CHAPTER 3

THE MULTIPLE VIEWS

This chapter discusses concepts of view points, views and concerns for Model-Driven Software Evolution based on IEEE 1471 standard. These concepts are used to propose multiple views for MoDSE. Proposed views are derived from the identified viewpoints which are identified based on concerns of the stakeholders\(^1\) in MoDSE. Analytical validation of proposed multiple views are explained at the end of the chapter.

3.1 NEED FOR MULTIPLE VIEWS

This section outlines the need for multiple views and three underlying concepts of proposed multiple views for MoDSE. There are two reasons for using multiple views [79]:

- The amount of information in a complex product is too large and diverse in nature to be displayed in one single graph. Information has to be broken down into smaller chunks that can be visualized and analyzed much easier. Different graphs (diagrams or models) can show different information, revealing structure that cannot be shown in one diagram.

- Different people are involved in the development process. This user group comes from different objective world and has a different background and task focus and sees the productive in a different way. So these people have different viewpoints and demand different views on the product data.

3.2 CONCERNS, VIEWS, AND VIEW POINTS

---

\(^1\) A stakeholder is an individual, team, or organization (or classes thereof) with interests and, or concerns relative to a system. (IEEE 1471 standard)
Views offer visual representations of a model. View is a vested interest to visualize how the system is used and evolved. Different definitions of View, Viewpoint and Concern are available in different context of the work. Here a view\(^2\) is defined as a visible projection of a model to fulfill a stakeholder’s concern, which is a model evolution task. Concerns are those interests which pertain to the system’s development and evolution [39]. IEEE 1471 Standard also allows for the definition of a view as well as viewpoints [39]. A viewpoint\(^3\) defines a number of important aspects and concerns that are addressed by that viewpoint.

A viewpoint establishes the conventions by which a view is created, depicted and analyzed. In this way, a view conforms to a viewpoint. View is a representation of a whole system from the perspective of a related set of concerns. Viewpoint is a specification of the conventions for constructing and using a view. So, views conform to viewpoints. Viewpoints are identified in such a way that they satisfy stakeholders concerns. Thus, views are constructed from the identified viewpoints. Views also address the stakeholder concerns. Therefore, three underlying concepts such as viewpoints, views, concerns and their relations are named as ‘Address’, ‘Satisfy’, ‘Conform’, ‘Construct ’ are illustrated in Fig 3.1.

### 3.3 PROPOSED VIEWS

MoDSE consists of multitude of modeling languages. So, there should be a model integration, mapping and transformations etc. between these different models as well as code also. The stakeholders have various roles as well as concerns regarding creation and evolution of models in the system. To resolve this, multiple views are proposed which addresses stakeholders concerns in the context of MoDSE. These proposed views are derived from the viewpoints which were identified by the stakeholders.

---

\(^2\)View is a representation of a whole system from the perspective of a related set of concerns. (IEEE 1471 Std).

\(^3\)Viewpoint is a specification of the conventions for constructing and using a view. (IEEE 1471 Std).
3.3.1 Context View

Models define what is variable in a system. Context of a model consists of surrounding all model and model elements related to it. The elements are scattered over different models. It often occurs that only a limited number of model elements can be viewed at a time and understanding the entire model and its elements at a stretch is not an easy task. To understand a model completely, it might be necessary to know its context. Therefore, Context View is proposed. The model element whose context is viewed in a context view is centered. All model elements that are directly related to the particular model elements are viewed around the model element, for example, class and its subclasses.

Any model can be expressed completely by representing all its surrounding model elements and other models with which it is connected or related. A scope of a model can be determined by knowing all its surrounding model elements and models. Thus, the identified view points are ‘Expressivity and Scope’. Context of a model is expressed in terms of surrounding model elements and other models which are connected to this model. So, context view is derived from the viewpoints such as expressivity and scope. The above proposed context view satisfy the concerns like complete
understanding of model and its surrounding elements, and impact analysis\(^4\) of model elements. Context view\(^5\) would be in the form as shown in Fig.3.2.

\[\text{Fig. 3.2 Context View}\]

### 3.3.2 Inter-Model View

Numbers of models are used to represent entire software system. All these models are connected to each other and each model consists of model elements which are also connected to each other. These relations determine how one model element or model is dependent on the other. These dependencies provide the overview of model and make it possible to integrate different models. Thus, identified viewpoints are ‘dependency’ and ‘integrity’. Inter-model relations between model elements and models are visualized in this view. Further it is possible to have an interaction and integration between models. Therefore, \textit{Inter model View} is proposed to satisfy the concerns like overview of a system,

\(^4\)Impact analysis is defined as process of identifying the potential consequences of a change, and estimating what needs to be done.

