Chapter II

REVIEW WORK
2.1 INTRODUCTION

A proper SE approach in LMS development ensures that it would reach its functional, social and technical objectives. Otherwise, the specification and design of the software would be susceptible to frequent modifications and would therefore increase the complexity and effort required for the development. For proprietary LMS software, this in turn would increase cost, and for free and open source applications, it would be technically more challenging to integrate different contributions made towards the development. SE process in general employs numerous well-supported and established methodologies (Pressman, 2005). In the context of LMS development, however, the case is not sufficiently illuminative since different SE methods and techniques are good at only some of the RE and designing issues but there is lack of any established methodology that could systematically handle different SE methods in LMS development (Calvo, 2003 and Gupta, 2013). LMS development entails particular attention in RE and software design phases for different reasons. Overlapping roles of users and mixed level of abstraction in the requirements should be handled efficiently specially for the model-based development of the system. The challenge of combining formal methods and UML within the development process is prominent due to absence of any standard framework. On the other hand, there has been a continuous effort from the researchers to support conventional LMS with additional features. Considering these issues, this chapter reports a review of the existing works on the SE methodologies suitable for LMS development. It also discusses on the endeavors towards some additional features of LMS.

2.2 REVIEW WORK

2.2.1 Requirement engineering methodologies

A good amount of research work has reported on the importance of different aspects of RE like early classification and mapping of the requirements, use of multiple models in RE, combining formal methods with UML specifications, and use of domain model.
Maguire, & Bevan (2002) has shown that a proper understanding of the requirements for LMS from different perspectives and their mapping with different user groups of LMS like learner, supervisor, author, and administrator helps to ensure that all the needs of the stakeholders are included in the specification. Avgeriou et al. (2003) has argued that mapping of the requirements to various quality attributes of the system are important to understand the impact and essentiality of any change in requirement specification. It is shown that such mapping is often mingled and inter-dependent. Almrashdeh et al. (2011) has reported a survey that was conducted to gather software requirements for LMS to provide desired functionalities to the learners. Different dimensions of the requirements are identified using a set of questionnaire and all requirements are analyzed from these perspectives. The five dimensions into which LMS requirements classified are instructional, technical, visual, administrative, and interactivity. While this work is largely focused on classification of functional requirements, the relationship of the requirements with quality criteria is missing. Although different SE methodologies are practiced in LMS development, all the development cycle begin with the RE phase that includes elicitation of user requirements, their analysis and subsequent specification (Hauge et al. 2006). The use of NL statements in all these phases often leads to ambiguous, erroneous, incomplete and inconsistent specification, which fails to address the end-user needs. Therefore, a major challenge in RE is to deal with the hidden ambiguity that arises due to the use of NL statements and mixed level of abstractions in defining the requirements (Bühne et al., 2004), (Hauge et al., 2006), and (Tjong, 2008). In order to meet the objective of a seamless, manageable, and traceable requirement specification for LMS systems, it is necessary that different requirements artifacts at different abstraction levels must be inter-traceable. Ensuring traceability of requirements at different levels of abstraction is a challenging task with many daunting trade-off issues, but it could be performed more easily and systematically with model-based approach for requirement analysis specially using formal methods (Avgeriou, 2003). Buhne et al. (2004) introduced an approach for modeling goals, scenarios, and requirements at different abstraction levels and applied it in the context of a development project. This work specifies the abstractions of requirements at three defined levels:
This work has listed the advantages of using different levels of abstraction, which includes making requirements traceable and manageable and changes could be easier to implement. In a similar work, Gorschek et al. (2006) has presented a model with four abstraction levels, which has been claimed to be a response to industrial need. This Requirement Abstraction Model (RAM) allows for placing the requirements at different levels and supports abstraction and breakdown of requirements to make them comparable to each other. Muhammad et al. (2007) have shown that RAM is adaptable and can be tailored to meet the needs of the various types of software including LMS. Since Software Requirement Specification (SRS) is the most important source of information for developing high-level tests, organizations willing to adopt the RAM model need to know the suitability of the RAM requirements for developing high-level tests. Muhammad et al. (2007) investigated this suitability by taking test cases from twenty randomly selected requirements of web-based LMS software. The results have shown that the requirements at lower-levels of abstraction are more suitable for developing executable tests, whereas it is hard to develop from requirements at higher-levels.

