

Extending VUML with Behavioral Modeling: A UML-Based Approach for Multi-View Object Specification

Ouali-Alami Chaimae, El Bdouri Abdelali, Lakhriissi Younes

SIGER Laboratory, Sidi Mohamed Ben Abdellah University, Fez, Morocco

chaimae.oualialami@usmba.ac.ma

Abstract. The VUML (View-based UML) profile makes it possible to create unified, multiview models for complex systems. Nevertheless, the work already done focuses mostly on structural issues and does not fully cover behavioral concerns. With an emphasis on multi-view objects, this study adds capabilities to the VUML profile to represent system behavior. We suggest modifying UML state machines to express view object behaviors and how they combine to form global multi-view object behaviors. Our method allows for the independent construction of view behaviors, followed by a methodical composition process. The outcomes demonstrate how our approach improves VUML's capacity to represent intricate, multi-view system behaviors while preserving concern separation. The work being done here helps to close the gap that exists between structural and behavioral modeling in multi-view systems, which has implications for developing complex software systems using model-driven development.

Keywords: Model composition, Behavioural, Fusion, Modeling, UML, VUML.

1. Introduction

In today's ever-changing technological landscape, the development of complex IT systems (El Hamlaoui et al., 2021), such as embedded systems or large-scale enterprise software, presents significant challenges. These systems must integrate a multitude of structural components and dynamic behaviors, making it difficult to create global models that capture all aspects simultaneously. Instead, the application is decomposed into several partial models, reducing the system's size and complexity. A compositional phase is then initiated to attain the final version of the application.

This paper addresses the challenge of behavioral specification within the VUML profile framework. The focus is on describing the individual behavior of multi-view objects, each comprising view objects encapsulating actor-specific data. This involves addressing two critical aspects: specifying the behaviors of view objects and composing these behaviors to form the global behavior of multi-view objects (Kriouile, 1995).

Complexity stems from the separate development of views in VUML and their subsequent composition to create multi-view object behaviors (Anwar, 2009). The proposed approach aims to strike a balance by allowing maximum freedom in view of development while providing means to facilitate fusion.

The View-based UML (VUML) profile has been developed to meet the need for a unified modeling approach that considers multiple perspectives within complex systems. Although VUML excels in structural modeling, including the decomposition of systems into manageable views, it has significant limitations regarding behavioral modeling, for the specification and integration of multi-view object behaviors.

This article seeks to fill a critical gap in the VUML framework: the effective specification and integration of multi-view object behaviors. The central question of this research is: how can VUML be improved to better support behavioral modeling, thus enabling a more comprehensive representation of complex systems? Our goal is to develop a method that allows the independent development of visual behaviors while ensuring their harmonious composition into a global behavior.

To address this issue, we suggest reusing UML mechanisms (Ober et al., 2006) for specifying behavior and communication between UML objects (Ouali-Alami et al., 2022). The approach involves using standard UML mechanisms to determine the behavior of VUML view objects and communication between views. This includes specifying the behavior of objects through state machines that communicate via signal exchanges or method calls. The technique is based on a separate description, followed by coordinating the state machines of view objects and the base.

Our approach utilizes existing UML mechanisms; state machines, to define and coordinate the behavior of the visible objects within the VUML framework. In doing so, we enable developers to independently design the behaviors of each view, which can then be systematically compiled into a global behavior model. This method not only preserves modularity and separation of concerns (Horman, 2004); but also, significantly reduces the complexity associated with integrating behavioral models into complex systems. The effectiveness of this approach is demonstrated through a detailed case study of the management of an automotive repair agency, showing tangible improvements in the system's behavioral modeling capabilities.

The document unfolds as follows: Section 2 presents some principles and definitions. Section 3 briefly explains viewpoint modeling based on the VUML approach. Section 4 provides an implementation where our approach will be applied. In this section, we delve into a current overall structural and

behavioral analysis of our case study.

2. Structure of a multi-view class

A multi-view class typically consists of a base class and a set of view classes. The base class includes structural and behavioral properties that are shared among all views. On the other hand, each view class represents a specific viewpoint or concern and contains attributes and behaviors specific to that view (Nassar, 2005). The views are extensions of the base class, allowing for the segregation of concerns and fine-grained access rights management at the class level. This structure enables the creation of a comprehensive and shareable model accessible from different viewpoints in a multi-view modeling approach.

The stereotyped data classes, labeled "base" and "view," portray the static aspects of the system (Lakhrissi et al., 2008; El Marzouki et al., 2017). Conversely, the dynamic behavior is illustrated by the state machines, denoted as "machine-base" and "machine-view," which are linked to these classes (Fig. 1).

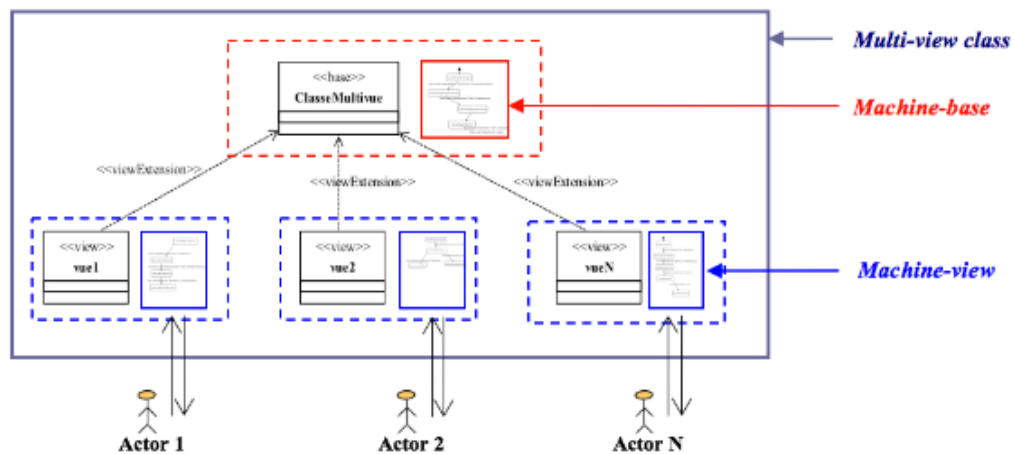


Fig. 1: Structure of a multi-view class.

