

# A Use Case Modeling Approach for Large-scale, Industrial, Network-based, Distributed, Real-Time Embedded Systems

Tao Yue

Simula Research Laboratory  
P.O.Box 134, 1325, Lysaker, Norway  
e-mail: tao@simula.no

Shaukat Ali

Simula Research Laboratory  
P.O.Box 134, 1325, Lysaker, Norway  
e-mail: shaukat@simula.no

**Abstract**—The transition from requirements expressed in natural language (NL) to a structured, formalized specification is an important issue to support requirements analysis and to generate an (initial) analysis model. The latter is also an important step of any (object-oriented) software development method, including the OMG’s Model Driven Architecture (MDA). Use case diagrams are commonly used to capture system requirements as use cases, relationships among them and interactions with actors. Using template and restriction rules is a common feature of NL analysis for reducing imprecision and incompleteness in use case specifications. Use case diagrams, together with template and restriction rules, form a use case modeling approach, which helps achieve better understandability of use cases and improved quality of derived analysis models. In this paper, we extend a general use case approach (RUCM) and use it in conjunction with the UML MARTE profile to model large-scale, network-based, distributed and real-time embedded (NDRTE) systems. One RUCM extension is proposed to achieve this objective: RUCM-NDRTE. The feasibility of RUCM-NDRTE has been evaluated by applying them to two industrial applications from communication, and maritime and energy domains.

**Keywords**—Use case modeling; model driven architecture; real-time embedded system .

## I. INTRODUCTION

Use case models (UCMods) are a widely used means for specifying functional requirements of systems and commonly used as a communication means between stakeholders. Use cases are generally textual-based and thus are ambiguous. To decrease such ambiguity, we proposed a Restricted Use Case Modeling approach, named as RUCM [1]. RUCM contains a use case template and a set of restriction rules for textual Use Case Specifications (UCSs), which is based on a systematic literature review [2] on transformations of textual requirements into analysis models. Using template and restriction rules is a common feature of natural language analysis approaches for reducing imprecision and incompleteness in UCSs. The main factor to consider is that the restricted natural language should be expressive and convenient enough for use by developers. We have conducted two controlled experiment to evaluate RUCM and the results demonstrate that RUCM has enough expressive power, is easy to apply, and helps achieve better understandability of use cases and improved quality of derived UML analysis models in [3].

A UCMod specified using RUCM is composed of one or more Use Case Diagrams (UCDs) and a set of UCSs for each use case in the UCDs, which are specified using the RUCM use case template and conformed to the RUCM natural language restriction rules. RUCM is a general approach and therefore it is meant to be extensible to specify UCSs of any application domain. In RUCM, we used UML UCD without any extensions. UCMods specified with RUCM can then be formalized using Use Case Metamodel (UCMeta) [4]. UCMeta can be considered as a formalization language of textual RUCM models, and also an intermediate model bridging the gap between the textual RUCM models and UML-based system analysis and design models (e.g., UML class and sequence diagrams). The formalized textual RUCM models (i.e., instances of UCMeta) can then automatically be transformed into UML analysis models such UML class and sequence diagrams [4], activity diagrams [5], and state machine diagrams [6].

Based on our experience of working with industries in the domains of communication systems and maritime and energy sectors, we observed a need to extend RUCM to address the special needs of capturing important characteristics such as specifying communication links and also their properties that connect different communication endpoints in UCDs. In addition, one of our industrial applications is to specify requirements in UCMods to facilitate robustness testing of communication and control systems [6]. Therefore, it is required to capture sufficient information in the UCMod so that this kind of analysis/testing can be facilitated. A UCMod specified using our approach, can later on be (either manually or automatically) transformed into other software artifacts (e.g., UML analysis models) to support different purposes such as providing an initial design of a system or generating test cases. This paper proposes RUCM-NDRTE, an extension of RUCM, which is used in conjunction with MARTE [7], a UML profile that supports modeling real-time embedded systems, to support use case modeling in the domain of large-scale, Distributed, Network-based, Real-Time and Embedded (NDRTE) systems. We applied RUCM-NDRTE to two industrial applications and we observed that it is applicable for large-scale NDRTE systems.