\(^5\)Context view and city view shown in Fig. 3.2 & Fig. 3.3.are drawn using Metric View Evolution tool.
dependencies between models and its elements, knowledge about impact analysis, interaction and integration of models.

3.3.3 City View

The entire system is visualized in terms of models. Especially in large systems, it can be very difficult to find a specific piece of information and is often spread over multiple models. During evolution of models understanding which model element or model has evolved is important. This can be achieved by tracing a particular model or model element. During evolution some new model elements and/or models may be added or deleted. For addition or deletion, it should be possible to check whether existing models in that system can be extendable or not. Thus, extendibility and traceability are identified as view points for this view.

This view is similar to a geographical map of a city. To represent all models of a system at a time; this view would be more useful. So, inter-model view is an instance of a City view. Therefore, city view is proposed to satisfy concerns like overview of models in an entire system, to search, trace, and highlight the change in models, to extend models by adding elements, and relationships between the models. City view of models would be in the form as shown in Fig. 3.3.
3.3.4 Metric View

In MoDSE, the models are the primary artifacts for evolution. So there is a need to have a set of metrics which measures and estimates quality, complexity, and impact analysis of models. Variety of metrics such as O-O metrics, versioning metrics etc., exists to measure various issues of evolution of models. So, metric set and data values which are produced from the metrics are identified as viewpoints to derive Metric View. Therefore, Metric View is proposed to satisfy the concerns like metrics for quality, complexity and impact analysis (change prediction and propagation) of the models, and data values. The discussion of metric set is out of the scope of this thesis.

3.3.5 Transformation View

Transformations map models to the respective next level, be it further models or source code. MoDSE required transformations due to different and interrelated models. There is a need to distinguish transformations like model to code, model to model (migration of models or mapping from higher level models to lower levels), and code to model. These transformations include platform independent models to platform specific models and platform specific models to platform independent models.
Model consistency is an important issue to be considered in transformations because of source and target models belongs to different platforms. Therefore, *Transformation View* is proposed and derived from the view point ‘consistency’. This view satisfies the Concerns like transformation techniques, rules, tools, transformation languages, model migration and mapping. Transformation techniques and languages are out of scope and not discussed here.

### 3.3.6 Evolution View

MoDSE requires multiple dimensions of evolution. Stakeholders need to know the evolution type for example, platform evolution. To know what kind of evolution, *Evolution View* is proposed. This view is the combination of all the above proposed views because change in a single model or model element may be the cause of the evolution. So, all the above proposed views provide information about evolution in different forms. Thus, viewpoints identified in the above proposed views are also applicable for evolution view. Evolution view satisfies concerns like trends, causes, dimensions for evolution, change in development platform, monitoring the quality and complexity of models at multiple dimensions.

### 3.3.7 Evaluation View

There is a need to validate and verify already evolved models. The information captured in all the above proposed views should be verified. This view is also responsible to collect and check the feedback of the participants in the evolution of models in MoDSE. Based on the stakeholders’ feedback, evolution process can be regulated. Therefore, *Evaluation view* is proposed and derived from the view point validity. This view satisfies the concerns like evaluation trends and techniques which validates the evolved models, improving the quality, controlling complexity of models, stakeholders’ feedback.
The above proposed views are sufficient for the stakeholders to gain enough knowledge and visualization of models during evolution process. Knowledge such as context, metrics, model interaction, integration, mapping and transformation etc., of models and model elements will be available for the user groups to produce more evolvable model based software systems. And all these views support the entire software development and evolution process.

3.3.8 Advantages of Proposed Views

- Provide visualization of evolution process.
- Stakeholders concerns will be addressed by these views and viewpoints.
- Stakeholders can gain enough knowledge required for evolution of models in MoDSE.
- Establish the communication between stakeholders.

3.4 VALIDATION OF THE PROPOSED VIEWS

The above proposed views are validated analytically to verify whether they are appropriate or not by considering Lehman’s laws of software evolution.