In the structure and behavior of a multi-view class, the terms "machine base" and "machine view" typically refer to state machines associated with the "base" and "view" classes (Dávid, 2016; Kühne, 2022), respectively. Here's a breakdown of their roles:

- Machine Base:

Role: The "machine base" represents the behavior of the base class within a multi-view class.

Functionality: It specifies the dynamic, transitions, and state changes that are specific to the base class (Hannousse, 2019).

Focus: The "machine base" is concerned with the behavior that is common to all views and actors associated with the base class.

Scope: It captures the core behavior shared among different perspectives or viewpoints.

- Machine View:

Role: The "machine view" is responsible for representing the behavior specific to a particular view or actor within the multi-view class.

Functionality: It details the dynamic aspects, transitions, and state changes that are unique to the view or actor associated with a specific perspective (Hannousse, 2019).

Focus: The "machine view" concentrates on the behavior that differentiates one view from another within the multi-view class.

Scope: It captures the specialized behavior tailored to the requirements of a particular viewpoint.

In summary, the "machine base" encompasses behavior shared across all views, providing a foundation for common functionality, while the "machine view" accounts for variations in behavior specific to individual perspectives or actors within the multi-view class.

For instance, in Figure 2, the state of the car "CarRepairing" (located at the center of the diagram) represents a multi-view state, with varied interpretations based on the actor type. A mechanic's perspective centers around faults, repairs, required tools, and spare parts. On the other hand, a workshop manager views the repair from a logistics standpoint, emphasizing track assignments, equipment reservations, and spare parts allocation. For a client, technical repair details are less relevant, and their interest lies in the repair contract specifics, associated costs, and the repair completion date. Meanwhile, the agency manager's concerns revolve around the financial aspects of the repair, encompassing actual costs, estimated completion times, and contract arrangements with the client.

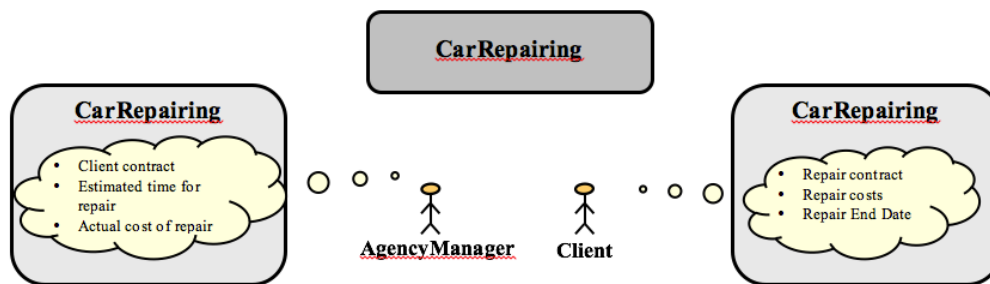


Fig. 2: Multi-View machine - Example of the Car Repairing

3. VUML Approach

Existing work on VUML (View Unified Modeling Language) primarily focuses on extending classical UML to enable modeling from different perspectives, particularly in complex systems. However, these approaches have certain limitations, such as the difficulty in maintaining consistency among the multiple views, the lack of a standardized framework for integrating behaviors, and increased complexity when implementing models at a large scale. In parallel, other behavioral modeling approaches in UML have been developed, particularly using state diagrams, sequence diagrams, and activity diagrams to represent the interactions and transitions of the system. Extensions like UML-RT or SysML have also emerged to meet the needs of real-time systems and complex systems. However, these methods are often limited by their ability to effectively manage concurrent and distributed behaviors, especially in a multi-perspective context. Another major challenge lies in composing behaviors from multiple viewpoints, which raises issues of coherence and integration of perspectives. It is difficult to ensure correct synchronization of behaviors arising from different views while avoiding conflicts or redundancies. The management of model granularity and the coordination of various levels of abstraction further complicate the task, raising issues of scalability and traceability of the modeled systems.

The VUML (View-Based Unified Modeling Language) approach is a modeling methodology that emphasizes the creation and management of multiple views to capture different aspects of a complex

system. It is an extension of the Unified Modeling Language (UML) tailored to address the challenges of modeling large and intricate systems. Here are key aspects of the VUML approach:

- **View-Based Modeling:** VUML employs a view-based modeling strategy where different views of a system are created to represent various perspectives and concerns. Each view focuses on specific aspects of the system relevant to a particular stakeholder or aspect of the system.
- **Multiview Classes:** In VUML, the notion of multiview classes is introduced. A multiview class consists of a base class (shared by all actors) and multiple view classes (extensions of the base class), each specific to a particular viewpoint. This allows for fine-grained access rights management and segregation of concerns at the class level.
- **Separation of Concerns:** VUML emphasizes the separation of concerns in system modeling. By creating distinct views for different stakeholders, concerns such as requirements, design, and behavior can be addressed independently in each view.
- **Horizontal and Vertical Modeling:** The approach combines both horizontal and vertical modeling. Horizontal modeling involves creating models at each level of abstraction, while vertical modeling aligns with the principles of Model-Driven Architecture (MDA) (Essebaa et Chantit, 2008; kim et al., 2008).
- **Event Observation:** VUML introduces the concept of event observation as a first-class object interaction method. Events and probes are formalized to handle interactions between system components. Probes can be used to observe events and work with event data.
- **Formalization of Concepts:** VUML formalizes various concepts, including the definition of multiview classes, the use of probes, and the creation of a VUML Probe Profile to support event observation.
- **Decentralized Design Process:** The design process in VUML consists of decentralized phases, including the identification of actor desires, the creation of various PIM (Platform-Independent Model) models, and a composition operation to combine independently created design models into a global VUML design model (Anwar et al., 2011; Lakhrissi, 2008b).
- **Tool Support:** The approach may include tool support to facilitate the modeling process, ensuring consistency and manageability of models and code.

