A SOFTWARE ENGINEERING APPROACH FOR DESIGN OF EDUCATIONAL TECHNOLOGIES

Thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy
in
Computer Science & Engineering

by

SRIDHAR CHIMALAKONDA
200999011
`sridhar_ch@research.iiit.ac.in`

SOFTWARE ENGINEERING RESEARCH CENTER
International Institute of Information Technology - Hyderabad
Hyderabad - 500 032, INDIA
February 2017

International Institute of Information Technology

Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "*A SOFTWARE ENGINEERING APPROACH FOR DESIGN OF EDUCATIONAL TECHNOLOGIES*" by SRIDHAR CHIMALAKONDA has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Advisor: Prof. Kesav Vithal Nori

Dedicated to

matrudevo bhava, *Ammi*

pitrudevo bhava, *Dad*

acharyadevo bhava

The unique teacher of my life

Prof. Kesav Vithal Nori

The Man of Fundamentals

atithidevo bhava, *The entire universe*

Acknowledgments

It has been an incredible journey for the last several years towards learning, discovering, re-discovering and even re-inventing [...] while simultaneously encountering successes and failures along the way. The beauty of interdisciplinary research has enthralled me, so are its challenges.

Firstly, I would like to wholeheartedly thank my advisor Prof. Kesav V. Nori, for his valuable guidance throughout. Loved his intellectual insights with humour, very rare to be associated with someone like him, who is critically sound and fundamentally rooted. It gives me immense pleasure to thank Dr. Raghu Reddy, Head of SERC who has given technical feedback and primarily for his pragmatic support. Dr. Venkatesh Choppella for his valuable inputs and my colleagues Naveen Kulkarni, Kirti, Sai Krishna, Amulya (Gita) and all the lab members. Sai Gollapudi (Sai garu) requires a special mention for his valuable guidance not just for research but in general about life :) and Lalit bringing energy to the lab. Team from Virtual Labs for being so nice with me. Thanks to Pushpalatha garu of SERC for processing all my travel grants quickly.

Special thanks to Prof. Sridhar Iyer and Prof. Kannan Moudgalya and friends from IIT Bombay. Sandhya ma'am of IIIT-Hyderabad has always been supportive. The entire fraternity of IIIT-Hyderabad including Prof. P.J. Narayanan, Dr. Vasudeva Varma, Prof. Jayanthy Sivaswamy, Prof. Jawahar, Prof. Rajeev Sangal, Prof. Kamal Karlapalem who have supported me quite well during this journey and Prof. T.V Prabhakar for his moral support. Appaji Sir and Phani Sir for their amazing support!

I was lucky enough to travel to six international conferences during my PhD and to get extensive feedback on my work both from software engineering and educational technologies communities. I would like to particularly thank Professors David Notkin for his inspiring stories, Neno [Nenad Medvidovic], Andy, Andre, Tom, Jo Atlee, Laurie Williams, Helen Sharp, Margarett Brunette and friends Reid Holmes, Fritz, Michalis... and the entire software engineering community. Prof. Pankaj Jalote, Dr. Srinivas Padmanabhuni and the entire Indian Software Engineering (ISEC) community. Prof. Kinshuk for his valuable time for reviewing my work, Prof. Demetrios G. Sampson for his great suggestions for journal papers and Prof. Nian-Shing Chen for his boost up on everything. Many other well-wishers and friends from educational technologies community like Vincent, Panagiotis Zervas, Imran, Carlo Giovannela, Sabina, Jelena and others for their inputs and feedback on my

thesis. Especially the sessions with these people role playing as my committee members has triggered several research directions beyond my thesis.

Dr. Dan Hyung Lee, Head of Working Group 4 in ISO/SC7 [Software and Systems Engineering Standards] for nominating me as Co-Editor of two international standards in software product lines. His confidence in me has inspired me to continue my research in software product lines. Prof. Rajat Moona for allowing me to be Co-Editor from Bureau of Indian Standards.

Special thanks to Prof. TV. Prabhakar, Dr. Venkatesh Choppella and Prof. Sanjay Goel for their valuable suggestions as thesis committee members; especially for supporting both the directions of software engineering and educational technologies in my thesis.

Harshit Harchani, Shivam Khandelwal, my BTP students and Adithya Chimalakonda for their immense help in supporting the development of technologies as part of this thesis.

I would also like to thank Shri YSK Seshu Kumar, Joint Secretary and Deputy General of National Literacy Mission Authority and Chairman of CBSE. Sanukta Mudgal, Dr. Sayanna, SRC Hyderabad, A. Satyanarayana Reddy, Director of Adult Education, Telangana and Director, TSLMA for hosting the software generated from this thesis on official websites of Government of Telangana and facilitating field visit to villages.

Finally and most importantly, I would like to thank my parents and brother for their unprecedented love and support throughout the thesis (I cannot express my thanks through words here).

Oh My God!!! Many thanks to God!!!

More by divine intervention than by choice, I have also researched "*On the nature of relationships*" for around three years during my PhD, which has been a great learning as well. I would like to wholeheartedly thank my advisor once again for his immense support during these challenging times, without which I would not be here today!

Abstract

This thesis is driven by the idea of advancing computing to address critical challenges in the domain of education. In particular, focuses on facilitating the design of educational technologies for scale and variety, where scale is the number of systems to be developed and variety stems from a diversified range of instructional designs [consisting of varied goals, processes, content, teacher styles, learner styles and so on]. More specifically, we address this challenge in the context of adult literacy in India [287 million learners spread across 22 Indian languages and instructional designs developed by 32 state resource centers spread across India]. *How to address these challenges and design educational technologies that adhere to instructional design principles, provide flexibility and are less effort intensive?*

This thesis presents an approach for design of educational technologies for scale and variety in education. The crux of this approach is the application of software engineering ideas like *patterns*, *ontologies* and *software product lines* without losing focus on instructional design. This approach consists of (i) modeling different aspects of instructional design like goals, process and content using *patterns* and mapping them to commonly accepted approaches like Bloom's taxonomy and Merrill's first principles of instruction (ii) representing these patterns using an ontology based framework that systematically captures different aspects of instructional design (iii) based on the modeling of domain through patterns and ontologies, this thesis presents a pattern oriented software product lines approach for modeling a family of instructional designs and for further semi-automatically generating eLearning Systems based on these instructional designs. The core concern of quality of instruction is mitigated by deriving patterns and ontologies from well-established and field tested methodologies and instructional material under the aegis of National Literacy Mission Authority of Government of India.

Finally, we demonstrate how software tools built on top of our approach [`http://rice.iiit.ac.in`] can be used to semi-automatically generate eLearning Systems for flexible instructional designs and multiple Indian languages. We see that the proposed approach for scale and variety based on patterns, ontologies and software product lines could be applicable beyond adult literacy to the broad spectrum of education, and in particular to skill and school education.

Contents

List of Figures

Figure Page

List of Tables

Thesis Contributions in a Nutshell

The research in this thesis is motivated by the philosophy of *advancing computing research for society* [societal challenges $\twoheadrightarrow$ research challenges]. Specifically, this thesis focuses on extending and applying ideas from computing and software engineering towards design of educational technologies for scale and variety in the context of adult literacy in India [287 million learners, 22 Indian languages, varied instructional processes, content]. The research undertaken in this thesis has broadly contributed in the following directions:

Research/Publications

- *How to systematically model different aspects of Instructional Design to facilitate design of educational technologies?* - **Patterns** [IJAE 2017, ICALT 2014], **Ontologies** [ICALT 2013, ICALT 2012], **Systematic structures** [T4E 2012, ANQ Congress 2011], **Tools** [CHI 2013, ICALT 2013]
- *How to reduce effort during design of educational technologies for large scale & variety?* **Systematically organize instructional material** [T4E 2012], **Apply Software Product Lines** [ICALT 2012, ICALT 2013, PLEASE 2013 @ ICSE 2013]

Research/Standards & Industry - *To the following standards in ISO/IEC JTC1/SC7 - Software and Systems Engineering as Co-Editor* [ACM SIGSOFT SEN May 2016]

- ISO/IEC 26551, *product line requirements engineering*, *International Standard*, 2015
- ISO/IEC 26552, *product line architecture design*, work-in-progress since 2014
- ISO/IEC 26553, *product line realization*, work-in-progress since 2012
- ISO/IEC 26554, *product line verification and validation*, since 2012
- ISO/IEC 26555, *product line technical management*, *International Standard*, 2015

- *"A Reference Model for Design of Educational Technologies – Quality, Scale and Variety"*, a proposed draft standard based on my PhD thesis for Indian Committee of ISO/IEC SC36 [Information technology for learning, education and training]

Practical Impact & Open Source Software

- The prototype software developed as part of this thesis is available at `http://rice.iiit.ac.in`

- The mobile version of the generated software is deployed on Google Play Store and is available at `https://play.google.com/store/apps/details?id=iiit.rice.al.telugu`
- The software/app is listed in the official websites of Department of Adult Education of Government of Telangana at `http://tslma.nic.in/` and State Resource Center, Government of Telangana at `http://srctelangana.com/`

Societal Impact

- Recommendations on *(i) quality of education (ii) rapid development of educational technologies* as *Special Invitee*, Steering Committee, Education, *Planning Commission of India, headed by Prime Minister of India*, 2011
- *Designing Technologies for 287 million adult learners*, Thesis presentation to the team headed by *Director General*, NLMA, the highest authority for adult education in India and the Chairman of Central Board of School Education (then).
- The research done in this thesis was appreciated and all state directors were instructed to experiment with our work on the field | Collaborating since June 2016

Key Concepts, Definitions & Notation

We briefly define the concepts and working definitions used throughout the thesis as follows:

Instructional Design - Instructional Design is used as an umbrella term that can refer to an entire discipline or as a process, art, science or technology [5]. Merrill has defined instructional design as the practice of creating "*instructional experiences which make the acquisition of knowledge and skill more efficient, effective, and appealing*" [6]. Berger defines instructional design as a "*systematic development of instructional specifications using learning and instructional theory to ensure the quality of instruction*" [7]. From the definition of Berger, instructional design acts as a basis for quality of instruction. In a similar line, Carroll emphasizes that quality of instructional design leads to quality of instruction [8].

In this thesis, we consider *instructional design as an underlying structure consisting of different aspects of instruction like goals, process, content aimed at (i) providing a base for quality of instruction (ii) facilitating design of educational technologies*[1]

Quality of instruction - Based on Carroll's model of learning [8], we consider quality of instruction as the degree of impact of instructional design towards achieving learner's goals.

Educational Technologies - Educational technologies is a broad concept [9] sometimes used interchangeably with terms like learning technologies and instructional technologies. One common definition from AECT is "*Educational technology is the study and ethical practice of facilitating learning and improving performance by creating, using, and managing appropriate technological processes and resources*" [10]. According to Council for Educational Technology, UK, "*Educational technology can be considered as the systematic development, application and evaluation of systems, techniques and aids to improve the process of human learning.*"

We consider "*educational technologies as a set of processes, techniques, methods and tools that facilitate systematic development of eLearning Systems based on well-established instructional designs.*"

[1]We largely stick to the term "instructional design" in this thesis as generally used in North America and refer to the term "learning design" as used in Europe when referring to specific literature.

eLearning Systems - We consider eLearning Systems as a sub-class of educational technologies that are designed for improving learning and teaching in a particular context. Specifically, we consider eLearning Systems[2] as simple multimedia systems that use audio and visual aspects to teach reading, writing and basic arithmetic corresponding to physical instructional material in the context of adult literacy in India [11]. In essence, *educational technologies* facilitate systematic design of *eLearning Systems*.

Scale & Variety

Scale: We consider scale as the number of eLearning Systems to be developed. For example, it could stem from 22 Indian Languages or instructional material from several publishers. This differs from the traditional definition of scale, which generally refers to the ability of the system to handle exceeding users. *Variety:* eLearning Systems for flexible instructional designs like varying goals, instructional processes, content and so on. Facilitating design of eLearning Systems for different kinds of learners as well as different roles of teachers can also be considered as source of variety.

Patterns describe a generic solution to a recurring problem within in a certain context [12].

An ontology *"is a formal, explicit specification of a shared conceptualization"* [13].

A feature *is a characteristic or end-user-visible behavior of a software system* [14].

A Software Product Line is a paradigm for developing a family of similar but distinct software systems [15][2]. SPLs are mainly used to improve productivity during development of software systems while maintaining consistent quality [2].

"A software product line (SPL) is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission, and that are developed from a common set of core assets in a prescribed way" [16].

We use *ProcessPattern* and the acronym *pasi* (play, act, scene, instruction) interchangeably to refer to our proposed instructional process pattern explained in Section §3.5.2 and *ContentPattern* or *fcrmt* (facts, cases, rules, models and theories) to refer to a pattern to organize instructional material described in Section §3.5.3. We use the term Bloom's taxonomy for ease of reading but actually refer to Bloom's revised taxonomy [17].

Notation and symbols used in this thesis beyond feature models [18], component and UML diagrams are listed in Appendix §C.

[2]We also use the term *i*Primers for eLearning Systems in the context of adult literacy in India

CHAPTER 1

Introduction

In this chapter, we introduce our work that aims to address challenges in educational technologies by applying ideas from software engineering. We discuss the motivation and inspiration to our work that stems from grand challenges in computing and education in Section §1.1. We then present adult literacy case study in India as an instance of this challenge in Section §1.2. We present the problem statement of this thesis in Section §1.3 followed by the proposed contributions of this thesis in Section §1.4. Finally, we briefly define the research scope of this thesis in Section §1.5 followed by an outline of this thesis in Section §1.6.

1.1 Motivation & Inspiration

The role of technology in education has undergone a massive transformation in the 21^{st} century promising to facilitate anywhere, anytime learning to everyone [19][20]. Supporting this trend, a plethora of educational technologies like computer aided instruction, web based learning, game based learning, learning management systems, computer-supported collaborative systems, virtual learning environments have emerged for a wide range of environments and contexts across the globe. Even though these technologies vary on several dimensions, *software* is a central theme of many of these technologies. In addition, there is also a need to constantly improve these technologies catering to emerging trends like personalized learning, gesture based learning, augmented reality, gamification [21] and Massive Open Online Courses (MOOCs) [22]. This further increases the complexity during the design of educational technologies and makes it an incredibly hard challenge to customize and adapt these technologies as per the emerging trends.

On the other hand, despite significant progress in educational technologies, there is also a strong criticism that most of them lack a pedagogical or instructional design basis leading to poor quality of instruction [23] [24] [25]. These emerging trends and issues with current educational technologies present some key grand challenges for computing and engineering:

- Design of technologies to facilitate personalized learning

  "Everybody is a genius. But if you judge a fish by its ability to climb a tree, it will live its whole life believing that it is stupid!" - Albert Einstein

  - Advance personalized learning, Grand Challenge 14 from *Grand Challenges for Engineering, National Academy of Engineering of the National Sciences*[1]
  - Provide a teacher for every student, from Grand Challenge 3 from *Grand Research Challenges in Information Systems, Computing Research Association* [2]
  - A STEM Grand Challenge - Accelerate the development of problem solving, reasoning, and decision making skills [26]

- Improve quality of education using technology

  "Education is not the learning of facts, but training of the mind to think!" - Albert Einstein

  From a technological perspective, we construe these challenges as

"Design and customization of educational technologies for scale and variety while maintaining quality of instruction"

In the next section, we present the adult literacy case study in India as a sub instance of this challenge setting the context for this thesis.

1.2 A Challenge

How to facilitate design and customization of eLearning Systems to teach 287 million adult illiterates in India spread across 22 Indian Languages, who are beyond the age of schooling, earning their livelihood, who speak their native language, but cannot read or write?

The world has undergone a rapid transformation into digital age with over an estimated 7 billion mobile users and around 2.4 billion Internet users worldwide [27]. However, the same world has an estimated 775 million young people and adults who are unable to read or write even in the digital era [28]. Surprisingly, India itself has around 37% of them, who are beyond the age of schooling, speak their language, but cannot read or write and spread across 22 Indian Languages [28].

In addition, according to reports from Government of India, the present average of adult illiterates taught by instructors is around 10[3], whereas even assuming 200 adult illiterates per year for 5 years would still need a dedicated force of 287,000 instructors. The

[1]http://www.engineeringchallenges.org/challenges.aspx

[2]http://archive.cra.org/reports/gc.systems.pdf

[3]Personal Communication, State Resource Center Director, Telangana, India under the programme of "Each one, teach ten"

National Literacy Mission (NLM) of Government of India (GoI) has been striving to address this challenge since 1988 and has created a uniform methodology for teaching adult illiterates across India [29]. In the literature, there were several efforts of using technologies like radio, television and even mobiles to reach out to adult illiterates in India [30] [31].

A technology initiative by Tata Consultancy Services (TCS), an Indian Software Consultancy Services firm, as part of their Corporate Social Responsibility program consists of 9 eLearning Systems for 9 Indian Languages and has made around 120,000 people literate [1]. While these experiments have yielded significant productivity increase over Government of India efforts, with decreasing dropouts and increasing pass rates [1], the instructional design was constant and the eLearning Systems are monolithic in nature making their customization a hard task. With the success of these eLearning Systems on the field, we have focused on improving the productivity during the design of these eLearning Systems. To this end, we have applied the ideas of software reuse and software product lines and were able to reduce effort from 5 to 6 person years spread over 2 calendar years to 5 to 6 person months in 6 calendar months for developing an eLearning System [11]. But this approach is for automating a family of eLearning Systems based on a *fixed* instructional design whereas the dire necessity is to design eLearning Systems for flexible instructional designs and further customize them for 22 Indian Languages and variants. This presents the following key requirements and challenges for adult literacy in India, setting the context for this thesis:

- Facilitate design of eLearning Systems for 22 Indian Languages and dialects *(scale)*.
- Facilitate quality of instruction by basing the design of these eLearning Systems on instructional design.
- Facilitate the design of these eLearning Systems for flexible instructional designs (*varying* goals, processes and content) catering to the *varying* needs of 22 Indian Languages and variants.

Even though these challenges are specific to adult literacy, design of eLearning Systems for other forms of education like schooling, skills, engineering, and customizing them for varied contexts and delivering them in multiple languages makes it a grand challenge. Table 1 shows an example possibility of designing eLearning Systems for six subjects from K1 to K12 with each of them having varied goals/process/content to be delivered in 22 Indian Languages and variants. The problem in these cases is of *scale* and *variety* during the design of these eLearning Systems for varied instructional designs.

Table 1.1 Scope of Educational Technologies - An Example

Class/Subject	1st, 2nd, 3rd Language	Maths	Science	Social
K1	$l_1 ... l_{22}$ | *Varying goals/process/content*	$l_1 ... l_{22}$	$l_1 ... l_{22}$	$l_1 ... l_{22}$
K2	$l_1 ... l_{22}$ | *Varying goals/process/content*	$l_1 ... l_{22}$	$l_1 ... l_{22}$	$l_1 ... l_{22}$
K3	...	...	...	...
K12	$l_1 ... l_{22}$ | *Varying goals/process/content*	$l_1 ... l_{22}$	$l_1 ... l_{22}$	$l_1 ... l_{22}$

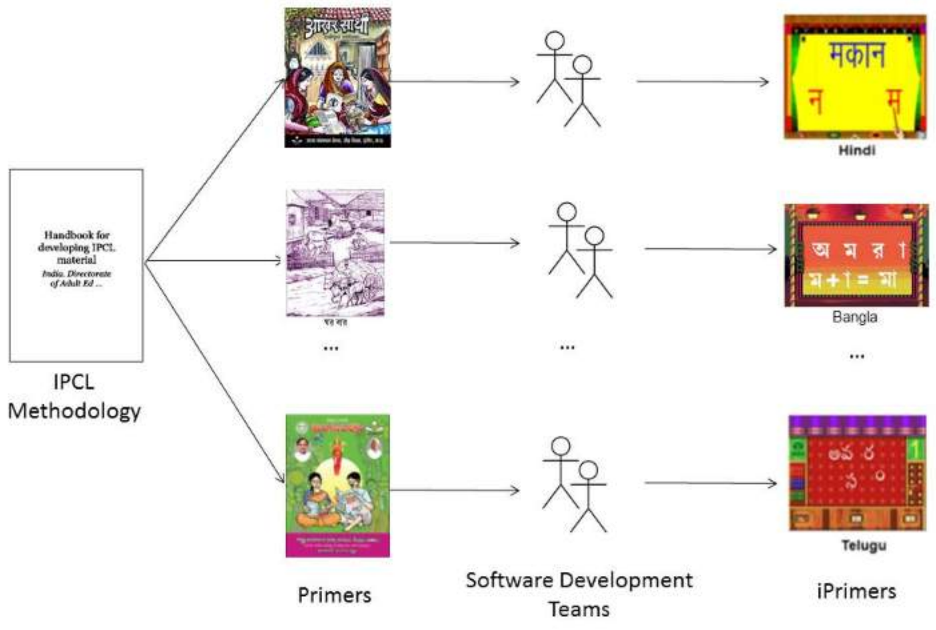

Handbook for developing IPCL material
IPCL Methodology
Primers
Software Development Teams
Hindi
Bangla
Telugu
iPrimers
...

Figure 1.1 Current approach for design of eLearning Systems in adult literacy

1.3 Problem Statement

This thesis explores the possibility of facilitating the design of educational technologies to create customizable eLearning Systems to address the challenges of *scale* and *variety* in the context of adult literacy in India. To this end, the current approach for developing eLearning Systems for adult literacy is presented in Figure §1.1. In this approach, the key input comes from a well-established methodology called Improved Pace and Content of Learning (IPCL) [29], from National Literacy Mission for teaching adult illiterates across India, which is then used as a base for producing instructional material for 22 Indian Languages. This approach is detailed in Section §2.1. With these inputs, eLearning Systems for 22 Indian Languages are developed by software development teams. The main problem with this approach is that these systems are *monolithic* in nature and any change at methodology, instructional material or software development process forces their re-development leading to massive efforts. In addition to the challenge of huge effort, this approach also does not support customization of eLearning Systems for varied instructional designs, which is at the crux of this thesis. This strongly calls for explicit modeling of instructional design and its variants to facilitate *scale* and *variety* inherent in the problem domain.

Instructional Design has gained significant role in the field of Technology Enhanced Learning as an underlying and complex discipline often involving multiple perspectives and connotations. In the context of this thesis, we look at instructional design as an underlying structure that encompasses principles of instruction to facilitate design of educational technologies. To the best of our knowledge, we are not aware of research focusing on modeling instructional design for adult literacy. However, there has been extensive research on modeling instructional design for the last several years resulting in a plethora of educational modeling languages (EMLs) [32] [33] [34] like poEML [35], PALO [36], Web COLLAGE [37] as a way to model and reuse aspects of instructional design. Sampson et al. presented an open access hierarchical framework for integrating open educational resources at different levels of granularity [38]. IMS-LD emerged as a standard for learning design [39] and then focus shifted to tools like LAMS [40] and LDSE [41] that aim to support teachers. A vision paper aimed to create an approach that integrates most of these tools towards an integrated learning design environment [42]. Despite this rapid progress, many researchers have pointed to several shortcomings of modeling and reusing instructional design like complexity of authoring, lack of adequate tool support, interoperability and inability to support teachers [43].

In addition, several researchers have used ontologies as a means to represent different aspects of instructional design. Ontologies have been proposed to capture educational content [36] as well as learning design using IMS LD [44]. LOCO [45] was presented as an ontology to bridge the gap between learning objects and learning designs through context. Extending these ontologies further, an ontology framework was developed for creating an intelligent learning organization [46]. However, these ontologies and tools based on them are tightly coupled with each other and do not support for modeling instructional design variants making it difficult for design of eLearning Systems for *scale* and *variety*.

Designing reusable learning objects is another direction that has received significant attention from several researchers [47][48][49] [50] [51][52] [53] in educational technologies but most of these efforts have not been very fruitful due to lack of emphasis on other aspects of instructional design [54] [55] [56] [57] [58]. A direction that is of interest to this thesis is the use of software engineering approaches for designing a family of eLearning Systems for *scale* and *variety* rather than focusing on individual eLearning Systems. A software product line for generic digital information products was presented in [59] and an approach for development of web-based eLearning Systems focusing on web services and XML was presented in [60]. In our prior work, we have developed an approach called TALES for automating the development of a family of eLearning Systems based on software product lines [11]. However, none of these approaches focus on modeling instructional design as a

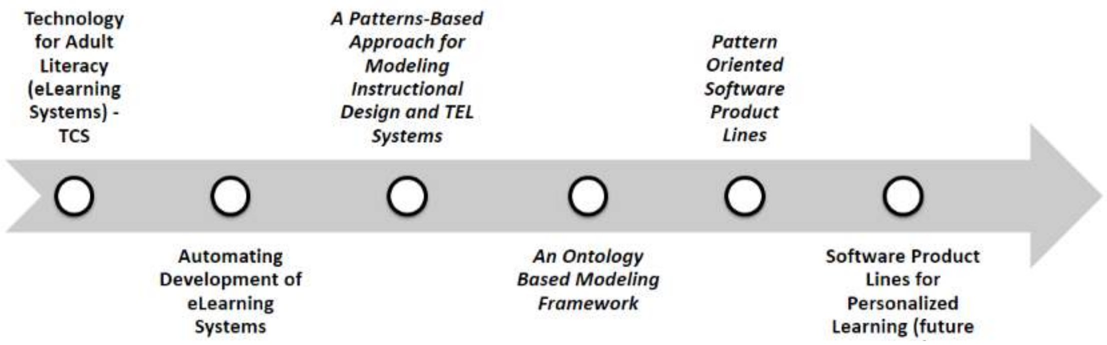

Technology for Adult Literacy (eLearning Systems) - TCS
Automating Development of eLearning Systems
A Patterns-Based Approach for Modeling Instructional Design and TEL Systems
An Ontology Based Modeling Framework
Pattern Oriented Software Product Lines
Software Product Lines for Personalized Learning (future

Figure 1.2 Design of educational technologies for scale and variety - Our research journey

basis for eLearning Systems leading to criticism from quality of instruction perspective and leave supporting flexible instructional designs as an open research problem.

