ISSN 1846-6168 (Print), ISSN 1848-5588 (Online)
https://doi.org/10.31803/tg-20241106104357

Original scientific paper
Received: 2024-11-06, Accepted: 2024-11-25

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

Chaimae Ouali-Alami*, Abdelali El Bdouri, Younes Lakhrissi

**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:** behavioural; fusion; model composition; modeling; UML; VUML

## 1 INTRODUCTION

In today's ever-changing technological landscape, the development of complex IT systems [1], 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 system's size and complexity are decreased by breaking the application down into many component models. A compositional phase is then initiated to create the application's final release.

The problem of behavioral specification in the context of the VUML profile framework is discussed in this study. The main goal is to describe the distinct behavior of multi-view objects, each of which is made up of view objects that contain data relevant to an actor. 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 [2].

Complexity stems from the separate development of views in VUML and their subsequent composition to create multi-view object behaviors [3]. 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 paper aims to close a significant hole 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 [4] for specifying behavior and communication between UML objects [5]. The method makes use of common UML methods to control how VUML view objects behave and how views communicate with one another. This involves using state machines that interact with one another through method calls or signal exchanges to indicate how objects should behave. The method is based on a distinct description, after which the base and view objects' state machines are coordinated.

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 [6]; 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 modeling of complex systems has seen considerable growth, with frameworks like SysML and UML-RT being widely used. SysML, an extension of UML (Unified Modeling Language), is specifically designed for modeling complex engineering systems, such as those encountered in automotive, aerospace, or cyber-physical systems. Although SysML is effective in capturing requirements and structuring systems, it shows its limitations in managing dynamic behaviors and synchronizing different views. In particular, SysML may lack flexibility when behaviors need to be modeled through evolving scenarios.

Moreover, UML-RT, a variant of UML adapted for real-time systems, excels in contexts requiring rapid responses, such as telecommunications, medical devices, or embedded systems. However, UML-RT presents difficulties in synchronizing states and transitions between its capsules, especially for complex multi-view systems.

The extended VUML approach proposed in this article stands out for its ability to model modularly and integrate multi-view behaviors coherently. This method offers greater flexibility and reduces synchronization conflicts, thus overcoming the limitations encountered by SysML and UML-RT in modeling complex systems.

Behavioral modeling allows for better management of complex systems by clearly defining the dynamic interactions between components. This improves flexibility by allowing different parts to evolve independently, while ensuring their smooth integration into the overall system. By anticipating and representing possible behaviors, this modeling facilitates the predictability of the system's responses to various scenarios, thereby reducing the risks of errors and incompatibilities. It also allows for simulating and testing behaviors before implementation, increasing the robustness and responsiveness of the system.

for example, in the field of Cyber-Physical Systems; In applications such as smart electrical grids, the absence of behavioral modeling can prevent the correct simulation of interactions between the grid and IoT devices. This could lead to errors in load management or risks of overloads, thereby disrupting the overall system.

The following is how the document develops: A few definitions and concepts are presented in Section 2. Viewpoint modeling based on the VUML technique is briefly explained in Section 3. Our method will be used in the implementation given in Section 4. We examine our case study's current general structural and behavioral analysis in this part.

## 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 [7]. 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.

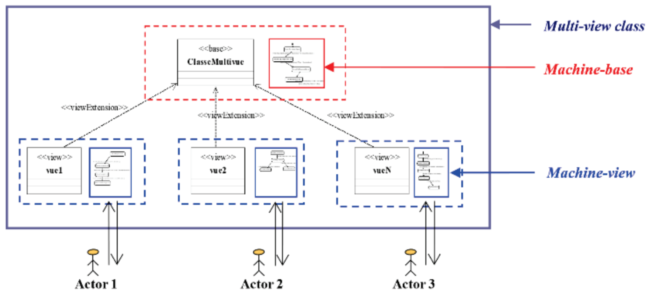

**Figure 1** Structure of a multi-view class

The stereotyped data classes, labeled "base" and "view," portray the static aspects of the system [8, 9]. 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).

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 [10, 11], 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 [12].
  - 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 [12].
  - 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.