Overall, the VUML approach provides a structured and systematic way to address the complexities of modeling large and diverse systems by employing multiple views and emphasizing the separation of concerns.

In simpler terms, the key concepts of VUML can be described as follows:

- **Actor:** represents a human or logical entity that interacts with the system.
- **Point of view:** Reflects an actor's perspective on the system or a part of it. Each actor is associated with a single point of view.
- **View:** A modeling entity, primarily static, that results from applying a point of view to a specific entity (like a class) or, in a broader sense, to the entire system.

The View-based UML (VUML) profile has been developed to meet the need to model complex systems by allowing a unified view from different perspectives. Previous research on VUML has mainly explored its application in structural modeling, where it excels in breaking down systems into separate and manageable views. However, although VUML offers powerful tools for structural modeling, it has notable limitations with regard to behavioral modeling. In particular, the specification and composition of multi-view object behaviors remain major challenges. Existing work has often failed to deal in depth with how individual behaviors of views can be coherently integrated to form a global behavior. This gap limits the effectiveness of VUML in capturing complex system dynamics, thus justifying the need to develop additional mechanisms to improve its ability to model complex behaviors in a multi-view framework.

4. Methodology

This methodology enhances VUML by introducing "machine-base" and "machine-view" state machines to capture the behaviors of the actors. It unfolds in three phases: identifying the needs of the stakeholders, creating PIM models corresponding to different viewpoints, and composing these models to form a unified design. The overall behavioral analysis and viewpoints ensure consistency among the perspectives of the different stakeholders. Finally, the merging of behaviors ensures a harmonious integration of interactions and workflows, creating a coherent system.

The VUML approach involves three main development phases:

Identification of Actor Desires (Global Analysis):

- The initial step focuses on recognizing the desires of the actors.
- The primary objective is to formulate a requirements model, typically represented as a UML use case diagram.

Creation of Various PIM Models:

- The second decentralized stage involves generating multiple PIM (Platform-Independent Model) models, each corresponding to a distinct viewpoint.
- These models encompass various UML diagrams such as class diagrams, state machines, and sequence diagrams.
- The output of this phase is a set of UML models that cater to the needs of different actors.

Composition Operation:

- The final stage is a composition operation, which entails combining independently created design models to formulate a comprehensive VUML design model.
- This phase is critical for achieving an integrated and holistic representation of the system, as the independently developed models are combined to create a unified design.

We propose a three-step approach: (1) global behavioral analysis, (2) view-specific behavioral analysis, and (3) behavioral composition (Ouali-Alami et al., 2022). In the first step, integrating behaviors from different perspectives into a complex system is a difficult process due to the challenges of conflict management, state synchronization, and preserving overall system coherence. Then we delineate the procedures that comprise the phase of global behavioral analysis. And analyzing and refining the state of the "base machine" according to the needs of the point of view, illustrating this process through specific interactions between the actors and the base machine, such as the client and the agency manager, before identifying of a design strategy in which the behaviors' composition is organized according to several stakeholder viewpoints, each of which focuses on a certain facet of the system's behavior (Nikiforova et al., 2016).

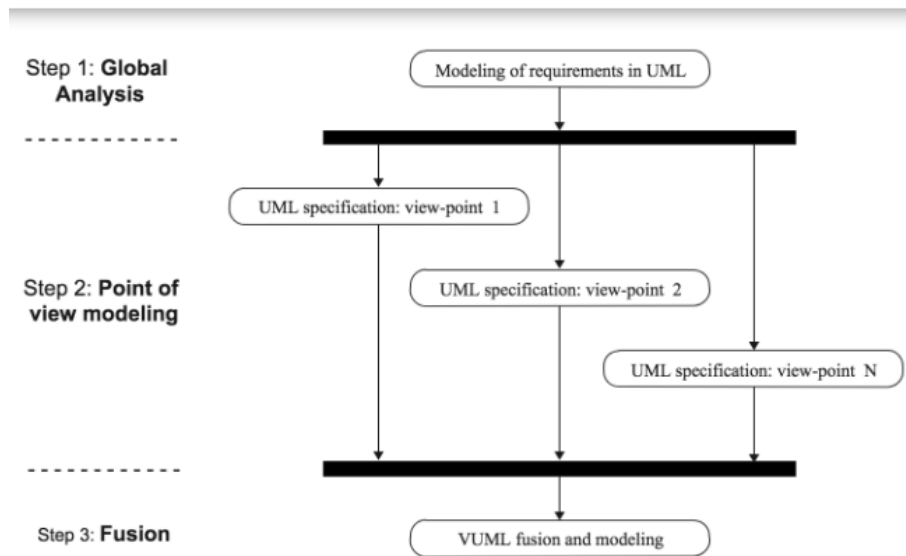


Fig. 3: Development phases of VUML Global behavioral analysis

In this subsection, we outline the adjustments implemented in the VUML approach to incorporate the behavioral aspects of the methodology.