To summarize, existing approaches in the literature focus on either modeling instructional design or on software reuse and not both presenting a strong motivation and need for this thesis.

With this background, the key research goal of this thesis is:

"To create an approach that facilitates the design of educational technologies to support scale and variety in education with a focus on adult literacy in India"

For the last several years, we have been working on creating several technological aids to support adult literacy in India with our research spanning across educational technologies [61][62], software engineering [63] and human computer interaction (HCI) [64]. We have briefly summarized some of these different perspectives in [65]. However, the focus of this thesis is on applying software engineering approaches and principles to accelerate the design of educational technologies for *scale* and *variety* based on well-established learning methodologies. The progress of our research is summarized in Figure §1.2 and is detailed as part of contributions later in this chapter.

1.4 Proposed Approach & Contributions

Design of educational technologies for *scale* and *variety* while maintaining *quality* is a major challenge requiring research from several disciplines like learning methodologies, educational technologies, software engineering and human-computer integration. However, the focus of this thesis is to address an instance of this challenge in the case of adult literacy from a technological perspective. To this end, we rely on the following inputs:

- An educational philosophy that provides a strong basis for learning and teaching.

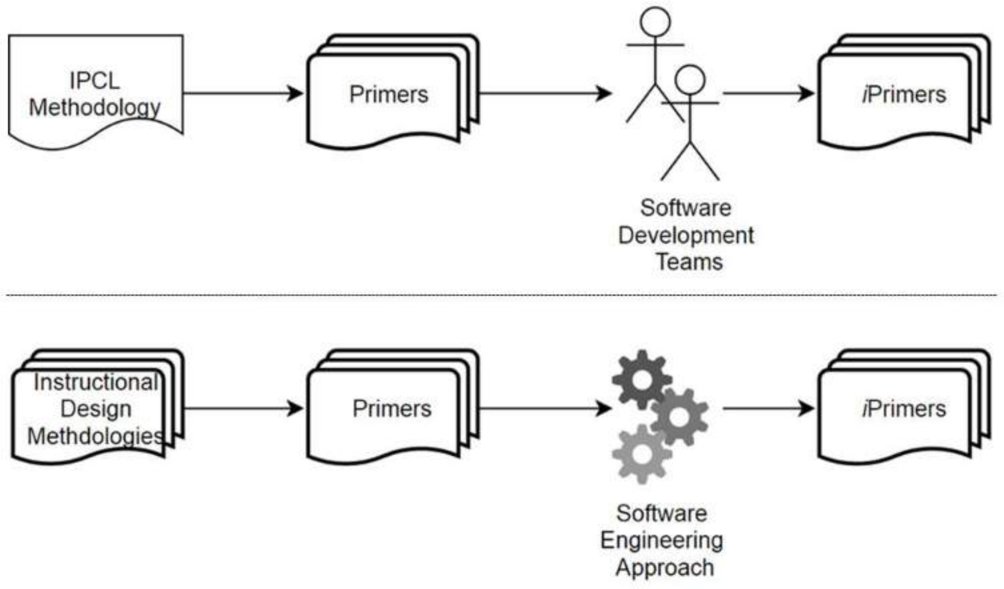

IPCL Methodology
Primers
Software Development Teams
iPrimers
Instructional Design Methdologies
Primers
Software Engineering Approach
iPrimers

Figure 1.3 Existing and Proposed Approach for Design of educational technologies for scale and variety

- Instructional material devised by domain experts based on the above methodology.
- Field tested eLearning Systems based on this instructional material.

Figure §**??** shows a simple schematic of the existing approach (top) and proposed approach (bottom) for design of *i*Primers for adult literacy in India. In the existing approach, individual software development teams develop *i*Primers for every primer and all these primers are based on a single instructional design methodology i.e., IPCL in the case of adult literacy in India. The core idea of the proposed approach is to systematically model different aspects of instructional design using *patterns*, concretely represent them using *ontologies* and then apply a *software product lines* approach for semi-automatically generating eLearning Systems for varied instructional designs and multiple languages. This allows for flexible modeling of instructional designs and creation of customizable *i*Primers. An overview of our proposed approach is presented in Figure §1.4. Even though this approach takes the same inputs as in Figure §1.1, the key difference is the design of educational technologies that can handle the scale and variety for flexible instructional designs instead of re-developing eLearning Systems for every new case and every change in the inputs. To this end, the following are the main contributions of this thesis:

Contribution §1 - Pattern-Oriented Design

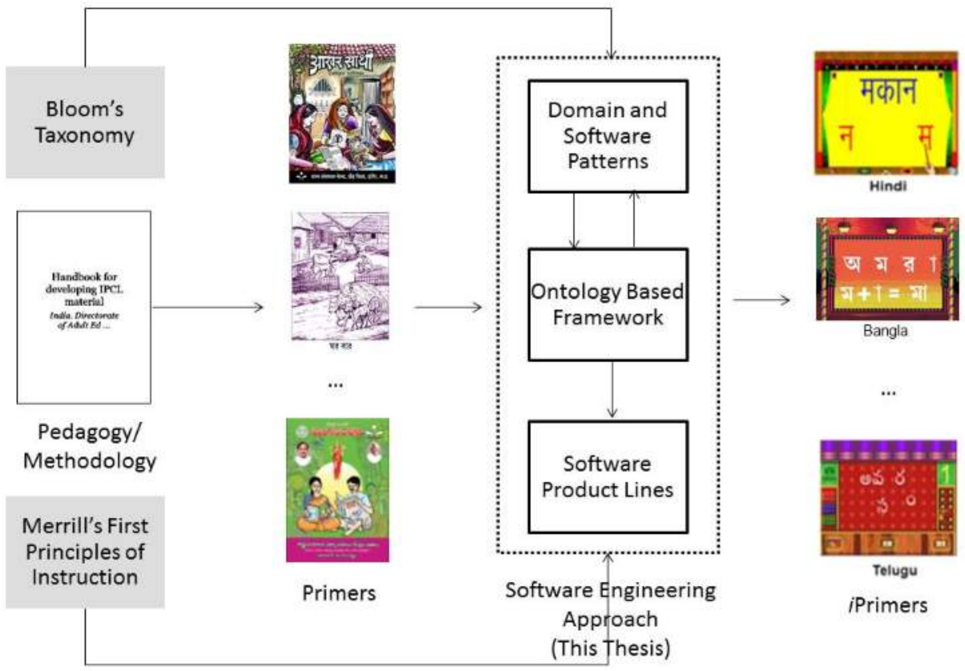

Bloom's Taxonomy
Handbook for developing IPCL material
Pedagogy/ Methodology
Merrill's First Principles of Instruction
...
Primers
Domain and Software Patterns
Ontology Based Framework
Software Product Lines
Software Engineering Approach (This Thesis)
मकान
Hindi
Bangla
...
Telugu
iPrimers

Figure 1.4 Proposed approach for design of educational technologies for *scale* and *variety*

One of the strong criticisms faced by several eLearning Systems today is the lack of an instructional design basis leading to poor quality of instruction [24][66]. In addition, there is also a huge scarcity of qualified teachers across India[4]. As expressed earlier in Section §1.3, the key goal of this thesis is to design educational technologies for *scale* and *variety* rather than a single eLearning System.

Problem: How to base design of eLearning Systems on instructional design while facilitating scale and variety?

We propose *patterns* and *pattern languages* as a first step towards modeling different aspects of instructional design based on commonly accepted principles and practices in the literature. While there is extensive work on patterns for instructional design like Pedagogy Patterns Project [67], E-LEN [68], the main focus has been on patterns from a pedagogy perspective than on designing technologies, which is the focus of this thesis. In addition, researchers have focused either on domain patterns (instructional design) or on technology patterns, and not both. In this context, we propose a patterns-based approach to design of educational technologies based on instructional design. The crux of this approach is to create a structure of solution in instructional design and integrate it with a solution in technology based on patterns. We propose a set of patterns for instructional design in the context of adult literacy in India. More specifically, we propose a pattern for modeling

[4]http://www.hindustantimes.com/india/not-in-the-class-a-story-of-india-s-missing-teachers/story-8HITtop9bbJbL18cGC5U7H.html

instructional material and map it to Bloom's taxonomy [17] and a pattern for modeling instructional process and map it to Merill's first principles of instruction [6].

For example, we proposed a structural pattern called *ContentPattern* for modeling instructional material indicating gradual knowledge progression as *(facts → cases → rules → models → theories)* and an example instantiation for adult literacy is: *ContentPattern* [1]:syllables (म, क, न) → words (नम, मकान) → rules (क +ा = का)) → phonetic model → eclectic method. Here, each of the aspects of *facts*, *cases*, *rules*, *models* and *theories* provide variation points such that they can be varied for different contexts. For example, the above example with variations for Telugu language can be syllables (మ, న, ం) → words (మనం, మరిది) → rules (న +ం = నం. This pattern is driven by goals, which are also modeled using a *GoalPattern* based on Bloom's taxonomy: *Capability (remember, understand, apply, analyze, evaluate, create), Condition, Criteria*. An instantiation is *GoalPattern* [1]: The learner should be able to remember syllables (म, क, न) and recognize them from a newspaper in less than a minute. This approach and patterns are detailed in Chapter §3 of this thesis.

Contribution §2 - An Ontology Based Modeling Framework

In order to achieve the goal of semi-automatically generating eLearning Systems, there is a need to concretely represent instructional design such that it can be processed by tools.

Problem: How to systematically represent different aspects of instructional design based on patterns?

There is an increasingly widespread use of ontologies in educational technologies with several applications; ranging from representation of domain knowledge [69] to the extent of personalization and generating adaptive content based on learner styles [70]. In this thesis, we explore the use of ontologies to systematically represent different aspects of instructional design like *context, goals, process, content, role, evaluation and environment* based on patterns. More specifically, we focus on the aspects of *goals*, *process* and *content* in this thesis. We rely on existing ontologies like ALOCoM [71] for content, IMS LD [72] for process, LOCO [45] for context, as inputs to our framework and adapt and extend them for our specific needs. This framework is detailed in Chapter §4 of this thesis.

For example, *goals* are specified using *GoalsOntology*. Some properties associated with this ontology are *description*, *priority*, *prerequisites* and so on. The *GoalsOntology* points to the *process* through which these goals will be achieved, the *content* that is required and the evaluation to be performed.

Contribution §3 - Pattern-Oriented Software Product Lines

Instructional Design lays a strong foundation for design of eLearning Systems with a pedagogical basis [73]. However, it is an extremely effort intensive activity to adapt and customize instructional designs to cater to varying needs.

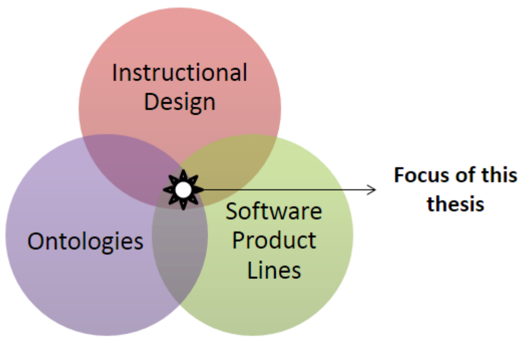


Figure 1.5 Focus of this thesis at the intersection of instructional design, ontologies and software product lines

Problem: How to model a family of instructional designs to drive creation of eLearning Systems for scale and variety?

In this thesis, we explore software product lines as an approach to address this challenge of modeling a family of instructional design variants as well as a family of eLearning Systems for adult literacy. The core idea of this approach is to look at modeling instructional design not as a single problem but as a family of similar but distinct problems. However, this critical challenge of reducing effort for creating and customizing flexible instructional designs is often considered as a one-off problem in the literature rather than a family of problems. In this approach, the key inputs for domain engineering phase of software product lines come from patterns and ontologies of instructional design, which are then customized during application engineering to facilitate the design of customizable eLearning Systems. We have elaborated this approach in Chapter §5 of this thesis.

Evaluation - We demonstrate the core contributions of this thesis through design of two prototype platforms (i) A platform for modeling instructional design variants (ii) A platform for semi-automatically generating eLearning Systems from instructional designs. Using these platforms, we have developed a few prototype eLearning Systems for multiple Indian Languages to demonstrate *scale* and *variety*. The platform uses principles of software product lines and is developed using basic web technologies like *Python*, *JavaScript*, *Jena API*, *XML Schema* and so on. Technical architecture is discussed in Chapter §6 of this thesis. Recently, we have semi-automatically generated two eLearning Systems for *Hindi* and *Telugu* languages.

1.5 Scope of Thesis

The grand challenges introduced in Section §1.2 require rigorous interdisciplinary research cutting across software engineering, educational technologies and HCI. However, this thesis takes a technological perspective and only focuses on facilitating the design of educational technologies for addressing *scale* and *variety* in the context of adult literacy in India. To this end, this thesis relies on IPCL, a well-established methodology for teaching adult illiterates from National Literacy Mission, instructional material that is based on this methodology. In addition, the focus of this thesis is not to study the effectiveness of eLearning Systems for literacy, as we rely on field tested eLearning Systems developed by TCS for 9 Indian Languages, which has made around 120,000 people literate [1].

The scope of this thesis is to investigate approaches to facilitate design of educational technologies for scale and variety and demonstrate it for adult literacy in India.

- This thesis works at the intersection of instructional design, ontologies and software product lines as shown in Figure §1.5.
- This thesis relies on well established learning methodologies rather than creating new ones.
- This thesis does not aim to conduct field studies with teachers or learners but relies on commonly accepted and field-tested approaches, and technologies.
- This thesis utilizes existing ontologies for instructional design as a basis for creation of proposed ontologies.
- This thesis leverages existing open source tools and creates research prototypes for demonstration of ideas.
- Modeling all dimensions of instructional design from all perspectives is an uphill task and in this thesis we confine to the main aspects of *goals*, *process* and *content* and briefly discuss other aspects in the generic context.

1.6 Organization of this Thesis

In this chapter, we have provided an overview of the thesis, its motivations and summarized the key contributions of this thesis. We briefly summarize the organization of this thesis as shown in Figure §1.6. In Chapter §2, we detail the adult literacy case as running example throughout the thesis. We then present related work of the thesis from an instructional design perspective and software engineering perspective and list unaddressed challenges from the literature that are relevant to this thesis. Chapter §3, Chapter §4 and Chapter §5 address the thesis problem stated in Section §1.3 by applying the ideas of *patterns*, *ontologies* and *software product lines* respectively. In Chapter §6, we present two platforms that support evaluation of the ideas in this thesis. In Chapter §7, we present the

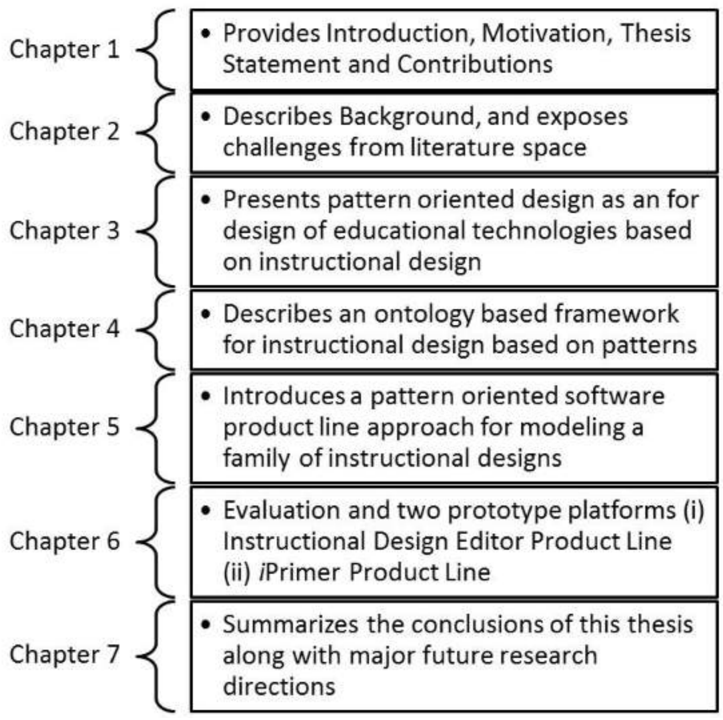

Chapter 1
• Provides Introduction, Motivation, Thesis Statement and Contributions
Chapter 2
• Describes Background, and exposes challenges from literature space
Chapter 3
• Presents pattern oriented design as an for design of educational technologies based on instructional design
Chapter 4
• Describes an ontology based framework for instructional design based on patterns
Chapter 5
• Introduces a pattern oriented software product line approach for modeling a family of instructional designs
Chapter 6
• Evaluation and two prototype platforms (i) Instructional Design Editor Product Line (ii) *i*Primer Product Line
Chapter 7
• Summarizes the conclusions of this thesis along with major future research directions

Figure 1.6 Structure of this thesis

conclusions of this thesis and provide some perspectives on future directions. A part of the work described in this thesis is based on previously published papers listed after §7.2.4.

CHAPTER 2

Background & Related Work

This chapter provides an overview of state-of-the-art in the context of this thesis and presents some background information. We present the background of adult literacy case study in Section §2.1 and summarize literature review in Section §2.2. We then analyze the existing work from the perspective of adult literacy in Section §2.3. In Section §2.4, we critically analyze existing work from educational technologies and software engineering viewpoints to address scale and variety. We end the chapter by presenting the open research problem motivating this thesis.

2.1 Background - The Adult Literacy Case Study

Literacy is one of the key factors in defining socio-economic progress of a country and is vital for a major country like India. On the other hand, India has the highest number of adult illiterates in this world amounting to 37% on this globe [28]. The National Literacy Mission of Government of India has been striving to address this challenge since 1988 [74]. According to NLM, literacy is defined as "*acquiring the skills of reading, writing and arithmetic and the ability to apply them to one's day-to-day life*" and more specifically the focus has been on *functional literacy* that includes:

- Self-reliance in 3 R's (*R*eading, w*R*iting, a*R*ithmetic)
- Becoming aware of the causes of deprivation and moving towards amelioration of their condition by participating in the process of development
- Acquiring skills to improve their economics status and general well-being

Teaching adults requires a different pedagogy and is often considered as andragogy in the field of learning sciences [75]. In [75], Merriam defines an adult learner as someone

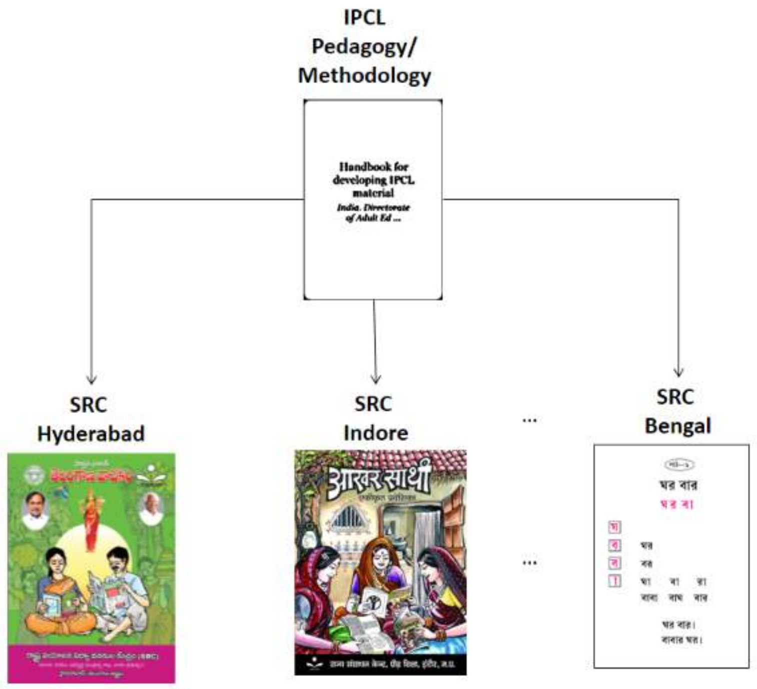

IPCL
Pedagogy/
Methodology
Handbook for developing IPCL material
India. Directorate of Adult Ed ...
SRC
Hyderabad
SRC
Indore
...
SRC
Bengal
...

Figure 2.1 Design of primers (instructional material) for literacy

who has (i) an independent view of learning (ii) extensive prior experience (iii) dynamic learning goals (iv) focus on application of knowledge for practical purposes and (v) gets motivated for internal factors. UNESCO has proposed *Belem* framework to support adult learning at a global level [76]. NLM at the national level has researched and proposed Improved Pace and Content of Learning (IPCL) as a uniform methodology for teaching 3*R*s to adult illiterates in India [29]. It was designed with the intent of providing a base instructional design for official Indian Languages and allows content creators to adapt it for varied contexts across India. Some key aspects of IPCL methodology are:

- Uniform basis for all official Indian languages
- Sustain learners' motivation
- Flexible for adapting the instructional material as per local needs
- Learning by entertaining
- Thematic focus (for different focus learners like women, farmers and so on)

The instructional material (called as *primer*) based on IPCL for different languages is subsequently produced by 32 state resource centers (SRCs) [1] across India as shown in Figure §2.1. Each of the primer is divided into three parts and each part is divided into

[1] http://www.nlm.nic.in/dir.htm

approximately 9 to 12 lessons and each lesson has content focusing on *R*eading, w*R*iting and a*R*ithmetic.

The instructional process using IPCL can be summarized as:

- Starting from common words and phrases known to people of a linguistic region, segment them to discover syllables
- Juxtapose syllables to form words through word play
- Segment syllables and continue the above till we have irreducible phonemes
- Identify the irreducible phonemes with phonetic alphabets
- Discover rules for combining alphabets to reconstruct syllables
- Discover rules to combine scripts of the phonetic alphabets
- Play with these rules
- The full alphabet is learnt by the end of the entire course

2.2 A Summary of Literature Review

Despite several advances of research in educational technologies and using ICTs for adult literacy [77], India is still home to the largest number of illiterates in this world [28]. There is a broad spectrum of work that could be related to research in this thesis. However, as shown in Figure §2.2, we consider related work from the perspectives of educational technologies and software engineering.

2.3 Approaches for Adult Literacy

The UNESCO CONFINTEA at an international level [78] and NLM at national level[2] have devised several initiatives to battle the challenge of adult literacy in India in the past several decades [74]. India has a long history of using ICTs for adult literacy [30] [79] [80]. A radio forum to reach adult learners was experimented as early as 1976 and satellite televisions are used in [30] with computer and laptop based solutions for literacy in [1] and [81] respectively. TCS, an Indian Software Consultancy Services Company has been contributing to adult literacy since 2001 [1]. They have developed eLearning Systems for 9 Indian Languages and also for languages like *Urdu*, *Moore*, *Spanish* and *Arabic* [1]. Figure §2.3 shows the screenshots of their eLearning Systems, which are used by teachers

[2]http://nlm.nic.in

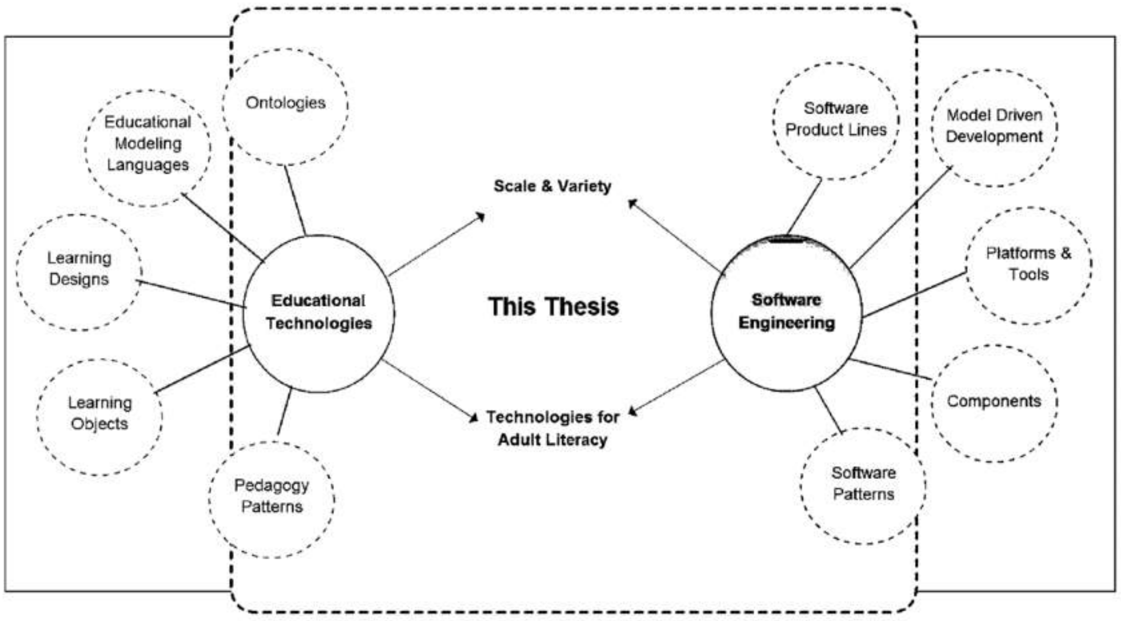


Figure 2.2 Gaps in Literature Review

in the classroom in conjunction to physical books. An approach based on same language subtitling for songs delivered on television increased motivation of learners [82] [83] but focused only on reading skills and is not interactive. The *Bridges to the Future Initiative* (BFI), is a multinational effort that aimed at improving literacy using technology [84] but the focus was on children rather than adult literacy population. It was designed only for one South Indian language (Telugu) because of resource constraints [84]. Hole-in-the-Wall project is another successful initiative based on minimal invasive education but was aimed at children [85].