Over the years, many research works have explored the possibility of using formal methods like VDM and Conceptual Graphs in requirement analysis (Dardenne et al., 1993), (Liu, 1993), and (Cox et al., 2001). However, formal modeling has not widely accepted and failed to gain popularity among the developers and still to come into practice. This is mainly because formal methods work in isolation and there is inadequate tool support for combining formal methods with conventional UML modeling. Another reason could be the apprehension of difficulty, to follow complex mathematical expressions and to relate abstract descriptions with real-world entities among the software engineers (Börger & Stärk, 2012). On the other hand, UML has become the de-facto standard of modeling requirements in a wide range of software applications due to its higher understandability and the ease of use. UML is semi-formal in nature and uses graphical and text-based notations, which makes it convenient to use but the same time it compromises with the level of preciseness in defining the requirements. In a work by Glinz (2000), the suitability of using UML in requirement specification is investigated with a case study of the Teleservices and Remote Medical Care (TRMCS), where various
In the context of software engineering, the review work on UML has identified several problems and deficiencies, particularly concerning use case models and system decomposition. Some of the deficiencies with use case diagrams include:

- It does not support any description about the structure of the use cases.
- It cannot describe the structural hierarchy of the use cases in an easy and straightforward way.
- A UML use case model cannot express the state-dependent behavior of the system adequately, thus cannot model a rich system context properly.

Considering these issues, many researchers have advocated combining formal methods like Conceptual Graph, VDM-SL, and Z with UML for the RE (Requirements Engineering) process. Conceptual Graph is a very useful technique for formal representation of the NL (Natural Language) written requirements to identify and characterize dependencies between the requirements. Cox et al. (2001) introduced a formal approach for modeling the dependencies identified between the requirements using Conceptual Graphs. This work has shown how such modeling is useful to determine the impact of any breach that could appear in the later stages of development if any requirement were misleading.

Different works by Woo et al. (2002), Joyner et al. (2002), and Robinson et al. (2004) have applied graph matching in the retrieval of sequence diagrams from the initial use cases. The query, repository, and sequence diagrams are represented as conceptual graphs in which UML meta-model elements are encoded as vertices and UML meta-model associations are encoded as edges. Joyner et al. (2002) proposed a graph-matching algorithm to find the similarity between different conceptual graphs. Given a query sequence diagram, the algorithm finds similar substructures in repository sequence diagrams.

In the field of formal specification languages, substantial work has been done on enriching traditional approaches of formal specification like Z and VDM (Gotel & Finkelstein, 1994), (Shroff & France, 1997), (France et al., 1997), and (France et al., 1998) for the RE (Requirements Engineering) need. Chantatub & Holcombe (1994) proposed some techniques for specifying and testing software requirements and design, which emerge from integrating three specification techniques namely:

- Informal specification technique (natural language).
- Semi-formal specification technique (graphical models, e.g., entity relationship diagram, data flow diagram, and data structure diagram).
- Formal specification technique (e.g., Z, VDM, OBJ).

In a work by Tsumaki & Morisawa (2000), a requirement-tracing...
CHAPTER II - REVIEW WORK

A framework for change tracking and influence analysis is proposed. Fitzgerald & Larsen (2006) have described methods for developing the semantics and tool support for VDM, and shown how to apply it to identify achievements and challenges in real systems, using lightweight but effective VDM-SL description.

In some recent works by Pohl (2013) and Dutta et al. (2013), it is discussed that for keeping the knowledge expressed in different representation formats (textual, graphical, semi-formal, formal etc.) consistent, different representations must be integrated and a transformation technique among informal, semi-formal and formal representations is required. However, none of this work has focused on any acceptable framework to alleviate the process of integrating these formal methods with UML.

Different works by Robinson & Volkov (1999), Bühne et al. (2004), Larman et al. (2005), and ShuiYuan et al. (2009) have shown that instead of building a single large requirement analysis model, a set of smaller models for different levels of abstraction is more advantageous. The requirement analysis model is then more useful to detect any underlying error at any level of abstraction, which might occur while detailing a requirement from higher-level of abstraction to lower-level of abstraction. It is also observed that a level-wise verification technique brings confidence to the analysis teams while detailing the requirements in the subsequent phases of modeling. In addition, any possible ambiguity or conflict of the requirements could be avoided while detailing the requirements to the desired level of abstraction, which would leave less chance for software designers to make erroneous assumptions.

DM provides an abstract description of the environment in which the system would be developed. Models evolved at different levels of abstraction, require an early identification and decomposition of objects, entities, events, processes etc. to the lowest level of granularity. DM can serve this purpose, which could be either built or reused from existing. Prieto-Díaz et al. (1990), has proposed a model for the domain analysis process and a library-based domain infrastructure to show how domain analysis could be integrated into the software development process. It describes domain analysis process as an ongoing process of continuous refinement. In a work by Berger et al. (2008), Application-based Domain Modeling (ADOM) is proposed and it is claimed that while developing a
particular application in the domain, a domain model is used as a reference and the application model can be validated against the relevant DM in order to detect completeness and correctness errors. Although, different methods could be used for domain modeling, most commonly used techniques are domain ontology and the Conceptual Class diagram.