The rest of the paper is organized as follow: Related work is presented in Section II. Background presented in Section III. Section IV presents RUCM-NDRTE including a discussion on the extended UML UCD notation and an

extension to RUCM use case template. Evaluation via two industrial case studies is presented in Section V. The paper is concluded in Section VI.

## II. RELATED WORK

It is a common practice to follow a use case template to structure UCSs, thereby helping their reading and reviewing. Various use case templates (e.g., [8-12]) have been suggested in the literature to satisfy different application contexts and purposes. In addition to capturing requirements, use cases can also facilitate the automated derivation of initial analysis models – one of our goals. The systematic review [2] we conducted to examine literature works that transform textual requirements into analysis models reveals that six approaches require use cases. RUCM template is based on the systematic review results and contains fields similar to those encountered in conventional use case templates but with a few variations on the structure of the flow of events. The RUCM template not only complies with conventional use case templates as much as possible but also facilitates the process of deriving analysis models. In this paper, we propose an extension to UML use case diagram and an extension to RUCM template and restriction rules, to support use case modeling of the NDRTE domain and we have not noticed any use case modeling approach reported in literature that has achieved the same or similar objective as ours.

## III. BACKGROUND

### A. Running example

The running example is a Video Conferencing System (VCS). The core functionality of the VCS is sending and receiving multimedia streams. VCS deals with establishing video conferencing calls, disconnecting calls, and starting/stopping presentation. It can also receive requests for establishing calls, disconnecting calls, and starting/stopping presentation from other video conferencing systems (Endpoints) participating in a videoconference. This running example will be used in the rest of the paper to illustrate our approach. Part of the use case diagram describing the functionalities of VCS is provided in Figure 1.

### B. Characteristics of NDRTE

Our objective is to devise a practical solution to specify requirements of large-scale NDRTE systems as UCMods based on their characteristics, which are briefly discussed below:

- Large-scale systems of systems are often network-based and distributed. Distributed systems/subsystems/components should have their functionalities captured as use cases and a mechanism to group these use cases separately for each system/subsystem/component is required.
- Communication between components in a network-based system is through message passing. This should be captured explicitly in UCMods.
- Embedded systems are computing systems with tightly coupled hardware and software integration. Therefore

actors (e.g., external instruments) of the UCMod for an embedded system must be captured.

- The correctness of real-time systems depends on functional and timing correctness. For these systems, timer is a commonly used actor, which periodically initiates the execution of a use case.

Notice that NDRTE is a complex domain and due to space limitation, we only present a few key characteristics above, which are important at the level of requirements specification.

### C. RUCM

Use case modeling approach RUCM encompasses a use case template and 26 well-defined restriction rules [1]. Rules are classified into restrictions on the use of NL, and rules enforcing the use of specific keywords for specifying control structures. Its goal is to reduce ambiguity and facilitate automated analysis. We performed two controlled experiments to evaluate RUCM in terms of its ease of use and the quality of the analysis models derived by trained individuals [1]. Results showed that RUCM is overall easy to use and that it results in significant improvements over the use of a standard use case template (without restrictions to the use of NL) in terms of the quality of derived class and sequence diagrams and the understandability of UCSs.