To harness the emergence of mobiles in developing countries for literacy, a few approaches based on mobile-tablets have been proposed mainly to impart reading skills [86]. A landscape research review of mobiles for reading outlines the need for extensive further research to validate the effectiveness of mobile technologies [86][87]. A study focusing on adult literacy using mobile phones shows promises as well as several challenges [88], mainly in terms of developing the mobile apps for varied languages. Visibility of alphabets on low-end mobile phones is another issue with use of mobile phones in the context of adult literacy eventhough it can be addressed if smart phones are available at lower cost in the future. A report from Indian journal of adult education summarizes the use of ICTs for adult literacy and their inadequacy for mass scale of India [80]. Initiatives like “EduTab” device focus on facilitating adult literacy in India [89], but their effectiveness from pedagogy perspective is unknown as they use their own instructional process and material. A review of tablet software for improving adult literacy suggests the use of games

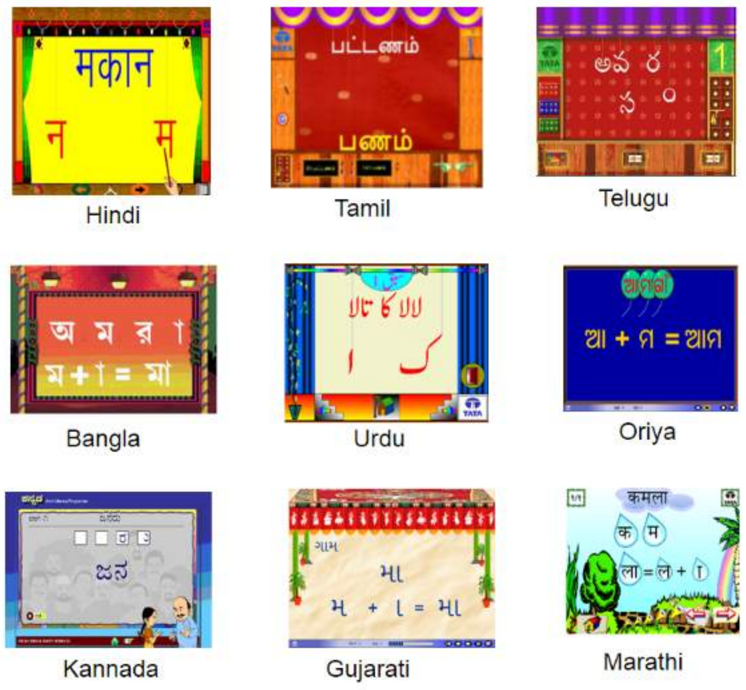


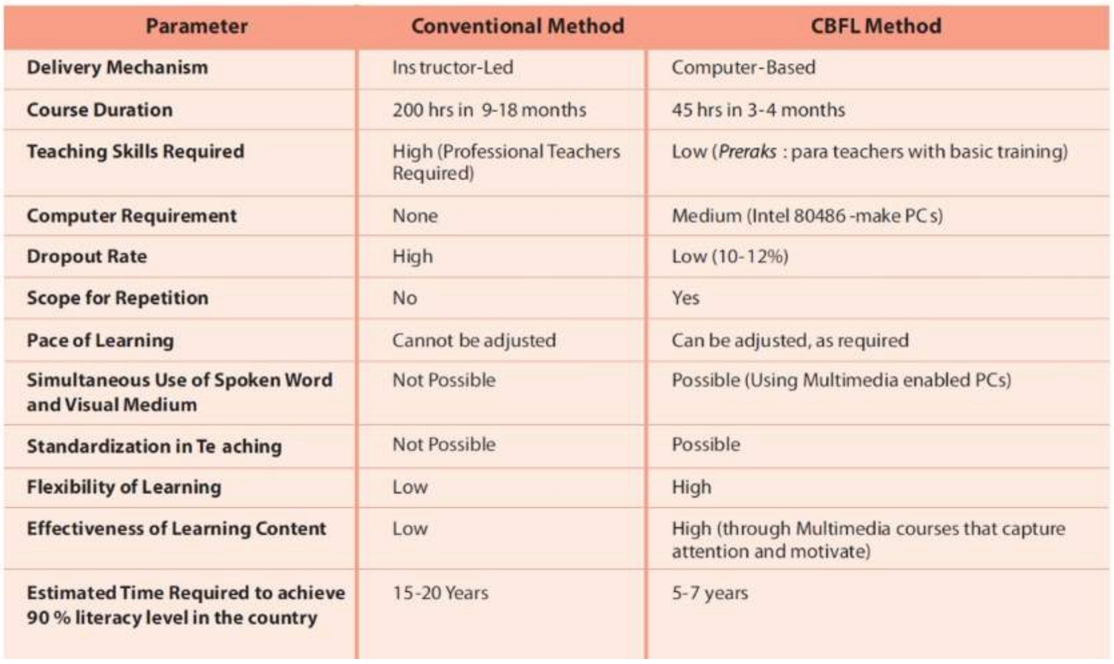
Figure 2.3 Adult Literacy eLearning Systems by TCS

for increasing learner engagement [90] but the tablet software is designed for English and requires significant development effort if it has to be designed for the scale and variety goal of this thesis.

A study from 2011 census data emphasizes the inadequacy of current programmes for adult literacy in India and anticipates that it might take 2050 to achieve 100% literacy rate using current approaches [91]. A research agenda post 2015 for learning and literacy lists 10 key priorities emphasizing the need for technology to be available in the local language of instruction particularly in developing countries [92]. The article also underlines the need for basis of pedagogy for ICT-based solutions for improving quality of instruction in literacy [92], which is severely lacking in most of the current technologies [92].

Of the existing approaches, we rely on TCS' technology for teaching adult illiterates as they are based on IPCL, field tested for 9 Indian Languages and made around 120,000 people literate in the last 15 years across India [1]. Figure §2.4 shows the comparison of results without and with technology for adult literacy through experiments by TCS. On the other hand, an analysis of the literature on technologies for adult literacy alleviates the following key concerns relevant for this thesis:

- Teaching adult illiterates requires a different pedagogy [93] [29]
- Lack of pedagogical basis for ICTs dents quality of instruction [23]



Parameter	Conventional Method	CBFL Method
Delivery Mechanism	Ins tructor-Led	Computer-Based
Course Duration	200 hrs in 9-18 months	45 hrs in 3-4 months
Teaching Skills Required	High (Professional Teachers Required)	Low (*Preraks* : para teachers with basic training)
Computer Requirement	None	Medium (Intel 80486 -make PC s)
Dropout Rate	High	Low (10-12%)
Scope for Repetition	No	Yes
Pace of Learning	Cannot be adjusted	Can be adjusted, as required
Simultaneous Use of Spoken Word and Visual Medium	Not Possible	Possible (Using Multimedia enabled PCs)
Standardization in Te aching	Not Possible	Possible
Flexibility of Learning	Low	High
Effectiveness of Learning Content	Low	High (through Multimedia courses that capture attention and motivate)
Estimated Time Required to achieve 90 % literacy level in the country	15-20 Years	5-7 years

Figure 2.4 Comparision of adult literacy statistics without and with technology, Source: TCS CBFL [1]

- Existing work focuses on designing educational technologies for one or few languages whereas the need is for approaches that work for the scale of 22 Indian Languages and variants

- Existing work does not address the needs of varying instructional *goals*, *process*, *content* and other aspects

2.4 Approaches for Scale and Variety

This thesis is primarily concerned with facilitating the design of educational technologies for scale and variety, where scale represents the number of systems to be developed and variety represents the different kinds of systems to be developed in education domain and more specifically in the context of adult literacy in India. As described in Section §1.5, this thesis cuts into the areas of instructional design, ontologies and software product lines.

The role of technology in education seems to be emerging since decades [94] and there has been an extensive literature on trying to understanding its varied roles, potentials, issues and challenges on the way [95][96] [97] [98] [99] and ways forward [20]. While there are several issues and concerns that restrict the use of technology in education [100] [66], one major concern is the immense effort for development of technologies based on instructional design [101] while another critical concern is diminishing quality of instruction [23]. In this section, we discuss existing literature related to these aspects from the perspectives of educational technologies and software engineering.

2.4.1 Patterns

Bednar et al. have stated that "... effective instructional design is possible only if the developer has reflexive awareness of the theoretical basis underlying the design . . . [it] emerges from the deliberate application of some particular theory of learning" [102]. This strong need to have a pedagogical basis for design of educational technologies has been emphasized in the literature by several researchers [103] [24] [104] [105] [106]. A critical and detailed analysis of the need to bridge learning theories and technology enhanced learning environments is elaborated in a 2014 journal article [107]. To address some of these concerns, patterns were proposed as a potential solution to capture best practices of teaching [108]. The core idea of *patterns* and *pattern languages* is the encapsulation, modeling and delivery of expert's knowledge and best practices to novices in a discipline. Essentially, patterns are derived from experiences and provide abstract representations of recurring solutions to recurring problems in a given context [12]. The roots of patterns are claimed to be in the field of architecture [12] and extensively practiced in software engineering mainly for improving quality of software design and facilitating reuse [109] [110].

The Pedagogy Patterns Project was a major effort to capture best practices in the area of teaching and learning as a way to document best advices for teachers and support quality of instruction [108] [111]. The E-LEN project is another initiative aimed at providing pedagogically-informed technology and experiences as pattern languages for new institutions mainly focusing on learning management systems [68]. There is also extensive work on patterns and pattern languages for different aspects of teaching and learning [112] [104]. A pattern language for creative learning is presented in [113] and for adaptive learning in [114]. Laurillard has created pedagogical patterns using a design science approach [115]. Patterns and pattern repositories for person centered e-learning were proposed in [116]. More recently, patterns derived from practitioner workshops were documented in [117]. While existing literature on patterns emphasizes the need for capturing best practices in teaching and learning, the focus has been mostly on pedagogy and technology aspects are largely ignored. Even in research that considered technology, there is a huge gap between domain (teaching and learning) and technology patterns motivating further research. The key focus in this thesis is to leverage the potential of patterns for modeling instructional design and use that base for design of educational technologies. Chapter §3 of this thesis elaborates our pattern-oriented design.

2.4.2 Learning Objects, Standards and Repositories

There has been extensive research on learning objects in the domain of technology enhanced learning to facilitate reuse [48] and reduce effort during development of educational technologies [19]. Literature has a number of definitions for learning objects [118] each of them using varied terminologies like learning components, knowledge components, content objects but a consensus on their definition is yet to be achieved [119]. It was observed by Reigeluth and Nelson that teachers often break instructional materials into small, reusable chunks such that they can reuse these materials for other instructional contexts [120]. A widely used definition describes learning objects as any kinds of resources (like books, audio video lectures, animations and so on) that can be reused to support learning [118]. However, along with significant research progress on learning objects, there is severe criticism on learning objects from many perspectives [56][58]. Even attempts to clarify the definitions of learning objects have led to the criticism that "there are said to be as many definitions of LOs as there are users" [56]. However, the emergence of learning object metadata standard from IEEE Learning Technology Standards Committee [121] led to a common definition across many sectors. It defines a learning object as "any entity, digital or non-digital, which can be used, reused or referenced during technology supported learning". A heavy focus on reuse of learning objects from a technology perspective [122] has led to severe criticism from learning perspective [54] [123] [55] [57] [124]. In addition, Wiley has pointed that automated assembly of learning objects is not always possible [125]. Another critical issue was the lack of explicit contextual information for describing, searching, adapting learning objects through learning object repositories [56]. There have been significant efforts that resulted in the creation of several learning object repositories worldwide [126]. McGreal has presented a typology of different types of learning object repositories [127]. A practical analysis of learning objects and repositories revealed that reusing learning objects is still an elusive concept and their benefits in practice still largely depend on the context [58].

2.4.3 Learning Designs, Standards and Repositories

Koper has stated that it is not enough to have learning objects and metadata, but instructional value of learning objects should also be considered for e-learning [128]. To address some of these concerns with learning objects, an initiative was started by the Open University of the Netherlands (OUNL) in 1998 which resulted in a formal educational modeling language [51]. EML was iteratively refined for about three years by a number of experts from diversified backgrounds such as educational technologists, ICT-experts, XML-experts, etc. A working definition was proposed where *"an EML is a semantic notation for units of learning to be used in elearning to support the reuse of pedagogical entities like learning designs, learning objectives, learning activities, etc."*. In this definition, a unit of learning describes the learning design, the resources and the services needed in order to achieve one or more interrelated learning objectives and can be viewed at different levels of granularities like course, module, lesson or curriculum. This idea of a formal educational modeling language has garnered interest from several quarters and a survey of EMLs has geared towards a clear specification of requirements for such a language. These efforts have translated into what is called as IMS Learning Design which was later adopoted as a standard [39] that aims *"to provide a containment framework of elements that can describe any design of a teaching-learning process in a formal way"*.

There have been several initiatives to improve educational modeling languages in several forms like a cognitive approach in PALO [36], a separation of concerns approach in poEML [35], a lesson markup language LMML [129], a visual language for designing instruction like E2ML [130], several approaches based on IMS-LD [72] [131], COLLAGE focusing on collaborative learning [132] and so on. These approaches were supported by tools like *ReCourse Editor* [133], *ReLoad Editor* [133], *ASK-LD Editor* [134] and so on. A classification framework for EMLs was presented in [33] and a number of visual editors for learning designs were reviewed in [34]. A classification and analysis of learning design tools into standards-based, generic form-based, theory-based, knowledge-based has motivated the need to have software design that is linked to theory and practice from a pedagogical perspective [135]. Based on lessons from these earlier efforts, tools like LAMS [40] and The Learning Designer [41] were designed to be intuitive to teachers. A team-based approach for modeling learning designs through an integrated learning design environment is proposed in [42][136]. To facilitate reuse of learning designs, several learning design repositories have emerged similar in spirit to learning object repositories [131]. Boyle proposed a layered approach towards integration of learning design and learning object perspectives [137]. Sampson and Zervas proposed a hierarchical framework and supporting tools to facilitate open access to education and learning in the context of open educational resources [38]. In his survey of ICT-based tools for supporting instructional designers and delivery of learning systems, Paquette underlines the need for a *knowledge based instructional design* and the need for *application of software engineering approaches* to address the increasing complexity of instructional design and engineering [138].

Amidst this rapid progress, researchers have continuously identified several challenges of IMS-LD [139], its expressiveness [140], conceptual complexity [141] [142], lack of authoring tool support [43][143]. Prieto et al. have used five learning design tools to model a sample *"healthy eating"* lesson and found that different tools are designed for different purposes with different levels of granularity for different stakeholders and have varied representations [144].

An analysis of existing literature on learning designs and tools reveals that most of the efforts in this direction are either too focused on pedagogy or on developing tools for specific contexts. It is here that thesis focuses on applying software engineering principles towards systematic modeling of instructional design.

2.4.4 Ontologies for Instructional Design

Since Gruber's definition of ontology as "*a formal, explicit specification of a shared conceptualization*" [13], ontologies have gained immense importance in several domains for its wide range of applications [145] [146] like in software engineering [147], enterprise modeling [148], requirements engineering [149]. These ontologies are of different kinds ranging from informal light weight ontologies to formal ontologies depending on the degree of formalism and the power of expressivity [150]. Happel et al. have discussed the advantages of ontologies over conceptual models and meta-models [151] as follows:

- Enable new and efficient way to the information reusability.
- Enable to extend easily.
- Provide consensus on the understanding of domain knowledge.
- Support better understanding of domain knowledge.
- Define problem and solution domain knowledge separately.
- Assist in analyzing the structure of domain knowledge.
- Facilitate a machine to use the knowledge in an application.
- Share common semantics among people and applications.

In the domain of education, ontologies have gained immense importance with applications ranging from explicit representation of domain knowledge to automatic generation of personalized content [152]. Mizoguchi and Bourdeau have identified four key requirements of instructional authoring systems (i) adaptivity (ii) explicit conceptualization (iii) standardization to facilitate reuse (iv) theory-awareness, and proposed knowledge and ontological engineering as a potential solution to cater to these requirements [153].

One particular use of ontologies that is of interest to this thesis is to model instructional design theories and learning design standards [154] during design of educational technologies. A 10-year research effort has resulted in creating a comprehensive ontology covering instructional design knowledge for various instructional theories and adhering to learning design standards [155]. SMARTIES is a scenario-based instructional authoring tool based on this ontology and advocates the design of educational technologies based on educational theories modeled as ontologies to facilitate quality of instruction. However, the inherent complexity of the ontology and SMARTIES tool made it tough for its practical usage [156].

While focusing on quality of instruction is one aspect, using ontologies in education to facilitate reuse is another critical research direction that received extensive attention in the literature [153]. Devedzic explored the notion of ontologies for intelligent tutoring systems (ITS) based on inspiration from software patterns in 1999 [157]. Ontologies to formalize learning object content models have been proposed in [158]. To facilitate flexible content reuse, the Abstract Learning Object Content Model (ALOCoM) ontology and a set of supporting tools were proposed in [158]. Amorim et al. have proposed a learning design ontology based on IMS LD specification through a set of 20 design and run time axioms [44]. The basic premise of this ontology was to explicitly and precisely address the drawbacks of IMS LD specification [44]. But isolated research on learning objects and learning designs have made reuse difficult motivating the need for a bridge ontology focusing on context [45]. A formal ontology was presented for representing instructional design methods and provides a rule catalogue to verify the conformance of ontologies for a particular instructional design theory [159]. An ontology and a software framework focusing on competencies was discussed in [160]. However, the creation of these ontologies is not based on patterns, which is the case in this thesis. Furthermore, the existing ontologies are not aimed at scale and variety, which is the need in this thesis.

2.4.5 Software Reuse for Design of Educational Technologies

Development of software components for the domain of education started way back in 1999 [161]. But the use of software engineering approaches in educational technologies has garnered significant attention with the advent of reuse of learning objects [47]. Design principles from software engineering were borrowed to facilitate reuse of learning objects [49]. However, this emphasis itself has led to severe criticism on software engineering being misused in the context of learning objects from a learning perspective [56]. Researchers have used model driven development to facilitate reuse of learning objects [162]. Dodero et al. further proposed a model-driven approach to learning design, a domain-specific language and a tool based on this approach to facilitate modeling of learning designs [163]. A model-driven development approach for learning design using the LPCEL Editor was proposed in [164]. But despite the advantages of these generative approaches, it was noted that the complexity of authoring process increases because of model development required from domain experts [165]. The term educational software engineering was coined in [166] but the focus has been on games for software engineering education.

One major direction of research that is relevant to this thesis is the application of software product lines for accelerating educational technologies. A software architecture for design of e-learning platforms is proposed in [60]. Pankrautius has proposed a software product line approach for digital information products in [59]. A software product line approach towards automating the development of a family of eLearning Systems for adult literary in India was proposed in [11]. Even though these approaches show significant productivity improvements in terms of reducing effort [11], none of them are based on principles in the domain of education creating a significant gap between instructional designs and the software platforms that are built on them.

2.5 An Open Research Challenge from Literature Review

Analyzing literature from the perspectives of educational technologies and software engineering for design of educational technologies for scale and variety, we see limited existing work for adult literacy case study as the focus was not on scale and variety, which is the crux of this thesis. Even when the research was applied for cases other than adult literacy, the focus has been on either educational technologies or on software engineering and not both, presenting the need for this thesis and hence the problem statement.

How to facilitate design of educational technologies for scale and variety in the context of adult literacy in India while maintaining quality of instruction?

CHAPTER 3

Pattern-Oriented Design

The first section §3.1 of this chapter starts by discussing core guiding principles in computing towards a patterns based approach. We then present an overview of the source of patterns, their evolution and how to document them in Section §3.2. How patterns can help scale and variety is discussed in Section §3.3 followed by a patterns based approach for design of educational technologies in Section §3.4. Section §3.5 presents the idea of pattern-oriented instructional design with its sub sections proposing patterns for goals in Section §3.5.1, instructional process in Section §3.5.2 and instructional material in Section §3.5.3. We end the chapter with Section §3.6 by discussing how the domain patterns can be connected to software patterns.

3.1 Guiding Principles

To support design of educational technologies for *scale* and *variety*, the proposed approach is based on the following underlying principles.

It is a dire necessity to improve flexibility and re-usability while reducing complexity during the design of educational technologies and some of these concerns are tackled by the software engineering community through a set of fundamental principles [167]. One principle that is of interest to this thesis is the notion of ***separation of concerns*** that helps in handling different dimensions of a system while improving re-usability and reducing complexity [168] [169]. This principle can be used to separate concerns in the form of layers either horizontally or vertically; views [170]; modules [171]; aspects [172]; patterns [169]; features [14] and so on. ***Modularity*** is a specialization of this principle that deals with separating software into components [171] while ***Abstraction*** is another specialization that hides complexity of the system enabling designers to focus on specific concerns [167]. Domain-driven design is another fundamental principle that tries to address complexity by emphasizing domain as the basis for software design [173]. We apply these principles throughout the thesis for systematically modeling instructional design in the form of patterns, identifying ontologies for these patterns and in modeling features of eLearning Systems from a software product line viewpoint. In the next section, we will summarize some key aspects related to patterns and get into the crux of our pattern-oriented design approach.

3.2 Source, Evolution and Structure of Patterns

Source and Evolution of Patterns - During the professional journey of an expert in a field, the expert encounters several recurring problems and different ways of solving those problems. When a group of experts in a particular field communicate, discuss, debate and document their experience, they realize that certain solutions work and do not work in particular contexts. This leads to the idea that the primary source of patterns is literature or experience, either own or documented experience in the form of best practices and guidelines. Granularity of a pattern is another critical aspect that has to be considered during the design of patterns. For example, in the field of software engineering, there are coding idioms (a kind of patterns), design patterns, architectural patterns, each representing an increasing granularity of abstraction. Pattern languages help in connecting these patterns by describing the relationships between them and how they can be integrated to address specific problems. The agreement on whether a particular knowledge is pattern or not usually comes through a consensus among a group of experts in the particular field. The Hillside group has been sponsoring a series of conferences along with workshops named Pattern Languages of Programs (PLoP) since 1994 [174]. These venues provide a forum for pattern authors to gather, discuss, learn and document patterns. This series has led to development of communities of patterns like *AsianPLoP*, *EuroPLoP*, *ScrumPLoP* in order to create patterns with a consensus from local communities. A pattern typically goes through the phases of *discovery*, *specification*, *validation*, *application* and *maintenance* and is continuously revised based on updated pattern knowledge [175]. Figure §3.1 shows a typical process during pattern development. In the domain of education also, patterns have mainly emerged from participatory pattern workshops [176].

In essence, patterns generally emerge from literature or experience, and are generally specified by an expert or pattern modeler. Once the pattern is specified, it is exposed to the community for review from experts in the field for a consensus of the pattern. This pattern is applied by pattern users in varied problem contexts and they provide feedback leading to update of pattern.