3.4.1 Analytical Validation

This section presents how the proposed multiple views were validated analytically against Lehman’s laws of software evolution.

3.4.1.1 Lehman’s Laws of Software Evolution

The eight laws of software evolution are proposed by Lehman during the 70s and 80s are described in this section [55,68,69]. Over the years, the laws have gradually become recognized as providing useful inputs to understanding of the software process and have found their place in a number of university software engineering curricula.

I. Continuing Change
A system must be continually adapted else they become progressively less satisfactory in use.

II. Increasing Complexity

As a system is evolved its complexity increases unless work is done to maintain or reduce it.

III. Self Regulation

Global system evolution processes are self-regulating.

IV. Conservation of organizational stability

Unless feedback mechanisms are appropriately adjusted, average effective global activity rate in an evolving system tends to remain constant over product lifetime.

V. Conservation of Familiarity

In general, the incremental growth and long term growth of system tend to decline.

VI. Continuing Growth

The functional capability of systems must be continually increased to maintain user satisfaction over the system lifetime.

VII. Declining Quality

Unless rigorously adopted to take into account for changes in the operational environment, the quality of a system will appear to be declining.

VIII. Feedback System

Evolution processes are multi-level, multi-loop, multi agent feedback system.
3.4.1.2 Analytical support

This section analytically describes how the proposed views are closer to the Lehman’s Laws of Software Evolution.

(i) Continuing Change

The first and fundamental premise of the MoDSE says that the evolution should be a continuous process. Generally stakeholders initiate the change. Aim of the context view is to provide different context of models and model elements in the system and continuous change can also be viewed. Hence, context view satisfies the first law. Inter model view satisfies the first law by providing continuing change in terms of dependency between models and model elements. City view also satisfies first law which provides information about change in models such as extendibility and traceability. Metric view provides numerical values which can be used to manage continuous change. So, metric view satisfies first law. Transformation view satisfies first law where different types of transformations like model to model or platform independent to platform specific model are required when change occurs. Evolution view provides causes for change. Hence, this view satisfies the first law. Evaluation view is responsible to check whether the changes have done properly or not. Therefore, this view also supports the first law.

(ii) Increasing Complexity

Complexity is one of the major issues in MoDSE. Any single complex model or model element can be visualized by knowing its context in context view. So, this view satisfies second law. A single change in a model or model element may affect the complexity of other models which are directly related to changed model. Relation or dependency of models can be visualized from inter model view. Hence, inter model view satisfies second law. Change in models may extend models existing in the system. Extendibility may affect complexity and it can be visualized from city view which satisfies second law. To control or maintain complexity, measures and metrics are required. Purpose of the metric view is to provide metrics set and data values which are used to estimate complexity. So,
metric view satisfies second law. Transformation view satisfies the second law where there will be an increase in complexity after transformation of models. Evolution view also has its own role to control and monitor complexity. As a system evolves its complexity increases unless work is done to maintain or reduce it. So, evolution view satisfies second law by providing causes and trends for the evolution. Evaluation view satisfies second law by verifying increased complexity from the stakeholder’s satisfaction and feedback.

(iii) Self Regulation

Evolution of a model-based system involves a team of stakeholders within the organization. Interests, goals, purpose, tasks and objective of the team differs from each other. They together establish systematic parameters for more evolvable software. They also determine growth and other development characteristics of the evolving product. Context view provides different context of the evolvable system. Overview and complete understanding of the system can be derived from the context, inter model and city views. Extendibility and traceability can be determined from the City view. Metric view provides parameters for systematic evolution, and it requires data values which help to estimate issues like quality and complexity etc. Evolution and evaluation views provide causes for evolution and user satisfaction in terms of feedback. Therefore, all the proposed views support third law.

(iv) Conservation of Organizational Stability

Proposed views are aim to provide knowledge during model driven software evolution and also to understand the evolution of models. It is still believed that the effort expended on system growth and evolution is determined by managerial decision. But these decisions are constrained by external forces, trade unions or the availability of skilled personnel for example. In any practical situation the level of activity is not decided exclusively by management edict but by host of feedback inputs and controls [69]. However, Evaluation and Evolution Views provide the information about evolution of
models and stakeholders feedback which is important for managerial decisions. So, these two views satisfy the 4th law. But proposed views are not intended to measure organizational stability or invariant work rate in terms of available resources in the organization. So, this law is not applicable for the proposed work. This is shown in Table 3.1 as ‘NA’ (not applicable).