Table 1 Use case Disconnect

<i>Use Case Name</i>	Disconnect		
<i>Brief Description</i>	User disconnects an Endpoint participating in a conference call.		
<i>Precondition</i>	The system is in a conference call.		
<i>Primary Actor</i>	User_VCS	<i>Secondary Actors</i>	EndPoint
<i>Included Use Case</i>	None	<i>Specialized Use Case</i>	None
<i>Extending Use Case</i>	None	<i>Invoked Use Case</i>	None
<i>Basic Flow</i>	1) User_VCS sends a message to the system to disconnect an Endpoint. 2) The system VALIDATES THAT Endpoint to be disconnected is in the conference call. 3) The system sends a disconnection notification to Endpoint VIA Ethernet NETWORK. 4) Endpoint sends an acknowledgement message back to the system VIA Ethernet NETWORK. 5) The system VALIDATES THAT The conference call has only one EndPoint. 6) The system disconnects Endpoint. <i>Postcondition:</i> The system is idle.		
<i>Specific Alt. Flow (RFS Basic flow 2)</i>	1) The system sends a failure message to User_VCS. 2) ABORT. <i>Postcondition:</i> The system is in a conference call.		
<i>Specific Alt. Flow (RFS Basic flow 5)</i>	1) The system disconnects Endpoint. 2) ABORT <i>Postcondition:</i> The system is in a conference call.		

Table 1 is an example of UCS documented with RUCM. Use case *Disconnect* contains one basic flow, two specific alternative flows (first column). The two specific alternative flows are used to branch from the basic flow under specific conditions. RUCM specifies three different types of alternative flows. Specific and bounded flows indicate from which step in which flow of reference they branch whereas a global flow can branch from any step. For instance, the specific alternative flow in Table 1 branches from Reference Flow Step (RFS) 2 in the basic flow, and the condition for branching is in that step: the software fails to VALIDATES THAT some condition holds. Restrictions to NL take the

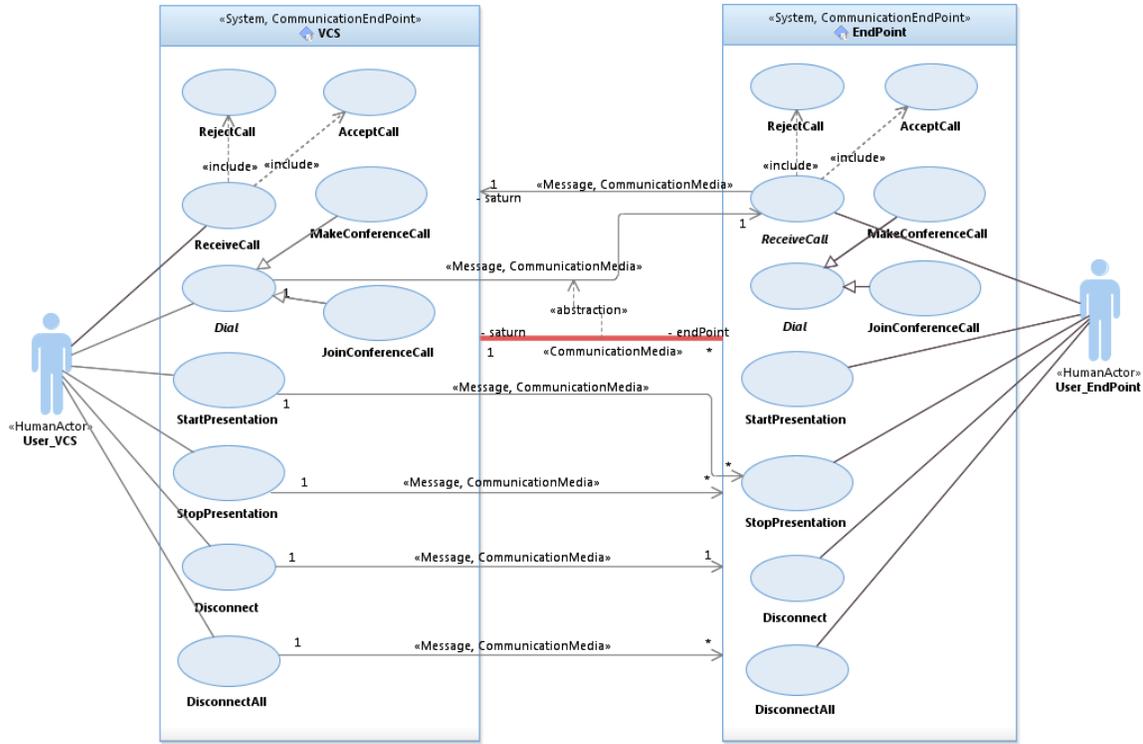


Figure 1 Use Case Diagram of a Video Conference System (Partial)

form of keywords, like VALIDATES THAT. The four highlighted fields are newly introduced fields for RUCM-NDRTE and will be discussed in Section IV.