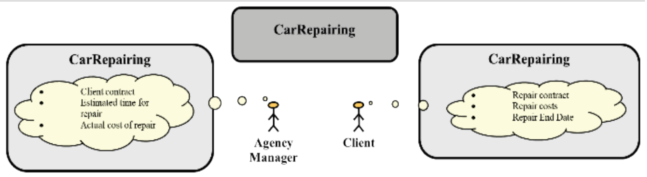

**Figure 2** Multi-View machine - Example of the Car Repairing

For example, the car's "CarRepairing" state (in the diagram's center) in Fig. 2 depicts 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.

## 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) [13, 14].
- 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 [15].
- 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 following is a description of the main ideas of VUML:

- Actor: symbolizes a logical or human being interacting with the system.
- Point of view: represents the viewpoint of an actor on the entire structure or a portion of it. A single point of view is linked to every actor.
- 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.

Three primary stages of development are included in the VUML approach:

**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 [16] (Tab. 1).

**Table 1** Methodology of Behavioral Analysis and Design

| Phase | Sub-steps | Objectives | Tools and representations |
|---|---|---|---|
| Global Behavioral Analysis | - Identify the needs of the stakeholders (analysis of the desires of the stakeholders)<br>- Create a needs model in the form of a use case diagram<br>- Specify the overall behavior using state machines | - Understand the overall behavior of the system<br>- Ensure a coherent basis for specific viewpoints | - UML use case diagrams<br>- UML state machines compliant with the UML Omega standard |
| Analysis/Design by Viewpoint | - Generate PIM models for each viewpoint<br>- Include UML diagrams: classes, state machines, sequences<br>- Refine the behaviors specific to each actor (design to each viewpoint ) | - Capture the specific behaviors of each actor<br>- Reflect the particular needs in the form of separate models | - UML diagrams (classes, sequences, state machines)<br>- UML Omega for complex transitions |
| Merging/Synchronization of Behaviors | - Identify and resolve conflicts between viewpoints<br>- Align the states and transitions between the actors<br>- Create a global model integrating all viewpoints | - Integrate the behavioral models of the actors into a coherent system<br>- Resolve conflicts and maintain overall coherence | - Workflow alignment diagrams<br>- IFx tool for formal verification<br>- State transition synchronization |

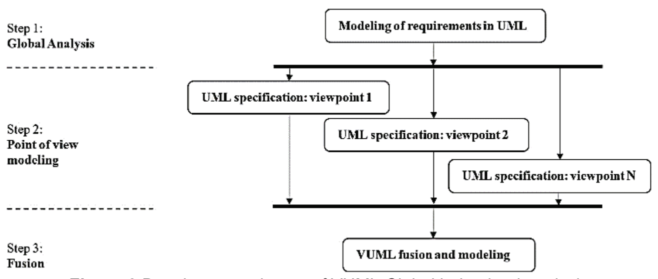
Step 1:
Global Analysis
Modeling of requirements in UML
UML specification: viewpoint 1
Step 2:
Point of view modeling
UML specification: viewpoint 2
UML specification: viewpoint N
Step 3:
Fusion
VUML fusion and modeling

**Figure 3** Development phases of VUML Global behavioral analysis

In this subsection, we describe how the VUML approach was modified to include the behavioral components of the technique.

## 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 [17]. 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 [18].

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 has a unique point of view. This point of view defines how an actor perceives and interacts with the system.

- 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 [19]. 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 [20]. 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 [21].

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 mentioned before, the state machines that are suggested to illustrate how views behave comply with the UML standard [22, 23]. 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 Omega UML profile serves as the basis for the syntax we use to represent transitions in the created state machines. With a collection of techniques tailored to communication and execution characteristics, this profile offers a comparatively extensive semantics.

Utilizing 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

Global analysis is the first step in the process, which acts as a central location for requirements modeling. Finding the demands of different actors and grouping them into functional units—represented as use cases—is the aim. An overview of the system's functions is shown in Fig. 4, which includes (1) agency monitoring, (2) people management, (3) material management, (4) financial management, and (5) automobile management. Every feature is broken down into smaller, more manageable functional parts that show the roles and responsibilities of each actor. The client, agency manager, workshop manager, and maintainer are the particular actors that are the subject of our investigation.