(v) Conservation of Familiarity

Software undergoes continuous change during its life time. Due to this continuous change it is not possible for stakeholders to retain familiarity of these changes for a longer period. All these proposed views support fifth law.

(vi) Continuing Growth

This law, Continuing Growth, appears little different to first law, Continuing Change. For example, accommodating a new requirement often necessitates substantial changes to system architecture and implementation, which leads to system uncertainties. Sixth law addresses change deriving from a different source. According to second premise of MoDSE, evolution should be done incrementally. So, evolution is a process of a continuing change and continuing growth also. Continuous growth can be visualized and traceable in inter-model and City views. So, these two views satisfy sixth law. Context of model which is continuously growing can be visualized by using Context view. So, context view satisfies sixth law. Metric view provides numerical values which represents growth (change) of models, computing change propagation, impact analysis, quality etc., whereby, metric view satisfies sixth law. Transformation view provides knowledge relevant to transformation of models from code to model, model to code, and model to model which indicates the continuous growth in transformed models. By this the transformation view satisfies sixth law. Evolution and Evaluation are also possible
to provide sufficient information regarding continuous growth of model. Thus, all the proposed views support sixth law.

(vii) Declining Quality

Quality is another major concern in MoDSE. Ultimately quality must relate to user satisfaction and also with stakeholders feedback. A system that has performed satisfactorily for some period of time suddenly exhibit unexpected behavior, unexpected results. There are several causes to explain this, but here it is considered as due to evolution of models. So, there is a chance of declining quality at least for a moment. Evolution view satisfies this law by providing causes for declining quality. But, if the evolution of models is handled properly then quality cannot be declined. Among the above proposed views, metric view has much responsible to achieve desired quality during evolution. Thus metric view satisfies seventh law. Evaluation view satisfies seventh law by identifying declined quality with user satisfaction and feedback. Source models are transformed to target models which are represented in transformation view. Due to the transformations there may be a change in models and there is a chance of losing quality in models. Hence, this view satisfies seventh law. Context view, inter model view, and city view are not intended to provide knowledge about quality. So, these three views do not support seventh law.

(viii) Feedback System

Systems are evolvable due to feedback of the stakeholders. Feedback is collected from different user groups. MoDSE is a feedback evolution process and it consists of multiple users, models, dimensions of evolution. Evolution and evaluation views capture stakeholders’ satisfaction and feedback also. Therefore these two views satisfy eighth law. The remaining proposed views such as context view,
inter model view, city view, metric view, and transformation view are not intended to collect feedback from the user. Hence, these views do not satisfy eighth law.

<table>
<thead>
<tr>
<th>Views</th>
<th>I</th>
<th>II</th>
<th>III</th>
<th>IV</th>
<th>V</th>
<th>VI</th>
<th>VII</th>
<th>VIII</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>NA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>V2</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>NA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>V3</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>NA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>V4</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>NA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>V5</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>NA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>V6</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>NA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>V7</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>NA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

Analytical Validation of the Proposed Views

Analytical support of the proposed views with laws of software evolution described above and same is shown in Table 3.1. The ‘Y’ represents proposed view which satisfy respective law, ‘N’ represents proposed view that do not satisfy respective law and ‘NA’ represents respective law that is not applicable for the proposed work. In Table 3.1, columns are numbered as I, II so on to represent eight laws of software evolution and rows are named as V1, V2 so on to represent seven proposed
views. From the values in Table 3.1 it is observed that majority of the entries are ‘Y’, which represents that most of the laws are satisfied by majority of the proposed views. Therefore, it is proved that the proposed views are very closer to the laws of software evolution and they are sufficient to gain information during MoDSE.

3.5 SUMMARY

This chapter proposes seven multiple views for Model-Driven Software Evolution. For each view, view points are identified. Each proposed view was derived by using these viewpoints. Different concerns of the stakeholders are satisfied by each view, providing sufficient knowledge regarding model-driven software evolution. Proposed views are validated analytically by using eight laws of software evolution is discussed. From the table it is observed that all the proposed views support majority of laws. Therefore, the proposed views are sufficient for the stakeholder to capture the information about models during evolution in the context of MoDSE. It is also observed that Lehmann’s laws are important and applicable for MoDSE.