#### IV. RUCM-NDRTE

Our RUCM-NDRTE use case modeling approach consists of a UML profile NDRTE, along with a subset of MARTE extending UML UCD (Section A), a set of advanced UCD modeling features (Section B), and an extension to RUCM including its template and restriction rules (Section C).

##### A. NDRTE Profile

NDRTE has three packages: actor classification, use case classification, and network and communication presented in Figure 2, Figure 3, and Figure 4, respectively.

###### 1) Actor classification

The UML UseCases package does not classify actors. The most basic classification defined in UCDs for actors are primary and secondary actors, as we defined in our general use case modeling approach RUCM. However, for many systems, especially NDRTE systems, it is desirable to characterize different types of actors. For instance, a typical control and communications system interacts with the instruments under its monitoring or control such as sensors and actuators. Based on our experience of working with two industrial NDRTE systems, we further classified actors into the following categories: 1) Human actor (e.g., Operator and Tester); 2) External system; 3) External instrument (e.g., sensors, actuators, camera, speaker, microphone); and 4)

Timer, representing a hardware or software entity that can periodically initiates use cases.

Each type of actor is specified as a UML stereotype to extend UML metaclass Actor, as shown in Figure 2, where abstract stereotype <<Actor>> is used to organize this set of actor types. Actor ExternalInstrument is further characterized based on its direction (through attribute direction : IODirection). For example, an external device such as a sensor should be an input device; whereas a speaker should be an output device. Actor Timer specializes the MARTE stereotype <<TimerResource>>.

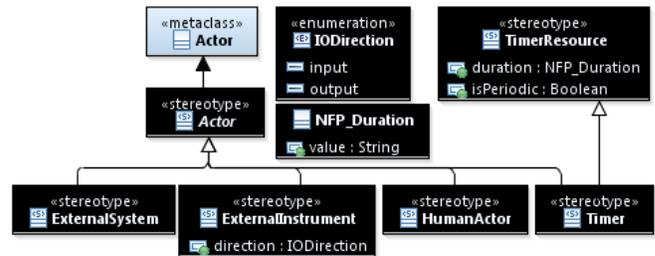


Figure 2 NDRTE Actor Classification Profile Package

###### 2) Use case classification

Because of the characteristics of NDRTE and the requirements in our particular application domains and also to capture network failure scenarios to support robustness testing, we classify the use case types into the following two types: network abnormal use cases and periodical use cases. Two stereotypes are defined to extend UML metaclass UseCase, as shown in Figure 3. An example is shown in

Figure 5, where use case *IntroducePacketLoss* is stereotyped with `<<NetworkAbnormalUC>>`.



Figure 3 NDRTE Use Case Classification Profile Package

Stereotype `<<PeriodicalUC>>` characterizes periodical use cases that are initiated by a Timer actor after a specific interval of time. This kind of use cases is common in real-time systems.

Model-based robustness testing of networked systems requires emulating hostile network situations, against which the system should be robust, i.e., it should not crash or halt. In the worst case situation, the system should go back to the most recent safe state. The hostile network situations are coined as network abnormal use cases in this paper. Tester is the actor who initiates all these use cases. Each abnormal use case is stereotyped as `<<NetworkAbnormalUC>>`, as shown in Figure 5. For example, use case *IntroduceCorruptPacket* emulates the corruption of certain percentage of packets in the network.

