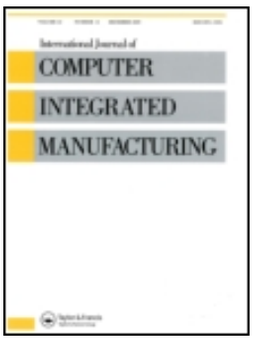

International Journal of Computer Integrated Manufacturing

ISSN: 0951-192X (Print) 1362-3052 (Online) Journal homepage: https://www.tandfonline.com/loi/tcim20

A case study: using UML to develop a knowledge-based system for supporting business systems in a small financial institute

W. W. C. Chung & J. J. F. Pak

To cite this article: W. W. C. Chung & J. J. F. Pak (2006) A case study: using UML to develop a knowledge-based system for supporting business systems in a small financial institute, International Journal of Computer Integrated Manufacturing, 19:1, 59-68, DOI: 10.1080/09511920500174380

To link to this article: https://doi.org/10.1080/09511920500174380

Published online: 19 Feb 2007.

International Journal of Computer Integrated Manufacturing, Vol. 19, No. 1, January-February 2006, 59–68

A case study: using UML to develop a knowledge-based system for supporting business systems in a small financial institute

W. W. C. CHUNG* and J. J. F. PAK

Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hong Kong

Financial industry is a core industry in Hong Kong. The business systems in financial institutes are widely applied throughout their front, mid, and back office. The system specialist who supports these systems must have finance and information-systems knowledge. This multi-disciplinary requirement leads to difficulties in finding capable candidates. It is reasonably assumed that a knowledge-based system (KBS) with the intelligence of a system specialist can assist a helpdesk to support the business systems. However, KBS development involves many known issues. In particular, the uncertainties of collected knowledge should be discussed and remodelled iteratively until the final knowledge set is close to the actual knowledge. Although the literature states that it is theoretically feasible to use Unified Modelling Language (UML) on knowledge acquisition, case studies on this area to date have been insufficient, hence this research experiment UML on developing a KBS for supporting business systems in a financial institute. The method of using UML diagrams is documented throughout the study. The results show that the UML was useful in different stages of KBS development. The case study articulates how the KBS helps to solve business system problems in the company. In summary, this paper can shed light on future research in using UML for KBS development.

Keywords: Business process management; Case-study research; Knowledge-based system; Unified modelling language

1. Introduction

1.1. Background of a small financial institute in Hong Kong

1.1.1. Definition of financial business. Businesses can be categorized into non-financial and financial business. They perform the following functions: (1) transformation from one type of financial asset into another preferable type of asset via markets; (2) exchange of financial assets on behalf of customers; (3) creation of financial assets for the customer and selling these to other market players; (d) provision of investment advice and management of the customers' portfolio (Eng *et al*. 1998).

In Hong Kong, there are more than 200 local banks and foreign banks, and more than 100 deposit-taking companies. The stock market there is one of the most active centres in Asia. This lures many foreign investment bank and fund houses to participate in the Hong Kong financial market and, thus, generate income for the region.

Inside a financial institute, the business operations can be classified into three main kinds: front-office operations, mid-office operations and back-office operations. This classification is based on the workflow sequence of business operations. Generally speaking, front office deals with customers, brokers, and other financial companies. Mid-office manages market risk, credit risk, business workflow, and financial reports. Back office is in charge of settlement, reconciliation, and position-keeping of cash, cheque, payment, securities and derivative.

*Corresponding author. Email: mfwalter@inet.polyu.edu.hk

International Journal of Computer Integrated Manufacturing
ISSN 0951-192X print/ISSN 1362-3052 online © 2006 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/09511920500174380

1.1.2. Importance of business systems in financial industries. Business systems are computer software designed for automating a particular business process. Many financial institutes rely on different kinds of business system to process their daily work. This can enhance the productivity and efficiency of the operations. Thus, business systems play an important role in financial institutes. However, in case of any malfunction, considerable problems can arise.

1.2. Difficulties in seeking system specialists in financial industries

1.2.1. Common problems of business systems. There are different kinds of business system malfunction. Common examples are: real-time market data feed stalled, system interfaces problem, new features/report requirement, transaction rejection, database corruption, systems reconciliation discrepancy, incorrect position or P/L, transaction duplication, insufficient training, batch job problem, client software corruption, operator fault, internal system data feed error, and slow response from network/server.

1.2.2. Value of a system specialist. A system specialist is a role dedicated to solving the above problems. Obviously, the problem domain involves both information technology and finance. Usually, the solution may need close collaboration with dealers, financial planners, settlement managers, software vendors, computer operators, project managers, and so forth. Thus, capable candidates should have multi-disciplinary experience in financial industries and in-depth IT knowledge so that they can make instant decisions to diagnose and solve the problems.

1.3. Using KBS to support business systems

On-the-job training for a helpdesk operator could be a feasible approach by which to nurture a new system specialist, but it takes time to develop and depends on the calibre of the trainee. Sometimes, a well-trained employee may leave the company after several years of services, and the quality of support may be difficult to maintain.