While modeling the requirements at different levels of abstraction, an ontology-based semantics of the entities, concepts, processes and objects is useful for ensuring consistency between the models. In the work of Montenegro et al. (2011), an ontology for the common modules of an LMS has been proposed. This work shows how to determine the domain and the scope of the ontology by reusing an existing ontology. It enumerates the important terms in the ontology by defining the classes and the class hierarchy. Linta et al., (2012) has presented the ontology for an E-learning system, using elements such as syllabus, teaching methodologies, learning styles and activities.

Conceptual class diagram is another useful technique for domain modeling. Baumeister et al. (1999) proposed a conceptual model for the navigational structure of hypermedia applications, which is consist of navigational class model represented with conceptual class diagram and a navigational structure model. The navigational class model specifies which classes and associations from the conceptual class diagram are available for navigation and the navigational structure diagram specifies how the navigation is performed. In a similar work, Hennicker & Koch (2000) have proposed a methodology for hypermedia design, which is based on use case analysis and a conceptual model of the application domain. Here, navigational classes can be mapped to conceptual classes. All attributes of navigational classes map directly to attributes of the corresponding conceptual class. However, the required information on the omitted classes can still be kept as attributes of other classes in the navigation space model.

Observations:

The above-discussed works strongly suggests that using multiple modeling techniques for requirement analysis is more suitable than using any particular method, when requirements are specified at different levels of abstraction. Therefore, a RE framework is required to facilitate the interoperability between
CHAPTER II- REVIEW WORK

the models, using the DM and to ensure traceability between the requirements at
different levels of abstraction. The framework should support for integrated use of
formal methods and UML.

2.2.2 Software architectural design

Similar to different levels of abstraction in requirement specification, software
design also evolves at different levels of abstraction, starting with software
architectural design at the most abstract level down to the detailed design level,
which is implementable. Different works of Medvidovic et al. (2000), Goulao &
Abreu (2003), and Garlan et al. (2010), has mentioned that although software
architectural design is a key component to the successful development of complex
software, its practice in reality has been largely ad-hoc, informal, and distinctive.
The need of expressive notations for complex software systems to represent their
architecture can be availed in different ways. One can use a specialized notation
for AD like (Architecture Description Language) ADL or can adapt a general-
purpose modeling notation, such as UML. The lack of tool support for ADL
hinders the acceptability of this formal method among the practitioners. On the
other hand, UML, with adequate tool support, is a widely accepted and practiced
method for designing at different levels of abstraction; still it is an open question
whether UML is sufficiently expressive for AD, as currently defined. Below we
discuss about existing works on the possible inclusion of formal methods to
support the standard UML based architectural design process.

Software architecture design has different architecture views like, conceptual
view, module view, code view, execution view etc. A conceptual view describes
the architecture in terms of domain elements and depicts the functional features of
the system by establishing the relationship between the elements. Hofmeister et
al. (1999) described two basic elements in the conceptual view: i) components
with ports through which all interactions occur, and ii) connectors with roles
to define how they can be bound to ports. The components and connectors are bound
together to form a configuration and each component can be decomposed into
other components and connectors. In order to bind together a port and role in a
configuration, the port and role protocols must be compatible. This work
(Hofmeister et al., 1999) introduces new stereotypes to UML Class/Object, Package, and Component Diagrams for the elements of the AD. Many researchers have also worked on how to use Architecture Description Language (ADL) as modeling notation to support software architecture design. Although, there is little consensus in the research community on how a software architecture should be modeled with an ADL (Medvidovic et al., 1999), researchers have proposed for modeling software architectures both with domain-specific and general-purpose ADLs. Acme (Garlan et al., 2010), Darwin (Magee and Kramer, 1996), C2 (Medvidovic et al., 1999), Rapide (Luckham et al., 1995) and Wright (Allan and Garlan, 1994) are some of the important ADLs which are commonly used in software architecture. Medvidovic & Taylor (2000) have presented a report on a commonly acceptable definition for ADL and a concise classification for ADLs. This work identifies that an ADL must explicitly model components, connectors, and their configurations; these elements, however, could be further broken down into their constituent parts depending on the application domain to which the architecture belongs. However, one major concern with ADLs is that, each ADL and supporting toolset operates in isolation, making it difficult to integrate those tools and share architectural descriptions. Garlan et al. (2010) proposed Acme representations that can provide a formal basis for the description and analysis of software architectures using a relatively simple core set of elements like Component, Port, Interface and Connector for defining system structure, coupled with the ability to extend their behavior using roles, configuration and protocols that are appropriate to the context of domain. It is claimed that Acme is advantageous over the other ADLs for its ability to serve as a common interchange format for different architecture design tools. This feature adds generosity to SE methodology to import and export AD to/from any other ADL.