Structure of Patterns - There are patterns everywhere and at different levels of granularity expressed in different ways by different experts in different communities [177]. This presents the challenge of what constitutes pattern knowledge and how to capture it. Tra-

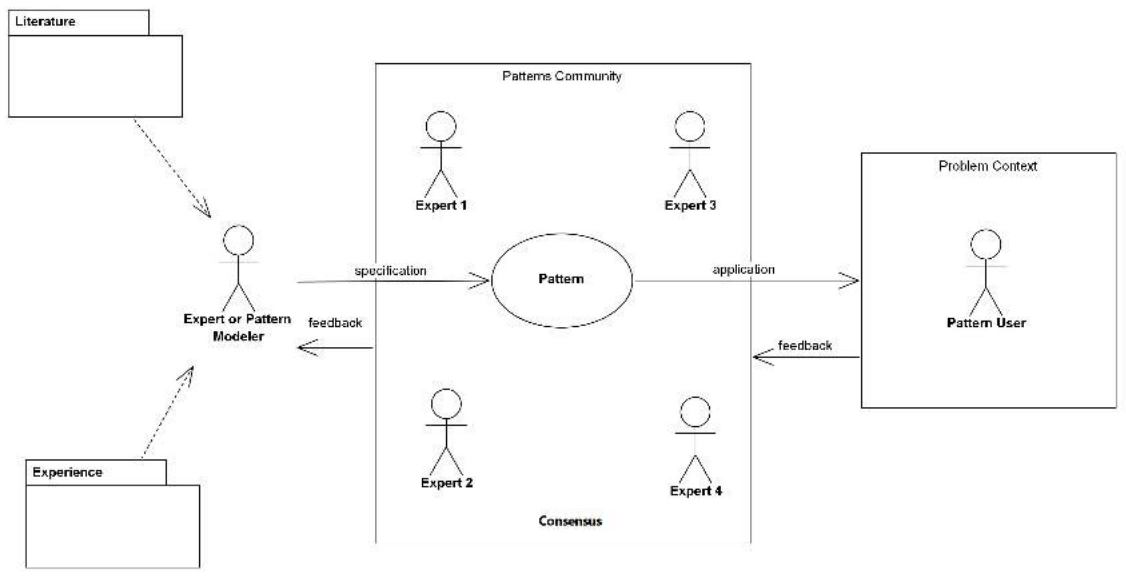
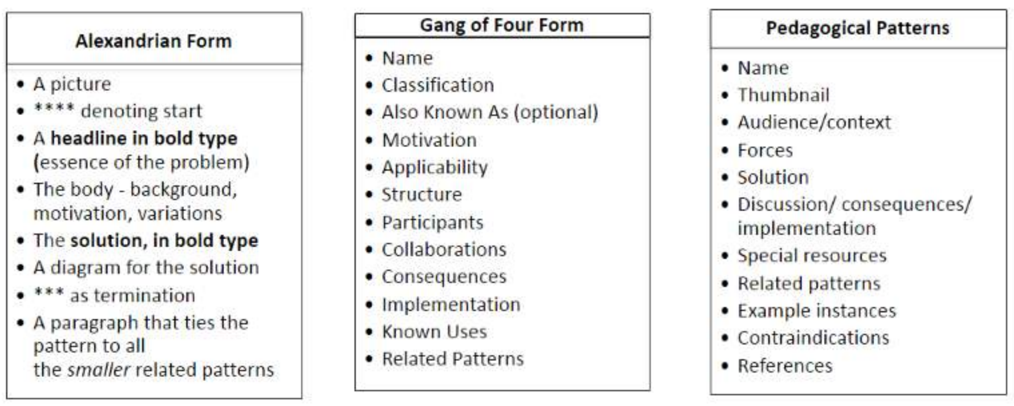


Figure 3.1 Overview of patterns life cycle



Alexandrian Form

- A picture
- **** denoting start
- A **headline in bold type** (essence of the problem)
- The body - background, motivation, variations
- The **solution, in bold type**
- A diagram for the solution
- *** as termination
- A paragraph that ties the pattern to all the *smaller* related patterns

Gang of Four Form

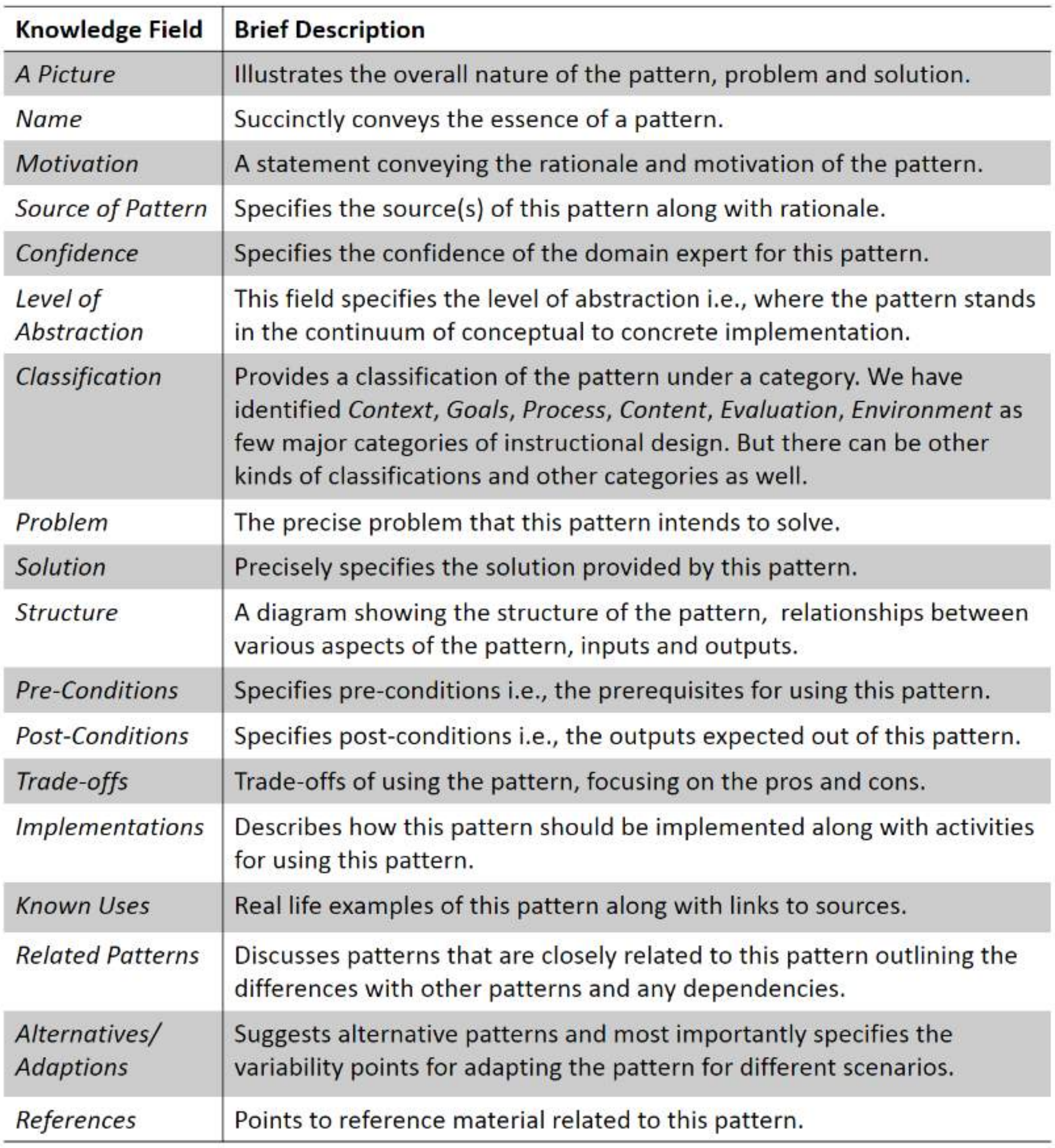
- Name
- Classification
- Also Known As (optional)
- Motivation
- Applicability
- Structure
- Participants
- Collaborations
- Consequences
- Implementation
- Known Uses
- Related Patterns

Pedagogical Patterns

- Name
- Thumbnail
- Audience/context
- Forces
- Solution
- Discussion/ consequences/ implementation
- Special resources
- Related patterns
- Example instances
- Contraindications
- References

Figure 3.2 Diversity of structure of patterns

ditionally, patterns are captured as a description of *context*, *problem*, *solution*. However, several researchers have proposed different structures for capturing pattern knowledge. Figure §3.2 shows examples of commonly used pattern structures from the domains of software engineering [109] and pedagogy [67]. Alexandrian form is the most commonly used format for representing patterns [12]. Using this form, Alexander has documented a pattern language comprising of 253 patterns for the domain of towns, buildings and construction [12]. The 23 Gang of Four (GoF) patterns are the most commonly used patterns in the domain of software design [109] and use the meta data as shown in Figure §3.2. The Pedagogical Patterns project uses the structure shown in Figure §3.2. These are just a few examples of pattern structures used by different communities. A pattern language for representing patterns itself has emerged from the community [177].



Knowledge Field	Brief Description
A Picture	Illustrates the overall nature of the pattern, problem and solution.
Name	Succinctly conveys the essence of a pattern.
Motivation	A statement conveying the rationale and motivation of the pattern.
Source of Pattern	Specifies the source(s) of this pattern along with rationale.
Confidence	Specifies the confidence of the domain expert for this pattern.
Level of Abstraction	This field specifies the level of abstraction i.e., where the pattern stands in the continuum of conceptual to concrete implementation.
Classification	Provides a classification of the pattern under a category. We have identified *Context*, *Goals*, *Process*, *Content*, *Evaluation*, *Environment* as few major categories of instructional design. But there can be other kinds of classifications and other categories as well.
Problem	The precise problem that this pattern intends to solve.
Solution	Precisely specifies the solution provided by this pattern.
Structure	A diagram showing the structure of the pattern, relationships between various aspects of the pattern, inputs and outputs.
Pre-Conditions	Specifies pre-conditions i.e., the prerequisites for using this pattern.
Post-Conditions	Specifies post-conditions i.e., the outputs expected out of this pattern.
Trade-offs	Trade-offs of using the pattern, focusing on the pros and cons.
Implementations	Describes how this pattern should be implemented along with activities for using this pattern.
Known Uses	Real life examples of this pattern along with links to sources.
Related Patterns	Discusses patterns that are closely related to this pattern outlining the differences with other patterns and any dependencies.
Alternatives/ Adaptions	Suggests alternative patterns and most importantly specifies the variability points for adapting the pattern for different scenarios.
References	Points to reference material related to this pattern.

Figure 3.3 A detailed pattern structure

3.3 Patterns for Scale and Variety

Figure §3.3 shows a detailed pattern structure derived from existing pattern representations. The key idea of this pattern structure is to capture as much information as possible such that this metadata could be used for searching and managing patterns and pattern repositories. In addition to the textual information as shown in Figure §3.3, patterns could be represented visually focusing on *structure* as well as *behaviour*. However, one critical requirement for patterns in this thesis is to facilitate domain experts to model pattern knowledge without overburden. We will motivate the need for modeling detailed pattern knowledge at a lower level of granularity in the form of ontologies in Section §4.1.

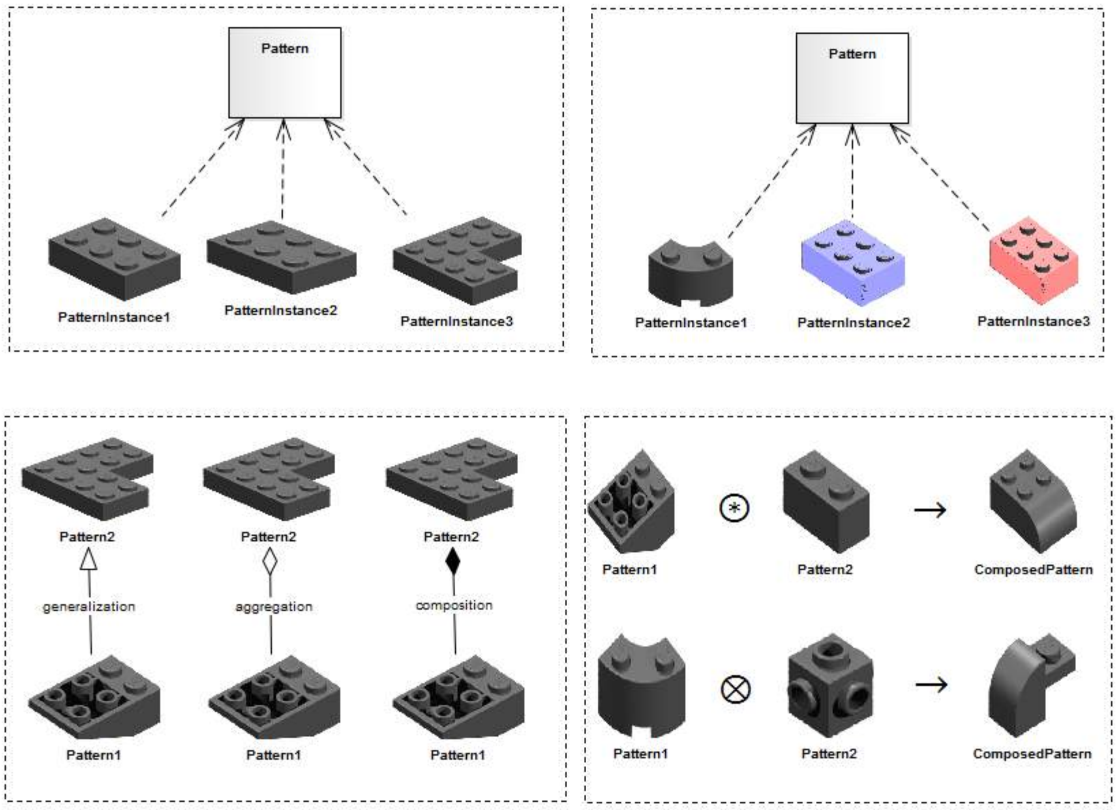


Figure 3.4 Some pattern variations, relationships and compositions

In his seminal work on patterns, Alexander emphasizes that a solution in pattern can be used "*a million times over without ever doing it the same time twice*". We extend this notion to include not just the solution but also the problem, its variations, representations of the pattern, different aspects of the solution and its variants to facilitate scale and variety.

There are several possibilities of creating variants of patterns. The simplest case being that a pattern can be instantiated multiple times as shown on the left hand side of Figure §3.4 or instantiated with simple variations as shown on the right hand side. In Figure §3.4, we show a simple example of different kinds of possible relationships between two patterns in a domain. If we change these relationships, we get a different view of the pattern in context leading to a variation. Similarly, patterns can be composed using different operators as shown in Figure §3.4. Changing the operators results in new composed patterns and variations. If we move beyond just two patterns and consider a large number of patterns, then different ways of composition with multiple operators leads to different ways of modeling the domain. Consider the case of an instructional design with 10 patterns, then different compositions of these patterns can lead to several instructional design variants. This leads to creation of varied instructional designs catering to the needs of specific requirements. The essence of this discussion is to emphasize that modeling domain in terms of *patterns* facilitates systematic creation of several variants, which is one of the primary goals of this thesis. We will elaborate more on these patterns with examples throughout this chapter. But formalizing the representation and composition of patterns is beyond the scope of this thesis and is outlined as future work.

3.4 A Patterns Based Approach

The notion of patterns has its roots in the field of Architecture [12] but was adopted in other disciplines like software engineering [109] and interaction design [178] among others. Whilst there exists several definitions and views, *patterns are primarily concerned with the idea of finding recurring solutions to recurring problems in a certain context*. According to Alexander, the emphasis has to be on pattern languages that facilitate the assembly of patterns in order to create numerous possible solutions, rather than patterns themselves [12]. Buschmann et al. have distilled existing literature on patterns and proposed the following uses in their Pattern Oriented Software Architecture (POSA) series [179][110]:

- *Capturing, Documenting and Communicating Experience*
  - Patterns document existing best practices built on tried and tested design experience
  - Patterns identify and specify abstractions that are above the level of single objects, classes, and components
  - Patterns provide a common vocabulary and shared understanding for design concepts
  - Patterns are a means of documenting software architectures
  - Patterns capture experience in a form that can be independent of specific project details and constraints, implementation paradigm, and often even programming language
- *Construction of Systems*
  - Patterns support the construction of software with well-defined properties
  - Patterns help in building complex and heterogeneous software architectures. Every pattern provides a predefined set of components, roles and relationships between them

In this thesis, we apply the idea of patterns primarily for (i) *capturing experience* (ii) *providing instructional design as basis for educational technologies* (iii) *facilitating reuse during design of educational technologies*. Specifically, we use patterns for modeling domain (instructional design) and software. We also look at patterns as a central way to encapsulate

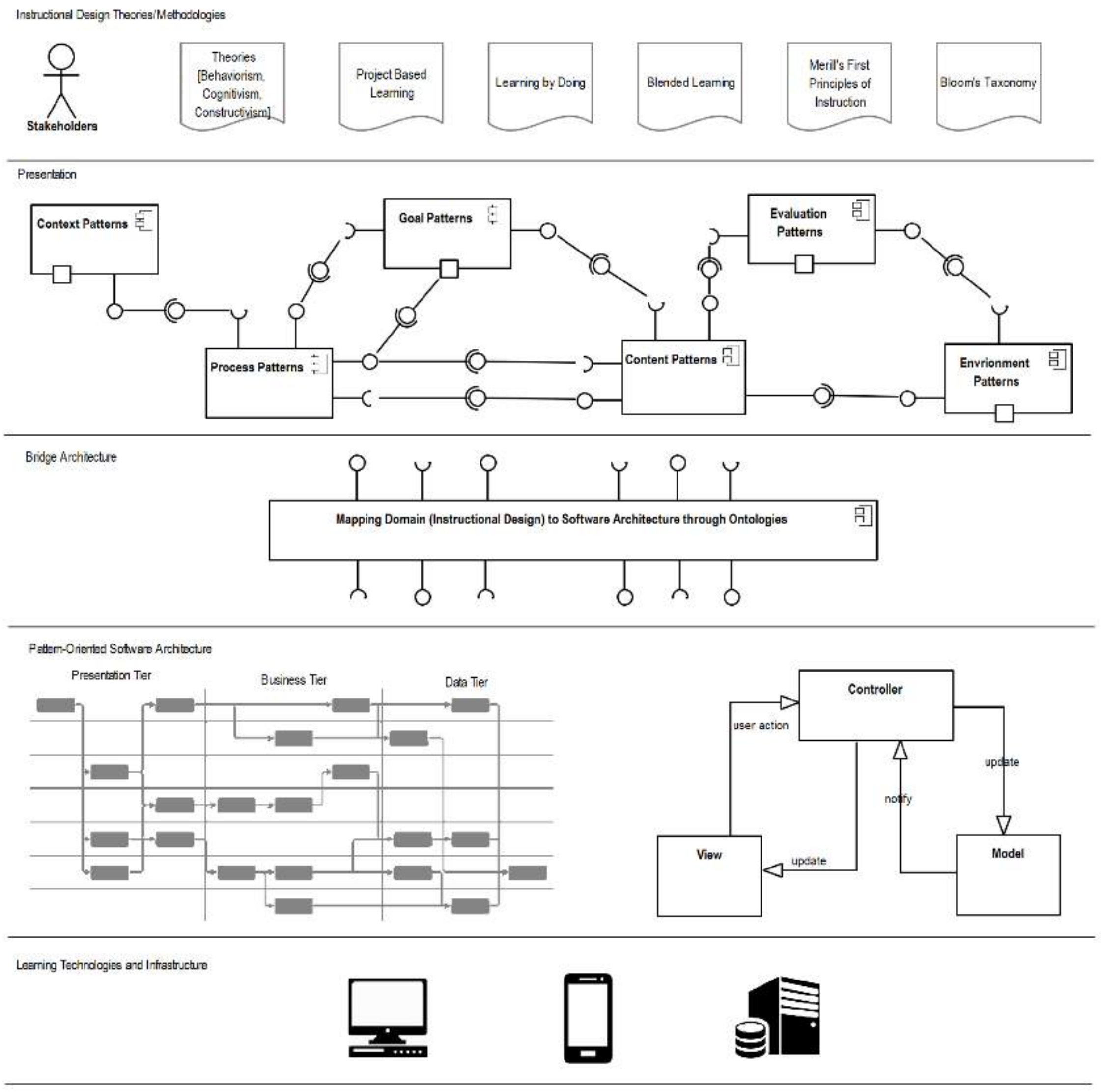


Figure 3.5 Architecture of patterns based approach

commonly understood knowledge of experts (instructional designers, software architects) and facilitate use of this experience by naïve professionals. This is extremely important in a discipline like TEL with huge scarcity of expert teachers at all levels of education. Figure §3.5 presents the core architecture of our proposed patterns based approach to design of educational technologies. This architecture stems from fundamental principles in software engineering and integrates multiple architecture styles (Layered, Domain-Driven and Component-Architecture). This architecture shows a holistic perspective (top-down and bottom-up) and tries to integrate patterns in both domain as well as software through five layers.

A *Instructional Design/Methodologies* - lays a pedagogical foundation for design of educational technologies

Educational experts have long emphasized that developing educational technologies without strong instructional basis is futile and can lead to poor quality of instruction [24] [180]. Most of the educational technologies today require huge effort from teachers in configuring the technologies rather than on focusing instruction [181]. This is further aggravated with a huge dearth of qualified teachers. The first layer from the top in Figure §3.5 focuses on providing a pedagogical foundation for design of educational technologies. We rely on well-established principles and practices from a pedagogical perspective in this thesis and specifically focus on how we can structure these practices towards systematic design of educational technologies. For the case of adult literacy in this thesis, we rely on IPCL approach detailed in Section §2.1 as a pedagogical foundation. This pedagogy is further integrated with other commonly accepted approaches likes Merrill's principles of instruction from teaching or instructional process perspective [6] and Bloom's taxonomy from learning perspective. The rationale for these principles is presented in Section §3.5.1 and Section §3.5.2 of patterns.

B *Pattern-Oriented Instructional Design* - aims to model instructional design using patterns [discussed in Section §3.5]

C *An Ontology Based Modeling Framework* - that acts a bridge between domain and software platforms [elaborated in Chapter §4]

The key purpose of this bridge layer is to connect from instructional design (domain) to educational technologies (software) through a common interface. We used ontologies as the primary mechanism to represent knowledge from patterns and to connect with the rest of the software architecture [182]. We extended the existing literature on instructional design, and proposed an ontology based framework called ID*ont* for systematically modeling instructional design [182].

D *Pattern-Oriented Software Architecture* - models the software platforms for educational technologies using patterns

In their seminal work that inspired successful use of patterns in software engineering, Buschmann et al. emphasize that the primary purpose of architectural patterns is to create a fundamental structural organization schema for software systems [179]. In our approach, instructional architecture patterns derived from POID and software architecture patterns that drive POSA provide this structure for instructional design and educational technologies. We use common interfaces to bridge these different types of patterns at different levels of abstraction *(instructional architecture patterns $\rightarrow$ instructional design patterns $\rightarrow$ software architecture patterns $\rightarrow$ design patterns)*. Essentially, POSA represents the architecture of educational technologies. In the fourth layer and on the left hand side of Figure §3.5, we have a three-tier architecture of a TEL system that implements two important architectural patterns from [179] (i) From Mud to Structure (Layers) – allows controlled decomposition of overall system (ii) MVC and several design patterns (Strategy, Composite, Factory and so on). On the right hand side is an MVC architectural pattern that is derived from instructional architecture patterns and instructional design patterns in POID.

E *Technologies, Platform and Infrastructure* provides the delivery platform with a set of technologies and tools. In this layer, a set of technologies and tools are designed to support the proposed architecture and facilitate the semi-automatic development of eLearning Systems.

We will elaborate POID and POSA in the rest of the chapter.

3.5 Pattern-Oriented Instructional Design

The changing landscape of educational technologies requires a diversified range of instructional designs catering to multiple perspectives and views from a diversified range of stakeholders [73]. This is primarily because there is no "one-size-fits-all" solution for all needs. In fact, even the problem itself varies depending on who is viewing it, where does it come from, how critical is it? what are the short term and long term needs? how to address them? their capabilities and resources available at that point of time. This presents a definite need for systematic instructional design such that it can be flexibly modified as per changing requirements. Several researchers have tried to address these concerns through a number of Educational modeling languages (EMLs) [35] [36] [37] [34][183][40][42][136]. Despite significant research, modeling instructional design remained an open research problem with several challenges [42] [43][184] [185]. Our analysis of the state-of-the-art in learning design also revealed that the goal of end-to-end automation (from pedagogy to technology) through tools has resulted in too generic, too complex specifications leading to slow progress in this field and is in line with existing research [181][180].

Instead, we learn from the community and attempt to present a pointed approach towards design of educational technologies for a family of instructional designs in the context of adult literacy in India. *Our work fundamentally deviates from the state-of-the-art as we focus on not one instructional design but on a family of similar but distinct instructional designs.*

One interesting approach that has been undermined and largely unexploited in TEL is the use of patterns and pattern languages for modeling instructional design. There has been a strong recognition for use of patterns in the community but mainly from a learning sciences perspective [178][111]. The Pedagogical Patterns Project provides a huge collection of patterns focusing on pedagogy and does not explicitly consider the use of technology and software patterns [67]. Patterns for person centered e-learning were proposed in [175] based on action research but are not designed for scale and variety. Pattern languages for different aspects of educational design [186] also ignored the notion of TEL. Mor has discussed the notion of identifying design narratives as a basis for capturing design knowledge, developing design patterns based on that and showing the practical use of these patterns through design scenarios [187]. More recently, Laurillard has proposed teaching as a design science with the idea of supporting computational and collaborative building of pedagogies [115]. In an attempt towards supporting teachers in practical usage of patterns, a collection of patterns have been presented in [117].