To provide clarification, we will use the "cars management" feature as an example (Fig. 4). The administration of the agency's vehicles includes duties including registration, supervision of each vehicle's expertise and repairs, and final verification testing. In the "save" use case, the customer gives details regarding their vehicle, works with the agency manager to carry out expert and repair contracts, and verifies the repair by giving their automobile one more functioning test. While the workshop manager is responsible in supervising the automobiles' maintenance chores, maintainers are involved in technical aspects such as expertise, repair, and testing.

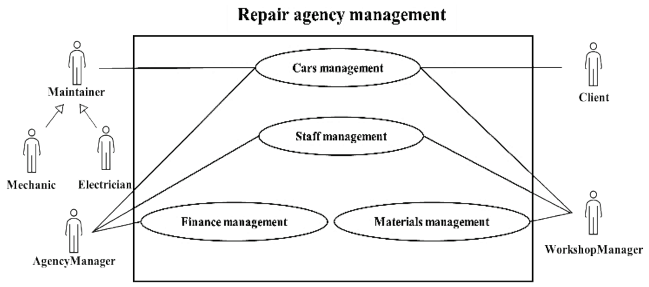

**Figure 4** Repair Agency Management" use case

### 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. The following classes were found to be multi-view in our case study: 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 is crucial to remember that throughout the full analysis by point of view in the next

phase of the method, this list of multi-view classes and their classification as static or reactive may change.
- 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: When the vehicle breaks down, the owner contracts with the agency and gives them the details they need to register the vehicle. The first step in the agency's repair procedure is to bring the vehicle to the garage, where mechanical and electrical specialists will inspect it for problems. A repair contract is created based on the findings of the experts, and the identified flaws are fixed by the maintainers. Testing is the last phase to make sure the vehicle operates as intendedThe several probable states of an automobile in the garage are included in this notional life cycle.

The initial state, OutOfOrders, signifies the car's breakdown condition, marking the beginning of the maintenance journey. This state is crucial as it sets the context for subsequent processes. Following this, the InGarageRouting state illustrates the logistics involved in transporting the vehicle from its location of failure to the garage, emphasizing the importance of efficient routing in minimizing downtime.

Once the car is in the garage, it enters the InExpertiseProcedure phase, where expert assessments are conducted to detect faults. This stage is critical, as it lays the groundwork for informed decision-making regarding the necessary repairs. The subsequent InRepairProcedure state is multifaceted, comprising two essential components: the negotiation of repair contracts based on expert evaluations and the technical repair itself. This duality highlights the complex interplay between administrative and technical aspects of automotive repair.

Finally, the process culminates in the InTest state, where the vehicle undergoes rigorous testing to confirm its operational readiness before exiting the garage. The Exit state signifies the conclusion of the car's lifecycle within the garage, encapsulating the entire process from breakdown to resolution.

The base-state machine of the Car class is shown in Fig. 5. This device provides a logical temporal sequence of an object's states, encapsulating its life cycle inside the system. In the second step of the process, the designers then integrate this machine into the lexicon for shared and reusable reasons.

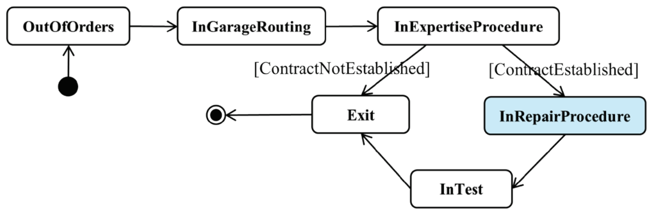

**Figure 5** "machine-base" of Car Class

### 5.3 Analysis/Design of Behavior by Point of View

The behavioral specification phase is started based on a particular viewpoint when the structural model for that viewpoint has been built (see section 4.2). The "machine-base" states found in the first phase are analyzed from a single point of view. A sub-machine may be produced from each basic state, which can then be refined and specialized to meet the needs of the particular point of view. We choose to use our example to restrict the illustration:
- One part of the "machine-base." We'll focus on the InRepairProcedure stage, when the vehicle goes through a sequence of steps to return to its typical operational condition.
- The client and the agency manager are the two actors to whom this is given.