On the other hand, UML is also widely used to represent the software architectural models. There has been a continuous effort to find out new approaches to combine ADLs (especially ACME) and UML for AD. Real-Time Object-Oriented Modeling (ROOM) (Selic et al., 1994) is a well-known object-oriented modeling that combines a variant of component diagram with variant of state transition diagram and fulfills the main requirements for an ADL. Rumpe et
(1999) has shown how elements of the component-based modeling language ROOM could be integrated with UML. ROOM uses a variant of the component-port-connector model for designing software architecture in conformance with ADLs. In a work, Garlan et al. (2002) have examined the space of possible mapping of ADL to UML. They have described the principal strategies for representing architectural structure in UML with the benefits and limitations of each strategy. This work has also identified the aspects of architectural description that are intrinsically difficult to model in UML using these strategies.

Goulao and Abreu (2003) have suggested an extension of the UML class diagram notation, which allows the component-based description of software architectures. This work proposes new UML modeling techniques to map from Acme to UML and thus allowing architectures to be expressed in a mainstream notation. The feasibility of this mapping is demonstrated through several examples.

In addition to depicting an abstract view of the system, AD is also useful for the design verification with some specified standards (Mokarat & Vantawood, 2013). Ontology is a lightweight formal method, which is commonly used in AD for this purpose. Kim & Fox (2002) have shown that a specific organization's quality management standard, such as the ISO 9000, can be represented using ontology model, and then compared to as a reference model for goodness measure for business processes and organizational structures. This model is represented using common entities, attributes, and relationship, comprising general schema or data model, which are formally defined and constrained. In this work, ISO 9000 standard is represented as inference rules that are applied to facts about an organization's quality management processes and structures. Finally, the conformance or nonconformance to requirements is automatically inferred using the proposed methodology.

In the domain of LMS, LTSA provides a generic specification of a high-level architecture for information-technology-supported learning, education, and training systems. Some endeavors have been made to develop LMS based on the LTSA guidelines (Peredo et al., 2006), (Sengupta et al., 2009). Finke (2004) has discussed on how LMS could be benefitted by focusing on technical standardization issues such as Sharable Content Object Reference Model
CHAPTER II- REVIEW WORK

This paper proposes an LMS architecture, which is based on standardized Reusable Learning Objects (RLO) and it allows for learner-centric and personalized learning processes. The perspectives of cognitive and pedagogic processes are also included within the architecture. The requirements are examined for a pragmatic architecture, which holds the potential to combine technical constructs (RLOs) with process-oriented pedagogical concepts. However, methodology related to verification or checking conformance of the system to LTSA is not addressed in this paper.

Pahl & Holohan, (2004) have advocated that as LTSA supports the taxonomy functionality for syntactical aspects and the logical theory function for semantic aspects, it would be useful to enhance the LTSA definition by ontology-based semantic descriptions. This work has introduced a framework for semantic service description and composition. In another work, Choe & Kim (2005) deals with cost-effective reusability of LOs along with their dynamic sequencing plan. An UML-based ontology for educational content model is introduced to the IEEE LTSA model for this purpose. This proposed enhanced LTSA model is claimed to be a solution to develop efficient e-learning systems.

Observation:

AD works as an intermediate between requirement specification and implementable detailed design of software. The above-discussed works show that integrating ACME style within UML is very useful as the abstract level design of components and their inter-relationships could be further decomposed and detailed towards the implementation in the subsequent phases of development. Moreover, the formal description of the design could be tested for conformance with any applicable standard, like LTSA for LMS.

2.2.3 Software detailed design

Software detailed design is a very mature discipline with well-established methodology being practiced by the developers, especially UML, which has undoubtedly become the de-facto industry-accepted method for detailed design. Therefore, instead of exploring conventional UML based designing, here we
CHAPTER II- REVIEW WORK

A report on some of the works that have used formal methods to supplement UML in software design.

Although UML is a widely accepted modeling and designing technique, it lacks in preciseness and formality. Moreover, using different diagrams (Use Case, Sequence, Activity, State Chart, Deployment etc.) at different levels of abstraction brings additional concern to prove consistency among them. Although Use case diagram is good for any level of abstraction, but due to its text based descriptions it fails to depict a precise and complete view of the system. Conversely using a formal modeling technique can solve this problem since methods like VDM, Z come with specification languages, which are intended to alleviate the rigor of formalism for using them in software specification. Formal methods are also used to improve the consistency, traceability and verifiability of the design components created with semiformal UML diagrams. But using VDM-SL alone to model all types of requirements at every level of abstraction, results in a complex specification, which is not only difficult to analyze but also requirements are hard to trace between different levels of abstraction. Consequently, many research works have been carried out to integrate formal methods with semi-formal or graphical methods in software development and to ensure consistency between different design artifacts (Dascalu, 2001). Dascalu, (2001) has shown an integration technique for graphical-semi formal representations with formal notations for construction of time-constrained system. The graphical notations employed are a subset of UML whereas Z++ is the choice for formal notation. The translation between UML to Z++ is performed in a pragmatic and systematic way with a detail algorithm being proposed. It promotes a lightweight practical specification, which is reliable as well as supports rapid development.