By definition, a knowledge-based system (KBS) can enable an agent to reason about the world and determine appropriate action (Lam 2003). It can be valuable to develop a KBS with the knowledge of system specialists, which can assist a helpdesk operator to learn and solve the business system problems.

1.4. Challenges of developing a knowledge-based system (KBS)

1.4.1. Difficulties in knowledge-based system development. Structured KBS development methodologies were relatively immature in comparison with other traditional branches of engineering. Examples are GEMINI, POLITE, and KADS. Many knowledge engineers still preferred to use a prototyping approach (Ross 1992).

Until now, the process of knowledge acquisition remains a great barrier in knowledge engineering. One of the key issues is to solicit the expert to articulate their knowledge clearly, which may be tacit or intricate (Afshar 2002). Besides, knowledge workers need to represent the captured knowledge visibly. The representation should also be discussed and remodelled iteratively until the final knowledge set is close to the expert's mental model. Although Håkansson (2001) suggested using Unified Modelling Language (UML) to represent the expert's domain knowledge, case studies in this area are rare. Thus, the case study in this paper could bridge the research gap between theory and practice.

1.4.2. Adapt unified modelling language (UML) to develop KBS. The objective of this paper is to articulate a way for using UML diagrams to develop a knowledge-based business system for a small financial institute. The guru of case studies research, Robert Yin (Yin 1989), stated that single-case design is justifiable provided the case is a rare event. As UML is rarely adopted in developing KBS, a single-case study in this area should be an interesting case to report.

2. Literature review

2.1. Object-oriented development and UML

UML is the blueprint of a software system. It is a standard language to specify, visualize, and model the artefacts of a software system. It provides the ability to capture the characteristics of a system by using notations.

The origin of UML is archaeological and comprehensive. Three software-engineering gurus contributed significantly: Grady Booch, Ivan Jacobson, and Jim Rumbaugh. Booch was strong in system design, but he neglected analysis (Booch 1994, Booch *et al.* 1999). Rumbaugh was strong in system analysis, but he put little emphasis on design stage. Jacobson *et al.* (1999) emphasized user experience. Their early works together established the base for UML in 1990s.

The underlying premise of UML is that no one diagram can capture the different elements of a system in its entirety. Hence, UML is made up of eight diagrams that can be used to model a system at different points of time in the software life cycle of a system. The eight UML diagrams are:

- Use case diagram: a collection of system functions, showing which actors and processes interact with each use case.

- Class diagram: shows the classes in a system and variety of relationships (association, aggregation, and inheritance) among classes.
- State diagram: shows the event-ordered behaviour of an object.
- Activity diagram: emphasizes the flow of control among objects and models the functions of a system.
- Sequence diagram: displays object interactions and messages in time-sequence order.
- Collaboration diagram: displays object interactions in event-sequence order under a relatively complicated circumstance.
- Component diagram: shows physical components, interfaces, and usage dependency between components.
- Deployment diagram: shows how physical components are distributed across the physical environment in which the system runs.

2.2. *UML for knowledge-based system development*

UML diagrams could be adapted for developing knowledge-based systems (Håkansson 2001). The adapted features of the diagrams are:

- Use case diagram: treats knowledge workers' task as a use case.
- Class diagram: classes represent questions, rules, and conclusion.
- Sequence diagram: emphasizes the time ordering of rules and solution.
- State diagram: shows the event-ordered behaviour of rules and solution.
- Activity diagram: models the functions of a system and emphasizes the flow of control among rules.
- Collaboration diagram: emphasizes the structural organization of the rules that draw conclusions.

3. Case-study research

By writing up a case-study report, the real-life context can be described. A case study is also appropriate for explanatory reasons to answer research questions about the 'How' and the 'Why'. In particular, it addresses (1) why a set of decisions were taken, (2) how they were implemented, and (3) what the result was. Hence, we would follow the case-study protocol to validate our theory applied on the adopting organization. The quality of our research design could be judged by the validity and reliability.

Considering the construct validity, appropriate operational measures were built for the study. The proposal of our research is 'The UML approach is useful for developing KBS in a small financial company'. Yin (1989) stated that single-case design is justifiable, provided the case is a rare event. As UML is rarely used under this context, a single-case study would be of added value.

To maintain the internal validity, the credibility on the causal relationship was established in this research. In fact, one of the analytic tactics, so-called explanation building, was used in this case study. This advocates iteratively monitoring the causal relationship between the intervention and the research proposition. Thus, the differences of using and not using UML diagrams were documented throughout the project to validate whether the approach is beneficial to the development of KBS.

To keep the external validity, the findings were generalized beyond this single case study. This was achieved by replication of the theory in other areas. Tactically, the approach was synthesized in the form of lecture notes and group exercises so that students could understand and replicate the theory. The students' answer should be documented to support further research. In addition, to uphold the reliability, the company background and details of the UML adoption were articulated clearly through the case study.

