to the relatively static class diagram, are concerned with functionality. This makes sense at the level of a system but, as noted above, will not cover all types of organisational activities, with their interactions between interested parties. The different form of thinking is illustrated in the problems that students often have to overcome when learning the principles of use cases: it takes time to appreciate that, although there are ovals with arrows entering them and leaving them, this does not depict a flow of data and that the connections between actors and use cases are of a different nature to those linking, say, external entities and processes on data flow diagrams.

The tendency to regard use cases as the initial driving force of many object-oriented design projects leads to their being used as the initial focus of the effort needed to derive the classes and objects that will form the basis of the new application. What follows, then, is an illustration of the thinking that needs to be gone through in connecting a RAD and a set of Use Cases, accompanied by some guidelines which have resulted.

3. 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. It has been used before within the Systems Modelling Group's work on the links between process and software specification models [13] and so constitutes a well-understood set of activities covering the selection and initial enrolment of research students through to the formal authorisation by the Research Degrees Board of particular research projects by those same research students.

For the purpose scope of this paper, its utility is that it contains the type of unstructured decision and the communication between participants in the process that require the type of modelling technique which is derived from analysis of organisations rather than of software. To determine whether or not an applicant is a suitable research student, for example, depends on a variety of factors, the candidate's ability and motivation, the ability of staff to supervise research in the proposed subject area not just immediately but also over a period of years. Similarly a review process, although it takes place within a formal timetable, will require qualitative judgements which will not be reducible to predefined procedures.

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

4.1 Role Activity Diagrams and Use Cases

We have chosen to use Ould's **R**ole **A**ctivity **D**iagrams (RADs) [14, 15] to model our process initially. The basic concepts of RAD were first introduced by Holt et al [7], and later enriched by Ould [14]. 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.

A RAD comprises a set of activities, decisions and transactions. The last of these are essentially interactions between two "roles", i.e. participants in the process. This gives us a starting point for considering the derivation of use cases from the RAD. It makes sense to consider the process modelled in the RAD as acquiring information through the transactions in order to make the decisions which lead to changes of state. The use of information would be illustrated as a set of use cases, each one supporting a different aspect of the process.

A use case is itself defined as a provider of output of measurable value to an actor. This gives us a point of contact with the philosophy that underpins process modelling, which is to consider a process as something which produces an output of benefit to some internal or external customer [4]. The use cases, then, support the provision of information to actors in such a way that the overall benefits of the process are realised, which implies the following with respect to the elements of a process defined in a RAD:

i) each use case should embody at least one transaction – otherwise there is no beneficiary of the activities that the information supports.
ii) each use case should support at least one activity leading to a change of state – otherwise the beneficiary would not receive any information.

This still leaves the question of what is involved in identifying the relevant transactions and the activities leading to state change and how they are to be grouped together in order to produce a meaningful set of use cases. In practice, the grouping tends to follow the path of establishing how particular customers within the overall process are to be supported. For each use case, it ought to be possible, even at an early stage, to envisage an implementation class which will form the basis for the interface with which the actor interacts: a form or other type of screen layout.

What follows is an explanation of how the use cases were derived for the case study process.

4.2 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 offers a place to the candidate student through to the student's formal registration on a PhD programme.

There are four roles in this process: *Director of Studies* (DoS), *Student*, *Research Admin*istrator (RA) and the *Research Degree Board* (RDB). The early stages of the

process cover interview and the making of offers to candidate students: successful candidates complete and RD0 form that leads to enrolment. 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. This form is created and considered by periodic meetings of the RDB. Different activities then follow the RDB's decision, either continuation to registration if the project is approved or revision of the RD1 if not. When the outcome is "approved", the registration process is completed.

Figure 1 to go here.

5. The derivation of use cases from transactions and states

The use cases are developed from transactions and states derived from the RAD. A systematic approach to this is as follows:
i) Start reading the RAD horizontally (flows of the RAD) visiting roles with transactions stated as imperative verbs (e.g. interview, enrol, send, etc.).
ii) Record a transaction identifying its nature and initiator, for example *Transaction 0:* **Interview the Student** by the DoS in the RAD Process Model in figure 1.
iii) Observe the effect of executing this transaction on a state change in the same column of the role being visited or other roles in the same flow. For example, *State 0:* Student RD0 Form Completed.
iv) Continue for the next transaction.