In our analysis of literature, we have observed that patterns and pattern languages are still not widely used either in instructional design or in TEL because of several reasons including:

- Existing approaches focus on patterns and pattern languages either for communication or engineering purposes and not both
- Current approaches focus on patterns mainly from a pedagogical perspective rather than their structure and provide minimal support towards design of educational technologies
- Lack of bridge between domain patterns (instructional design) and technology patterns (software)

To summarize, existing approaches focus on patterns either in domain or in software and not both. Most importantly, none of the existing works focus on *scale* and *variety*, which is the main goal of this thesis. Paquette has summarized the extensive work and progress in the field of instructional design and concluded the strong need for researching and applying ontological engineering and software methods for instructional design and engineering [138]. On the other hand, from a software engineering perspective, domain engineering is a critical activity to address complexity, reuse and evolution needs of software systems [188]. In this thesis, we take a cue from Apel et al. [18] and Czarnecki et al. [189] and consider domain as an area of knowledge:

- that covers the desired requirements of the systems in that area
- includes a set of concepts and terminology understood by practitioners in that area
- and includes the knowledge of how to build software systems (or parts of software systems) in that area

Modeling and structuring domain is a fundamental step that acts as a basis towards facilitating reuse and in this thesis we are concerned with the domain of instructional design.

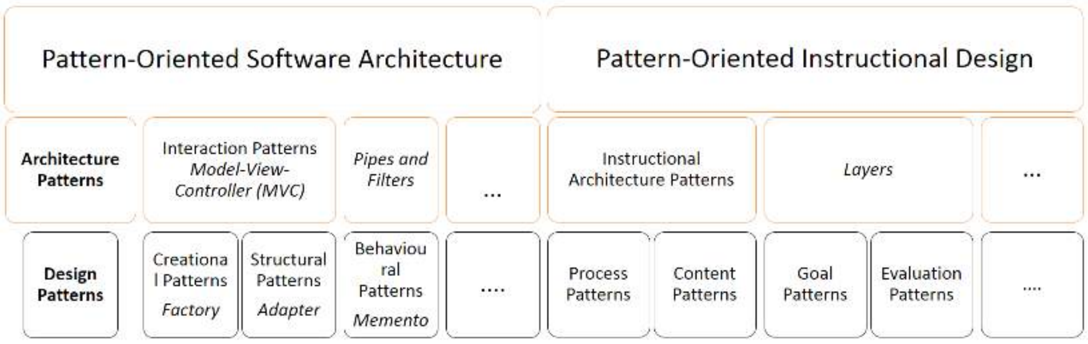

Pattern-Oriented Software Architecture
Pattern-Oriented Instructional Design
Architecture Patterns
Interaction Patterns Model-View-Controller (MVC)
Pipes and Filters
...
Instructional Architecture Patterns
Layers
...
Design Patterns
Creational Patterns Factory
Structural Patterns Adapter
Behavioural Patterns Memento
....
Process Patterns
Content Patterns
Goal Patterns
Evaluation Patterns
....

Figure 3.6 Comparision of Pattern-Oriented Software Architecture and Pattern-Oriented Instructional Design

We propose Pattern-Oriented Instructional Design as *a domain engineering activity* towards design of educational technologies based on instructional design. POID aims at designing a solution in the problem domain in the language of instructional designers and teachers. The key input for POID comes from pedagogies or learning methodologies that provide a basis for TEL. The purpose of this layer is two-fold (i) to structure instructional design for reuse (ii) to facilitate flexible modeling of instructional designs such that educational technologies based on these instructional designs have a pedagogical basis and are prepared for facilitating automation.

We base the design of POID on Pattern-Oriented Software Architecture (POSA) [110] to model instructional design aspects as patterns. Figure §3.6 shows a high level diagram of how POID corresponds to POSA with different levels of granularity. For example, on the left hand side, we have design patterns at the bottom and then architecture patterns. On similar lines, we have instructional design patterns at the bottom and instructional architecture patterns on top of them.

Figure §3.7 shows an overview of Pattern-Oriented Instructional Design. The core instructional design requirements stem from different kinds of instructors and learners, which are then used as input to compose different patterns liks *goals*, *process*, *content* and so on to create a specific instructional design. *A Pattern-Oriented Instructional Design is an integration of patterns at instructional architecture level and instructional design patterns towards designing a specific instructional design for a specific set of educational requirements.* Here, we consider instructional architectural patterns as high-level organizing structures that address recurring high-level problems in instructional design. For example, Can we have a pattern that allows different evaluations for the same instructional goals? These patterns are essentially an integration of instructional design patterns, which focus on a specific aspect like goals or process and so on. One popular example of an architectural pattern for interactive systems in software engineering is the Model-View-Controller (MVC) that allows flexible design of user interfaces [190]. Can we have these kinds of patterns for

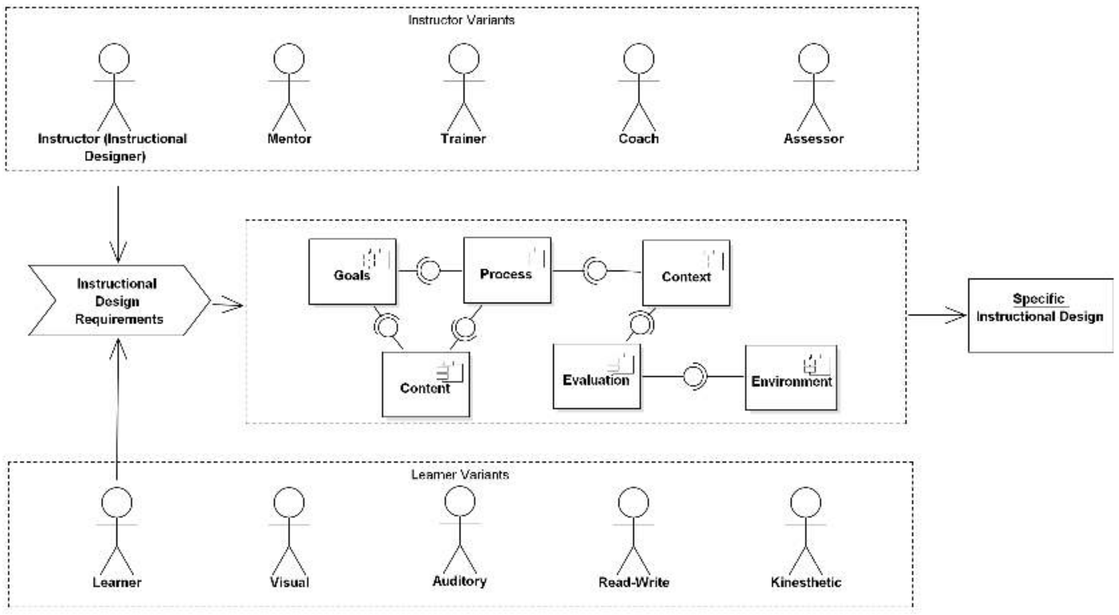


Figure 3.7 An overview of Pattern-Oriented Instructional Design

instructional design? It is here our POID approach is a direction integrating patterns at a higher level of abstraction.

The POID process starts by understanding instructional design and then progresses towards a detailed analysis of the instructional design to identify patterns in the domain. Then these patterns are continuously refined and relationships between them are identified and established. Figure §3.8 shows a classification of patterns in POID into categories of *Context*, *Goals*, *Process*, *Content*, *Evaluation* and *Environment*. This list is primitive and is designed such that it can be adapted and extended by instructional designers to support evolution. Each of these categories have patterns that aim to address a particular aspect of instructional design. For example, *Goals* in instructional design can be represented in various forms like Bloom's or Gagne's taxonomy or ABCD technique, and each of them can have their associated patterns, from which the teacher or instructional designer chooses patterns for their particular context. In our approach, every pattern provides and requires a set of interfaces that clearly define the boundaries of that pattern and how that pattern communicates with the rest of the patterns. The focus in this thesis is not just on patterns or pattern categories, but on integration of these patterns towards specific instructional designs.

Figure §3.5 shows two possible instructional architecture patterns; one that integrates goals and evaluations (goal provides an interface ↔ evaluation requires an interface). The core idea of this pattern is to provide a flexible architecture that allows changing of goals and evaluations in an instructional design with less effort. Another possible pattern could be an integration of processes and content. This kind of abstraction leads to POID, which in itself is a technology independent solution in the language of problem domain. However, it

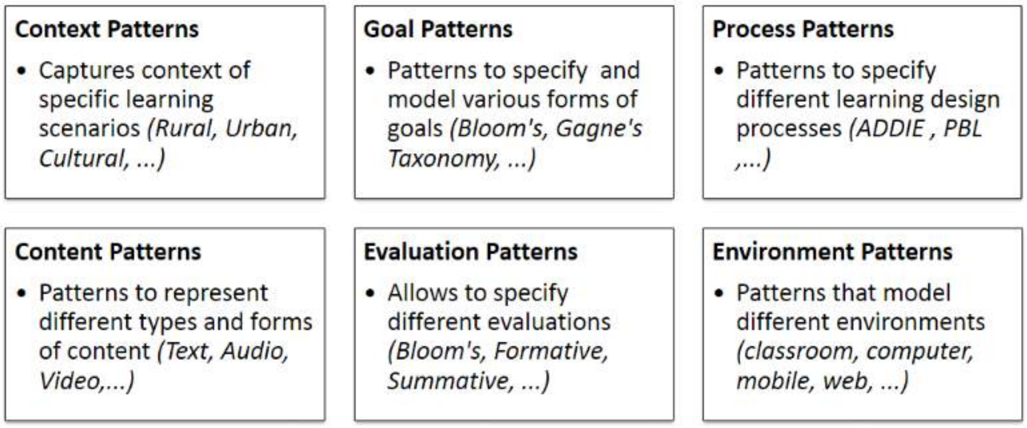

Context Patterns
• Captures context of specific learning scenarios (Rural, Urban, Cultural, ...)
Goal Patterns
• Patterns to specify and model various forms of goals (Bloom's, Gagne's Taxonomy, ...)
Process Patterns
• Patterns to specify different learning design processes (ADDIE , PBL ,...)
Content Patterns
• Patterns to represent different types and forms of content (Text, Audio, Video,...)
Evaluation Patterns
• Allows to specify different evaluations (Bloom's, Formative, Summative, ...)
Environment Patterns
• Patterns that model different environments (classroom, computer, mobile, web, ...)

Figure 3.8 A classification of patterns in instructional design

is important to note that patterns were never proposed to be specific and complete solutions rather provide a basic structure of a generic solution to a family of problems [110], which have to be further adapted and implemented for a specific context. Finally, Alexander hints that the problem of scale and variety can be addressed using patterns in his seminal work [12] as:

"The elements of this language are entities called patterns. Each pattern describes a problem that occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice."

This emphasizes that patterns provide an opportunity to find a generic solution to a problem and then find several customized solutions to a family of similar problems. Even though patterns are generic in nature, they have to be discovered from literature and experience for a particular context. In this thesis, we use our decade-long experience in the case of adult literacy in India. *We conducted a workshop in collaboration with Tata Consultancy Services and in association with NLMA to discover our patterns with the community towards a consensus*[1]. The workshop consisted of directors of State Resource Centers representing officers who are responsible for creating instructional process and material for teaching adult illiterates based on local requirements. Specifically, we introduced our patterns for instructional process and content and incorporated their feedback. We also held interviews with the directors of SRCs to figure out their use of technology for adult literacy especially to gather what kinds of technologies have been accepted and what are the hindrances to the use of technologies. One key learning that came out of that workshop is a strong requirement that the technology has to be customized as per local needs. One director specifically noted that "the instructional process that you follow in a state like *Bihar* and the

[1] at TCS, Hyderabad in 2011

one in a state like *Tamilnadu* differ and the technology has to be adapted as per varied needs". In addition to our experience, we also rely on commonly accepted approaches like Bloom's taxonomy [17], Merrill's first principles of instruction [6] as a basis for the patterns proposed in this thesis.

In the following section, we detail some of the instructional design patterns we discovered in the context of adult literacy in India.

3.5.1 A pattern for modeling goals

We assume that any instruction is goal-driven; making it critical to explicitly state instructional goals. Several terminologies like "learning goals", "learning outcomes", "learning objectives" have been in use to discuss goals. But the crux is to explicitly state and represent these goals such that they become clear to different stakeholders like teachers, learners, policy makers and so on. In our case, for design of educational technologies while adhering to instructional design principles, they can also help in tracking the progress of learners, evaluation and in helping teachers to improve instructional strategies. We advocate a *goal-driven approach* throughout this thesis as it is critical and essential for any instructional design.

For example, if we consider the representation of goals in instructional design, several variations are possible. In the simplest case, a goal can be represented using Bloom's taxonomy and several instances of this goal can be created. If we extend the scenario and think of two goals as part of instructional design, the two goals can be associated in several ways.

The crux of most commonly used Bloom's taxonomy is to provide a classification for organizing educational objectives based on thinking models and to facilitate better communication among stakeholders in education [17]. This taxonomy has evolved since its inception but was originally divided into three domains: *cognitive*, *affective*, *pyschometer* with the first two domains focusing on acquiring knowledge and attitude and the third domain on skills to put that knowledge to constructive use [17]. Each of these domains are further divided into levels that indicate progress *"from simple to complex and concrete to abstract"* [17]. In revised Bloom's taxonomy, goals are organized in the increasing order of complexity in six levels: *remember*, *understand*, *apply*, *analyze*, *evaluate*, and *create* from a cognitive domain perspective. ABCD model proposed by R. Mager is another commonly used framework for writing learning objectives [191]. In this model, a learning objective consists of four components: [*audience*- who] will be able to [*behavior*-perform] [*condition*-constraints] [*degree* - level of quality]. The point of this discussion is to emphasize that there are several ways of modeling goals motivating the need for several patterns for goals in different contexts. In the context of this thesis, based on these inputs and our experience of designing eLearning Systems for adult literacy in India, we propose a pattern to model

Knowledge Field	**Brief Description**
A Picture	
Name	GOALS PATTERN [REASONING, BLOOM'S TAXONOMY, SCALE & VARIETY]
Motivation	This pattern is an attempt towards making goals explicit and documenting them for communication and facilitating automation during design of educational technologies.
Source of Pattern	This pattern emerges from Bloom's taxonomy for structure of goals and IPCL pedagogy for adult literacy.
Confidence	*High*, Bloom's taxonomy is a widely used approach for modelling goals.
Level of Abstraction	This pattern starts with conceptual level but look for implementation section for details of concrete implementation.
Classification	*Goals*
Problem	Concise representation of goals and communicating them to various stakeholders is a serious concern during design of educational technologies.
Solution	Model goals using Bloom's taxonomy focusing on cognitive dimension remember, understand, apply, analyze, evaluate and create.
Structure	The [learner] [beginner, medium, advanced] will be able to [perform] [level] under [constraint/conditions]
Pre-Conditions	Teacher should know the specific goals that are modeled by this particular pattern.
Post-Conditions	The specific goal(s) modelled using this pattern along with its relationship to other goals.
Trade-offs	This pattern helps in systematically structuring and organizing the entire goals in a particular course/lesson but at the same time it might be cumbersome for detailing all the data related to goals as there can be a large number of goals in a particular lesson. The value add of this pattern comes handy when goals are similar and goals have to be defined for not just one course but a family of courses like adult literacy case study.
Implementations	The goals should be filled by using a template based on this pattern, probably a wizard that will help teachers in filling the goals.
Known Uses	Bloom's taxonomy has been successfully used in several cases in the literature but not enough cases of applying it to adult literacy and none specific to adult literacy in India.
Related Patterns	Goals set the direction for entire instructional design. This pattern is closely related to other patterns, mainly with the PROCESS PATTERN [SCALE & VARIETY] which has activities towards achieving the goals, CONTENT PATTERN [SCALE & VARIETY] that helps in organizing the instructional material, EVALUATION PATTERN [SCALE & VARIETY] that is closely associated in evaluation learning with respect to the targeted goals.
Alternatives/ Adaptions	Some alternatives are GOALS PATTERN [ABCD MODEL], GOALS PATTERN [GAGNE'S TAXONOMY] that help in explicitly modelling goals but use different structures and driven by different theories.
References	Refer Bloom's Taxonomy, Revised Bloom's Taxonomy

Figure 3.9 A sample structure and partial instance of *GoalsPattern*

goals based on these inputs. Figure §3.9 shows our proposed pattern for modeling goals based on Bloom's taxonomy.

An example of a goal from IPCL methodology and its modeling using Bloom's taxonomy is as follows:

"Reading aloud, with normal accent, and at a speed of 30 words a minute, a simple passage on a topic of interest to the learner"

A refinement of this goal for specific multiple Indian languages is as follows:

- The learner will be able to *recognize* three syllables "म", "क", "न" in an existing paragraph
- The learner will be able to *recognize* three syllables "క", "ఢ", "◌ా" in an existing paragraph

3.5.2 A pattern for modeling instructional processes

Instructional process is one of the critical aspects of instructional design as it facilitates the fulfillment of goals through a systematic process. However, most of the times it is not explicitly modeled by making it difficult for design of educational technologies. In this section, we will look at a commonly accepted way of teaching in the context of adult literacy in India based on IPCL methodology [29] and present a structure for organizing that knowledge into a pattern. We discuss the instructional process in detail along with teaching philosophy as it forms the basis for a pattern that could be instantiated thousands of times for all Indian languages and dialects.

The structure of instructional process from Improved Pace and Content of Learning (IPCL) for adult literacy [29] can be summarized as follows:

- There are three sets of primers (instructional material), each primer being an improvement on the other in terms of progression.
- Each primer is an integrated one in the sense that it combines workbook, exercise book, tools of evaluation of learning outcome, certification, etc.
- The primers are prepared through workshops involving creative thinkers, writers, linguists and artists, pretested and their suitability affirmed before actual application.
- There are exercises at the end of every lesson and three tests in each primer.
- The entire exercise is based on the principle of self-evaluation and confidence (rather than competition) oriented evaluation. The tests are intended to be simple, non-threatening and participative.

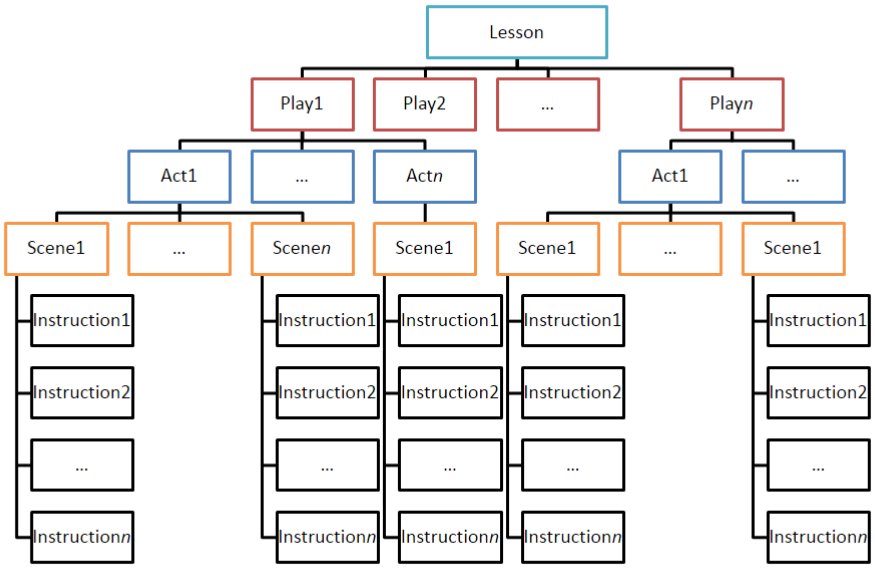


Figure 3.10 Structure of instructional process

- Every learner is free to attain the desired level according to his/her leisure and convenience. It is, however, expected that a learner should be able to complete all the three primers within the overall duration of 200 hours spread over 68 months.

We construe teaching as a process to help the student move from known (cognitive foundations) to build the understanding of, or the derivation of, the previously unknown in a stable and confident manner. Processes are best organized when they have clearly articulated goals. According to NLMA, the overall goal for literacy is the ability to read and write so as to fulfill normal (functional) tasks in day-to-day living in an informed and enabled manner. This needs social awareness of regional and national identity, the needs of a civil and healthy society, and aspirations of the individual, the family, the village, the local region and the country as a whole. All of it cannot be achieved in one go. So the initial focus is on the 3Rs (*R*eading, w*R*iting, a*R*ithmetic). The idea is to provide the basics of 3Rs such that the rest can be learnt through other sources of learning.

For adult learners, literacy instruction can be facilitated by choosing to deploy a socially or functionally relevant theme to provide the corpus for teaching the 3Rs. This theme is only a vehicle to carry the instruction of the 3Rs, that is, what are learnt are the 3Rs, with the side-effect that a meaningful social awareness is also created. The theme provides us with familiar words and phrases which provide the cognitive foundations on which we build further understanding. The end goal of this whole instruction is to understand the real foundations of language (for the purposes of reading and writing) that underlie the cognitive foundations arising from everyday experience, the fundamental sounds depicted by the alphabet, their associated visual representation through script, the sound formation rules that build complex sounds of syllables from the basic or pure sounds of the alphabet along with script formation rules, juxtaposition of syllables to form words, and juxtaposition of words to form phrases and sentences.

This large goal is gradually broken down into simpler sub-goals that all add up. The initial pace of learning is very slow, because the learners consciously understand that there is substructure below the cognitive foundations that come from everyday experience, and the learners are also subconsciously being exposed to a systematic method of discovering this substructure in a reasoned manner. The pace of learning can be increased after this basic approach is assimilated.

Before we start the instruction for any goal, there should be a motivating prelude, that provides an inspiration, so that the need to learn is 'bought' by the learners. The prelude need not be philosophical or lofty, but could come from role models within their own community, thereby giving cognitive credence to the served goal. After the instruction there could be a postlude that summarizes what has been learnt, so as to keep the focus clear and sharp all through.

We consider a subject is a set of topics. The subject and each of the topics have learning goals and respective prerequisites. A set of lectures focus on a topic. Topics could have sub-topics with respective learning goals and prerequisites. This goes down recursively till there are basic instructions to be provided to elucidate the fundamentals of the subject or topic. Basic instructions have no further prerequisites, from within, though they could well have such prerequisites for the course as a whole or depend on knowledge presumed to be known to students. Basic instructions are elemental, or atomic in relation to teaching; as are fundamentals of the subject atomic with respect to learning. Basic instructions are delivered from first principles as it were; fundamentals form the first principles for reasoning in the subject.

This systematic method of pedagogy for adult literacy starts with what is already known through adult illiterate's use of language, segmenting the known words and cognizing the newly uncovered (syllable) structure, learning to recognize the new in all contexts and situations through play, and moving on to previously unknown but now cognized, rationalized, known and internalized entities or ideas.

On an average, from our experience of analyzing IPCL approach, existing eLearning Systems for *Hindi*, *Telugu*, *Urdu* and *Kannada*; each one of them has about 20,000 visual components, 2,500 aural components, and 750 instructions arising from 24 to 28 lessons based on primers from SRCs and NLM. These can be systematically organized in the following way:

- Each lesson is treated as a *play* or a drama
- Each *play* consists of a sequence of *acts*
- Each *act* consists of a sequence of *scenes*

- Each *scene* consists of a sequence of *instructions*
- Each *play*, *act*, *scene* and *instruction* has defined instructional goal
- An instruction is where the work of teaching is undertaken. plays, acts, and scenes are conceptual organizational structures. They provide convenient points in teaching for conveying motivation to learn, inspirational messages, putting forth exemplary role models amongst learners who have gone on to achieve. There are, on an average 750 instructions in a given eLearning System based on adult literacy primers

The detailed instructional process is as follows:

- The process starts from commonly known words to speakers of a language, segments the words phonetically into syllables, then segments the syllables into simple sounds, which are then identified as alphabets of the language.
- The visual pattern of the word as a whole is presented again and again till it is cognized with no conscious effort.
- Aural segmentation is carefully orchestrated by the instructor, so that the phonetic alphabet is discovered.
- Simultaneous visual segmentation of the word into syllables builds the correspondence between the sounds of the syllables and their scripted visual form. Again this is repeated till it sticks in the memory of all students. This instruction can be adapted based on learners' pace.
- The teacher identifies the alphabets as new facts derived from known words and support activities are required to make these discoveries stick in the memory of learners.
- Comparing words amounts to comparing syllables in previously seen words. Syllables that have similar sounds in them are then segmented based on the similarities; this exposes the dissimilarities, as well as leading to the discovery of phonetic alphabets due to the similarities.
- Abduction is used to identify the cause that leads to dissimilarities.
- Concatenation of sounds provides a basis for phonetic composition. These rules are inferred from examples. Concurrent rules are inferred for visual composition of scripted sounds to form syllables.
- The set of phonetic alphabets discovered so far, and the set of rules for composition of sounds to form syllables and words, both in the visual and aural forms, form a relatively complete world of all words that can be deductively derived from them.