3) Network and communication

NDRTE systems communicate with each other via communication channels (Section III.B). Conventional use case modeling approaches cannot be applied to model such systems.

We introduce stereotype `<<System>>`, which is applied on components of a system. A system component communicates with another component of the system through network. In UCDs, the network communications are captured as associations between two components. In this context, the two components are communication endpoints and the network is the communication media. Therefore, as shown in Figure 4, we introduce stereotype `<<CommunicationEndpoint>>` and apply it on components stereotyped with `<<System>>`. We also introduce stereotype `<<CommunicationMedia>>` and apply it on the communication network. These two stereotypes are adapted from the MARTE profile [7]. We simplified the properties of the corresponding MARTE stereotypes since we introduce them to UCDs and therefore it is important to keep the level of abstraction. Enumeration *CommunicationMediaKind* shows a network classification. When stereotype `<<CommunicationMedia>>` is applied on a network connection, the value of the attribute *communicationMediaKind* of the stereotypes indicates the type of the network (e.g., Ethernet).

An example of applying these network and communication stereotypes are provided in Figure 1. The system VCS communicates with multiple EndPoints via a group of network connections, which is represented as the association between the VCS and EndPoint components. This ‘abstract’ communication link is associated to concrete ones through UML Abstraction. For example, VCS dials an EndPoint via the network connection and therefore triggers use case *ReceiveCall* of the EndPoint by sending messages. This concrete network connection is captured as an association between use case *Dial* of VCS and *ReceiveCall*

of EndPoint. This association is stereotyped with `<<Message>>` and `<<CommunicationMedia>>`. We introduce stereotype `<<Message>>` and its attribute *messageKind* to characterize messages transferred from one system to another through communication channels. There are three types of messages: *synchronous*, *asynchronous*, and *reply* as shown in Figure 4. For example, use case *Dial* of VCS sends synchronous messages to *ReceiveCall* of EndPoint. Asynchronous messages are transferred from use case *StopPresentation* of VCS to system EndPoint. Reply messages are sent from use case *ReceiveCall* of EndPoint to VCS to notify VCS whether it accepts or rejects the conference call. To avoid cluttering the diagram, we do not show the attributes of stereotypes in the UCDs.

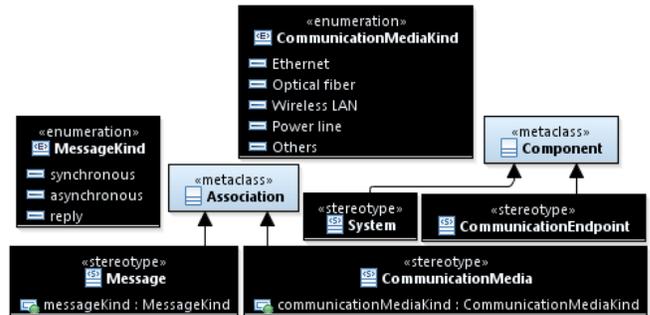


Figure 4 Network and Communication Profile

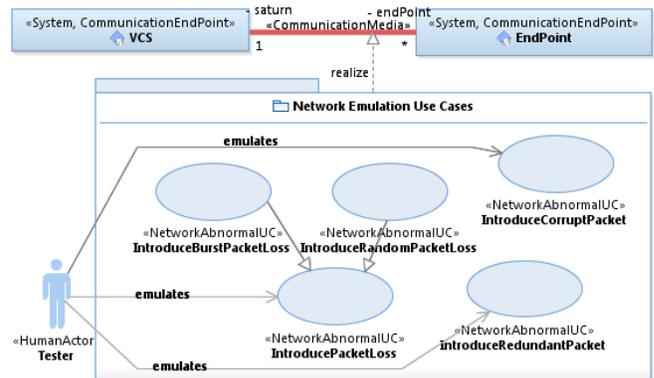


Figure 5 Network Emulation Use Cases

B. Advanced Use Case Diagram Modeling features

The most commonly used elements in UCDs include use cases, actors, associations between actors and use cases, extend and include relationships between use cases, and generalization between use cases. The UML UseCases package semantically supports more advanced UCD modeling features, which are rarely used in the literature. We use these features as part of our RUCM-NDRTE since we found them very useful in our context.