4.1. Global behavioral analysis

Global behavioral analysis in VUML refers to the examination and understanding of the overall behavioral aspects of a system represented in the VUML modeling approach (Wong et al., 2015). It involves analyzing the interactions, dependencies, and dynamic behaviors of various components within the system to gain a comprehensive view of how the system functions as a whole. This analysis helps in identifying potential issues, ensuring consistency across different viewpoints, and refining the system's behavior to meet the specified requirements. The global behavioral analysis is an integral part of the VUML approach to ensure a thorough understanding and effective representation of both structural and behavioral aspects in the modeling process.

4.2. Behavioral analysis/design by point of view

Behavioral analysis/design by point of view in VUML (View-Based Unified Modeling Language) involves focusing on the specific behaviors and interactions of system elements from the perspective of individual actors or stakeholders. The VUML approach recognizes that different actors have distinct viewpoints, and each viewpoint is associated with a particular set of concerns and behaviors (Asteasuain et Braberman, 2017).

Here's an overview of Behavioral Analysis/Design by Point of View in VUML:

- **Actor-Centric Approach:** VUML emphasizes an actor-centric modeling approach where each actor (human or logical entity interacting with the system) has a unique point of view. This point of view defines how an actor perceives and interacts with the system.
- **Viewpoints:** A viewpoint in VUML corresponds to the modeling of a specific aspect of the system from the perspective of an actor. It includes both structural and behavioral elements, providing a holistic representation of the system for a particular actor.
- **Behavioral Analysis:** The behavioral analysis involves capturing the dynamic aspects of the

system, such as interactions, state transitions, and workflows, based on the concerns of a specific actor. This analysis ensures that the system behaves following the expectations and requirements of that actor.

- **Designing for Specific Concerns:** Behavioral design within a specific viewpoint considers the unique concerns and requirements of the associated actor. It includes the definition of states, transitions, events, and operations that are relevant to that actor's perspective.
- **Consistency Across Viewpoints:** While each actor's viewpoint is unique, VUML also emphasizes the importance of maintaining consistency across different viewpoints. This involves ensuring that the behaviors specified for each actor align cohesively to create a coherent and functional overall system.
- **Iterative Process:** Behavioral analysis and design in VUML are iterative processes. As the understanding of each actor's needs evolves, the corresponding viewpoint can be refined, and the overall system behavior can be adjusted to accommodate changes.
- **By adopting a behavioral analysis and design approach by point of view, VUML enables a more focused and actor-specific representation of system behavior. This helps in creating models that accurately reflect the intended functionality and interactions from the perspective of different stakeholders in the system.**

By adopting a behavioral analysis and design approach by point of view, VUML enables a more focused and actor-specific representation of system behavior (El Marzouki et al., 2016). This helps in creating models that accurately reflect the intended functionality and interactions from the perspective of different stakeholders in the system.

4.3. Behavioral Fusion/Synchronisation

The Composition of behaviors from multiple perspectives poses significant challenges in the modeling of complex systems (Nikiforova et al., 2017). Each view often represents a specific perspective, capturing aspects of the system, such as actors' interactions or internal processes. However, integrating these disparate behaviors into a coherent global behavior is far from trivial. Key challenges include managing conflicts between behaviors, synchronizing states between different views, and preserving overall system coherence. In addition, interview interactions can lead to unexpected emerging behaviors, making it difficult to predict and verify overall behavior. These challenges underline the importance of developing robust behavioral composition mechanisms that effectively coordinate the dynamics between different views while the integrity of the overall model (El Marzouki et al., 2017). Behavioral Fusion/Synchronization in VUML (View-Based Unified Modeling Language) refers to the process of combining and harmonizing the behavioral aspects of multiple viewpoints or actors within a system. In VUML, each actor has a unique viewpoint that captures its specific concerns and behaviors. Behavioral fusion/synchronization is crucial for ensuring that the overall system functions cohesively when considering the perspectives of different actors.

Here are key aspects of behavioral fusion/synchronization in VUML:

- **Integration of Viewpoints:** VUML recognizes that a complete system involves the collaboration of multiple actors, each with its viewpoint. Behavioral fusion involves integrating the behavioral models associated with each actor's viewpoint into a unified representation of system behavior.
- **Coherence and Consistency:** The goal of behavioral fusion/synchronization is to achieve coherence and consistency across different viewpoints. It ensures that the behaviors specified for each actor complement each other and collectively contribute to the intended system functionality.
- **State and Event Alignment:** Fusion involves aligning the states and events defined in different viewpoints. This ensures that the system transitions seamlessly between states based on the

interactions and events triggered by different actors.

- **Workflow Alignment:** Fusion addresses the alignment of workflows and sequences of activities across different actors. It ensures that the overall system workflow is coherent and aligns with the expectations of all stakeholders.
- **Conflict Resolution:** In cases where conflicts arise between behaviors specified in different viewpoints, synchronization aims to resolve these conflicts. This may involve negotiation, compromise, or the introduction of additional mechanisms to reconcile conflicting requirements.
- **Iterative Process:** Like other aspects of VUML, behavioral fusion/synchronization is an iterative process. As the understanding of system requirements and actor viewpoints evolves, the behavioral models may need to be adjusted and synchronized to reflect the most up-to-date specifications.
- **Fusion at Different Levels:** Behavioral fusion can occur at different levels, ranging from the synchronization of high-level system behaviors to the alignment of detailed state transitions and interactions between specific elements in the system.

By incorporating behavioral fusion/synchronization into the VUML approach, the modeling process aims to create a comprehensive and integrated representation of system behavior. This ensures that the system, as perceived from the diverse viewpoints of different actors, functions harmoniously and meets the overall objectives and requirements of the stakeholders.