- The complete alphabet of the language is known at the end of the Literacy instruction. Similarly, the complete set of rules for composition of sounds and their scripted forms are also known at the end. It is this completeness that allows induction to be the basis for their confidence that they can read or write anything expressed in their language. Inference and induction lead to deduction as a means for a complete approach to understanding the cause-effect relationships between phonetic alphabet and their scripted representation, and composition thereof.

- In the end, given the relationship between pure phonemes and alphabet, and rules for script and aural composition, given a scripted word, we can deduce the sound of such a word; equally, when a word is uttered, we can deduce the scripted version from the same relationships and rules.

While the number of lessons might vary, we discovered that the following types of standard acts in IPCL methodology:

- To introduce new sounds

- Use these new sounds to form words, both familiar and new

- Recapitulate sounds that have been learnt in previous lectures

- Combine previously learnt sounds with newly introduced sounds in this lesson to form words, both familiar and new

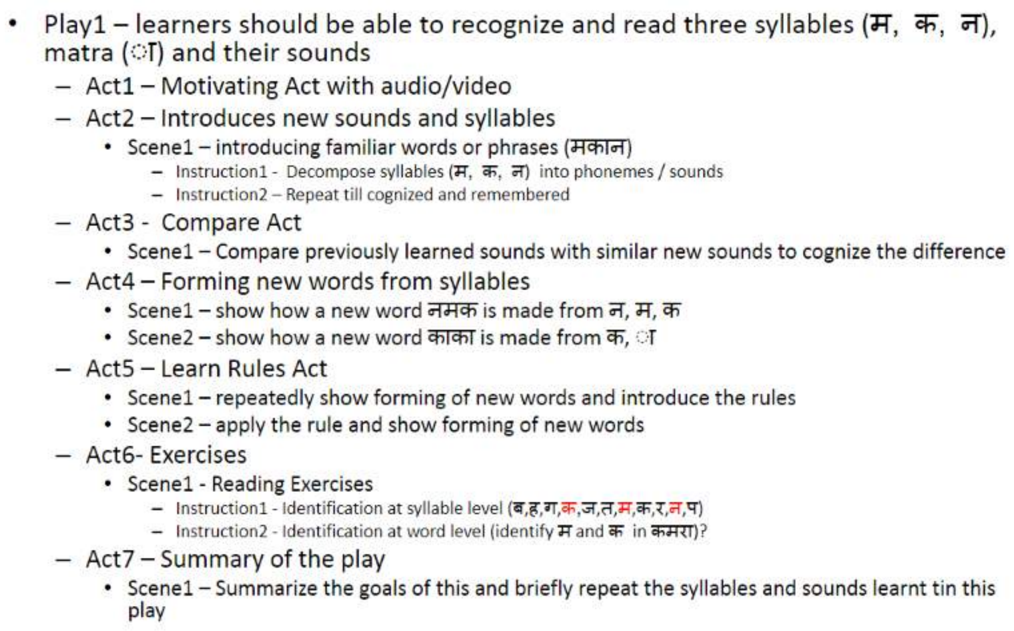
- Compare previously learned sounds with similar new sounds to cognize the difference

- Learn to create new sounds from old sounds from the above experience

- Combine newly created sounds with sounds learnt so far, including this lesson, to create words, both familiar and new

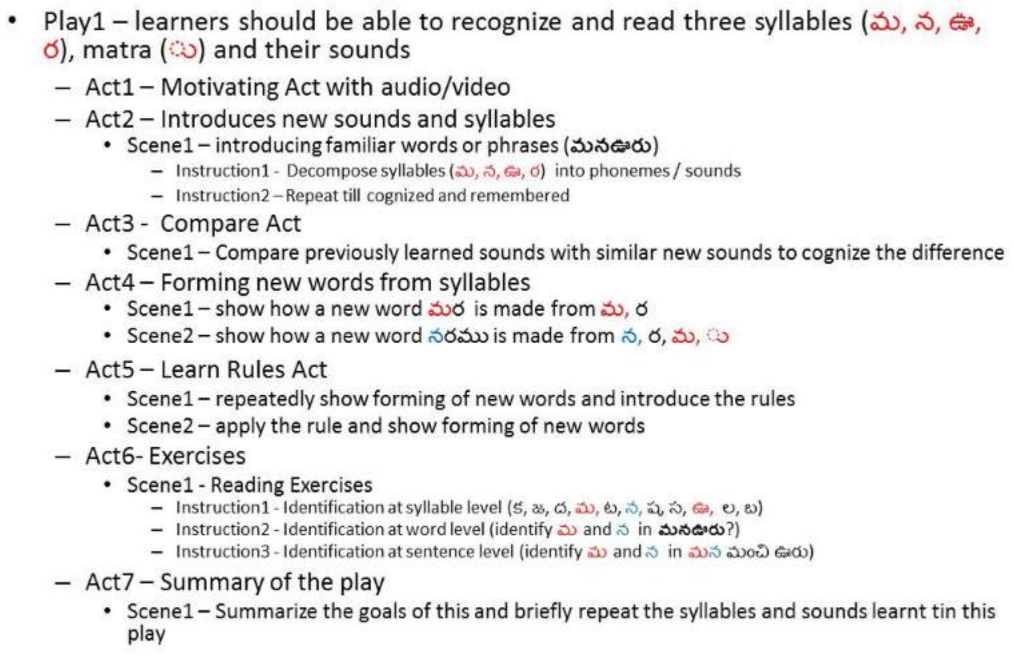
- Exercises, through jumbled words, and reading words not covered so far, but using only the sounds learnt so far

Figure §3.10 shows organizational structure of a *lesson* using the *pasi* pattern. The number and order of the plays, acts, scenes, instructions is not strictly fixed even though guidelines can be framed. For example, the first *play*, *act* and *scene* focus on providing motivation to the learner and the last *instruction* might be a summary of what has been learnt so far in a particular lesson. Figure §3.11 shows a fractional part of an example *play* based on this pattern for the *Hindi* language. In this example, there are several *acts* each having its respective goals, and consisting of specific *scenes* and further *instructions*. For example *Act4* deals with the goal of teaching how to form new words from syllables with two *scenes*



- Play1 – learners should be able to recognize and read three syllables (म, क, न), matra (ा) and their sounds
  - Act1 – Motivating Act with audio/video
  - Act2 – Introduces new sounds and syllables
    - Scene1 – introducing familiar words or phrases (मकान)
      - Instruction1 - Decompose syllables (म, क, न) into phonemes / sounds
      - Instruction2 – Repeat till cognized and remembered
  - Act3 - Compare Act
    - Scene1 – Compare previously learned sounds with similar new sounds to cognize the difference
  - Act4 – Forming new words from syllables
    - Scene1 – show how a new word नमक is made from न, म, क
    - Scene2 – show how a new word काका is made from क, ा
  - Act5 – Learn Rules Act
    - Scene1 – repeatedly show forming of new words and introduce the rules
    - Scene2 – apply the rule and show forming of new words
  - Act6- Exercises
    - Scene1 - Reading Exercises
      - Instruction1 - Identification at syllable level (ब,ह,ग,क,ज,त,म,क,र,न,प)
      - Instruction2 - Identification at word level (identify म and क in कमरा)?
  - Act7 – Summary of the play
    - Scene1 – Summarize the goals of this and briefly repeat the syllables and sounds learnt tin this play

Figure 3.11 Part of an example *play* in *Hindi* language based on *ProcessPattern*(*pasi*)



- Play1 – learners should be able to recognize and read three syllables (మ, న, ఊ, ర), matra (ు) and their sounds
  - Act1 – Motivating Act with audio/video
  - Act2 – Introduces new sounds and syllables
    - Scene1 – introducing familiar words or phrases (మనఊరు)
      - Instruction1 - Decompose syllables (మ, న, ఊ, ర) into phonemes / sounds
      - Instruction2 – Repeat till cognized and remembered
  - Act3 - Compare Act
    - Scene1 – Compare previously learned sounds with similar new sounds to cognize the difference
  - Act4 – Forming new words from syllables
    - Scene1 – show how a new word మర is made from మ, ర
    - Scene2 – show how a new word నరము is made from న, ర, మ, ు
  - Act5 – Learn Rules Act
    - Scene1 – repeatedly show forming of new words and introduce the rules
    - Scene2 – apply the rule and show forming of new words
  - Act6- Exercises
    - Scene1 - Reading Exercises
      - Instruction1 - Identification at syllable level (క, జ, ద, మ, ట, న, ష, స, ఊ, ల, బ)
      - Instruction2 - Identification at word level (identify మ and న in మనఊరు?)
      - Instruction3 - Identification at sentence level (identify మ and న in మన మంచి ఊరు)
  - Act7 – Summary of the play
    - Scene1 – Summarize the goals of this and briefly repeat the syllables and sounds learnt tin this play

Figure 3.12 Part of an example *play* in *Telugu* language based on *ProcessPattern* (*pasi*)

illustrating how new words are formed from already learnt syllables. This *ProcessPattern* has several sources of variation for instruction process. The variations can be the *number* of plays, acts, scenes, instructions; the *order* of them, the specific play, act, scene or instruction, the content used and other aspects of instruction providing customization for scale and variety. Figure §3.12 shows the same example in Figure §3.11 with variations introduced for *Telugu* language.

While the *ProcessPattern* provides a goal-driven structure for modeling a pedagogy at a high-level, it does not include a strong philosophical attitude of instructional process and does not specify how these goals have to be achieved. It is here we analyzed the literature in instructional design and found that there are several perspectives of instructional design. There are also several ways of modeling instructional processes based on different instructional design theories or methodologies. Merrill has analyzed existing instructional design models and proposed that the following fundamental principles are critical to any instructional design [6].

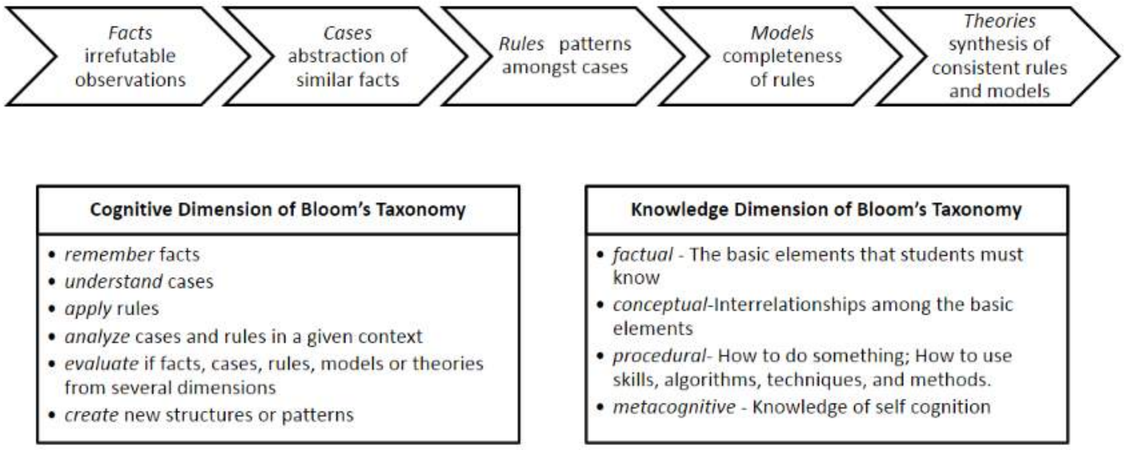
- Activation principle - reaching out to what students know
- Application principle - exercising their new knowledge
- Integration principle - accumulating or integrating what they have learnt recently with what was learnt in the past
- Demonstration principle - showing how this new knowledge can be used
- Task Orientation principle - getting students to solve problems

Each of these principles (activities) are repeatedly used in a specific order in the instructional process to fulfill goals. In addition to these principles, Merrill also proposed a deeper sub-cycle *structure–guidance–coaching–reflection* that strengthens these activities. For example, a structure has to be provided for the learner as part of instruction while applying activation principle and necessary guidance has to be given to the learners during a demonstration activity.

Generally, the *ProcessPattern* involves some or all of these principles at different levels of granularity but the application of these principles becomes explicit for tasks at *instruction* level. So, for example *Scene1* of *Act2* introduces words that are familiar to the learners essentially involving activation principle whereas learners have to use application principle in *Scene2* of *Act5* to form new words from existing syllables. Similarly, other instructions in the instructional process can be mapped to principles.

3.5.3 A pattern for modeling instructional material

Instructional material is at the center of instruction and we discovered a pattern for content as shown in Figure §3.13. This pattern is primarily derived from IPCL, scientific


Facts irrefutable observations
Cases abstraction of similar facts
Rules patterns amongst cases
Models completeness of rules
Theories synthesis of consistent rules and models

Cognitive Dimension of Bloom's Taxonomy

- *remember* facts
- *understand* cases
- *apply* rules
- *analyze* cases and rules in a given context
- *evaluate* if facts, cases, rules, models or theories from several dimensions
- *create* new structures or patterns

Knowledge Dimension of Bloom's Taxonomy

- *factual* - The basic elements that students must know
- *conceptual*-Interrelationships among the basic elements
- *procedural*- How to do something; How to use skills, algorithms, techniques, and methods.
- *metacognitive* - Knowledge of self cognition

Figure 3.13 Pattern structure and mapping to cognitive and knowledge dimensions of Bloom's taxonomy

method but most importantly driven by our future need to facilitate reasoning in the subject. Can learners provide rationale and reasoning for their answers?

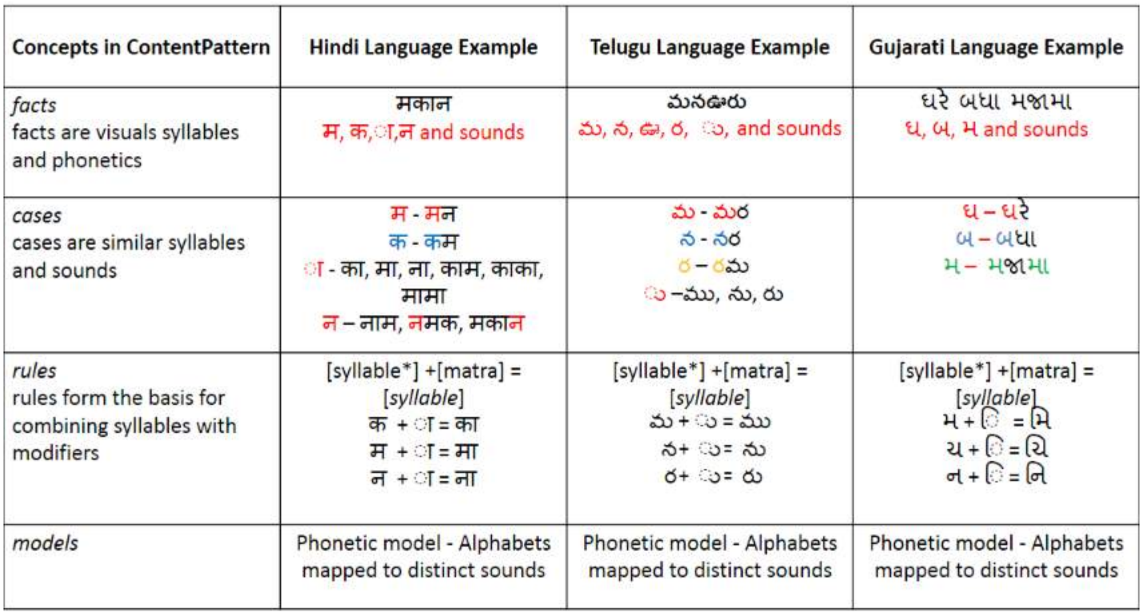
Figure §3.13 shows the progression of content from simple facts to be remembered to foundations in the subject. The mapping of this pattern to cognitive and knowledge dimensions of Bloom's taxonomy is also shown in the figure. Here, we describe the underlying foundations for this pattern from the literature:

- The pattern starts with facts, which are irrefutable observations and presumptions that relate to what the learners know.

- Then, we organize facts and search for similarities between them and this abstraction leads us to form 'typical' cases by abduction [192] (inferring the cause from its plausible observed effect), patterns that hold across facts, thereby reducing the number of facts to remember; thus we can inferentially reason about classes of facts [193].

- We organize the cases found so far. We look for patterns amongst these cases and form a rule of thumb [194]. Here, if the rule of thumb applies, we have a heuristic to support our 'inferential' reasoning – without being able to assert that the inference will always hold in all situations the rule of thumb is applied. When we can formulate the completeness of this collection of patterns, we move from a heuristic to rules – complete knowledge pertaining to an area of discourse. With this organization, we can reason from a specific situation and 'deduce' consequences.

- The completeness of rules that pertain to a class of situations, i.e., applicable to every instance in that class of situations, gives us a 'model' for deductive reasoning for the class of situations of interest [195]. The model, its complete collection of rules, is



Concepts in ContentPattern	Hindi Language Example	Telugu Language Example	Gujarati Language Example
facts facts are visuals syllables and phonetics	मकान म, क,ा,न and sounds	మనఊరు మ, న, ఊ, ర, ు, and sounds	ઘરે બધા મજામા ઘ, બ, મ and sounds
cases cases are similar syllables and sounds	म - मन क - कम ा - का, मा, ना, काम, काका, मामा न – नाम, नमक, मकान	మ - మర న - నర ర – రమ ు –ము, ను, రు	ઘ – ઘરે બ – બધા મ – મજામા
rules rules form the basis for combining syllables with modifiers	[syllable*] +[matra] = [*syllable*] क + ा = का म + ा = मा न + ा = ना	[syllable*] +[matra] = [*syllable*] మ + ు = ము న+ ు= ను ర+ ు= రు	[syllable*] +[matra] = [*syllable*] મ + િ = મિ ચ + િ = ચિ ન + િ = નિ
models	Phonetic model - Alphabets mapped to distinct sounds	Phonetic model - Alphabets mapped to distinct sounds	Phonetic model - Alphabets mapped to distinct sounds

Figure 3.14 Example of *ContentPattern* instances for *Hindi*, *Telugu* and *Gujarati* languages

an expression of 'applicable' knowledge. Its completeness guarantees satisfiability through deduction, a primitive of computation, along with unification, when considering logic as a model of computation.

- A set of 'models' that have consistency, integrity constraints that persist across models, gives us a theory for the domain of discourse, giving rise to many plausible applications of knowledge, individually, or in combination as made possible because of the inherent integrity amongst them.

  Essentially the process continues (abduction, inference, induction) till the subject is unfolded. Here, soundness comes from relating to factual knowledge. Completeness and consistency aid us in moving from empirical knowledge to formal knowledge, from inferential reasoning to deductive reasoning in the domain of discourse. Using this approach, the learners are exposed to the build up to the model rooted in their factual observations. They have been given a basis for its completeness and taught the reason of its adequacy to cope with all situations that can arise in the class of problems they are expected to cope with and solve. This understanding is reinforced through adequate worked out examples, and are given sufficient exercises to practice working up solutions starting from first principles, and deduction based on the logic of the model that was taught. Figure §3.14 shows instances of this pattern for *Hindi*, *Telugu* and *Gujarati* languages.

The *ProcessPattern* and *ContentPattern* are closely connected and can be considered as an instructional architecture pattern. In this pattern, teaching is structured, not simply as a sequence of lectures, but as a sequence of *models* as needed by a *theory* to be taught. Here, we consider teaching to be a like a *play* uncovering each *model* as the instruction unfolds, with learner participation! Each *model* in turn would be a sequence of topics. Uncovering a *model* is done in a succession of *acts*, with each *act* uncovering a topic pertinent to some *model*. Each *act* is presented as a succession of *scenes*, with each *scene* focusing on *cases* or *rules*. Finally, each *scene* is delivered as a succession of basic *instructions*, which uncover and play with *facts*.

3.6 Pattern-Oriented Software Architecture

Software architecture deals with most of the design decisions concerned with *structure*, *behavior*, *interaction*, *qualities*, and *implementation* of a software system [188]. In this thesis, we are interested in identifying the variants that are possible in each of these aspects to support the desired requirements of scale and variety in educational technologies. In general, architectural patterns and reference architectures provide a baseline and a set of guidelines for creating specific software architectures tapping the variabilities of multiple related systems in a particular domain [188]. At a high-level, customization of software can be done at design level, requiring explicit modifications to architecture and design of software and also from a user perspective, requiring configurations [196]. Morche [196] lists three common ways of tailoring software applications from literature:

The users can *customize* the application by configuring a set of pre-defined options mostly related to user interface and existing functionality. These options can range from themes or colors in the user interface to turn off or turn on features of the existing system. For example, an instructor might configure the theme of the eLearning System based on local context and culture, evaluation in terms of multiple choice or fill in the blanks and so on.

In *integration*, users can have configuration options to integrate functionality that is outside the system. For example, a teacher might want to integrate learning management system like *Moodle* into the current system. This might require tweaking of components, extending interfaces and fixing interoperability issues. A GLUE-architecture was proposed to support integration of tools into virtual environments [197].

When additional new features have to be added, the system has to be *extended* by adding new code, re-writing and sometimes even re-designing some parts of the system. Extending the system becomes an expensive activity if the current system is not designed for extensibility. Extensions can emerge from new requirements from domain or because of new techniques and technologies available to design software. For example, if a new accreditation rule requires the instructional goals to conform to a particular standard, then the system designed for extensibility should have a basic module for evaluation which could be extended to add/remove features required as per new accreditation requirements.

In essence, variability is a broad concept and can range from user needs, market segments, customer profiles which can be addressed through a wide variety of artifacts that are generated throughout the software development life cycle. In this thesis, we attempt to address variability in instructional design as well as in software through patterns and software product lines as detailed in Chapter §5.

The core idea of POSA is to create software architecture using patterns such that these patterns can address variability and map to patterns in POID. Architecture patterns can further consist of design patterns with each of them addressing variability at different levels of granularity. In this section, we briefly provide examples for both architecture and design patterns.

We illustrate the idea of POSA through a commonly used pattern called Model-View-Controller (MVC). Kranser and Pope have described MVC for interactive user interface applications [190]. The key purpose of MVC pattern is to facilitate separation of the interactive application into three parts or components to efficiently address changing requirements. Models primarily represent the underlying application domain knowledge and act as core structure for Views and Controllers. Instructional design is the underlying domain in this thesis and as such forms the basis for models. The representation of this model itself can vary based on how the domain is modeled. As emphasized by Kranser and Pope, the focus should be on modeling specific information about the application domain such that it can drive the other two parts. Views primarily focus on the user interface, graphical elements and what the users view as part of the system. A typical user interface consists of hierarchical views and the data for these views is fed from the model behind them. This is quite useful as a model can have many views associated with it facilitating variability from a user interface perspective. Consider a scenario where an eLearning System uses the same model but can be viewed using several user interface metaphors. The variabilities can range from simple color or themes to complex modifications emerging from instructional design. The order of these views itself can be a source of variability. Controller is the third component of MVC pattern that acts as the interface between models and views.