Mota et al., (2004) has pointed out that graphical specification with UML must to be formally verified, before the implementation phase, in order to ensure development of reliable systems. This work presents a protocol interface for integrating Computer Aided Software Engineering (CASE) using UML and formal verification techniques. It uses automatic property extraction from UML diagrams and keeps track of the aspects of system development, which are verified by suitable mechanisms based on First Order Logic.
Sengupta & Bhattacharya, (2006) have proposed a method to ensure consistency between different UML diagrams with help of a defined set of consistency rules. Z notations and XML are used to analyze different UML constructs like Class Diagram, Use Case Diagram, Activity Diagram and Sequence Diagram. In this work, Z equivalent structures of different UML diagrams are proposed and consistency between the UML diagrams and their relationship are verified. This work tries to ensure traceability of requirements by verifying the diagrams by its Z notation based formal syntactic.

Larsen, (2009) has contributed on identifying mapping potential between VDM++ and different UML constructs like Class diagrams and Sequence Diagrams. An abstract syntax tree for VDM is used as an essential part of the model transformation. A tool is built to support bidirectional transformation rules for UML and VDM. The class diagrams are exchanged between tools using XML based standard. This work also proposes a model transformation between sequence diagram and VDM++ traces.

Observations:

Most of the above-discussed works depend heavily on the model transformation between formal and semiformal techniques, but lack in any commonly acceptable method for integrating their use in the development process. It is obvious from the above-discussed work that the main challenge is to ensure consistency between different design-constructs of UML and formal methods. However, an integrated approach could be also helpful to check traceability of the requirements from design phase to the RE phase.

2.2.4 Interface requirements

This section reports on some of the endeavors made towards formal modeling of interface requirements using VDM-SL and FSM. The main objective behind the Model-driven approach in SE is to ensure that the end product meets all the user's requirements and the development process is more quick, efficient and accurate (Jin, 2008). Since all contemporary LMS are web-based applications, the requirement and design artifacts involves obvious blend of functional requirements and interface-requirements. The reason for such blending is that the
behavior of user interface cannot be properly understood in isolation; instead, it requires analysis of the type of data processing taking place at input and output of the functional requirements (Weyers, 2011). In general, interface-requirements, being Non-Functional Requirements (NFR), are hard to define in a formal way. The case of web-applications is even more complex due to the event driven nature of such applications. The sequencing of the activities, required to implement the requirements, cannot be predefined as the users of the web-application determine their (activities) order of occurrence in the system. Moreover, some of the interface-requirements appear as an integral part of the functional requirements. On the other hand, inefficiency of formal specification languages like VDM-SL to handle interface requirement is well known. Liu (1993) highlighted some of the limitations of VDM-SL as a specification tool for interface requirements. This includes a) lack of data types to handle complex applications (specially web-application), b) pre-post condition based specification against operational constraints, c) lack in modeling concurrent execution, and iv) lack in event driven modeling.

Atterer (2000) has presented a conversion model for requirements from high-level definition to low-level definition through reification being done by VDM-SL. Since, interface requirements come into play at the lower-level design (less abstract) of the system, this conversion model from high abstraction level to low abstraction level is particularly important for formal modeling of interface requirements. It is observed that VDM-SL is particularly suitable where a) operations cannot be explicitly specified, b) the structure of the system is not modular, and c) the features are at higher level of abstraction in the specification. Paiva et al. (2003) has presented an approach to integrate formal methods for user's interface requirements within the software development process. This approach covers the requirement and design specification by means of formal models. It also facilitates conformity testing of the user interface implementation with respect to the requirement specification. These conformity tests are detailed through a state transition model with an abstraction mapping function between detailed (implementation) to abstract (specification) states and operations.
Bowen (2007 and 2008) has shown how interface design could be incorporated within a formal software development process without changing the nature of user-driven and user-centered design. This work demonstrates use of a refinement process as a formal method for transformation of some higher-level design to something less abstract. The process is repeated until it leads to a concrete implementation. It is observed that formal modeling of interface requirements can help to ensure consistency across different design elements at different levels of design, and perhaps most importantly, can help to incorporate the interface design process into a larger, formally based, software development process.

Besides VDM-SL, Finite State Machine (FSM) based modeling of interface requirements has also been endeavored by many researchers. Meudec (1998) has highlighted the need of extra data types to deal with the detailed level of specification. This work presents a rationale for using tests derived from high-level formal specifications and then set to find an efficient technique for the generation of adequate test sets from specifications written in our study language, VDM-SL. It formalizes the traditional high-level partitioning technique for test cases generator prototype, and extends it to take the semantics of VDM-SL fully into account.