As previously noted, the state machines proposed to depict the behavior of views adhere to the UML standard (Lakhrissi et al., 2007; Lakhrissi et al., 2008c). While UML provides diverse concepts for specifying behavior, their semantics often remain insufficient or vague. The drive to make UML a universal modeling standard capable of analyzing/designing any domain has resulted in ambiguous and nonspecific semantics. The syntax we employ for expressing transitions in the developed state machines is derived from the Omega UML profile. This profile offers relatively rich semantics, providing a set of mechanisms specific to communication and execution aspects.

The use of the UML Omega profile and the IFx tool is fully justified in the context of the behavioral modeling of complex systems, particularly those requiring rigorous and formal verification.

The UML Omega profile is an extension of UML designed to model real-time, distributed, or critical systems, providing mechanisms to specify complex behaviors with formal rigor. It stands out for its ability to capture temporal, concurrent, and synchronous aspects, making it particularly suitable for modeling systems where timing accuracy and concurrency management are essential. This not only allows for a better representation of the systems in question but also integrates formal verification techniques in the early design phases, thereby limiting costly errors during the development stage.

The IFx tool, for its part, is used to perform formal verification of models specified in UML Omega. It allows for the simulation, verification, and analysis of the properties of complex systems through rigorous formal semantics. By utilizing techniques such as model checking, IFx helps to detect potential errors, such as unwanted race conditions, violations of timing constraints, or deadlock situations, even before the deployment phase.

Thus, in this context, the UML Omega profile ensures accurate and formal modeling of complex behaviors, while IFx provides the necessary tools for validating these models, ensuring the robustness and reliability of the modeled systems.

5. Results - Case study

In this section, we demonstrate the application of the afore aforementioned to our case study.

The results of applying the proposed approach to our case study reveal several significant advancements. The global structural analysis has made it possible to model the needs of the various stakeholders by organizing them into functional units represented by use cases. (e.g., car management). The global behavioral analysis identified reactive multi-view classes and detailed their lifecycle (e.g., the states of the Car class, from diagnosis to repair). Then, the behavioral design phase from the perspective of each actor (e.g., Client, Agency Manager) refined these states. Finally, the behavioral composition has allowed for the integration of these multi-view behaviors while ensuring coherence and a smooth transition between states. Although this approach has improved clarity and precision in modeling complex behaviors, it has introduced increased complexity in coordinating signals between the states of different machines.

5.1. Global Structural analysis

The initial phase of the process involves global analysis, serving as a centralized stage for requirements modeling. The objective is to identify the needs of various actors and organize them into functional units, presented as use cases. Figure 4 provides an overall depiction of the system's functionalities: (1) car management, (2) personnel management, (3) material management, (4) financial management, and (5) agency oversight. Each feature is further detailed into manageable functional units, indicating the responsibilities assigned to each actor. Our study focuses on specific actors: the client, agency manager, workshop manager, and maintainer.

For clarity, we will elaborate on the "cars management" functionality as an illustrative example (Fig. 4). The management of the agency's cars encompasses tasks such as handling registration, overseeing expertise and repairs for each vehicle, and conducting final verification tests. In the "save" use case, the client provides information about their car, collaborates with the agency manager in executing contracts for expertise and repair, and validates the repair by conducting a final functional test on their vehicle. Maintainers play a role in technical phases like expertise, repair, and testing, while the workshop manager is involved in overseeing maintenance tasks for the cars.

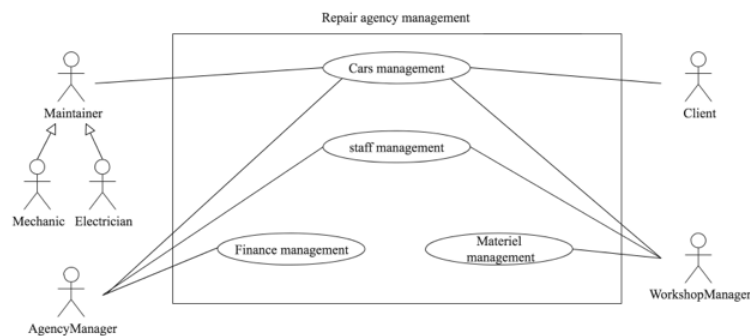


Fig. 4: Repair Agency Management" use case.

5.2. Global behavioral analysis

In this section, we outline the steps involved in the global behavioral analysis phase.

- In this phase, we **identify reactive multi-view classes** based on the comprehensive structural analysis of the case study. The detailed examination of use cases and user requirements, through tools like sequence and activity diagrams, aids in pinpointing classes with potential multi-view and reactive behavior. In our case study, the classes identified as multi-view include Car, Expertise, Breakdown, Repair, and Contract. For simplicity, we attribute reactive behavior specifically to the Car class, while categorizing others as data classes (static). It's important to note that this list of multi-view classes and their classification as static or reactive may evolve

in the subsequent phase of the approach during the detailed analysis by point of view.

- In this step, we **identify potential states for each reactive multi-view class** by considering their entire life cycle. Taking the Car class as an example, the life cycle involves several stages: the car breaks down, the owner initiates a contract with the agency, and provides the necessary information for car registration. The repair process at the agency begins with bringing the car to the garage, followed by mechanical and electrical expertise to identify faults. Based on the expertise results, a repair contract is established, and maintainers address the detected defects. The final step involves testing to ensure the car functions correctly. This nominal life cycle encompasses the various possible states of a car in the garage.

We summarize these states as follows:

- OutOfOrders: the initial condition of the car.
- InGarageRouting: the state that represents the process of bringing the car from its breakdown location to the garage.
- InExpertiseProcedure: the state representing the expertise operation on the car to detect faults.
- InRepairProcedure: after fault detection, the car enters the repair procedure, consisting of two crucial parts: (i) negotiating the repair contract based on the expert assessment results, and (ii) the technical repair, representing the action of fixing the faults detected in the assessment phase.
- InTest: the final step before leaving the garage, involves testing to ensure the proper functioning of the car.
- Exit: the state representing the end of the car's life cycle in the garage.