It is not uncommon to integrate several architectural patterns while designing a software application to address varied requirements [110]. For example, Figure §3.15 shows a simple overview of how MVC pattern can be integrated into a layered architectural pattern. Here the user interface, business logic and data are separated into three layers Presentation, Business and Domain. MVC pattern is spread across *presentation* and *business* layers. The MVC pattern itself is a composed pattern consisting of several design patterns and can be implemented in several ways leading to variations like Hierarchical MVC, PVC [198]. Several design patterns are used to implement MVC and its variations. The model part of the MVC pattern is part of the domain strongly mapped to data parts of the domain i.e., pattern-oriented instructional design. A simple way to vary the application is to change these data parts in the POID without changing the interfaces exposed to POSA. Even MVC itself uses *Observer* pattern to notify views and controllers, providing a variation point. Composite pattern is commonly associated for constructing user interface elements in a hierarchical way and exposing only the top level view for the entire architecture leaving flexibility to change user interface elements. *Strategy* pattern is commonly used to alter between different controllers and use concrete strategies at different points of time to facilitate different behaviors for different triggers in the software application. One key difference in this layered MVC architecture as shown in Figure §3.15 is the use of components instead of classes emphasizing the need to model every class as a component with explicit interfaces to facilitate variability. The domain layer emphasizes the strong role of domain in this architecture than just traditional databases. The data is mapped to patterns in the domain as this can allow traceability of changes from domain to software. A 10-step process for MVC pattern [179] along with potential variabilities is shown below:

1. Separate human-computer interaction from core functionality
   inputs, output behaviours, accessor functions

2. Implement the change-propagation mechanism
   publish-subscribe pattern, specific implementations

3. Design and implement the views
   appearance of views, display procedures, parameterized views, multiple draw methods

4. Design and implement the controllers
   specific behaviours for user actions, event handling

5. Design and implement the view-controller relationship
   initializations for which factory method pattern could be used, hierarchy of views and controllers

6. Implement the set-up of MVC
   initializations, events

7. Dynamic view creation
   components for managing views

8. Pluggable controllers
   different controllers

9. Infrastructure for hierarchical views and controllers
   composite pattern, chain of responsibility pattern

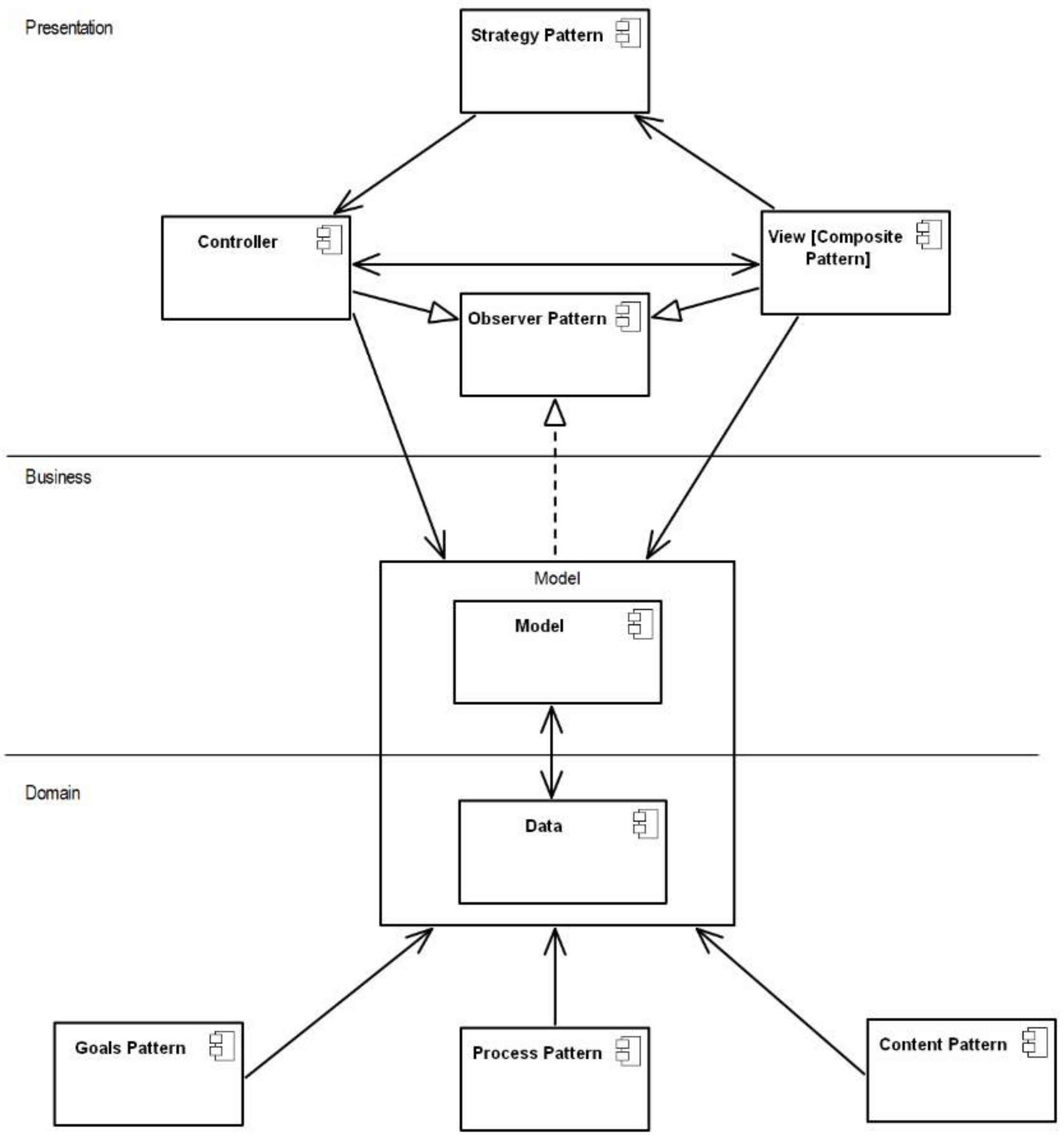


Figure 3.15 Layered Model View Controller Pattern

10. Further decoupling from system dependencies
    bridge pattern, higher levels of abstraction

Today, there are several web frameworks like *Django*, *Symfony*, *Rails* that are based on MVC pattern but most importantly implement their own variation, addressing specific requirements. Most of the variations are possible by tweaking one or more of the steps in the above process. For example, there could be hundred controllers in a large scale application providing variability or the need to create varied interaction themes with users can be addressed by tweaking step 9. According to [110], MVC pattern references twelve design patterns like *Facade*, *Observer* and so on.

POSA can also be designed by integrating several design patterns. Gamma et al. list 23 design patterns and their relationships in a single diagram [109] and Zimmer provides a succinct classification of relationships between various design patterns into three layers: [design patterns specific to an application domain], [design patterns for typical software problems] and [basic design patterns and techniques] [199]. To summarize, the core essence of this chapter is to show how varying different aspects in patterns can facilitate variability needed for scale and variety in educational technologies.

In the next chapter, we present how this abstract knowledge of patterns can be concretely represented using ontologies. In Chapter §5, we discuss how these patterns form the basis for domain engineering phase of software product lines for instructional designs and eLearning Systems.

CHAPTER 4

An Ontology Based Modeling Framework for Instructional Design

The fundamental idea of domain knowledge and what kind of domain knowledge is required for adult literacy is discussed at length in Section §4.1. We then briefly discuss the broad spectrum of ontologies and their development process in Section §4.2 and Section §4.3 respectively. We focus on the breadth of ontologies in the domain of instructional design in Section §4.4. In Section §4.5, we present our ontology based framework for modeling instructional design. Within this section, we detail ontologies for modeling goals, instructional process and instructional material in sub sections §4.5.1, §4.5.2 and §4.5.3. We end the chapter by concretely presenting a domain ontology for adult literacy in Section §4.6.

4.1 Domain Knowledge

John McCarthy has envisioned that an intelligent way of building systems should focus on the *knowledge* that is required to represent system's inputs and methods through which possible conclusions can be automatically derived from that knowledge [200]. Newell has proposed the need to have a *knowledge level* focusing on specifying the world independent of *symbol level* that focuses on implementing the behaviour of the system [201].

> "The Knowledge Principle: A system exhibits intelligent understanding and action at a high level of competence primarily because of the *specific knowledge* that it can bring to bear: the *concepts*, *facts*, *representations*, *methods*, *models*, *metaphors*, and *heuristics* about its *domain* of endeavor." -Lenat and Feigenbaum [202]

The *ContentPattern* proposed in Chapter §3; consisting of *facts*, *cases*, *rules*, *models* and *theories* can be considered as a part of domain knowledge in the above statement. Feigen-

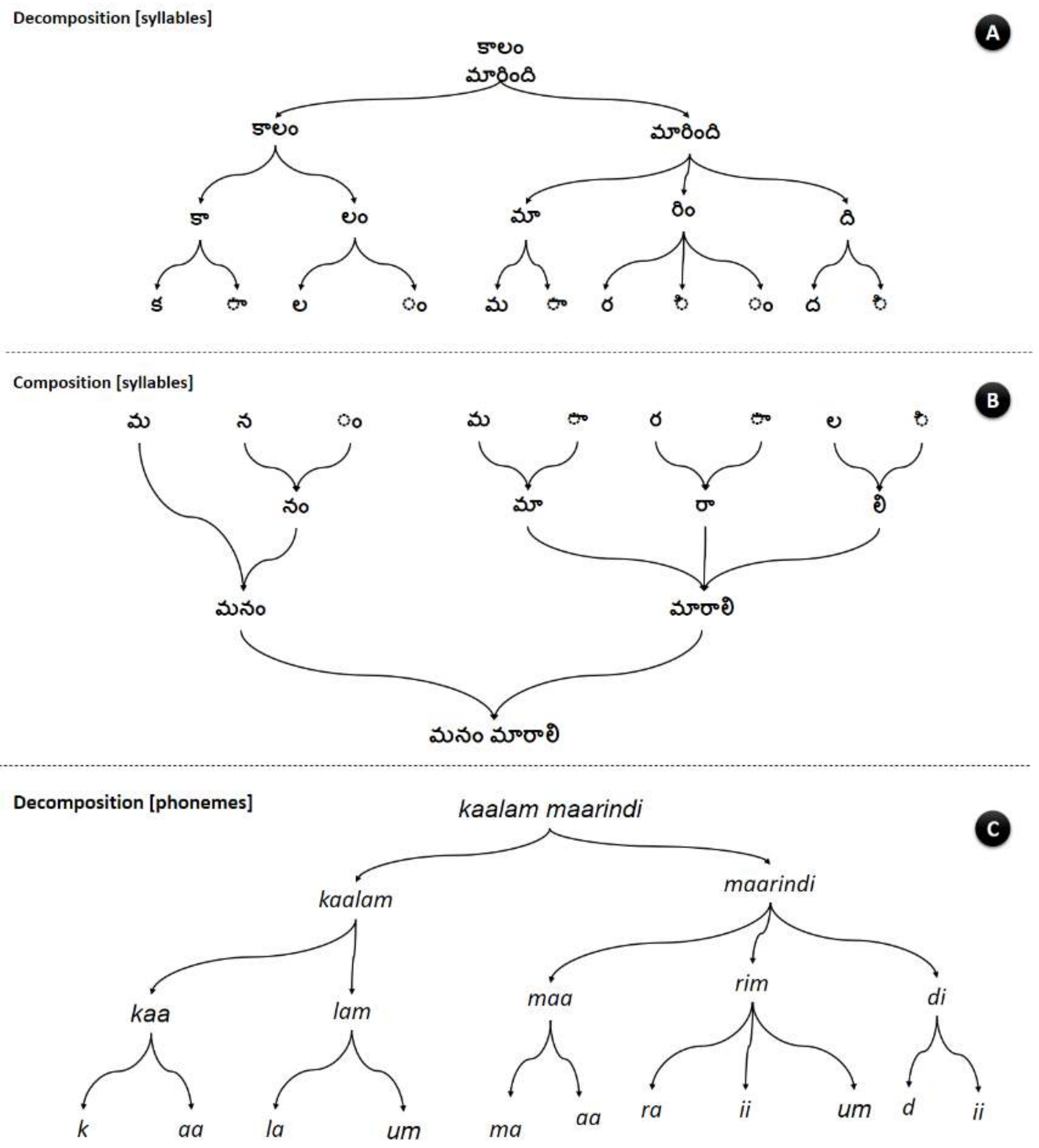
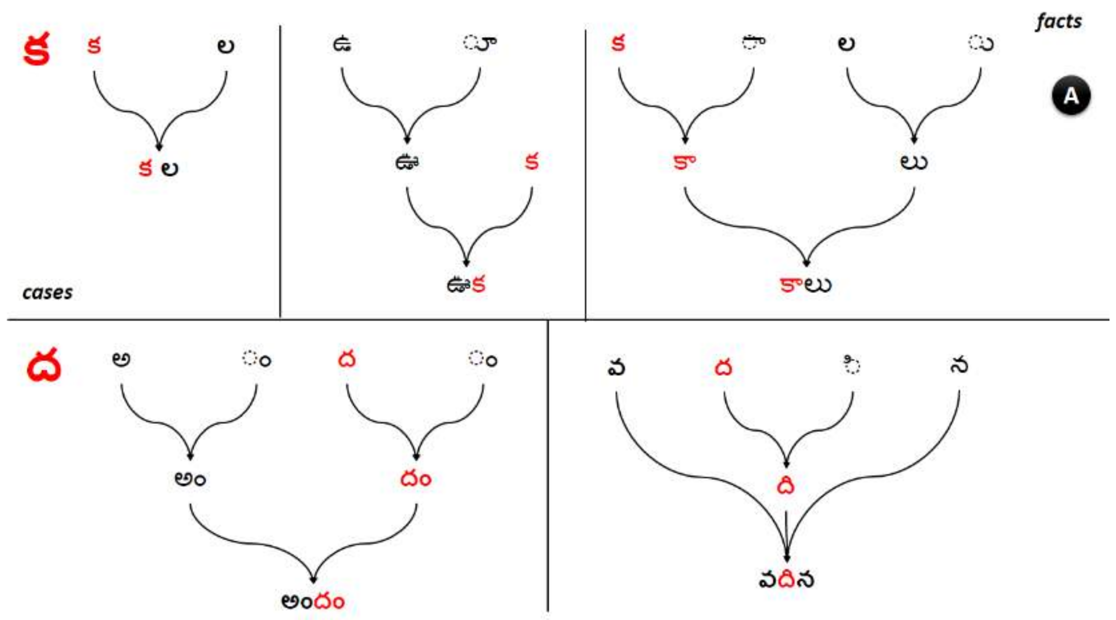


Figure 4.1 Top-down decomposition of sentences to syllables [A], Bottom-up composition of words and sentences from syllables [B], Top-down decomposition of sentences to phonemes [C]

baum coined the term *knowledge engineering* and proposed knowledge base should be a fundamental basis that stores expertise of human experts in solving real-world problems [202]. There are two primary directions of research related to knowledge engineering one in the field of Artificial Intelligence primarily for automatic reasoning and expert systems and in computer science to represent different aspects of the system. Several researchers have figured out multiple ways of representing knowledge like concept maps, topic maps, ontologies, first order logic and so on [203][204].

What is the specific domain knowledge in this thesis and how to represent it? In order to address this question, we start with concrete examples from adult literacy case study setting the context for the rest of the chapter. We introduced IPCL in Section §2.1 as a well-established approach for teaching adult illiterates in India. This approach suggests the use of *eclectic method* for teaching reading skills, comprehension, problem solving and facilitates learning through interpretation of contents in the context of life [78]. We consider "context of life" as mentioned in IPCL approach as learners' prior knowledge. Ecletic method primarily differs from traditional methods as it does not start with alphabets but instead uses familiar and known words to learners, decomposes them to syllables and phonemes as shown in Figure §4.1[A][C].

These syllables and phonemes are further synthesized to form words and in the end, the alphabet is learnt as in Figure §4.1[B]. The patterns of decomposition (top-down) focuses on cognitive abilities of learners whereas the patterns of composition (bottom-up) can facilitate reasoning of the subject knowledge. We have depicted examples of this process in Figure §4.2 for the *Telugu* language based on the primer (instructional material) and IPCL approach. In this figure, a sentence in *Telugu* language కాలం మారింది is first decomposed into two words namely కాలం, మారింది. Each of these words are further decomposed till the syllables are obtained. Similarly, the same sentence is also decomposed into phonemes representing the sounds of the sentence, words and syllables respectively. On the other hand, Figure §4.2[A] shows the composition process which uses the syllables and phonemes to form words and sentences. Specifically, words కల, ఊక and కాలు are formed by composing the syllable క and other relevant syllables and words అందం, వదిన are formed from their respective syllables with the learnt syllable ద in red color. This hierarchy of decompositions and compositions forms the basis for learning new syllables and phonemes in a language as shown in Figure §4.2[B]. Several compositions are possible V-Vowel+Vowel Modifier, C-Consonant+Vowel Modifier, C+V-Consonant+Vowel, Vowel Modifier, C+C-Consonant+Consonant, C+C+V-Consonant+Consonant+Vowel and there will be several constraints on these compositions as well. For example, only one modifier might be allowed after a consonant. Discussing that in detail is not within the purview of this thesis. But this approach of learning alphabet from known words and sentences and later synthesizing new words from syllables and phonemes was empirically established by IPCL as a successful approach for adult literacy in India [29][205]. Most importantly, this approach works



B

క	ం	ి	ద
• కల	• మనం	• కంకి	• అందం
• ఊక	• రసం	• వరి	• దారం
• కాలు	• కాలం	• రవిక	• దారి
• కాలువ	• వారం	• విలువ	• వదిన
• కునుకు	• సంసారం	• మురికి	• మరిది
		• మినుములు	• కందులు

Figure 4.2 Composition of cases from facts [A], Possible set of cases for a lesson in *Telugu* language primer [B]

for the scale and variety of 22 Indian languages and varied instructional designs. This specific knowledge has been abstracted into instructional design knowledge and patterns as shown in Chapter §3.

Generalizing from this specific knowledge, the domain knowledge representation in this thesis should be:

- in synergy with instructional design
- machine-processable
- facilitate reuse and semi-automatic design of applications (eLearning Systems)
- able to support sharing of knowledge between different applications and tools

More specifically, the focus of this thesis is to represent a *subset of domain knowledge* towards developing families of eLearning Systems for adult literacy in India rather than on discovering methods for representing entire instructional design or language learning.

Ontologies is one of the widely used methods to represent domain knowledge that suits the above requirements. Over the years, ontologies have garnered mainstream attention for its wide range of applications in several domains [145] [206]. However, the term ontology itself has several connotations and is understood in a variety of ways in the literature [145][207][208]. A commonly used definition of ontology in computer science comes from Gruber [13], where he defines an ontology as "*a formal, explicit specification of a shared conceptualization*". This definition was further characterized and explained by several researchers [145] [207] as:

- *formal*, meaning that an ontology should be represented using a formal language processable by machines and tools
  This characteristic will allow us to represent instructional design knowledge in machine-processable form to facilitate automation
- *explicit*, by using different types of primitives and precisely stating different concepts and axioms defining the ontology
  Instructional design for adult literacy is embedded in IPCL and primers; ontologies will help in making it explicit
- *shared*, the ontology is meant for a group of stakeholders within a community belonging to a specific domain or sub-domain
  How to share IPCL across all Indian languages and How to specify this knowledge to software engineers?
- *conceptualization*, represents a specific view of a domain through various abstractions
  a view of instructional design for adult literacy in India

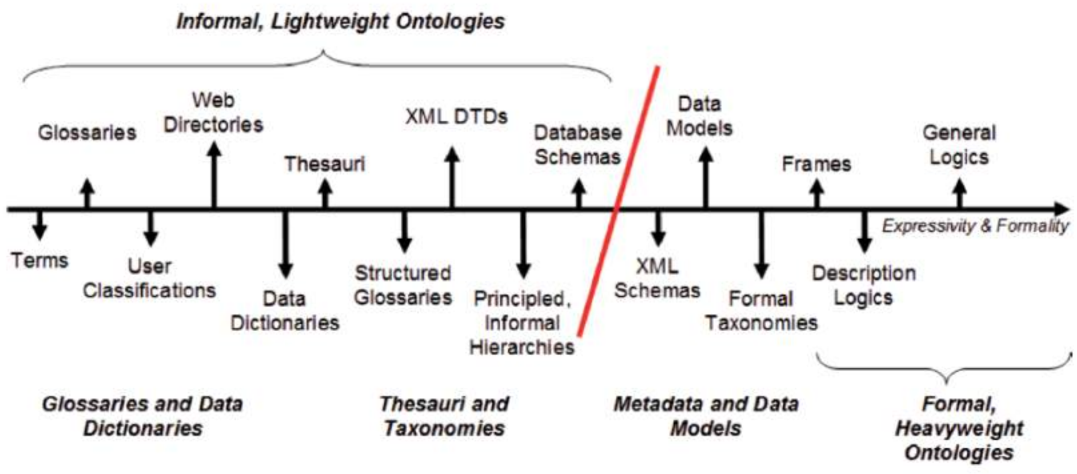


Figure 4.3 A spectrum of ontologies

Fensel attributes the popularity of ontologies is due to the promise of providing "*a shared and common understanding of a domain that can be communicated between people and application systems*" [209], which can be construed as communication and automation requirements needs of this thesis.

4.2 A spectrum of ontologies

There has been a significant growth of research in ontologies in the past decade resulting in a variety of approaches for ontological engineering [210][211][70] [148][212][149]. In particular, ontologies have been used for representing different aspects of a specific domain using a wide range of different mechanisms [213]. At the core of ontologies is the identification of concepts generally represented as a hierarchy of classes and sub-classes and different relationships between them. Each of the these classes typically have associated properties and can also have a set of constraints. Instances of these ontologies are called as individuals and represent a specific instance of a particular domain represented by the ontology. For example, an ontology for instructional design at a higher level can be defined using concepts like *goals*, *process*, *content* and so on and each of these sub-classes can be further defined. An instance of this ontology can be a specific instructional design for teaching a specific course. We discuss this ontology in detail in the rest of the chapter.

Diversified needs emerging from different domains gave raise to a spectrum of ontology kinds [213][150] as shown in Figure §4.3. These kinds of ontologies vary based on the degree of specification detail, formalism and expressiveness power as we move from one end to the other end of the spectrum. A detailed description of this spectrum is given in [213][208]. In essence, there are informal or lightweight ontologies on one end, primarily geared towards some sort of communication and on the other end, formal ontologies help in automated reasoning of knowledge [150]. This thesis falls in the middle and mostly uses OWL/XML Schemas to address the primary needs of communication and automation. They also provide a mechanism to use instructional design as a basis throughout the design of educational technologies. In addition, two key future research directions that motivated the need for ontologies in this thesis are:

- Design of personalized learning environments for a diversified range of learners, teachers and subjects in India and across the globe
- Design of technologies that allow students to explicitly justify their answers through *reasoning* and provide a debugging environment through automated reasoning

Additionally, *scope* is another critical factor that can be used to classify ontologies at different levels of granularity. A distinction is made between *upper ontologies* that describe general-purpose concepts and their relationships, *domain ontologies* that define domain-specific concepts, *task ontologies* that specify domain-specific activities and *application ontologies* that instantiate domain ontologies and integrate task ontologies for a particular application [214].

We use OWL 2, a W3C recommendation that refines and extends OWL, the Web Ontology Language for representing knowledge in the semantic web [215]. OWL 2 is based on earlier version of OWL, extends RDF and is also compatible with XML. According to Krotzsch, OWL serves as a descriptive language for expressing expert knowledge in a formal way and as a logical language for drawing conclusions from that knowledge [216]. Accordingly, OWL 2 allows ontology engineers to represent knowledge using various representations like RDF/XML, OWL 2 XML, Functional Syntax, Macnhester Syntax, Turtle as shown in Table §4.1 with each of the methods having different expressive power and reasoning abilities [215]. The choice of ontology representation is primarily decided by ontology engineers based on the requirements and needs of the domain [214]. We confine ourselves to the descriptive use of ontologies and use OWL/XML for representing ontologies in this thesis.

4.3 Development of Ontologies

Ontology development has matured in the last few decades from being a research topic to even the discipline of ontology engineering [211] and is often seen as one of the cornerstones of semantic web today [215]. There are several surveys in the literature focusing on ontological development methodologies [217] [218] [219][220]. Most of these methodologies like Cyc, *METHONTOLOGY*, *TOVE* are based on standard development processes described in terms of phases with several activities in each of the phases. A deeper analysis

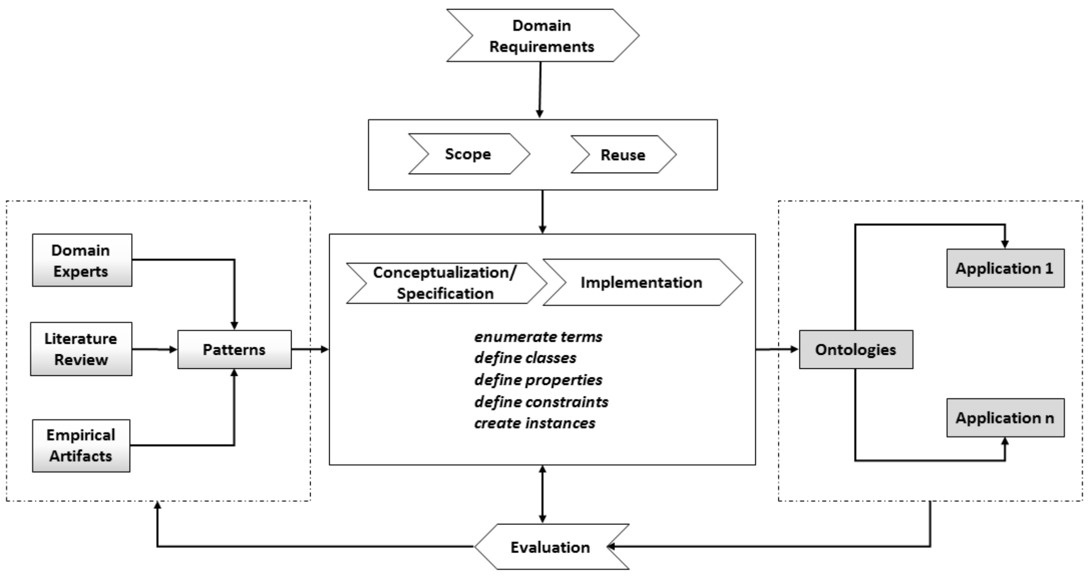


Figure 4.4 Ontology Development Process

Table 4.1 Syntax variations in OWL 2 Web Ontology Language

Name of Syntax	**Specification**	**Purpose**
RDF/XML	Mapping to RDF Graphs	Interchange (can be written and read by all conformant OWL 2 software)
OWL/XML	XML Serialization	Easier to process using XML tools
Functional Syntax	Structural Specification	Easier to see the formal structure of ontologies
Manchester Syntax	XML Serialization	Easier to read/write DL Ontologies
Turtle	Mapping to RDF Graphs, Turtle	Easier to read/write RDF triples

of methodologies for building ontologies is presented in [217] and from an evaluation perspective in [221]. There is a consensus in the literature that none of the existing approaches are fully mature when compared with software engineering and knowledge engineering methodologies [218] [222]. METHONOLOGY is one of the oldest and matured approach for ontology development [217] but without support for collaboration. A software engineering approach for developing ontologies is proposed in [223]. A six-stage ontology development method for engineering design was proposed by Ahmed et al. in [224]. Ontology development methods supported with tools became quite popular in the recent times and protégé is one of the exemplar examples to illustrate this case. In their Ontology Development 101, Noy and McGuinness proposed an iterative approach for building ontologies consisting of several activities that need not be sequential (i) determine scope (ii) consider reuse (iii) enumerate terms (iv) define classes (iv) define properties (v) define constraints (vi) create instances. An important conclusion from their work is "*there is no single correct ontology for any domain. Ontology design is a creative process and no two ontologies designed by different people would be the same*" [225].