Andrews et al. (2005 and 2010) has shown how to build hierarchies of FSMs to model the subsystems of web-applications, and then generated test cases as subsequences of states in the FSMs. These subsequences are then combined and redefined to form complete executable tests. Some constraints are used to select a reduced set of inputs to avoid the state space explosion. In similar work, Jeff Offutt & Wu (2010) have discussed on three aspects of web applications that are very useful for creating FSM model, (1) an extremely loose form of coupling that features distributed integration, (2) the ability of the users to change the potential flow of execution directly, and (3) the dynamic creation of HTML pages. This model is based on the notion of atomic sections, which are the logical part of a web page. This paper represents a new theoretical model that captures dynamic aspects of the presentation layer of web applications and a testing methodology for the verification of the design. However, integration of the server-side scripting, like JSPs and Servlets, which can dynamically create user interface, is
CHAPTER II- REVIEW WORK

not considered in this work. In another work, Mesbah et al. (2012), has proposed a state-flow model for handling UI requirements. It tries to find out an automated testing policy for AJAX applications by modeling specific fault states like DOM validity, error messages, discoverability, and back button compatibility.

Observations:

The above-discussed works shows that it is better to incorporate interface requirements within the functional requirements model than to handle it separately. Applying formal methodology for interface requirements not only helps to incorporate them within the requirement analysis and design model, in addition, model based test cases could be generated. However, the main challenge is the inability of formal methods like VDM and Z to handle high-level abstract requirement definition and low-level detailed design (including user interface) of the software with same efficiency.

2.2.5 Additional features of LMS

The online learning process with help of LMS already employs many useful features that supplement the traditional process of classroom teaching. However, there is a continuous effort among the researchers to introduce new exciting features to LMS in order to make it more efficient, adaptive and intelligent. Several research works have been carried out to help the learners and the instructors with learners' portfolio, appropriate learning path, and customized learning content. Below we discuss on some of the research works related to learner modeling, learning path and adaptive learning.

Learner modeling or Learners' portfolio s are expressed in different forms in different research works, depending on purpose and individual or programmatic design. Lee et al. (2005) has described an investigation process of the learners' portfolio in the e-learning environment. It adopts data mining techniques to establish the most adaptive learning path pattern for each cluster of learners, which provides a scaffolding to guide each cluster of learners. In another work by Yen et al. (2005), an advanced Petri Net model based analysis of the workflow is
proposed for a web-based multiple participants' virtual environment. Behaviors of the students are supervised by an intelligent system agent with help of a generic interface for scaffolding, which is based on virtual reality and real-time communication technologies. A work by Chen et al. (2008) has shown how learners' behaviors on a website are recorded as learning portfolios and analyzed for behavioral diagnosis or instructional planning. A student model is then built accordingly with help of the analytical results of learning portfolios and a concept map of the learning domain. Romero et al., (2008) has mentioned that existing LMS software like Moodle, Sakai, Atutor record learner's portfolios but only collect modest activity log data, without directly providing instructors sufficient help for analyzing learner behaviors. Although Moodle facilitates several reports about learners' activities, but those are according to the predefined categories, not on instructors' customized need.

A substantial work has been done to construct individualized a Learning Path for learners in order to make learning more efficient. In the work of Zhao & Wan (2006), a graphical structure of semantically connected nodes is used to find the shortest learning path between two nodes. The arrowhead weight mechanism and a proposed algorithm are used for the adjacency matrix to optimize the cost, time and effort. In another approach by Viet et al. (2006), construction of a learning path for each learner based on the record of learner's profile is described with help of some attributes of LO. This work is focused on learning path selection based on weight of the LOs and learner modeling. Hong et al. (2007) have contributed on developing e-learning systems with personalize learning mechanisms to assist online Web-based learning and adaptivity to provide learning paths in order to promote the learning performance of individual learners. In the proposed method, a pretest through the computerized adaptive testing (CAT) is first applied to collect the incorrect learning concepts of learners. Then the difficulty parameter of courseware and the concept relation degrees of courseware are simultaneously considered to determine the fitness function of the proposed algorithm in order to generate personalized learning path for individual learner. Pedrazzoli (2009) has shown an AI based tutoring system, which is able to identify, monitor and adapt the student's learning path, according to her knowledge base, learning habits and preferred learning style. The AI tutoring facility (eTutor), coupled with the LMS,
CHAPTER II - REVIEW WORK

is intended to support a human tutor with helpful activity data, available from the integrated Behavior Recorder Controller (BRC) system, allowing to confirm or manually modify actions suggested by the system (eTutor).

Adaptivity in conventional e-Learning systems has legacy from the existing works on ITS. Researches with ITS have a very long history started back in 1970s, much before the popular LMS came into existence (Wenger, 1987). Ample amount of contribution have been made since then in the mechanism of student modeling, pedagogy and domain modeling. A new generation of LMS, known as iLMS, that fits the basic features of ITS within LMS, has been proposed in many research works (Brusilovsky 2003 and Moodie, 2010). In a work by Simic et al. (2004), integration of the Semantic Web technologies in intelligent learning systems is proposed for iLMS architecture. This system offers a web-based environment for the developing and using online courses where user's environment is the adaptive attribute for the system.