4. Analysis and result

4.1. *System analysis*

4.1.1. Use case analysis. By definition, a use case diagram could illustrate the interaction between the users and the system functions. Under the circumstances of developing a knowledge-based system for supporting business systems, the use cases could be divided into two parts namely knowledge representation and knowledge acquisition. By drawing the use case diagram (figure 1), the role and responsibility of the system specialist (knowledge owner) and helpdesk officer (KBS user) on the KBS can be shown clearly. The system specialist is in charge of in-depth investigation on the cause and solution of the problems. They are the knowledge provider. The helpdesk operator is responsible for daily general support of IT matters. They should be the one who searches for solutions in the system.

Usually, indicating the interactions (verbs) between actors and system can help to identify use case. The five use cases between the system specialist and the KBS are:

- Add New Department;
- Add New Business Operations;
- Add New Symptom;
- Add New Rules;
- Add Solution.

On the other hand, there are another five cases around the helpdesk operator. They are:

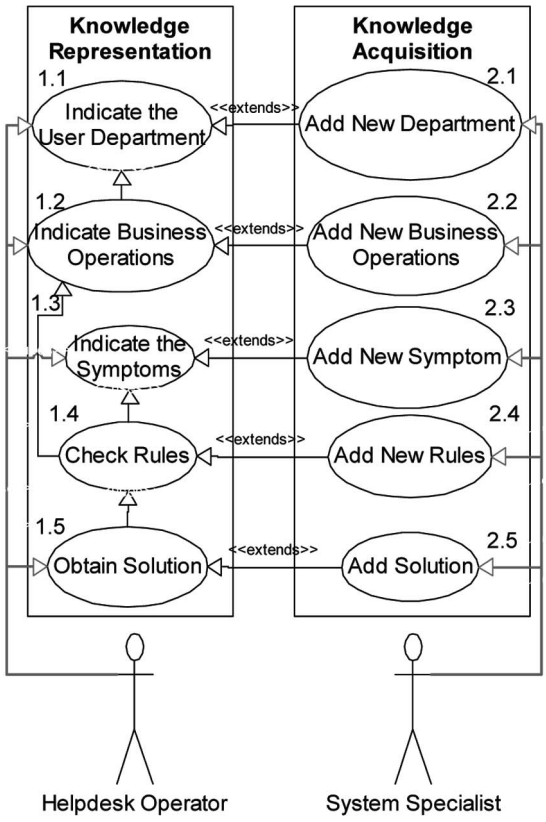

Figure 1. Use case diagram of the knowledge-based system.

- Indicate the User Department;
- Indicate Business Operations;
- Indicate the Symptoms;
- Check Rules;
- Obtain Solution.

The use case ' Obtain Solution' uses another system use case 'Check Rules', which also needs to use the case 'Indicate the Symptom' to work out the solution. There are five 'Extend' relationships in the use case diagram, which links up the cases of:

- 'Indicate the User Department' & 'Add New Department';
- 'Indicate Business Operations' & 'Add New Business Operations';
- 'Indicate the Symptoms' & 'Add New Symptom';
- 'Check Rules' & 'Add New Rules';
- 'Obtain Solution' & 'Add Solution'.

These are related by the 'Extend' rather than the 'Include' arrow because the knowledge workers' use cases are optionally useful for helpdesk operators' use cases only. They are useful when the system has a request for a new department, new business operations, new rule, new symptoms, or new solution.

The detail for each use case can be analysed further in a bid to extract more information from the use case diagram. The result (tables 1 and 2) can be tabulated in use case specifications.

4.1.2. Activity diagram. The diagram in figure 2 shows how the system taking different paths performs activities leading to different results. It also describes how the system responds to a user's request. It depicts the start state, activities that the system performs, and decisions that affect the workflow. The activity diagram describes the workflow

Table 1. Use cases of a helpdesk operator.

Function	1.1 Indicate the user department
Description	Helpdesk operator select a department type to narrow down the scope
Inputs	Helpdesk operator
Outputs	A department type (front, mid or back office)
Pre-condition	The suitable department type exists
Post-condition	The embedded business operations are defined beforehand
Side-effects	NA
Mini-spec	If the suitable department type exists, then select it
Function	1.2 Indicate business operations
Description	Helpdesk operator select a business operation
Inputs	Helpdesk operator
Outputs	A business operation
Pre-condition	The suitable business operation exists under the selected department
Post-condition	List the possible symptom for selection
Side-effects	NA
Mini-spec	If the suitable business operation type exists, then select it

Table 2. Use cases of a system specialist.