Figure 5 illustrates the base-state machine of the Car class. This machine encapsulates the life cycle of an object within the system, providing a logical temporal sequence of its states. Subsequently, this machine is incorporated into the glossary for shared and reusable purposes in the second phase of the process by the designers.

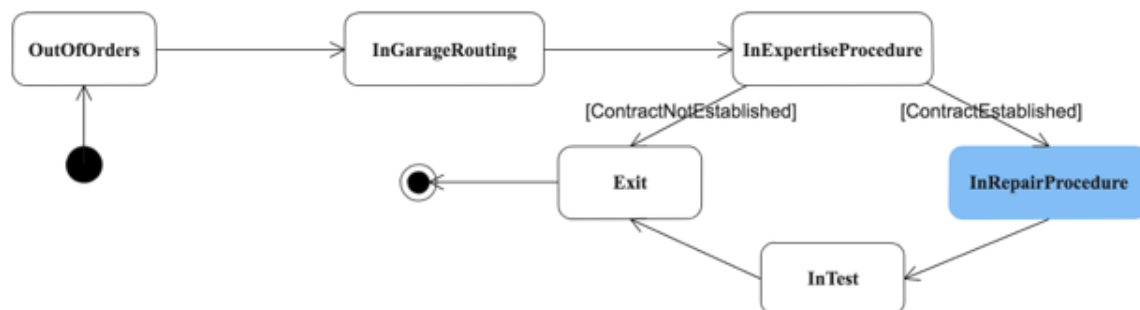


Fig. 5: "machine-base" of Car Class

5.3. Analysis/Design of Behavior by Point of View

After constructing the structural model for a specific viewpoint (cf. section 4.2.1), the phase of behavioral specification is initiated based on that viewpoint. It involves a single-view analysis of the "machine-base" states identified in the initial phase. Each base state can lead to a sub-machine, refining and specializing it according to the requirements of the specific viewpoint. We opt to limit the illustration using our example:

- A segment of the "machine-base." The state we will elaborate on is the InRepairProcedure state, during which the car undergoes a series of actions to restore its normal operating state.
- This is presented to two actors: the client and the agency manager.

Client Perspective: Derived from the class diagram crafted for the Client Perspective (refer to section 4.2.1), specifically focusing on the repair and contract negotiation process, we extract the portion of the SM_Client_Car state machine related to the InRepairProcedure state (Fig. 6).

The transition in the SM_Client_Car view from the ExpertiseProcedure state to the RepairProcedure state is activated by the Repair() signal requested by the client actor. However, during this step of the process, the analysis/design by point of view is conducted in a decentralized manner. Consequently, the SM_Client_Car state machine relays this signal to the base machine without concerning itself with which objects will receive the signal. The same principle applies when an actor anticipates a signal from another entity. In such cases, the developer from this viewpoint assumes that the signal originates from the "machine-base." This is exemplified by the form(opt) signal, which initiates the transition between the WaitFormAgency state and the WaitFillFormClient state in Fig. 7.

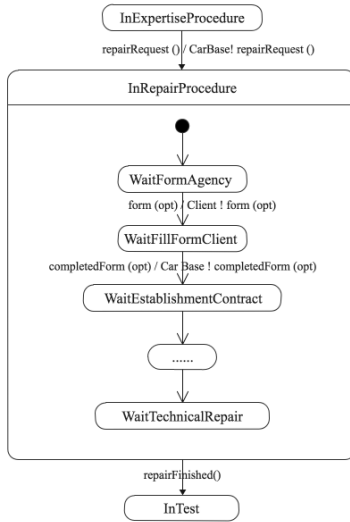


Fig. 6: Refinement of the InRepairProcedure State for the Client Viewpoint

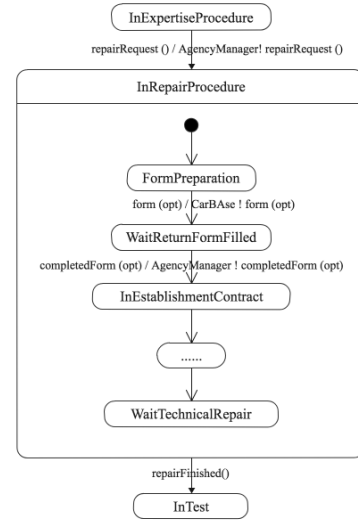


Fig. 7: Enhancement of the InRepairProcedure state for the AgencyManager Perspective

The perspective of the Agency Manager involves enhancing the InRepairProcedure state, as illustrated in Figure 7. This enhancement follows the same principle of centralizing communications within the machine base. Specifically, when the SM_AgencyManager_Car receives the signalRepair() from the "machine-base," it relays the signal to the agency manager actor. Subsequently, the SM_AgencyManager_Car compiles a form that includes a list of failures for repair and the associated options for addressing each failure. Once the form is prepared, the agency manager dispatches it to the SM_AgencyManager_Car, triggering a transition to the WaitReturnFormCompleted state upon receipt of the form(opt) signal.

5.4. Behavioral Composition

The term "behavioral composition model by point of view" isn't a standard phrase, and its meaning might depend on the context in which it's used. However, I can offer an interpretation based on common concepts in software engineering.

In the context of software design or system architecture, "behavioral composition by point of view" could refer to a method where the composition of behaviors is guided or structured based on different perspectives or viewpoints. Here's a breakdown:

Behavioral Composition: This generally involves combining the behaviors of individual components,

objects, or modules to create a larger system behavior. It's about how different pieces of functionality come together to form a coherent and functional whole.

Point of View: In the context of software engineering, a "point of view" often refers to a particular perspective or set of concerns related to a system. Different stakeholders or participants in the development process may have distinct viewpoints, such as end-users, developers, system administrators, etc.