After this analysis of a RAD process model, we expect to have two sets:
i) Set of Transactions: $T = \{T_0, T_1, .. , T_i, \dots , T_m\}$, $0 \leq i \leq m$
ii) Set of States: $S = \{S_0, S_1, \dots, Sj, \dots, Sn\}$, $0 \leq j \leq n$
In the case study example, the outcomes are 18 Transactions and 16 States (see Appendix 1 for the full derivation of transactions and states in the case study): for example the transaction "Enrol Student" can be translated directly from the RAD and it leads to a change of state for "Student", i.e. the student becomes "enrolled".

Grouping the transactions into subsets which will result in some measurable value for a specific actor will allow us to represent use-cases. In the case study process, a first analysis leads to the definition of two core use-case associated with {T0 .. T5} and {T6 .. T18}. These may be thought of as:

i) UC1: Manage Research Development Stage 0 (handle application and enrolment)

ii) UC2: Manage Research Development Stage 1 (manage student record during registration)

Between them, these two parts of the model cover the different stages in the "life" of the customer of the overall process: the student. In the first use case, the student moves from being an applicant to the university to embarking on a course of research study. The second use case covers the student's transition from fledgling to fully-accepted researcher within the faculty. In other words, each core use case covers a major state changes relating to the "customer" of the process as a whole. From here, it is possible to further partition the activities within the process with a view to supporting the information use and decision-making among the internal participants.

A good question to ask here is: which actors are associated with UC1 and UC2? Clearly, the role RDB has no association with UC1, and hence our thoughts converge on DoS and RA as the potential beneficiaries in the process model.

There are two options:

(1) associate a generic role (for example RD0 Operative) to UC1.

or

(2) partition UC1 into three use-cases:

(a) Actor: DoS, UC1.1: Perform_DoS_RD0_Activities: to include transactions covering application and interview. (Transactions T0-T1)
(b) Actor: RA, UC1.2: Perform_RA_RD0_Activities: to include transactions covering the creation of a student record and the student's initial enrolment. (Transactions T2, T4, T5)
(c) Actor: Student, UC1.3: Perform_Student_RD0_Activities: acceptance of an offer. (Transaction T3)

The first approach is probably appropriate if one is trying to develop applications which can be applied in a variety of similar situations in which the details of implementation and allocation of roles will vary. In some universities, for example, the data entry at enrolment is undertaken by the students themselves, while in others an administrator performs this activity. It would then be possible to retain features of the original situation through identifying roles such as "Student" or "Research Administrator" as specialisations of the more general actor, RD0 Operative. Here, a generic role for the actor will be suitable because of the variety of possible implementation details and will meet the principles of reusability that underpin object-oriented methods such as UML. The second, however, may be more helpful when a specific situation in a particular organisation is being analysed: here, reusability may not be as much of an issue because there is no intention to broaden the application to other places. An illustration of the use case model that might result is illustrated, using standard UML conventions [2] in Figure 2:

Figure 2 to go here

The same issue is raised in a more obvious form with the second use case, because some activities are undertaken in parallel, through collaboration, even, by both the Student and the Director of Studies. In order to maintain the conventions of UML, a single generic actor should be conceived, with the roles of Student and the Director of Studies being specialisations of it. Thus, the use case covering the management of the student record during registration, can be broken down into more specific use cases as follows:

(a) Actor: RD1 completer, UC2.1 Collect RD1 data: the initial process of requesting the proposal details be collected. (Transactions T6-T8)

(b) Actor: RDB, UC2.2 Support RDB decision: provision of information for the degree board meeting to take place and have its results recorded (transactions T9-T11).

(c) Actor: RA, UC2.3 Administer RDB decision: actions that result from the decision-making by the RDB (transactions T12-T14).

(d) Actor: RD1 completer, UC2.4 Request revision of RD1: an extension of UC2.3 in the event of reworking of the research proposal being required. (Transactions T15-T17)

The model would be illustrated as in Figure 3:

Figure 3 to go here.

What is noticeable is that this distinction illustrates the human choices that have to be made is developing a use case, even where the derivation of transactions and state changes has been begun in a relatively mechanical way.

6. Conclusion

The paper has illustrated the derivation of use cases from a process model, working within the conventions of both techniques while remaining within the spirit of the process approach to organisational analysis, the focus on defined outputs for the benefit of customers. Moving forward would mean deriving classes, using the preceding analysis as a starting point. Some progress on this can be seen at this stage, the state change indicators giving an idea of attributes for objects which will be present within the application. However, more investigation will be required in order to determine whether the foundation of the analysis in process modelling affects the development of further UML models. It is possible that further iteration between the specifications of functionality and the existing RAD would produce a suitably rich and well-informed basis for the development of software support.