Function	2.1. Add new department
Description	Add new department into knowledge base
Inputs	System specialist
Outputs	A new department type
Pre-condition	The new department type does not exist
Post-condition	Add new business operations to the department type
Side effects	NA
Mini-spec	If the department type does not exist, then allow the user add a new one
Function	2.2 Add new business operations
Description	System specialist enter the new business operations
Inputs	System specialist
Outputs	Business operations
Pre-condition	The business operation does not exist
Post-condition	Assign the new business operation to a department
Side effects	The corresponding solution should be defined afterwards
Mini-spec	If the suitable business operation type does not exist, then allow the system specialist add a new one

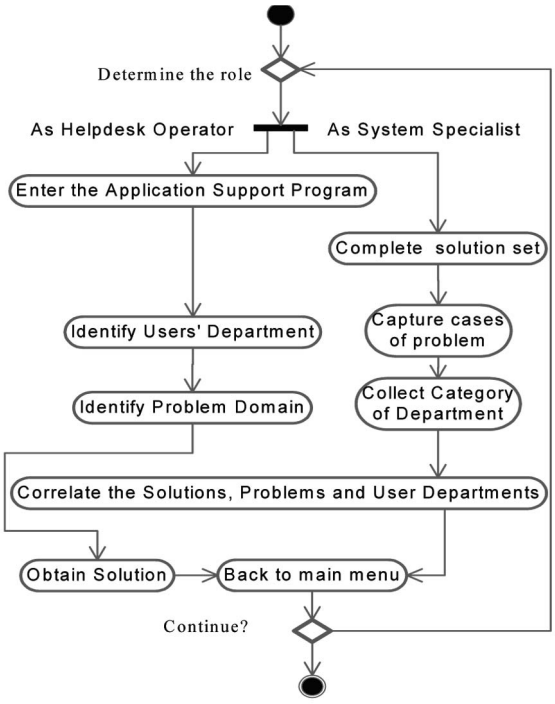

Figure 2. Activity diagram of the helpdesk operators and business system engineer.

of the knowledge-based system. To enable the helpdesk officer to go through the diagnostic process, the system needs to define the boundary of the problem by capturing the user department. Then, the system has to know the problem domain, which contains the problematic process and related symptoms. Finally, the system should be able to propose a solution.

To enable the system specialist to transfer their knowledge, the system has to capture all the available solutions, cases of problem and department category first. Then, it should request the system specialist to correlate them.

Through creating the activity diagram, the key players, the helpdesk operator and the system specialist, can understand the system requirements at a precise level. They can, thus, make valuable comments before the latter stage of KBS development.

4.2. System design

4.2.1. Class diagram. The class diagram (figure 3) can show the static structure and relations of the rules. A class is a group of objects (nouns) that have something in common. It captures a particular abstraction and also provides a template for object creation. Each attribute inside the class box is a set of choices for decision-making. We can treat these as decision boxes. The first decision is

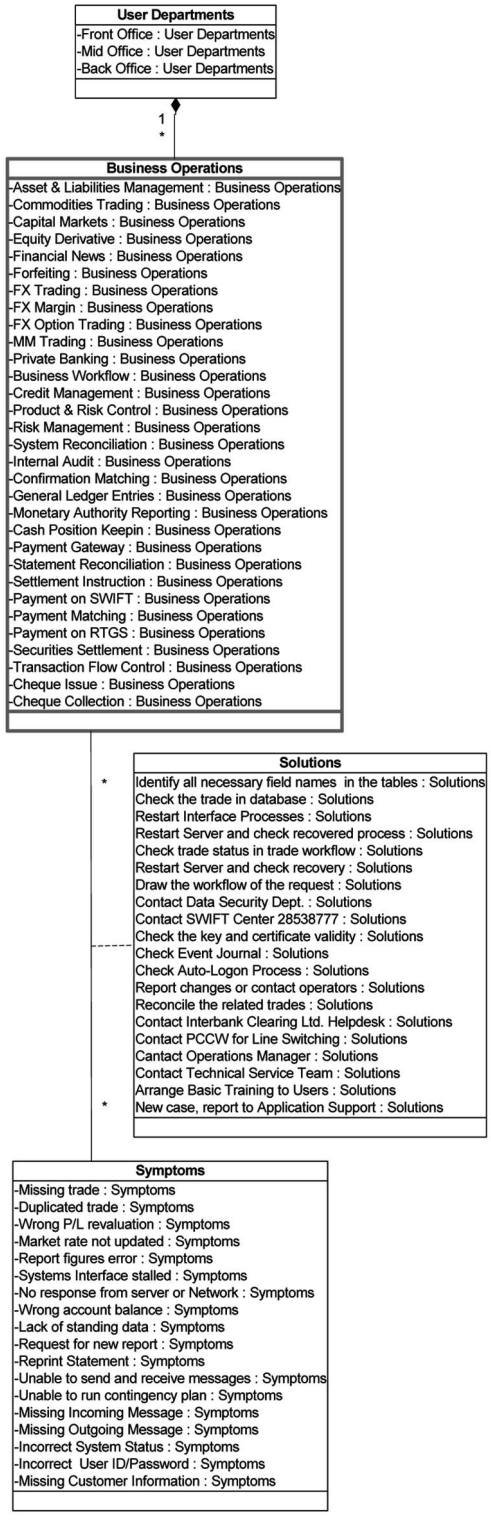

Figure 3. Class diagram for the knowledge-based system.

to determine the category of the user (problem originator). Then, the next decision is to identify the problematic business operation of the user. Finally, the general

symptoms are listed for a user's selection in a bid to determine the appropriate solution. Sometimes, users may not find a suitable choice from the general symptoms list. In this case, the system may suggest other more specific symptoms for that particular business operation. For example, in the case of a SWIFT payment problem, there are several specific symptoms to address the actual problem encountered. The user can then select a more specific symptom from the 'Specific Symptoms of SWIFT Payment' list and obtain accurate answers from the knowledge-based systems.