Client Perspective: We extract the part of the SM_Client_Car state machine associated with the InRepairProcedure state (Fig. 6) from the class diagram created for the Client Perspective (see section 4.2), which focuses on the repair and contract negotiation process.

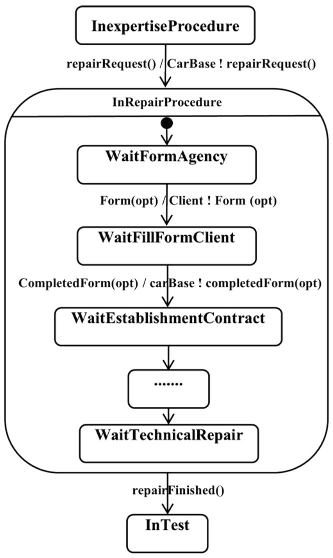
InexpertiseProcedure
repairRequest() / CarBase ! repairRequest()
InRepairProcedure
WaitFormAgency
Form(opt) / Client ! Form (opt)
WaitFillFormClient
CompletedForm(opt) / carBase ! completedForm(opt)
WaitEstablishmentContract
.......
WaitTechnicalRepair
repairFinished()
InTest

**Figure 6** Enhancement of the InRepairProcedure State for the Client Viewpoint

The Repair() signal, which the client actor requests, initiates the change in the SM_Client_Car view from the ExpertiseProcedure state to the RepairProcedure state. However, the analysis/design by point of view is carried out in a decentralized fashion throughout this process stage. Therefore, without worrying about which objects will get the signal, the SM_Client_Car state machine transfers this signal to the base machine. When one actor waits for a signal from another entity, the same logic is applicable. From this perspective, the developer in these situations presumes that the signal comes from the "machine-base." The form(opt) signal, which starts the changeover from the WaitFormAgency state to the WaitFillFormClient state in Fig. 7, serves as an example of this.

As seen in Fig. 7, the Agency Manager's viewpoint entails improving the InRepairProcedure state. The same idea of centralizing communications inside the machine base is used in this improvement. The signalRepair() is received by the SM_AgencyManager_Car from the "machine-base," and it is then sent on to the agency manager actor. The SM_AgencyManager_Car then creates a form with a list of the failures that need to be repaired along with the choices for fixing each one. After preparing the form, the agency manager sends it to the SM_AgencyManager_Car, which, upon receiving the form(opt) signal, initiates a transition to the WaitReturnFormCompleted state.

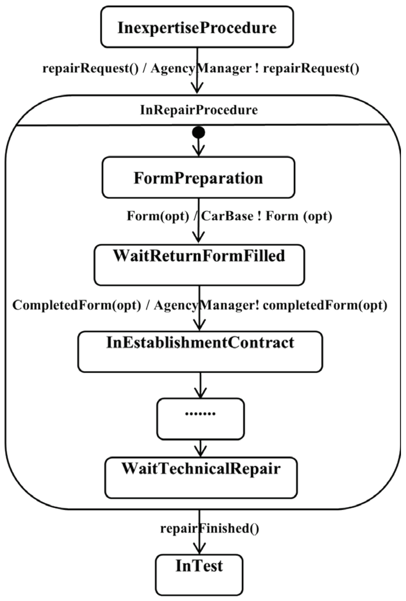

**Figure 7** Enhancement of the InRepairProcedure state for the AgencyManager Viewpoint

## 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.

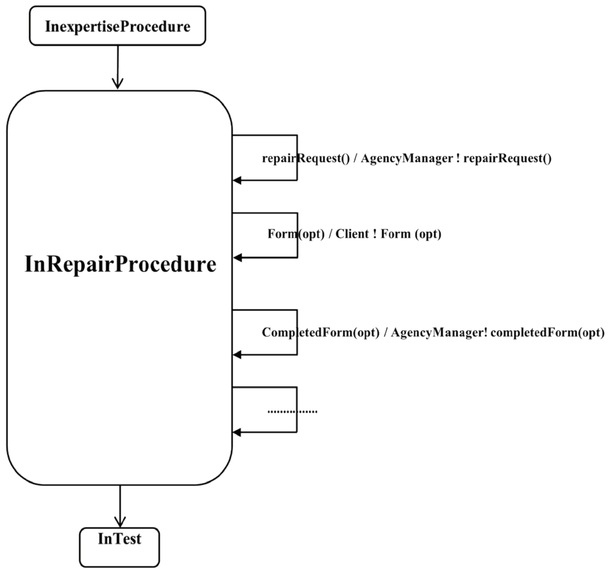
InexpertiseProcedure
InRepairProcedure
repairRequest() / AgencyManager ! repairRequest()
Form(opt) / Client ! Form (opt)
CompletedForm(opt) / AgencyManager! completedForm(opt)
...............
InTest