But this paper has also highlighted difficulties in the derivation of use cases from process models. The notion of an actor is not as clear as might be thought at first sight. There is no simple mapping of Roles in process models on to Actors in use case diagrams, since whether a role plays an active part in the final application is a relatively specific implementation decision. This arises from the more general point that RADs and use cases work on different principles as modelling techniques and require different forms of thinking to make sense of them.

The application of an object-oriented approach to software development, then, is appropriate to the application of a process approach to organisational analysis, but one-to-one matching of elements between the two sets of conventions is not feasible. What is involved is more of an activity of interpretation of the requirements of support for the process, expressed in terms which are appropriate to UML modelling techniques. This interpretation requires design choices on the part of the developers which will be related to not only the expressed needs of the potential user community but also derived from an awareness of the wider organisational context in which the process takes place and how it may be developed (even re-engineered) in the future.

Acknowledgements. The authors would like to acknowledge the contributions of other members of the Systems Modelling Research Group at UWE, namely Ian Beeson, Stewart Green and Jin Sa, to the discussions of the material and ideas in this paper. They would also like to thank the anonymous reviewers for very helpful comments.

7. References

[1] I. Beeson, S. Green, J. Sa, A Sully, Linking Business Processes and Information Systems Provision in a Dynamic Environment, Information Systems Frontiers 4-3 (2002) 319-331.
[2] G. Booch, J. Rumbaugh, I. Jacobson, The Unified Modeling Language User Guide, Addison-Wesley Longman, Reading, MA, 1999.
[3] A. Cockburn, Writing Effective Use Cases, Addison-Wesley/Pearson Education, Upper Saddle River NJ, 2001.
[4] T.H. Davenport, Process Innovation, Harvard Business School Press. Boston MA, 1993.
[5] H.E. Eriksson, M. Penker, Business Modelling with UML, John Wiley & Sons, Inc., New York, 2000.
[6] M. Fowler with K. Scott, UML Distilled, 2nd ed., Addison-Wesley/Pearson Education, Upper Saddle River NJ, 2000.
[7] A.W. Holt, H.R. Ramsey, J.D. Grimes, Coordination System Technology as the basis for a programming environment, Electrical Communication 57-4 (1983), 308-314.
[8] I. Jacobson, The Road to the Unified Software Development Process, Cambridge University Press, Cambridge, 2000. Revised and updated by Stefan Bylund.
[9]I. Jacobson, M. Christerson, P. Jonsson, G. Overgaard, Object-Oriented Software Engineering: a Use Case Driven Approach, Addison-Wesley / ACM Press, Reading MA, 1992.
[10] I. Jacobson, M. Ericsson and A. Jacobson, The Object Advantage: Business Process Reeingineering with Object Technology, Addison-Wesley / ACM Press, Reading MA, 1994.
[11] P. Kruchten, The Rational Unified Process: an Introduction, 2nd ed. Addison Wesley Longman, Reading, MA, 2000.
[12] C. Marshall, Enterprise Modelling with UML, Addison Wesley Longman, Reading, MA, 2000.
[13] M. Odeh, I. Beeson, S. Green, J. Sa, Process Modelling Using RADs and UML Activity Diagrams: An Exploratory Study, The 3rd International Arab Conference on Information Technology, ACIT2002, December 16-19, 23, 2002, Doha, Qatar.
[14] M.A. Ould, Business Processes - modelling and analysis for re-engineering and improvement, John Wiley & Sons, Chichester, 1995.
[15] M.A. Ould and T. Huckvale, Process Modelling: What, Why and How, K. Spurr, P. Layzell, L. Jennison and N. Richards (eds), Software Assistance for Business Re-Engineering, John Wiley, Chichester, 1993, pp. 81-98.