In terms of relationships, that between department class and business operation class is a composition type. A composition relationship indicates that an object is owned by a greater whole (Arrington and Rayhan 2003). The contained object (business operation) may not belong to more than one composition relationship, cannot exist independently of the whole, and is destroyed when the whole is destroyed. That is, the business operation would not exist if the department were closed. In other words, it is not an aggregation relationship because of the following two assumptions:

1. A business operation is owned by, but not part of, a department.
2. A business operation does not exist independently without a department.

The second relationship is that between business operation class and symptom class. This is an association relationship, which indicates a long-term relationship between the classes. In other words, a business process keeps a reference to another symptom whenever it is needed. In contrast, there is neither a 'part of' nor an 'owned by' relationship between them. Thus, it is neither an aggregation nor a composition relationship. Also, it is not a dependency relationship, as the relationship of the business operation class with symptom class would not end after method calling. In fact, the relationship between them builds up an association class—a solution.

A business process and a symptom can yield a solution. For example, the solution 'Check the trades in database' is always a solution of the business operations 'FX Trading' and the symptom 'Missing Trades'.

In terms of multiplicity, one department may have many business operations, but one business operation belongs to only one department. In contrast, one business operation may have many symptoms, and one symptom may relate to many business operations.

4.2.2. Sequence diagram. The classes, so-called event owners in the sequence diagram (figure 4), were extracted from the previous class diagram. The sequence

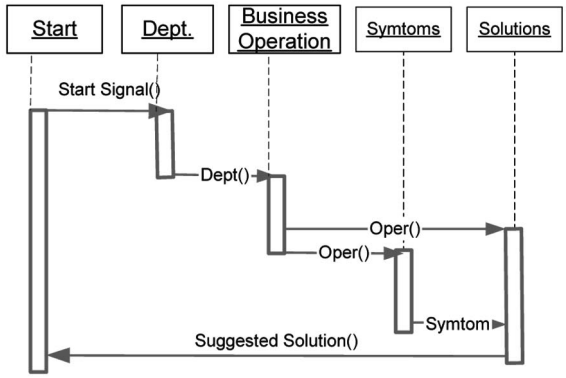

Figure 4. Sequence diagram of the knowledge-based system.

diagram was used to illustrate when and how long a class (head box) is triggered for action. It also showed the processes (arrow) used to trigger the event owners. The sequence of the processes started from top to bottom of the diagram. For instance, the StartSignal() process was the first process, whereas AffectDepartment() was the last process.

4.2.3. State chart diagram. State chart analysis is not necessary for all classes. To use this diagram smartly, user should select the complicated or interested classes for state analysis.

The state chart diagram shows the mode and the flow of queries in the solution class. The figure 5 illustrates the state changes during figuring out the solution in the knowledge-based system. The main difference between the sequence diagram and the state chart diagram is the scope. The sequence diagram focuses on the time order among all the classes, whereas the state chart diagram emphasizes the event-driven order with a class.

4.3. System implementation

4.3.1. Component diagram. The component diagram is a static model that shows the dependency among executable components of this program. As shown in figure 6, the business rules and the interfaces of this project are stored in a component called AppsSupp.xrp. XRKB is a package that enables the programmer to create, edit, and test a project. It also offers many built-in functions for common usages of projects. Based on the programmer's request, the XRKB package would obtain business rules and interface information from AppsSupp.xrp to perform the required tasks.

The back-end database used for storing the configurations, variables, and rules is an MS Access database. The database name of this project is AppsSupp.mdb. XRKB would update the latest project information into AppsSupp.mdb after getting a saving request.

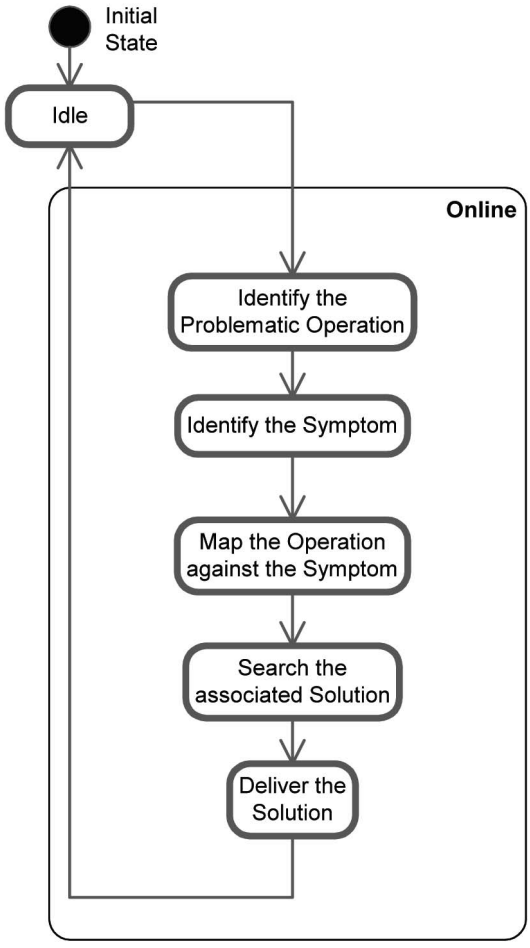

Figure 5. State diagram of the solutions class.