**Figure 8** Including Synchronization Messages for the InRepairProcedure State (Extract) in the Machine Base.

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.

Tab. 2 compares the proposed VUML method to other behavioral modeling frameworks, such as standard UML and SysML. It highlights the improvements brought by VUML in terms of viewpoint management, behavior fusion, and formal verification.

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.

**Table 2** VUML in comparison to other frameworks

| Criteria | VUML (Proposed) | Standard UML | Other (e.g., SysML, BPMN) |
|---|---|---|---|
| Points of View | Explicit integration, distinct actors | Generic approach | Limited to structural aspects |
| Merging Behaviors | Synchronization, conflict resolution | Weak support | Underdeveloped fusion mechanisms |
| Formal Verification | Support with IFx | Not supported | Limited to external tools |
| Complex Systems | Optimized for critical systems | Less suited to complex needs | Limited support for rich interactions |
| Clarity and Precision | Reduction of ambiguities | Possible ambiguities | Variable depending on the context |

## 6 CONCLUSION

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 [24]. 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 previously said, the original method only uses UML principles. Finding two different kinds of specialized machinery linked to either a "base" class or a "view" class for a given multi-view class is the main result of this method. According to the UML2.0 specification, these state machines are used to represent the dynamic behavior displayed by instances of the multi-view class under consideration.

A "machine-base" is intended to create coherence and coordination among the "machines-view" and express the collective behavior of the actors involved, whereas a "machine-view" depicts the life cycle of an object-view. A multi-view machine is a combination of a "machine-base" and its dependent "machines-view".

By supporting autonomous development in the second phase, this strategy tackles the problem. The "machine-base" idea was introduced in order to ease this, as the second phase required to be directed by a pre-established model. During the global modeling phase, the "machine-base" is created to express the aggregate behavior of the actors. When creating behaviors from other angles, this behavior is thought of as a model to follow.

During the decentralized modeling stage, the machines-view and object-view are defined independently, according on the "machine-base" state. The integration of incomplete behaviors from various perspectives is then achieved by adding signal exchanges in the fusion phase. The coordination of machines with the object-view and object-based states is made possible by these interactions.

However, as it grows, this UML-based method has drawbacks, especially in two crucial areas:

- 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: It becomes challenging when multi-view classes and their "machine-bases" must be identified early in the global analysis phase. Developing a state machine that encompasses an object's life cycle necessitates a detailed analysis of multi-view object use cases, which is problematic for big systems.
- Behavior Composition Challenges: Complexity of Integration: Many adjustments and changes may be required at the base machine and vision machine levels when integrating independently created vision machines. Maintaining the coherence of the entire system throughout the composing process when scaling necessitates considerable work.

This framework could transform software engineering by improving the modeling of complex behaviors, particularly in critical and collaborative systems. For example, case studies in the fields of industrial automation or intelligent transportation systems could validate the effectiveness of the method for synchronizing multi-view behaviors and resolving conflicts. These validations would demonstrate how this approach ensures consistency, reduces errors, and optimizes design in environments with high interconnectivity and 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 and could include applications in demanding fields where the management of multi-view behaviors is crucial. Furthermore, additional research on the optimization of synchronization mechanisms and adaptation to large-scale systems would provide increased robustness to the method.

## 7 REFERENCES

[1] 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. https://doi.org/10.5381/jot.2021.20.2.a2

[2] 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. (in French)

[3] 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). (in French)

[4] 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. https://doi.org/10.1007/s10009-005-0205-x

[5] Ouali-Alami, C., El Bdouri, A., Elmarzouki, N., & Lakhrissi, Y. (2022). View-based Modelling: Behaviour Specification based on UML Concept. https://doi.org/10.5220/0010730100003101

[6] Horman, Y. & Kaminka, G. A. (2004). Improving Sequence Learning for Modeling Other Agents. *Proceedings of the AAMAS 2004 Workshop on Learning and Evolution in Agent-Based Systems*.

