CHAPTER 1

INTRODUCTION

Software systems become more complex, due to this better abstraction and more maintainable systems are sought. The development and maintenance of software systems mainly depends on abstractions. Identifying right abstractions to represent high-level requirements and designs for development of a new system is a major problem. During maintenance, the problem is to identify abstractions that are encoded in an existing implementation in order to make modifications. The gap between design and implementation leads to increase complexity further decreasing productivity and maintenance.

To reduce this gap, recent software development is becoming more model centric than program centric. Because of reusable nature of models, role of models are increasing and becoming eminent in software development. More maintainable software systems can be achieved with the help of model centric approaches. This is the reason model centric approaches and modeling languages have gained a much broader use in software development. In software evolution as well as in software development model centric approaches like Model Driven Architecture (MDA), Model Driven Development (MDD), and Model Driven Engineering (MDE) etc. have gained more significance and popularity. Model Driven Engineering (MDE) is unification of initiatives that aim to improve software development by employing high-level, domain specific models in implementation, integration, maintenance and testing of software systems [13]. The major goal of MDE is to reduce the development and maintenance effort by working at model instead of code level. Thus models have become integral part of the software system and evolution process.

Software models are often informal sketches that are discarded once the code is written. While this approach may be sufficient for code-centric software development, with a model-driven
approach, models themselves become the primary software development artifacts [15]. While MDE promises to improve productivity and maintainability, widespread adoption and scaling to large software systems needs research into evolution of model-based systems, scope and expressivity of modeling languages, and into interaction and integration of models. Thus, MDE requires a new paradigm for software evolution which could lead to an emergence of Model-Driven Software Evolution (MoDSE). Deursen et al., have also brought a research proposal which is entitled as ‘Model-Driven Software Evolution’ [1]. Hence, Model-Driven Software Evolution (MoDSE) is conspicuous.

1.1 IMPORTANCE OF EVOLUTION

Software evolution is concerned with complete life cycle of software systems, from initial development to maintenance. Evolution also includes adding new features, improving old features, and bug fixing. In traditional software evolution, the development platform is fixed. Migration to a new platform is a sporadic event. Since an MDE platform hardwires many more architectural and design decisions than a traditional development platform so, platform evolution is a requirement for MDE. Thus, multiple dimensions are required in MDE such as regular evolution, meta-model evolution, platform evolution and abstract evolution [1]. These evolutions target the changes in modeling languages, underlying infrastructure, and code generators. Changes can be made for modeling languages and/or new modeling languages may be added but the development platform is fixed. Sometimes changes are required to modeling language itself to improve the expressivity. Such changes may require model migration. Existing models may remain unaffected by such changes, if modeling language abstracts over the specifics of target platform. After introducing new languages, the old system should be migrated to make use of it.

The two fundamental premises of Model-Driven Software Evolution insist that evolution should be a continuous process. The development of software system is a continuous search for recurring
patterns, which can be captured using domain-specific (modeling) languages. After developing a number of systems using a particular meta-model, new pattern may be recognized that can be captured in a higher-level or richer meta-model. Reengineering of legacy systems is a continuous evolution which is to be done incrementally and it should be a special case of model-driven paradigm. Therefore, capturing the information such as how models are evolved, what kind of change leads to an evolution etc. is essential irrespective of multiple dimensions described above. This thesis considers understanding evolution of models in the context of MoDSE. This kind of information is considered as concerns and/or various activities of MoDSE which is very much essential for the stakeholders involved.

1.2 MOTIVATION

Any software system adopts modeling languages for software development process. A single model is not sufficient and several models of different modeling languages as well as some code in programming languages are essential to develop an entire software system. Due to variety of models and modeling languages, there is a need to have model interaction, integration, mapping and transformation among the diverse models. In this manner model-driven engineering abandon from traditional software engineering with its massive development platform. In place of one or two programming languages MDE bring forward a multitude of languages that are themselves artifacts of development process. When models are incrementally introduced in legacy system, models need to interact with legacy code. Thus, there is a need to have an interaction between model and code. This kind of interaction can be provided through an interface in any modeling language. Hence, there is a need to have an interaction between models.

Model-Driven Software Evolution requires many types of transformation like model to code, model to model, and code to model. These transformations are done by using transformation
language by the choice. Therefore, major stakeholder\(^1\) concerns such as model interaction, model integration, model mapping, model transformation, and model merging etc are considered in this thesis. These concerns are accomplished by using tools, formalisms and transformation languages [3, 4, 5, 13, 15, 19, 36, 65, 69, 77]. For example, IBM Rational Rose modeler tool, Enterprise Architecture (EA) tool, OMG’s Meta Object Facility (MOF), Query/View/Transformation (QVT), Syntax Definition Formalism (SDF), and Extended Markup Language (XML) etc.

Visual modeling languages have limited modularity such as, representing entire software system in a single diagram. So, models have to be developed as independent parts of software system that can be easily integrated in large systems. Moreover, there should be possible views on models to capture the information during evolution of models. For example, Static view to represent structure of software system and dynamic view to represent behavior of software system are used in unified modeling language (UML). Thus, this thesis focused on multiple views which are proposed for model driven software evolution (MoDSE) to understand various concerns. Information gained from proposed multiple views are important for stakeholders. Several stakeholders (roles) will be involved in the process of model based evolution. Each stakeholder typically has interests in, or concerns relevant to that software system. An ability of models to evolve gracefully is becoming a concern for many stakeholders. So, there is a need to have viewpoints and multiple views which captures these stakeholders concerns. Multiple views provide a means to visualize complex information and also a way to fulfill interests of different user groups which are addressed in the thesis.