Appendix 1: Transformation of Process into Tasks and States

DoS	RA	RDB	Student
Transaction 0: Interview Student			State0: Student Interviewed
Transaction 1: Send RD0 to RA	State 1: RD0 Received		
	Transaction 2: Send Offer Letter to Student		State 2: Offer Letter Received
	State 3: Acceptance Letter Received		**Transaction 3: Send Acceptance Letter**
	Transaction 4: Create Student Record		State 4: Student Record Created
	Transaction 5: Enrol Student		
	State 5: Student Enrolled		
Transaction 6: DoS to Complete Student RD1 Form			**Transaction 7: Student to Complete RD1 Form**
State 6: Student RD1 Form Completed			State 6: Student RD1 Form Completed
Transaction7: Send Completed Student RD1 Form	State 7: Completed Student RD1 From Received		
	Transaction 8: Send RD1 to RDB	State 8: Completed Student RD1 Form Received	
		Transaction 9: Evaluate Completed Student RD1 Forms	
		Transaction 10: Completed Student RD1 Forms Evaluated	
	State 11: Outcomes of Evaluation of Completed Student RD1 Forms Received	**Transaction 11: Send Outcomes of Evaluation of Completed Student RD1 Forms**	
	Transaction 12: Record Outcomes of Evaluation of Completed Student RD1 Forms		
	State 12: Outcomes of Completed RD1 Form Recorded		
State 13: RD1 RDB Outcomes Received by DoS Approved, **HALT**	**Transaction 13: Send RDB Outcomes to Student** **Transaction 14: Send RDB Outcomes to DoS**		State 14: RD1 RDB Outcomes Received by Student Approved, **HALT**

Transaction 15: Revise RD1 Form with Student			**Transaction 16: Revise RD1 Form with DoS**
State 15: Student RD1 Form Revised			State 16: Student RD1 Form Revised
Transaction 17: Send Revised RD1 Form to RA			
	Continue from above: (RA) **Transaction 18: Send Student RD1 Form to RDB**		