The last element, MS Outlook, is represented as a node of run-time computation resource. MS Outlook is necessary because after completing a helpdesk call, a standard email with the details required would be sent to a dedicated mailbox. In this case, XRKB would trigger the MS Outlook function built in to the PC.

4.3.2. Deployment diagram. During run-time processing, as seen in the deployment diagram in figure 7, only AppsSupp.xra, Xrclient.exe, Xrserer.dll, and XrXB_Srv.dll are needed in the client PC. The design-time components, such as AppsSupp.xrp, XRKB package, and AppsSupp.mdb are not necessary. In other words, users do not need to install the expert system tools package and copy the related program source/database to the target PC during deployment. Xrclient.exe is the main executable program to execute the program. During execution, it obtains the business rule logic and interface information from the exported package called Apps-Supp.xra.

Apart from the AppsSupp.xra program, Xrclient.exe needs Xrserver.dll and XrKB_Svr.dll components to start the program. The user only need copy the above files into a dedicate directory and create a shortcut with the following typical content. 'c:\program files\XRKB\Xrclient.exe c:\program files\XRKB\AppsSupp.xra' Moreover, as per the program design, e-mail is needed after closing each helpdesk call. Thus, MS Outlook should be available on the client PC to enable the above service. The program would call the MS Outlook automatically.

4.4. Case study of using the KBS

The company in this case study was a foreign-owned investment bank, which has been operating in Hong Kong

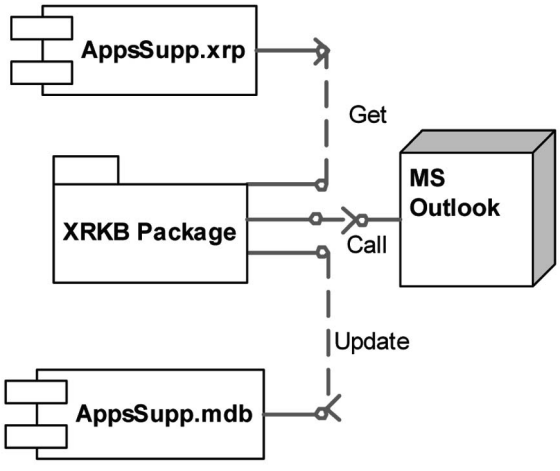

Figure 6. Component diagram of the KBS.

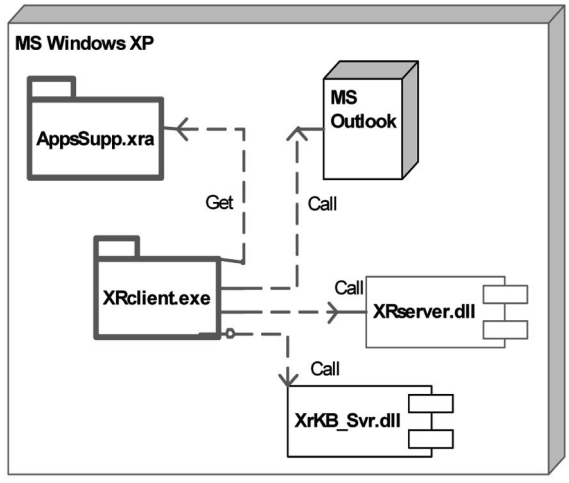

Figure 7. Deployment diagram.

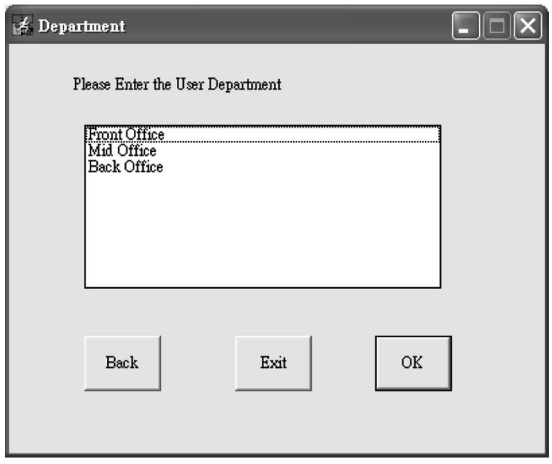

Figure 8. Case study—department enquiry.