Concept of view points, multiple views, stakeholders and concerns were described for software architecture in IEEE 1471 standard which is a technology-free [39]. It just provides a set of concepts with some recommended practices. Thus, to derive multiple views for MoDSE these set of recommended guidelines of IEEE 1471 is used as a back ground. In the

\(^1\) Stakeholder and User are used interchangeably in entire thesis
context of MoDSE authors have extended stakeholders list in IEEE 1471 standard such as Users, Acquirers, Designers, Developers, Tester and Maintainers. These stakeholder roles have an interest in software design and evolution of models. Obviously stakeholders will have their own viewpoints and views. Views differ from one stakeholder to another with respect to their roles during the evolution process. These kinds of views and view points were proposed by many researchers in traditional software evolution context but not in the context of model evolution [24, 48, 78, 81, 92].

There are number of frameworks that exist in literature for comparison and assessment of the various CASE tools. Comparison of these tools is essential to understand their differences, to ease their replication studies, and to discover what tools are lacking. Such a comparison is difficult because there is no well-defined comprehensive and common comparative study for different category of tools. A comparative framework was derived for design recovery tools and for software architecture visualization tools [104, 49]. These frameworks compare and assess the tools in context of stakeholder interest. This thesis derives such type of framework for comparison and assessment of tools in context of MoDSE. From this assessment, it is easy to know that how a selected tool satisfies the stakeholder concerns. Thus, motivation for this work lies in above mentioned two frameworks [104, 49]. Combining visualization approach with MoDSE is essential to understand evolution of models in a better way. As such there is no framework exists in the literature to evaluate tools which are useful for MoDSE and also to understand evolution of models with respect to stakeholder perspectives.

There is an increasing need for more disciplined techniques and engineering tools to support a wide range of model evolution activities, including model-driven software evolution, model differencing, model comparison, model refactoring, model consistency, model versioning and merging, and (co-)evolution of models. Many CASE tools have been evolved due to wide usage of model driven approaches. Tools are needed at different stages
of model driven evolution. So, here the question arises “how do you choose the right tool?”. Tools do much work in model driven approaches [7]. So, it is very much essential to choose tools carefully. To select right tools a recommendation system ‘mROSE’ is presented in the thesis which is conceptualized as a form of stakeholder interests and/or concerns. Such a perspective allows users to anticipate, explain, and evaluate different experiences and consequences following the introduction and intention of tools. The recommended items of interest to users based on preferences they have expressed, either explicitly or implicitly. Ever-expanding volume and increasing complexity of information has therefore made such systems essential tools for users in a variety of information seeking activities.

1.3 CONTRIBUTION OF THE THESIS

Software artifacts are subject to many kinds of changes, which range from technical to rapidly evolving technology platforms. These modifications include changes at all levels, from requirements through architecture and design, source code, documentation and test suits. They typically affect various kinds of models including data models, behavioral models, domain models, or source code models. In this context, models can play an important role. They can help and guide software evolution and can enforce and reduce critical risks and important resources (e.g., costs, personnel, time) involved in software evolution, by employing high level abstractions. Models can thus help to direct evolution. Models are the primary artifacts of model driven approaches and play a key role. The research projects such as MoDSE project at Delft University of technology, Netherlands, ARC project at University of MONS, Belgium and MODEVO project at University of Limerick, Ireland are focus to develop an approach includes methods, techniques, tool support, and stakeholder’s concerns, views etc. to understand MoDSE [1,26,60]. So, there is an increasing need for more disciplined techniques, approaches and engineering tools to support a wide range of model evolution activities including model-driven software evolution, model difference, model comparison, and model merging etc. Hence, the research is mainly focused on understanding evolution of models in various
evolution activities which is not covered in the above mentioned research projects. Contribution of the thesis is as follows.

- To understand evolution of models multiple views are proposed. These views are useful to view models in various aspects. The reason for multiple views is to consider different aspects of stakeholders and their various roles. Above proposed views are validated analytically using Lehman’s laws of software evolution.

- Above proposed multiple views are considered as key areas in the framework. Proposed framework is used to understand evolution of models by answering various queries that can be posed by the stakeholders. To answer the queries six level Likert scale is used as quantitative measures. Tools and stakeholder assessment were done to evaluate the proposed framework.

- Tools play a vital role in evolution of models. So, selection of tools became a tedious job. Therefore, a recommendation system is proposed to recommend appropriate tools for various activities of model-driven software evolution.

- Proposed recommendation system is implemented and named as ‘mROSE’ which consists of four major components. Longitudinal and Laboratory user studies are used to evaluate mROSE. Performance of mROSE is evaluated using performance metrics.

1.4 ORGANIZATION OF THESIS

The rest of the thesis is organized as follows.

- Chapter 2 presents literature survey on importance and understanding evolution of models in MoDSE. It includes need for multiple views, framework to understand evolution of models, role of automated tools in evolution of models, and brief introduction of recommendation systems for software engineering.
• **Chapter 3** addresses the proposed multiple views to view evolution of models in various aspects. It also discusses the analytical validation.

• **Chapter 4** presents the derivation of the framework. It also discusses evaluation of the proposed framework in two ways such as stakeholder assessment and tool assessment.

• **Chapter 5** presents a recommendation system named as mROSE which generate recommendations about tool selection for the appropriate concerns/activities of MoDSE.

• **Chapter 6** demonstrates the implementation of mROSE recommendation system.

• **Chapter 7** addresses the evaluation process of mROSE. Performance metrics are used to evaluate the performance of mROSE. An evaluation process includes Longitudinal and Laboratory User Studies. It also discusses results of the user studies.

• **Chapter 8** concludes the thesis and gives the directions for future research work.