1) Group use cases for subsystems

In UML, a component “represents a modular part of a system that encapsulates its contexts” and “a component is modeled throughout the development lifecycle and successively refined into deployment” [13]. In our context, we use components to represent subsystems of a distributed system and group all the use cases of each subsystem. For

example, as shown in Figure 1, two components are used to group use cases of subsystems VCS and EndPoint.

2) *Group use cases for separation of concerns*

In UML, packages are used to group model elements and provide namespaces for the grouped elements. In our context, a UML package is used to group a set of use cases, which represents a group of functionalities for separation of concerns (see Figure 5 for an example).

3) *Connect actors, use cases, and packages*

An association can be used to connect an actor to a use case, which is very commonly used when a UCD is designed. In our context, we also use associations for various purposes.

**Connect two use cases:** Such associations should be stereotyped with `<<Message>>` and `<<CommunicationMedia>>` (Section IV.A) to specifically define that there are messages transferred from a use case of one system to a use case of another through a communication channel.

**Connect a use case to a component:** As shown in Figure 1, use case `StopPresentation` is associated to the `EndPoint` component, which indicates that when VCS stops its presentation, a message will be sent to all the EndPoints that VCS have been given presentation to. Therefore the multiplicity of the association on the `EndPoint` side is many (shown as \* in the figure).

**Connect two components:** When an association is used to connect two components representing two subsystems, it is an “abstract” association in the sense that it shows that two subsystems are connected through network. As discussed in Section IV.A, stereotype `<<CommunicationMedia>>` should be applied on such associations to characterize the type of the network (e.g., Ethernet).

4) *Model relationships between associations and packages*

To show the dependency between an “abstract” communication link (association between two components) and concrete message transferring between use cases via the communication link, we propose to use UML Abstraction to link the abstract communication link to the concrete ones, as we discussed in Section IV.A.

To show the dependency between an “abstract” communication link (modeled as an association between two components) and a package grouping a set of use cases, we propose to use UML Realization to show the dependency of the association and the package. As shown in Figure 5, the abstract communication link between VCS and EndPoint is linked to the `Network Emulation Use Cases` package which contains all the network abnormal use cases through a `<<realize>>` dependency.

C. *Extension to RUCM*

1) *Extension to RUCM use case template*

Four new fields are introduced to the RUCM template: *Included Use Case*, *Extending Use Case*, *Specialized Use case*, and *Invoked Use Case*, respectively, indicate the use cases that are included by the use case, the ones that extend

the use case, the use cases that specialize it, and the use cases in another system that are initiated by it via the network connections between two systems. Table 1 and Table 2 provide the UCSs of *Disconnect* (system use case) and *Dial* (abstract use case, invoking use case `ReceiveCall` of `EndPoint`). All fields of the template of RUCM-NDRTE are shown only in Table 1. Empty fields are omitted for the specifications of other use cases.

Table 2 Use case Dial

<i>Use Case Name</i>	Dial
<i>Brief Description</i>	User_VCS dials the system.
<i>Precondition</i>	The system is powered on.
<i>Primary Actor</i>	User_VCS
<i>Invoked use case</i>	USE CASE <code>ReceiveCall</code> of <code>EndPoint</code>
<i>Basic flow steps</i>	1) User_VCS dials the system. 2) The system invokes <code>ReceiveCall</code> of <code>EndPoint</code> VIA Ethernet NETWORK. 3) <code>EndPoint</code> sends a reply message to the system VIA Ethernet NETWORK. 4) The system VALIDATES THAT <code>EndPoint</code> accepted the conference call. 5) The system establishes the conference call. <i>Postcondition:</i> The system is in a conference call.
<i>Specific Alt. Flow (RFS Basic flow 4)</i>	1) The system goes to idle. 2) ABORT. <i>Postcondition:</i> The system is idle.