Bringing these concepts together, "behavioral composition model by point of view" could imply a design or modeling approach where the composition of behaviors is structured or organized according to different perspectives or viewpoints. Each viewpoint might focus on specific aspects of the system's behavior, and the behavioral composition is done in a way that aligns with these different perspectives.

This could be relevant in situations where a complex system needs to be understood, developed, or maintained by different teams or individuals with specialized concerns. The design might involve composing behaviors from the viewpoint of each stakeholder or participant in the development process.

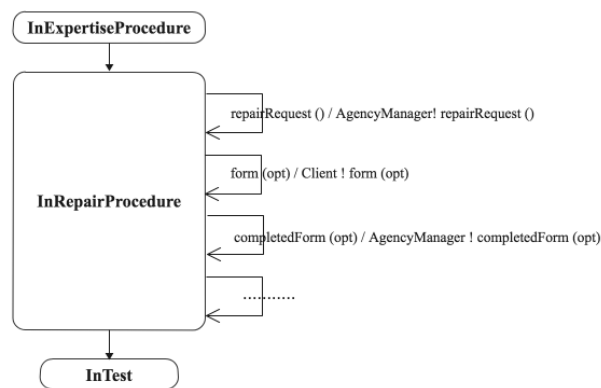


Fig.8: Adding Synchronization Messages in the Machine Base for the InRepairProcedure State (Extract).

Compared to existing VUML approaches, our method has made it possible to model more accurately the complex behaviors of multi-view objects in the way state transitions were managed between different views.

The proposed approach has shown significant advantages in terms of clarity and consistency in multi-view behavior modeling. However, it also has limitations, including increased complexity in the coordination of signals between the different states of the machine.

Compared to existing VUML approaches, the proposed method shows significant improvements in terms of accurately modeling the complex behaviors of multi-view objects. While traditional VUML approaches primarily focused on the separate structural and behavioral modeling of viewpoints, our method allows for a more nuanced management of state transitions between different views, thereby providing better clarity and coherence in the composition of behaviors. However, this increased precision also introduces an additional complexity, particularly in coordinating signals between the different states of the machines, which can make the process more challenging to manage than in traditional VUML approaches.

This work has important implications for software engineering and model-driven development. (MDD). By integrating a multi-view approach for modeling complex behaviors, it promotes better management of complexity in software systems. This method allows for a more coherent and detailed representation of the interactions between the different actors, which enhances the modularity and reusability of the models. Furthermore, by facilitating the synchronization of states and behaviors in multi-view systems, this approach contributes to the creation of more robust and adaptable software, a crucial asset for large-scale projects where multiple perspectives need to be harmonized.

6. Conclusions

This article makes a notable contribution to the VUML approach by proposing a systematic method for the specification and composition of multi-view object behaviors. This advance improves complex systems modeling by effectively integrating behavioral aspects while preserving the separation of concerns. The implications of this method for VUML and behavioral modeling open new perspectives in the field for the development of complex software systems. For future research, it will be crucial to address the challenges of scalability concerning large-scale system management and to explore solutions to further automate behavioral composition.

This method allows a clear separation of behavior development by point of view while providing effective mechanisms for their composition into coherent behaviors of multi-view objects.

This research expands the capabilities of VUML by introducing rigorous mechanisms for behavioral modeling, offering a new path for managing complex systems with multiple perspectives.

As just explained, the initial approach relies exclusively on UML concepts. The primary outcome of this approach involves identifying, for a given multi-view class, two distinct types of specialized machinery associated with either a "base" class or a "view" class. These state machines, aligned with the UML2.0 specification, serve the purpose of capturing the dynamic behavior exhibited by instances of the considered multi-view class.

A "machine-view" illustrates the life cycle of an object-view, while a "machine-base" is designed to establish coherence and coordination among the "machines-view" and articulate the collective behavior of the actors involved. The combination of a "machine-base" and its dependent "machines-view" is referred to as a multi-view machine.

This approach addresses the issue by advocating for independent development in the second phase. To facilitate this, the second phase needed to be guided by a pre-established model, leading to the introduction of the "machine-base" concept. The "machine-base" is crafted during the global modeling phase to articulate the collective behavior among actors. This behavior is regarded as a template to adhere to when formulating behaviors from various perspectives.

In the decentralized modeling phase, the machines-view linked to the object-view are delineated separately, aligning with the state of the "machine-base." Subsequently, in the fusion phase, the amalgamation of partial behaviors from different viewpoints is accomplished by introducing signal exchanges. These exchanges enable the coordination of machines with the states of objects-view and object-base.

Nevertheless, this UML-based approach encounters limitations as it scales up, particularly in two key aspects:

-Behavior Specification Challenges: Interconnected Views: Difficulties arise in ensuring the

independence of view development due to intricate connections between them. This interdependence poses a risk during scaling, making it challenging to implement the approach without modifying the development of other views to gather necessary information.

Early Identification: The necessity to identify multi-view classes and their "machine-bases" early in the global analysis phase becomes problematic. Creating a state machine covering the life cycle of an object requires a thorough study of multi-view object use cases, which, for large systems, becomes impractical.

-Behavior Composition Challenges: Integration Complexity: Integrating separately developed vision machines may necessitate numerous modifications and adaptations at both the vision machine and base machine levels. During scaling, significant efforts are required to maintain the coherence of the entire system during the compositional process.

Although our approach fills an important gap in VUML, it also reveals challenges related to its application to large-scale systems and the management of increased complexity. In the future, work should focus on developing tools to automate certain parts of the composition process, as well as exploring techniques to better manage complexity in large-scale multi-view behavioral models. Furthermore, it would be relevant to study the integration of this approach with other model-driven development methodologies in order to enhance its effectiveness and applicability.