Since an LMS application can be invoked virtually from everywhere and with different equipments, adaptation to user's environment has become an important issue. Many works have also preferred to adapt to the user platform (Abdel, 1996), (Brusilovsky, 1998). Simple adaptation to the platform (hardware, software, network bandwidth) usually involves selecting suitable types of multimedia components (i.e. HTML versions, image, graphics, audio, video, animation etc.) to present the content. More advanced technologies can provide considerably different interfaces to the learners at different platforms, like responsive web pages and AJAX-based content adaptation. In adaptive hypertext document, links are established in such a way that the user can explore, browse and search for not only a particular item but can also access information regarding relevant/associated issues.

In modern LMS applications, different parameters are used to fetch valuable information for providing adaptivity. Kobsa et al. (2007) and Busilovsky & Milan (2007) have suggested adaptation techniques based on user data, usage data, and environment data. User interest is another important attribute, which is frequently used for adaptivity in current LMSs. With the rise of adaptive bookmarking...
Observations:

The above-discussed work reveals that there is a continuous effort among the researchers to enrich conventional LMS with additional features in terms of adaptivity and personalization. Personalization either can be facilitated for a predefined problem like managing learner's profile, finding learning path, etc. or can be provided on-demand with help of runtime scaffolding. Contemporary adaptive systems focus on two distinct areas of adaptation: content level adaptation or adaptive presentation and link level adaptation or adaptive navigation support.

2.2.6 Summary of reviewed works

In the reviewed works, we have come across different formal methods for serving different purposes in SE – VDM-SL, Z as specification language, Conceptual Class Diagram and Ontology for domain modeling, Conceptual graph for modeling NL statements, etc. However, their existence is more in the research works and designing critical systems, and less in the development of software systems, like LMS. On the contrary, UML has become the de facto modeling standard for any phase of software development. As pointed out by many researchers, both UML and formal methods has their own advantages and limitations, and they work perfect only under certain scenarios.
established that integrated use of both UML and formal methods, if managed well, is better than using any one of them. Unlike UML, which is useful for modeling every level of abstraction in RE and design, there is no single formal method, which is such versatile in all situations. Therefore, use of multiple formal to supplement UML is required to fulfill the goal of seamless development of LMS, where requirements are traceable and verifiable at every level. However, such integrated approach entails pertinent frameworks at RE and software design phases, where different artifacts of different methods can inter-communicate to serve a common goal and be consistent to each other. A RE framework should be able to model requirements at different levels of abstraction. Similarly, in the design phase, three different levels—AD (abstract level), detailed design level, and interface design levels are prominent with their intrinsic features. Unlike the case of RE, where all levels of requirements are stated in NL, in the design phase three different frameworks are required to model those three levels.

Although contemporary LMS software, with the help of new edge web-technologies, are rich in terms of user friendly features, there is a chance of improvement on incorporating the features of traditional intelligent tutoring systems within the LMS. Some of the challenging features like personalized learning, learning path recommendation and adaptive hypermedia require particular attention.

Considering these gaps in the field of SE methodologies for LMS development and providing additional features, we below set the objectives of this research work.

2.3 OBJECTIVES OF THIS RESEARCH WORK

To develop a Requirement Engineering Framework for LMS that supports

i) Integrated use of Formal and semi-formal methods for requirement analysis

ii) Ensuring traceability of the requirements at different levels of abstraction
CHAPTER II - REVIEW WORK

To develop a Framework for AD for LMS that supports
i) Integrated use of Formal and graphical methods in AD
ii) Automated conformance verification with Learning Technology standard

To develop a Framework for detailed design for LMS that supports
i) Integrated use of Formal and semi-formal methods in Software Design
ii) Checking consistency between design elements developed from
different methods
iii) Ensuring traceability of the requirements to the design artifact

To develop a Framework for interface requirements for LMS that supports
i) Formal modeling of the interface requirements
ii) Model based verification of the design

To propose some novel techniques towards additional features of LMS
i) Personalized Learning
ii) Efficient Learning Path
iii) Adaptive Hypermedia

2.4 SCOPE OF WORK

To meet the above stated objectives, this research endeavors to find out suitable
SE methodologies for LMS development. Pertinent frameworks are proposed for
RE and software design phases. Some contributions are also made towards
enriching LMS with some advanced features.

Requirement Engineering:

and formal techniques like conceptual graph, use case diagram, and VDM-SL specification are used to model the requirements. A RE framework is proposed to alleviate the use of different methods in a collaborative way. The LMS domain is represented using conceptual class diagrams and the ontology. In addition, a verification model is introduced to ensure traceability between the requirements at different levels of abstraction.