2) *Extension to RUCM restriction rules*

We define a convention that ‘the system’ and ‘the network’ are used to denote the system under design and the network connecting the system to other systems. This convention is specified as a restriction rule and should be followed when a UCS is specified. For example, phrase ‘the system’, instead of phrase ‘VCS’, should only be used to denote the system VCS. By doing so, ambiguity can be reduced and therefore it helps generating higher quality of analysis models.

Two new keywords `VIA <Type> NETWORK` and `PERIODICALLY` are specified for RUCM-NDRTE. `VIA <Type> NETWORK` is used to indicate which step of a flow of events transfer a message from a communication endpoint to another one. ‘Type’ is a variable and should be replaced with a specific type of network communication such as Ethernet. For example, step 3 of the basic flow of the UCS of use case `Dial` (Table 2), contains this keyword and show that the system invokes `ReceiveCall` of `EndPoint` through the Ethernet network. Keyword `PERIODICALLY` is used in an action step of flows of events to indicate the step in which a use case is periodically executed.

V. EVALUATION

A. *Empirical evaluation of RUCM via controlled experiments*

We have conducted two controlled experiments to evaluate RUCM in terms of its applicability and impact on the quality of manually derived UML analysis models with UCMods written in RUCM as input. More specifically we wanted to answer these two research questions: 1) do users find RUCM too restrictive or impractical in certain situation? 2) do the rules and template have a positive, significant impact on the quality of the constructed UML analysis models? The experiment results (reported in [3]) show that RUCM is easy to apply and the RUCM results into significant improvements over traditional approaches (i.e.,

with standard templates, without restrictions) in terms of the quality of derived class and sequence diagrams.

Regarding the RUCM-NDRTE approach, it only extends use case diagrams and has very limited extension to the RUCM template and restrictions (Section IV.C). Hence, in terms of describing UCSs, RUCM-NDRTE should be very similar to RUCM. However, as discussed in Section IV.A, the NDRTE profile is introduced to extend the UML use case diagram notation and it should be evaluated to test its applicability. In the future, we plan to conduct empirical studies to evaluate the NDRTE profile.

*B. Industrial case studies*

Two large-scale NDRTE systems, respectively from the communication domain, and the maritime and energy sectors, are used to evaluate our RUCM-NDRTE. One is a video conference system (VCS) and the other is a subsea oil production system (SOPS). Table 3 presents the characteristics of the RUCM models of these two systems. VCS contains four systems/endpoints, which have the same functionalities. These functionalities are modeled as the same set of use cases. Each endpoint has 10 use cases; in total the whole system contains 40 use cases. Each of the VCS endpoint is operated by a human actor. A timer is needed to periodically initiate the adaption of call rate in case of problems with network. All the ten use cases of each system are specified using our approach.

SOPS has four different types of systems, three of which are located above sea level and the other is located in subsea. These systems have distinct functionalities and are connected through different types of communication mediums. Due to the reason that we had no access to all the requirements of these systems, we were not able to specify the UCSs of all the systems. Only 12 out of 65 representative use cases were specified. Notice that both VCS and SOPS have eight common network abnormal use cases since both of these systems employ same type of Ethernet communication medium. VCS has two extra abnormal use cases, which are specific to video conferencing protocols, i.e., H323 and SIP.

Table 3 Characteristics RUCM models

System	Criteria*					
	1)	2)	3)	4)	5)	6)
VCS	4	40	10	10	4 human actors and 1 timer	10
SOPS	4	65	12	8	2 human actors, 4 timer, 2 external instruments, and 1 external system	13

\*1) # of systems of system; 2) # of use cases; 3) # of UCSs; 4) # of abnormal use cases, 5) # of actors; and 6) # of periodical use cases.