[7] Nassar, M. (2005). Analyse/conception par points de vue: le profil VUML (*Doctoral dissertation*). (in French)

[8] Lakhrissi, Y., Coulette, B., Ober, I., Nassar, M. & Kriouile, A. (2008). Démarche VUML Statique et Dynamique. (in French)

[9] El Marzouki, N., Lakhrissi, Y., Nikiforova, O., El Mohajir, M. & Gusarovs, K. (2017). Behavioral and Structural Model Composition Techniques: State of Art and Research Directions. *Transactions on Computers, WSEAS*, 16, 39-50.

[10] 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.

[11] Kühne, T. (2022). Multi-dimensional multi-level modeling. *Software and Systems Modeling, 21*(2), 543-559. https://doi.org/10.1007/s10270-021-00951-5

[12] Hannousse, A. (2019). Dealing with crosscutting and dynamic features in component software using aspect-orientation: requirements and experiences. *IET Software, 13*(5), 434-446. https://doi.org/10.1049/iet-sen.2018.5324

[13] 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*, 195-207. Springer, Cham. https://doi.org/10.1007/978-3-319-46568-5_20

[14] 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*.

[15] Anwar, A., Dkaki, T., Ebersold, S., Coulette, B. & Nassar, M. (2011, April). A formal approach to model composition applied to VUML. In *the 16th IEEE International Conference on Engineering of Complex Computer Systems*, 188-197. https://doi.org/10.1109/ICECCS.2011.26

[16] El Marzouki, N., Lakhrissi, Y., Nikiforova, O. & El Mohajir, M. (2017, April). The application of an automatic model composition prototype on the-Two hemisphere model driven approach. In *IEEE International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS2017)*, 1-6. https://doi.org/10.1109/WITS.2017.7934673

[17] 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. https://doi.org/10.1007/s10009-014-0301-x

[18] Asteasuain, F. & Braberman, V. (2017). Declaratively building behavior by means of scenario clauses. *Requirements Engineering, 22*(2), 239-274. https://doi.org/10.1007/s00766-015-0242-2

[19] El Marzouki, N., Ņikiforova, O., Lakhrissi, Y. & El Mohajir, M. (2016). Enhancing conflict resolution mechanism for automatic model composition. *Applied Computer Systems, 19*(1), 44-52. https://doi.org/10.1515/acss-2016-0006

[20] 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*, 286-293. https://doi.org/10.5220/0006424902860293

[21] El Marzouki, N., Nikiforova, O., Lakhrissi, Y. & El Mohajir, M. (2017). Toward a generic metamodel for model composition using transformation. *Procedia Computer Science, 104*, 564-571. https://doi.org/10.1016/j.procs.2017.01.173

[22] 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, 5*(07), 2007-06. (in French)

[23] Lakhrissi, Y., Anwar, A., Nassar, M. & Kriouile, A. (2008). Composition des machines à états par point de vue dans VUML. In *Workshop JIMD, 3*(07), 2008-05. (in French)

[24] Ouali-Alami, C., El Bdouri, A., Elmarzouki, N., Korchi, A. & Lakhrissi, Y. (2022). Approaches to Design Complex Software Systems. https://doi.org/10.5220/0010740200003101

**Authors' contacts:**

**Chaimae Ouali-Alami,** PhD student
(Corresponding author)
SIGER Laboratory,
Sidi Mohamed Ben Abdellah University,
36000 Fez, Morocco
E-mail: chaimae.oualialami@usmba.ac.ma

**Abdelali El Bdouri,** PhD student
SIGER Laboratory,
Sidi Mohamed Ben Abdellah University,
36000 Fez, Morocco
E-mail: abdelali.elbdouri@usmba.ac.ma

**Younes Lakhrissi,** Professor (Full)
SIGER Laboratory,
Sidi Mohamed Ben Abdellah University,
36000 Fez, Morocco
E-mail: younes.lakhrissi@usmba.ac.ma