References

- Anwar, A. (2009). *Formalisation par une approche IDM de la composition de modèles dans le profil VUML* (Doctoral dissertation, Thèse de doctorat, Université de Toulouse).
- Anwar, A., Dkaki, T., Ebersold, S., Coulette, B., & Nassar, M. (2011, April). A formal approach to model composition applied to VUML. In 2011 16th IEEE International Conference on Engineering of Complex Computer Systems (pp. 188-197). IEEE.
- Asteasuain, F., & Braberman, V. (2017). Declaratively building behavior by means of scenario clauses. *Requirements Engineering*, 22(2), 239-274.
- Dávid, I. (2016). A multi-paradigm modeling foundation for collaborative multi-view model/system development. In *Proceedings of the ACM Student Research Competition at MODELS 2016*, October 3-4, 2016, St. Malo, France/Gray, Jeff [edit.]; et al.
- El Hamlaoui, M., Ebersold, S., Bennani, S., Anwar, A., Dkaki, T., Nassar, M., & Coulette, B. (2021). A Model-Driven Approach to align Heterogeneous Models of a Complex System. *The Journal of Object Technology*, 20(2), 1-24.
- El Marzouki, N., Lakhri, Y., Nikiforova, O., EL MOHAJIR, M. O. H. A. M. M. E. D., & Gusarova, K. (2017). Behavioral and Structural Model Composition Techniques: State of Art and Research Directions. *Transactions on Computers*, WSEAS, 16, 39-50.
- El Marzouki, N., Nikiforova, O., Lakhri, Y., & El Mohajir, M. (2016). Enhancing conflict resolution mechanism for automatic model composition. *Applied Computer Systems*, 19(1), 44-52.
- El Marzouki, N., Nikiforova, O., Lakhri, Y., & El Mohajir, M. (2017). Toward a generic metamodel for model composition using transformation. *Procedia Computer Science*, 104, 564-571.
- El Marzouki, N., Lakhri, Y., Nikiforova, O., & El Mohajir, M. (2017, April). The application of an automatic model composition prototype on the-Two hemisphere model driven approach. In 2017 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS) (pp. 1-6). IEEE.

- Essebaa, I., & Chantit, S. (2017). QVT transformation rules to get PIM model from CIM model. In *Europe and MENA Cooperation Advances in Information and Communication Technologies* (pp. 195-207). Springer, Cham.
- Hannousse, A. (2019). Dealing with crosscutting and dynamic features in component software using aspect-orientation: requirements and experiences. *IET Software*, 13(5), 434-446.
- Horman, Yoav, and Gal A. Kaminka. "Improving Sequence Learning for Modeling Other Agents." *Proceedings of the AAMAS 2004 Workshop on Learning and Evolution in Agent-Based Systems*. 2004.
- Kim, W. Y., Son, H. S., Park, Y. B., Park, B. H., Carlson, C. R., & Kim, R. Y. C. (2008). Automatic MDA (model driven architecture) transformations for heterogeneous embedded systems.
- Kühne, T. (2022). Multi-dimensional multi-level modeling. *Software and Systems Modeling*, 21(2), 543-559.
- Kriouile, A. (1995). VBOOM, une méthode orientée objet d'analyse et de conception par points de vue. Unpublished doctoral dissertation, University Mohammed V, Rabat, Morocco.
- Lakhrissi, Y., Nassar, B. C. I. O. M., & Kriouile, A. Démarche VUML Statique et Dynamique.
- Lakhrissi, Y., Ober, I., Coulette, B., Nassar, M., & Kriouile, A. (2008). Prise en compte des aspects comportementaux dans la démarche de modélisation de VUML. In *ERTSI, associé à la conférence INFORSID, Fontainebleau* (Vol. 27, No. 05, pp. 2008-27).
- Lakhrissi, Y., Ober, I., Coulette, B., Nassar, M., & Kriouile, A. (2007, July). Vers la notion de machine à états multivue dans le profil VUML. In *Workshop WOTIC* (Vol. 5, No. 07, pp. 2007-06).
- Lakhrissi, Y., Anwar, A., Nassar, M., & Kriouile, A. (2008). Composition des machines à états par point de vue dans VUML. In *Workshop JIMD* (Vol. 3, No. 07, pp. 2008-05).
- Nassar, M. (2005). Analyse/conception par points de vue: le profil VUML (Doctoral dissertation).
- Nikiforova, O., El Marzouki, N., Gusarovs, K., Vangheluwe, H., Bures, T., Al-Ali, R., ... & Leon, F. (2017). The two-hemisphere modelling approach to the composition of cyber-physical systems. In *Proceedings of the 12th International Conference on Software Technologies* (pp. 286-293).
- Nikiforova, O., El Marzouki, N., Kunicina, N., Vangheluwe, H., Florin, L., & Iacono, M. (2016). Several Issues on Composition of Cyber-Physical Systems Based on Principles of the Two-Hemisphere Modelling. In *Proceedings of the 4th Workshop of the MPM4CPS COST Action* (pp. 44-55).
- Ober, I., Graf, S., & Ober, I. (2006). Validating timed UML models by simulation and verification. *International Journal on Software Tools for Technology Transfer*, 8(2), 128-145.
- Ouali-Alami, C., El Bdouri, A., Elmarzouki, N., & Lakhrissi, Y. (2022). View-based Modelling: Behaviour Specification based on UML Concept.
- Wong, P. Y., Bubel, R., de Boer, F. S., Gómez-Zamalloa, M., De Gouw, S., Hähnle, R., ... & Sindhu, M. A. (2015). Testing abstract behavioral specifications. *International Journal on Software Tools for Technology Transfer*, 17(1), 107-119.