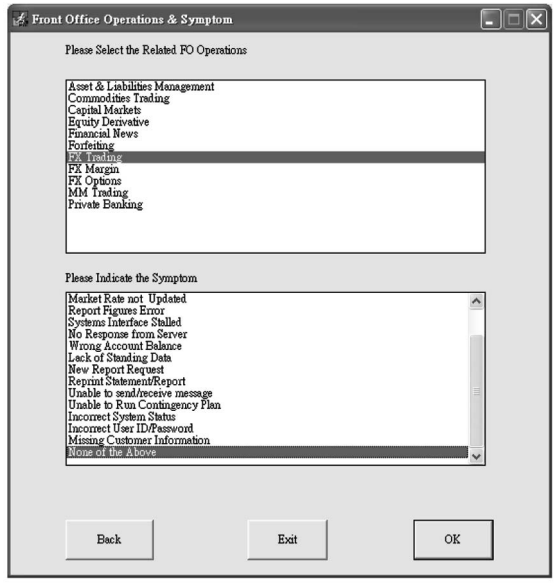

Figure 9. Case study—operations and symptoms.

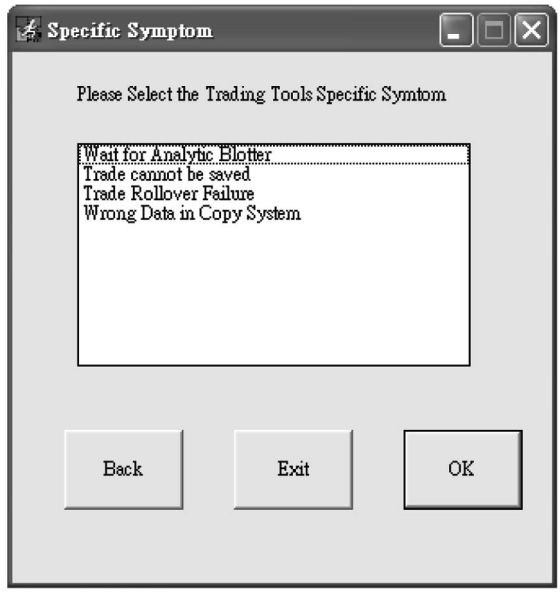

Figure 10. Case study—specific symptom.

for 10 years. It had one local office and employed more than 100 staff. The Hong Kong office was the IT centre of the Asian-Pacific region of the group. There were four system analysts and four helpdesk officers to support all kinds of IT system. The main users of this KBS were IT staff. The method of usage was as follows. First, the system administrator had to create all the user names and user id in the system. Their name was used for generating a recommendation report at latter stage. After selecting the user id, the IT staff had to input the user department as shown in figure 8. The user is one who raises the request or mentions the problem. The user department is the department to whom the users work with. The front office department normally conclude the dealing related operations, whereas mid-office involves all the risk management and financial control activities. Finally, back office mainly focuses on deal settlement and payment-related activities.

After selecting the department, the IT staff should choose the process that is most related to the user's problem. As shown in the figure 9, the IT staff need to select one of the front operations for further analysis, like asset and liabilities management, commodities trading and capital markets. Different departments would have different choices of process. The IT staff can click 'Back' to go one step backward if they found they made a wrong choice (department) on last step.

Similarly, the IT staff would select the relevant process under the mid-office or back-office operations menu. They are allowed to select only one process. If they need to solve more than one process at the same time, they should handle the cases one by one.

Then, the IT staff should ask the user about the specific problem and corresponding phenomenon. They could choose the symptom that best matches to the phenomenon. Sometimes, none of the general symptoms is suitable. In this case, the IT staff could select 'None of the above' from the general symptoms list. Then, a specific symptom may come up for further selection. As shown in figure 10, if one selects 'none of the above, in the FX Trading of the Back Office operations, the next step would be 'Trading Tools Specific Symptom'. An even deeper question would be asked if necessary.

After entering all the necessary information into the knowledge-based system, it would check the predefined rules and derive the suggestion for the user. The result is presented as a report on the screen. Then, the call can be sent to the MS Outlook Express to record the incident. The users have to select 'Transmit' in a bid to release the mail. The name of the operator, date & time, department, operations, symptom, and solution of the problem would be written on the email for further investigation.

5. Discussion

5.1. Hidden successful factors of system specialists

Although the KBS can help the helpdesk officers to solve some of the business system problem, it cannot replace the function of system specialists. During the knowledge acquisition, some of the core competences of a system specialist were identified. Apart from the requirement of technical and financial knowledge, the project identifies other factors, which may be treated as 'hidden success factors' of a system specialist. These factors are detailed below.

5.1.1. Human network. As a system specialist, they may often face a new problem that they have never encountered. In this case, the support should be able to make use of the specialists human network to acquire relevant information and knowledge. The network can include colleagues, vendors, ex-colleagues, classmates, academic professors, friends, or even relatives. The better the human network, the easier it is to source information and solve problems. This is one of the reasons why a KBS can hardly replace a system specialist. Instead, the system can record the source of information for the next usage.