Software Architectural Design:

In order to introduce formalism to the standard graphical representation of AD using UML, we propose a new set of stereotypes that extends the UML component diagram to depict AD of a system based on Acme style. One prime objective of such AD is to automate the conformance verification of the high-level design of LMS to LTSA conformance rules. The Acme code, equivalent to the graphical AD style, serves as a formal basis of the AD in the verification phase. First, the LTSA conformance rules, originally stated in NL, are represented formally in the form of conceptual graph diagrams and then used in an algorithmic process for checking conformance with the AD specification. The domain knowledge, represented by ontology plays an important role in this verification. The proposed methodology also defines a measure for the conformance level to understand the extent to which an AD artifact conforms to the LTSA standard.

Software Detailed Design:

In the detailed design phase, we propose a methodology to bridge the semantic gap between requirement specification and design artifacts. This work is a continuity of our work on RE phase where we ultimately produce a requirement specification using combination of use-case, VDM-SL and conceptual graph. A design framework is proposed to integrate the use of different methodologies to convert the requirement specification of the RE phase into an implementable and verifiable design artifact. The initial descriptions of the requirements at the specification using use-case diagrams are elaborated in the design phase in terms of Class diagram, Sequence diagram, and Activity diagram, where each of them representing different perspectives of modeling. We use an ontology description of the domain, which depicts the basic hierarchical structure of the domain.
CHAPTER II- REVIEW WORK

components in the form of object, process, and entity, to avoid conflicting assumptions between requirement specification and design artifact. The VDM-SL design specification is the extension of the VDM-SL requirement specification where the operations and data types of the latter are elaborated in accordance with the ontology. We propose for two types of verification on the design artifact produced at this level. Since the semi-formal UML diagrams and the formal VDM-SL specification are disjoint in nature, we first propose a verification technique to check consistency between the design artifacts. Secondly, we propose a model based verification approach to ensure traceability of the requirement between the design component and the requirement specification.

Designing Interface Requirements:

Considering the web-based nature of LMS applications, we introduce a methodology to perform formal representation and verification of the interface requirements at the design level. We propose two levels of add-ons to VDM-SL for modeling functional and interface requirements in an integrated way. The level-1 specification is more suitable for functional requirements at the requirement analysis model, while the interface requirements are stated in an abstract way. In contrast, the level-2 specification is more suitable for the compound requirements, i.e., interface-requirements being stated in a compound way with functional requirements at design level. At this level, interface requirements are described in a more detailed way, keeping the functional requirements intact. We propose a method for creating the level-1 specification and then convert it into level-2 specification, using a proposed conversion technique. We also introduce a technique to develop a FSM model from the level-2 specification. This model is then used for verification of the design against the requirements' specification by generating the test cases from the model.

Additional features in LMS:

Learning through hyperlinked media often leads to confusion as the reader reaches to multiple unknown terms while trying to understand a particular term. It becomes even harder when one tries to understand those unknown terms by following further such links. Consequently, they get confused about the learning path (where to initiate from and what is the sequence to reach to the target). We
propose two generic approaches towards the solution of this problem. Our first approach uses data mining techniques to generate FP from a graph representing a set of hyper-linked key words. This graph is loaded with weighted edges by the usage of those terms in contact-mode learning environment. Then our proposed mining algorithm search for any hidden association between the terms and use those associations to build a learning path. The second approach proposes a data structure for nodes representing a term. A tree is constructed from the web-based repository of the definitions. It represents the dependencies among the terms related to a definition. Then we propose a priority based post order traversal algorithm to recommend a learning path for a particular target term.

We endeavor to enrich the personalized learning environment of an LMS by introducing two different approaches – first, by a scaffolding technique for individual support to the learner, and then by recommending a personalized learning path to guide the learner. The proposed scaffolding is developed by using Online Analytical Processing (OLAP) techniques on LMS usage data. It helps the learner and the instructor in using the learning tools like live tutorial and question answer session in the synchronous medium like chatting, video calling, conferencing etc. The scaffold provides us useful information to the instructor about the learner's behavior collected from her self-regulated learning environment to make the system more adaptive and personalized. The second approach extends the generic solution for learning path by using an Ant Colony Optimization based searching technique to find out a learning path that suits the criteria of an individual learner.

Finally, with the intend to make web-based learning more flexible and adaptive to the needs of individual learner, we have developed a small prototype of LMS, named CompTutor, to implement our proposed strategies on adaptive learning. It provides individualized tutoring to the learners with help of five components: student model, domain model, tutor model, communication model and interface model. In CompTutor, we focus on adapting three important tasks namely sequencing, navigation and learning style. The required recapitulation of knowledge for individual learner can be fetched from the learner model using a set of keywords identified as 'hitch-words' for that particular learner on that learning session. Then we control the navigation activity by adjusting the...
CHAPTER II- REVIEW WORK

The domain model provides all possible hyperlinks for a keyword and the tutor model adjusts the links based on the knowledge level of the learner. For adapting learning style, we provide two simple alternatives, 'concept followed by examples' and 'examples followed by concept'. The learning contents are then adjusted accordingly to suit the preferred learning style as chosen by the learner.