There was no difficulty observed when the approach was applied by the authors of the paper. The resulting UCModS were evaluated by two domain experts (one from each domain) and results show that RUCM-NDRTE is easy to apply and the readability and comprehensibility of the resulting UCMod were improved to compare with their current textual-based requirement specifications. However, in the future, we plan to conduct field studies with our industrial partners to evaluate the approach. Note that it is the first time two case studies were performed and are reported on two industrial systems, which is a very time

consuming endeavor and rarely seen in literature. It may not be very large, but it models parts of real commercial.

VI. CONCLUSION

UML use case diagrams are used to specify behavior of a system from the standpoint of external actors and are renowned to facilitate communication and common understanding between stakeholders of the system. Use case diagrams, however do not provide a template for specifying textual description of use cases. Many textual-based use case templates have been proposed in the literature for specifying detailed description of use cases. Since these templates are textual-based, and hence can be ambiguous. To reduce such ambiguity in use case specifications, we earlier proposed Restricted Use Case Modeling approach, named as RUCM. This approach is expressive and applicable enough to be used by requirement engineers as it has been empirically evaluated via two controlled experiments. In our previous works, we automatically transformed RUCM models into initial UML analysis diagrams. With our experience of working with industrial systems in communication, and maritime and energy domains, we encountered a need to extend RUCM. This extension to RUCM for large-scale, network-based, distributed, and real-time embedded systems is presented in this paper and is termed as RUCM-NDRTE. We evaluated the applicability of RUCM-NDRTE with two industrial case studies and results show that they easy to apply. In the future, we plan to conduct controlled experiments to further evaluate RUCM-NDRTE.

REFERENCES

- [1] T. Yue, L. C. Briand, and Y. Labiche, "A Use Case Modeling Approach to Facilitate the Transition Towards Analysis Models: Concepts and Empirical Evaluation," in *MODELS2009*, 2009, pp. 484-498.
- [2] T. Yue, L. C. Briand, and Y. Labiche, "A systematic review of transformation approaches between user requirements and analysis models," *Requirements Engineering*, vol. 16, 2011.
- [3] T. Yue, L. Briand, and Y. Labiche, "Facilitating the Transition from Use Case Models to Analysis Models: Approach and Experiments," *Accepted for publication in Transactions on Software Engineering and Methodology (TOSEM)*, 2011.
- [4] T. Yue, L. C. Briand, and Y. Labiche, "Automatically Deriving a UML Analysis Model from a Use Case Model," Simula Research Laboratory, Technical Report 2010-152010.
- [5] T. Yue, L. C. Briand, and Y. Labiche, "An Automated Approach to Transform Use Cases into Activity Diagrams," in *6th ECMFA*, Paris, France, 2010, pp. 337-353.
- [6] T. Yue, S. Ali, and L. Briand, "Automated Transition from Use Cases to UML State Machines to Support State-based Testing," in *7th ECMFA*, Birmingham, UK, 2011.
- [7] "The UML MARTE profile, <http://www.omgmarTE.org/>," ed.
- [8] I. Jacobson, M. Christerson, P. Jonsson, and G. Overgaard, *Object-oriented software engineering: a use case driven approach*: Addison-Wesley, 1992.
- [9] C. Larman, *Applying UML and Patterns - An Introduction to Object-Oriented Analysis and Design and Iterative Development*, 3<sup>rd</sup> ed.: Prentice-Hall, 2004.
- [10] A. Cockburn, *Writing effective use cases*: Addison-Wesley Boston, 2001.
- [11] D. Kulak and E. Guiney, *Use cases: requirements in context*: ACM Press, 2000.
- [12] P. Kruchten, *The Rational Unified Process: An Introduction*: Addison-Wesley, 2003.
- [13] OMG, "UML 2.2 Superstructure Specification (formal/2009-02-04)."