5.1.2. Project-management skill. Usually, a system specialist needs to handle projects periodically. They may have to cater for a large number of tasks at the same time. They should be able to prioritize the tasks and allocate their resources suitably. Tasks can be classified by importance and urgency. To minimize the impact of complaint from different users, the system specialist should handle the high-importance and high-urgency tasks, followed by high-importance and low-urgency tasks. It is those high importance tasks that would often need fire-fighting action when they become urgent later on. This may affect the schedule of the original planning and make the jobs unmanageable.

5.1.3. Communication skills. As mentioned before, a system specialist has to interact with many people. Communication skills should encompass both verbal and written communication. Verbal communication is more important at the very beginning of an event, as people do not know their role and responsibilities. As long as people gain a clearer picture, they should understand their strength/weakness and opportunities/threats of that event. Then, written communication becomes more effective to confirm the commitment from different parties.

Generally speaking, good communication skills cannot be imparted within a short period. Such skills depend on many personal factors, such as academic background, language ability, experience, exposure, attitude, friendliness, and so forth.

5.2. Evaluation of applying UML in KBS development

5.2.1. Using UML for rule-based knowledge analysis. Technically, workflow analysis can be illustrated clearly using the activity diagram. The expert can easily discuss the correctness of the sequence of the questions. Also, the use case diagram is feasible for analysing rule-based knowledge. It can clearly classify the actors and processes into a knowledge-acquisition category and knowledge-representation category. This can help the knowledge worker to understand the role of different actors and their corresponding processes involved in the system. Without these diagrams, the expert cannot understand the scope of the study and help to correct any misunderstanding at the beginning of the project.

5.2.2. Using UML for rule-based knowledge design. Rules and solution objects are feasible when using the class diagram to design the consultation questions. The attribute of each class of rules is a list of choices for end users. The associative class between the business operation class and the symptom class is the solution class. The diagram can show the relationships among classes accurately and, thus, accelerate the development process at a later stage. However, it is not an easy task to present all the possible choices in a class diagram. The sequence diagram and state diagram can illustrate time-order and event-order of knowledge representation embedded in the new KBS. To sum up, without these diagrams, the knowledge engineer may not be able to realize the knowledge structure and knowledge representation sequence before programming and implementation.

5.2.3. Using UML for rule-based knowledge implementation. The component diagram and deployment diagram are also used in this project. This is an innovative usage, as the literature has not previously mentioned their usage on KBS development. In fact, they are very useful to transfer the necessary knowledge to the developers and IT implementer, respectively. The component diagram can illustrate the static structure of elements used for editing the source code. It is useful for a knowledge engineer to maintain and update the inference engine and the knowledge base. On the other hand, the deployment diagram can instruct the IT implementer how to install the KBS onto a client's PC.

Since KBS would be distributed to many helpdesk operators' desktop computers, the run-time configuration should be illustrated clearly to shorten the time for installation and version upgrading.

6. Conclusion

In summary, this paper has articulated a way of using UML to develop a knowledge-based system for a small financial institute in Hong Kong. The case study explained how the product, a customized knowledge-based system, can assist helpdesk officers in supporting their business systems. The best practices of using UML to develop KBS were also consolidated as teaching materials for class discussion in the elective subject 'Electronic Business for Small and Medium Enterprises' at the MSc level at The Hong Kong Polytechnic University. They can be used as the guidelines for replication or further validation of the theory.

Acknowledgements

The authors acknowledge the support offered by a Central Grant of The Hong Kong Polytechnic University under Project Number G-YA19 and Department of Computer Sciences, The City University of Hong Kong.

References

Afshar, F., Capturing consensus knowledge from multiple experts, in *Proceedings of ES2002, the Twenty-Second SGAI International Conference on Knowledge Based Systems and Applied Artificial Intelligence*, 2002, pp. 253–263.

Arrington, C.T., Rayhan, S.H., *Enterprise Java with UML*, 2003 (OMG Press: Arrington, CT).

Booch, G., *Object-Oriented Analysis and Design with Applications*, 1994 (Addison-Wesley-Longman: Reading, MA).

Booch, G., Rumbaugh, J. and Jacobson, I., *The Unified Modelling Language User Guide*, 1999 (Addison-Wesley-Longman: Reading, MA).

Eng, M.V., Lees, F.A., Mauer, L.J., *Global Finance*, 1998 (Addison-Wesley: Reading, MA).

Håkansson, A., UML as an approach to modelling knowledge in rule-based systems, in *Proceedings of ES2001, the Twenty-first SGAI International Conference on Knowledge Based Systems and Applied Artificial Intelligence*187–200, 2001.

Jacobson, I., Booch, G., and Rumbaugh, J., *The Unified Software Development Process*, 1999 (Addison-Wesley: Reading, MA).

Lam, K.W., *Propositional Logic, Handouts of Artificial Intelligence of MSc in Computing*, 2003.

Ross, T., *Knowledge Based Systems: Survey of UK Applications*, 1992 (Department of Trade and Industry: London).

Yin, R.K., *Case Study Research: Design and Methods*, 1989 (Sage: Thousand Oaks, CA).