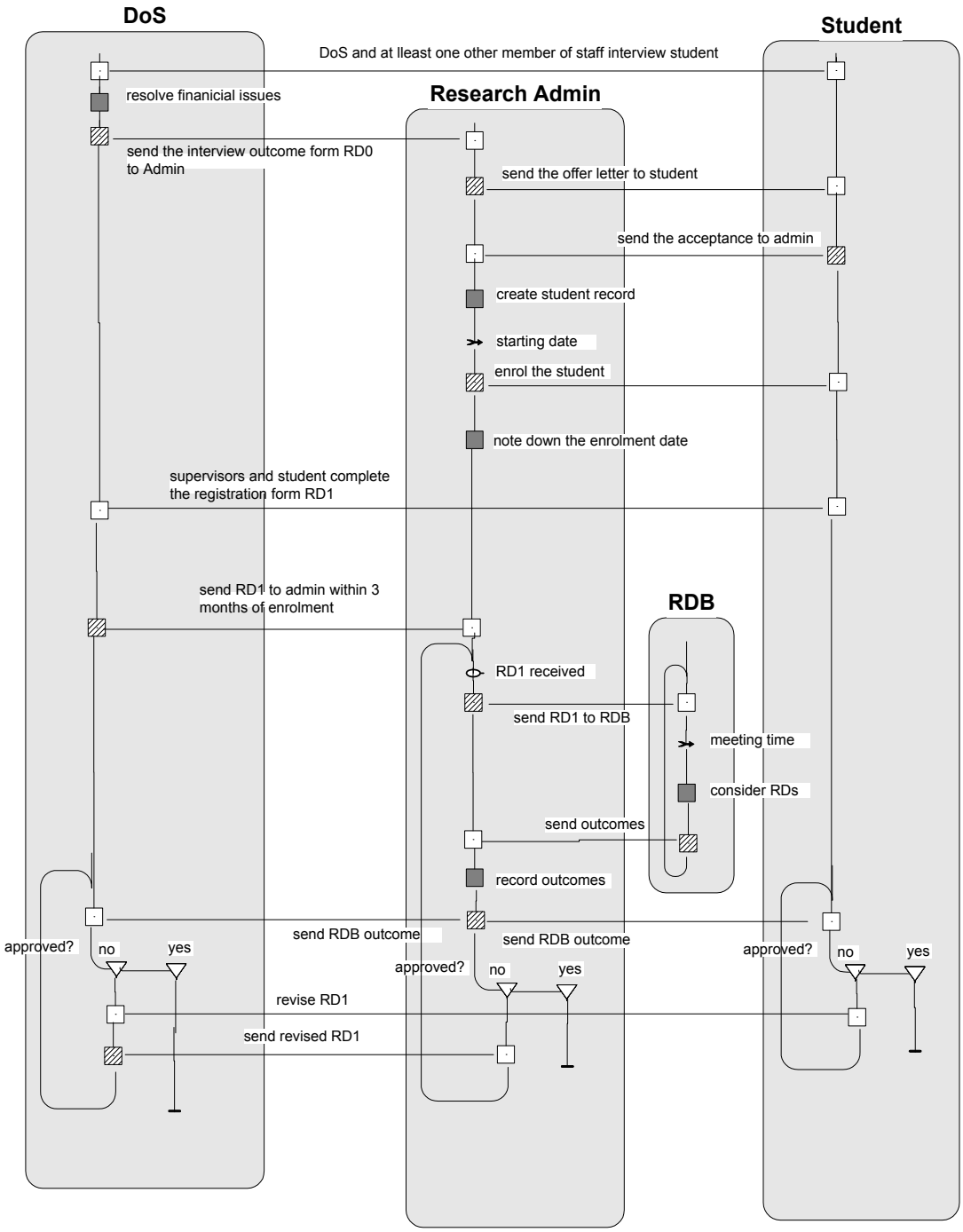

Figure 1: RAD for the Admission, Enrolment and Registration process.

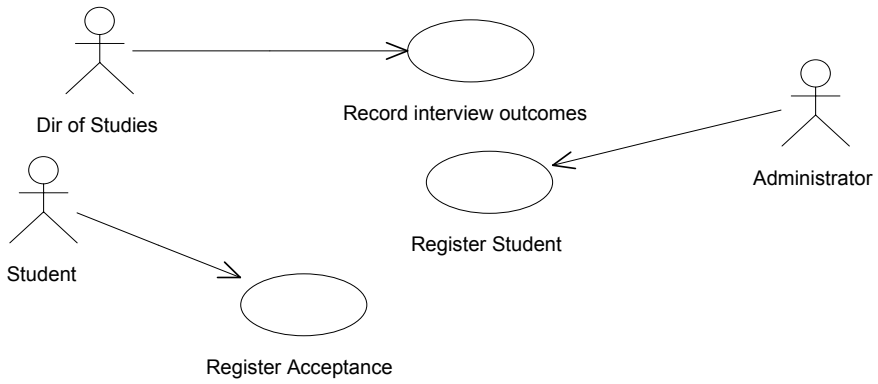

Figure 2 Use Case View of the Manage Research Development Stage 0, T0 to T5

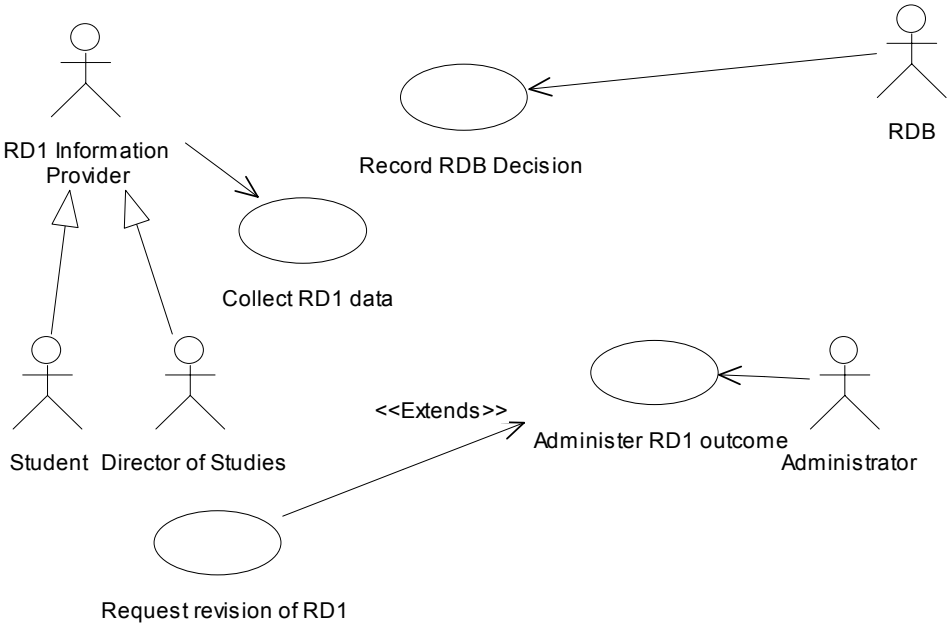

Figure 3 Use Case View of the Manage Research Development Stage 1, T6 to T17