We see three major directions for developing ontologies from a synthesis of the literature (i) manually by expert(s) for specific purposes following a varied set of processes from light-weight to a rigorous standard process (ii) semi-automatic way of developing ontologies, where a part of the ontology is developed manually and a part is automatically retrieved using text mining, natural language processing and other machine learning techniques (iii) fully automatic development, where the ontologies are derived using ontology learning approaches.

We follow a simple process for developing ontologies in this thesis as shown in Figure §4.4. The first step in the process is to determine the requirements from the ontology, which is driven by the set of eLearning Systems to be developed in our case. The next step is to figure out the scope of the ontology drawing a boundary for what is within and outside the scope. Once the scope is defined, the next step is to identify any existing ontologies that can be used for creating the ontology. There are several search engines like SWOOGLE[1] and ontology servers like OntoLingua[2]s for searching existing ontologies. We discuss the ontologies we adapted from existing literature in the next section. Once the suitable sources for ontologies are defined, the next step is to use a standard approach to identify the concepts, relationships between the concepts, define properties, constraints and instances using an appropriate representation language like OWL/RDF. An important distinction of this process from the standard ontology development methodologies is the use of patterns as one of the critical sources for building ontologies. The patterns themselves are discovered after extensive discussions with domain experts; rigorous analysis of literature and analyzing existing applications that are built in the domain. We have extensively discussed with domain experts from NLMA; analyzed literature on adult literacy and instructional design as a source for our patterns. We also analyzed several eLearning Systems developed by TCS for 9 Indian languages before creating the patterns. We use these patterns as one of the primary source for creating the ontologies. We also consider other literature from the instructional design space as input to our ontologies. The output of this entire exercise of conceptualization and implementation is a set of ontologies. The evaluation of this ontologies is carried out by developing a set of applications based on these ontologies and assessing whether the domain requirements have been met or not. Figure §4.5 shows a part of how we devised scope for our instructional design ontology framework for adult literacy.

[1]http://swoogle.umbc.edu/
[2]http://www-ksl.stanford.edu/knowledge-sharing/ontolingua/

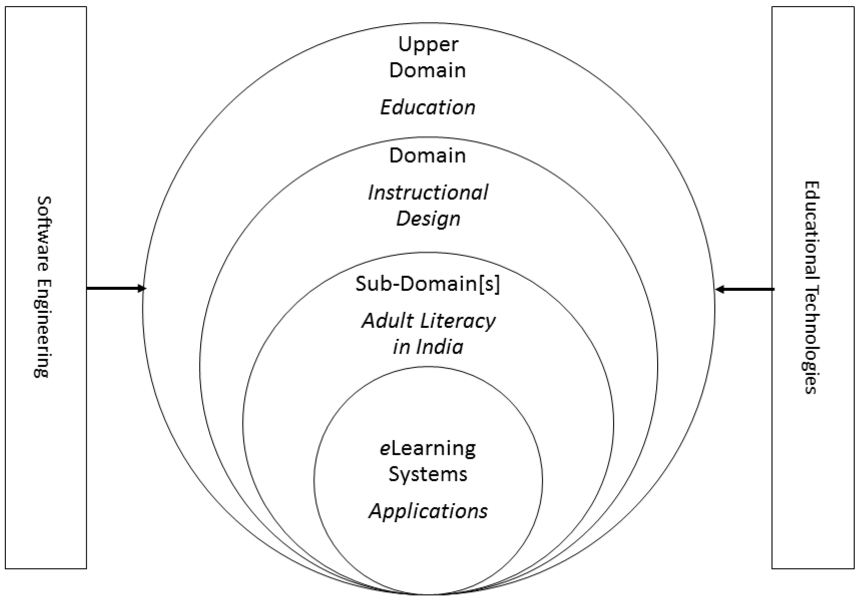


Figure 4.5 Scope of Ontologies in this thesis

4.4 Ontologies for Instructional Design

Ontologies are used in education domain for several applications ranging from representation of subject knowledge [226] to generation of content and assessment artifacts based on learners' styles [52] [153] [227]. Researchers have extensively worked on building systems based on ontologies primarily in the area of adaptive and personalized learning [152][228][229]. Ontologies are also applied in intelligent tutoring systems [230][231] [232] and for automatically generating multiple choice questions from domain ontologies [233]. Mizoguchi suggested several ways of how ontologies can help in addressing challenges in AI-ED domain [153] and reflected on their progress after 15 years in [212]. He suggested that there is long way to go for advancing methods for ontology development and their use in education [212]. Synthesizing the literature in this space, we see different kinds of educational knowledge [38] that can be modeling using ontologies:

- Curriculum, *selection of knowledge*
- Courses, *selection of specific knowledge pertaining to a subject*
- Instructional Design, *how this knowledge is taught*
  - context
  - goals

  - content
  - process
  - evaluation
  - environment

- Evaluation or Assessment, *to figure out if knowledge is acquired as per goals*
- Learner Styles, *knowledge of different kinds of learners*
- Teacher Styles, *knowledge of different kinds of teachers*

We do not attempt to model this entire knowledge but a subset of this knowledge within the scope of the thesis. We are interested in the development of ontologies for instructional design in the context of adult literacy in India. The key purpose is to create variants of instructional design catering to the the varying needs of eLearning Systems for adult literacy. We presented different categories of patterns in instructional design in Section §3.5 and in this section we present ontologies based on these patterns for different aspects of instructional design. We have extensively searched the literature to find ontologies that are relevant for our purposes.

What are the existing ontologies related to adult literacy in India?, and the domain of instructional design in general?

To the best of our knowledge, we could not find any ontologies that are even remotely connected to adult literacy in India. But we searched the literature for various ontologies focusing on different kinds of educational knowledge and give a few examples here. An ontology for literacy was proposed in the context of intelligent tutoring systems way back in 1999 [234]. We then looked into some upper ontologies and found an example curriculum ontology devised by BBC for the national curricula on UK focusing on three topics (Algebra, Geometry, Formula), level (KS1, KS2, KS3, GCSE) and different fields of study (Maths, English, Science) [235]. A comprehensive ontology that models several learning theories is presented in [236] where the idea is to have solid pedagogical basis for intelligent tutoring systems. Recently, Heiyanthuduwage et al. have analyzed 14 ontologies developed by different institutions for learning design and proposed an OWL 2 learners profile [237]. One of the earliest ontologies developed by Mizoguchi focuses on creating a task ontology to facilitate reuse of problem solving knowledge [69]. We came across several ontologies focusing on particular kind of instructional design; for example, a mobile learning ontology was designed for abductive science inquiry style of instruction [238]. An ontology for learning scenarios based on collaborative learning theories is given in [239] and one focusing on gamification is presented in [240]. There were other set of ontologies focusing on specific subject matter, like word problems in mathematics [241], software engineering body of knowledge [242]. In addition to these kinds of ontologies, there are different kinds of ontologies developed for learning content [158], learning design based on IMS LD standard [44], a context ontology for bridging the gap between learning content and learning design [243]. There were ontologies to represent learning object repositories [244] and learning design repositories [245] to facilitate search and retrieval of learning resources on the web.

However, none of these ontologies are connected to IPCL or patterns and hence do not directly meet the requirements in this thesis. We will discuss an ontology based modeling framework in the next section followed by domain ontologies for adult literacy.

4.5 IDont - An Ontology Based Framework for Modeling Instructional Design

Is teaching science the same as teaching mathematics? Does teaching in a country like US and India same? Can we use the same method for teaching different kinds of learners? The answer for most of these questions is no. There has been tremendous effort in trying to come up with several standards in the space of educational technologies like SCORM for learning objects [246], IMS-LD for learning designs [183], IEEE LOM for learning objects [48] and enormous research [185], platforms and tools [34] surrounding these standards. However, despite significant progress, most of the promises seem to be unfulfilled as discussed in Section §2.4.2 and in Section §2.4.3. We summarize the following major pitfalls:

- One-size-fits-all - There are hundreds of learning theories in the literature. Attempts towards coming up with a unified way of dealing with them turned quite complex denting the success of the initiatives.

- *End-to-end automation* - Several attempts have been made to automate different aspects of education, ranging from modeling learning theories to automatically generating learning environments and this focus on end-to-end automation turned futile in the most of the attempts [184]. For instance, in the case of IMS-LD, even though not stated explicitly, this goal of end-of-end automation resulted in complex authoring [185].

- *Administration and Management* - While it is important to handle and ease the job of teachers in administration and management activities for which several Learning Management Systems were developed, linking the instructional design to LMS has increased further complexity with the existing approaches.

We learn from these experiences and propose a framework for modeling instructional design using *ontologies* based on *patterns*. The design rationale for ID*ont* is as follows:

- *Simplicity & Separation of concerns approach* - We strongly advocate the separation of concerns approach to model ontologies. The core idea is to have smaller multiple ontologies for different aspects of instructional design such that they can be adapted, extended and reused in other contexts. Figure §4.6 shows how different aspects can be separated as components having explicit interfaces such that they can be connected with other aspects and customized for a specific learning situation.

- *Leverage and Reuse existing ontologies* - When designing new ontologies, ontological engineering suggests the utilization, adaption and extension of existing ontologies (learning objects, learning designs).

- *Technology Design* - The framework should support the creation of a platform and authoring tools to explicitly capture and model all the ontologies.

- *Extensibility and Customization* - These are two major criteria for the framework as most of the times the ontologies have to be customized and extended for the specific domain. There should be a provision in the framework such that existing ontologies should be replaced with custom ontologies with minimal effect on the overall framework.

- *Iterative and Collaborative Approach* - The design of this framework should follow an iterative approach and must consult different stakeholders (such as teachers, learners, instructional designers and so on) during the process.

- *Internationalization* is required for both ontologies and tools that support the creation of ontology instances.

The core premise of this framework is to systematically model instructional design using different aspects like *context*, *goals*, *process*, *content*, *evaluation*, *environment*. We distinguish between two kinds of instructional design knowledge, one is at a *conceptual* level that maps with existing learning methodologies and the other is at a *technical* level to facilitate semi-automation of eLearning Systems. In this context, the core idea of IDo*nt* is not to define complete ontologies but to point to several possible modular ontologies that are required for systematic modeling of instructional design. As such, most of the aspects of IDo*nt* are optional and can be configured based on specific purposes and learning situations. Figure §4.6 shows an overview of the IDo*nt* framework. The key inputs come from a set of instructional design requirements that drive selection of appropriate instructional design models, which are captured as patterns in our approach. We do not specify the exact ontologies for instructional design but have placeholders for different aspects. With the advent of several ontology repositories, an instructional designer or ontology engineer can either extract the required ontologies for specific instructional design model from existing literature or create a new one. This generic instructional design stitched from existing or new ontologies can

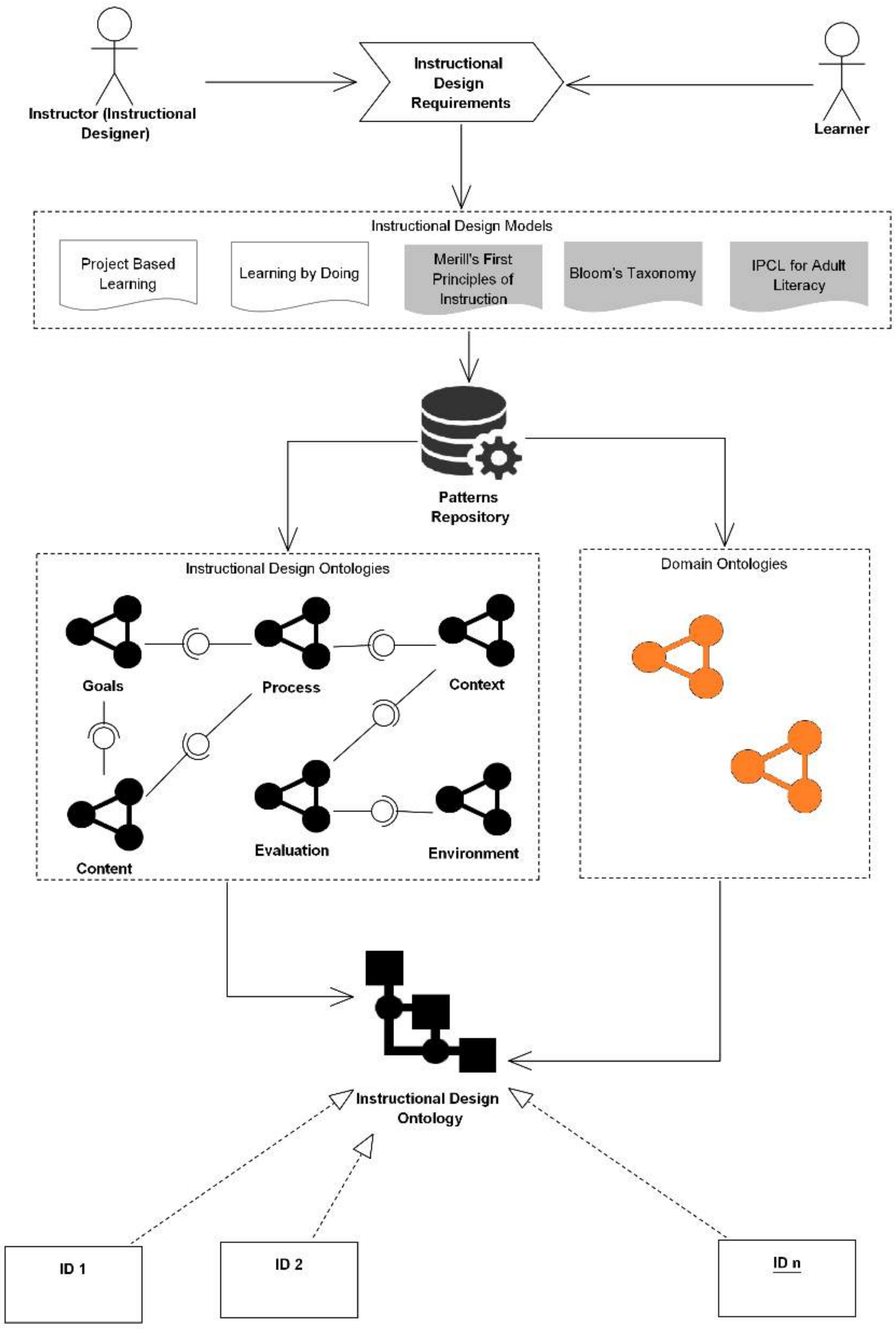

Instructor (Instructional Designer)
Instructional Design Requirements
Learner
Instructional Design Models
Project Based Learning
Learning by Doing
Merill's First Principles of Instruction
Bloom's Taxonomy
IPCL for Adult Literacy
Patterns Repository
Instructional Design Ontologies
Goals
Process
Context
Content
Evaluation
Environment
Domain Ontologies
Instructional Design Ontology
ID 1
ID 2
ID n

Figure 4.6 Overall Process of IDont Framework

be customized with domain ontologies and is further realized by specific instances like ID1, ID2 ... ID*n*.

Figure §4.7 presents an overview of ID*ont* framework for adult literacy. Even though we discuss several ontologies in the diagram, the focus of this thesis is on *goals*, *process* and *content* ontologies. We briefly explain the important ontologies of our framework as follows:

A. ContextOntology - Context plays a significant role in ID*ont* as it allows for modeling of various aspects related to a particular learning situation. The notion of learning context was proposed in LOCO ontology to bridge the gap between learning design and content consisting of domain specific information [45]. However, in this framework, we articulate context in a broader view that encompasses several pointers to all other ontologies. This is a meta-ontology that essentially captures the basic information related to all other aspects of instructional design such that each of these aspects can be potentially (re)used. As shown in Figure §4.7, *ContextOntology* has metadata associated with it along with context information related to various aspects of instructional design. *ContextOntology* specifies how a *ProcessContext* achieves *goals* using *ContentContext* delivered through *EnvrionmentContext* following *EvaluationContext* and performed by *RolesContext*.

B. GoalsOntology - This ontology formalizes the notion of goals (which can be instructional goals, learning goals or even learning outcomes). The details of how it is defined are left to the specific instance. Some properties associated with goals are *hasName*, *hasPriority*, *hasPrerequisites*, *hasEvaluation*, *isAchievedByProcess*. The *GoalsOntology* points to the process through which these goals will be achieved, target competencies, the instructional material that is required and the evaluation to be performed. Consider the scenario of creating goals for K12 students, and goals are prescribed by education boards. Teachers can potentially reuse these goals if captured in the form of *GoalsOntology*. As the evaluation related to these goals is separately captured, it can be reused as well. We prescribe to the idea of *goal-driven instruction* as part of our framework irrespective of instructional design models. We detail *GoalsOntology* in the latter part of this chapter in Section §4.5.1.

C. RolesOntology - The success of instructional design depends on several people who perform their roles in the process. Hence, it is important to capture the roles, their responsibilities and how the different aspects of instructional design should be adapted according to the needs of specific roles. The two most important roles are that of teacher and learner. The crucial knowledge about a learner is captured through learner profiles consisting of several attributes. Generic roles in instructional design are captured first and then modified based on specific learning situations. Mapping of goals with competencies of roles can also be done in this ontology. Most of the roles are further associated with teams and in that case the role of the team as well as individual persons is modeled separately along with roles performed by agents. This ontology should also have metrics to trace from goals to evaluation.

D. ProcessOntology - The crux of ID*ont* framework is the *ProcessOntology* that captures the instructional design process, and relates to all other ontologies and practically executes the process. In the literature, Learning Design is discussed heavily, in particular IMS Learning Design [39] and received criticism as well [185]. Ontologies for modeling learning design are presented in [44]. Based on our prior experience with adult literacy instructional design, IPCL and our future goal to introduce reasoning into adult literacy, we proposed the *ProcessPattern* - (*play*, *act*, *scene* and *instruction*) in Section §3.5.2. Each lesson is organized as hierarchy of *pasi* with instructions where concrete activities are performed based on Merill's principles of instruction in this particular instance. This instruction actually points to *ContentOntology* and associates required content for the respective instruction. This nomenclature allows us to systematically capture the knowledge of instructional design process and potentially reduce technological effort. This hierarchy has similarities to IMS LD but has variations to align with our patterns for adult literacy instruction. We will present the *ProcessOntology* in Section §4.5.2.

E. ContentOntology - This ontology allows for modeling of instructional material in a particular learning situation. There is extensive research on ontologies for learning objects and we use the ALOCoM ontology [71] as base for content aspect of our framework. However, for adult literacy instructional design, we have used *fcrmt* (facts, cases, rules, models, and theories) structure [11] as discussed in Section §3.5.3. So the *ContentType* of ALOCoM now includes *fcrmt*. The *ContentOntology* is closely associated with other ontologies and strongly with the *ProcessOntology*.

F. EvaluationOntology - What if the most common evaluations of instructional design are captured and an instructor can customize them based on his or her requirements? The main intent of this ontology is to capture evaluations as independent knowledge and link them with goals through *ContextOntology*. This separation makes it easier to perform different kinds of evaluations for the same set of goals. This ontology captures the details of evaluation and has a direct relationship with *GoalsOntology* which is connected with *ProcessOntology*.

G. EnvironmentOntology - The final execution of instructional design happens in an environment. Consider the scenario of a lesson to teach about shapes to K2 students. Considering that the students first listen about shapes in a *ClassroomEnvrionment* and if a teacher wants to use computers, then most of the other aspects of instructional design remain same except the environment which changes to *ComputerEnvironment*. Separating the environment from the rest of instructional design makes it easier to run the learning situation in different environments similar in spirit with software deployment.

H. DomainOntology - This ontology mainly articulates and customizes key aspects of instructional design with respect to a specific domain and provides a domain-specific version of ontology. In particular, the various sub-ontologies and properties of these ontologies will have detailed associations when mapped to a specific domain. For e.g. the *ContentOntology* will have strong co-relation and mapping with content tin the domain.

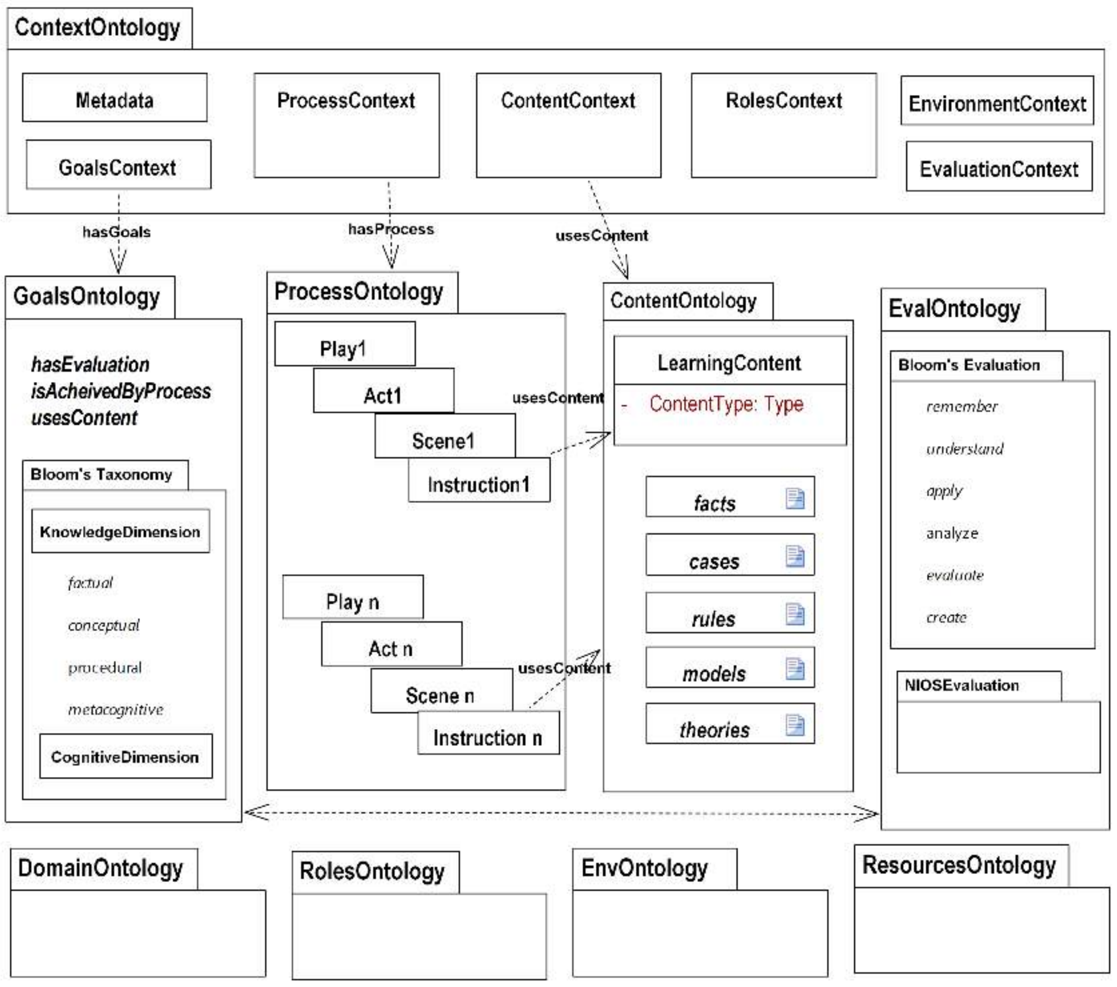


Figure 4.7 Instructional Design Ontology for Adult Literacy

There can be several other ontologies like *ActivitiesOntology* for capuring various activites that are suported in the instructional design, *WorkflowOntology* to model the tedious workflows in education, *FeedbackOntology* to capture continuous feedback of the instructional design, *OrganizationOntology* focusing on characteristics of the organization, *ResourcesOntology*, having pointers to specific resources like text, audio, video and so on. In our analysis of instructional design literature, we strongly see that it is virtually impossible to capture all kinds of instructional design models and any attempt towards it turns to be futile. However, the main intent of our framework is to use a separation of concerns approach to systematically capture various aspects of instructional design through ontologies.

The primary goal of ontologies in this thesis is to drive the semi-automatic creation of eLearning Systems based on accepted instructional design models.

We discuss the specific ontologies that are developed as part of IDo*nt* framework. Our attempt is never to be complete during the design of these ontologies but to design educational technologies in sync with instructional designs. We also include several entities in the

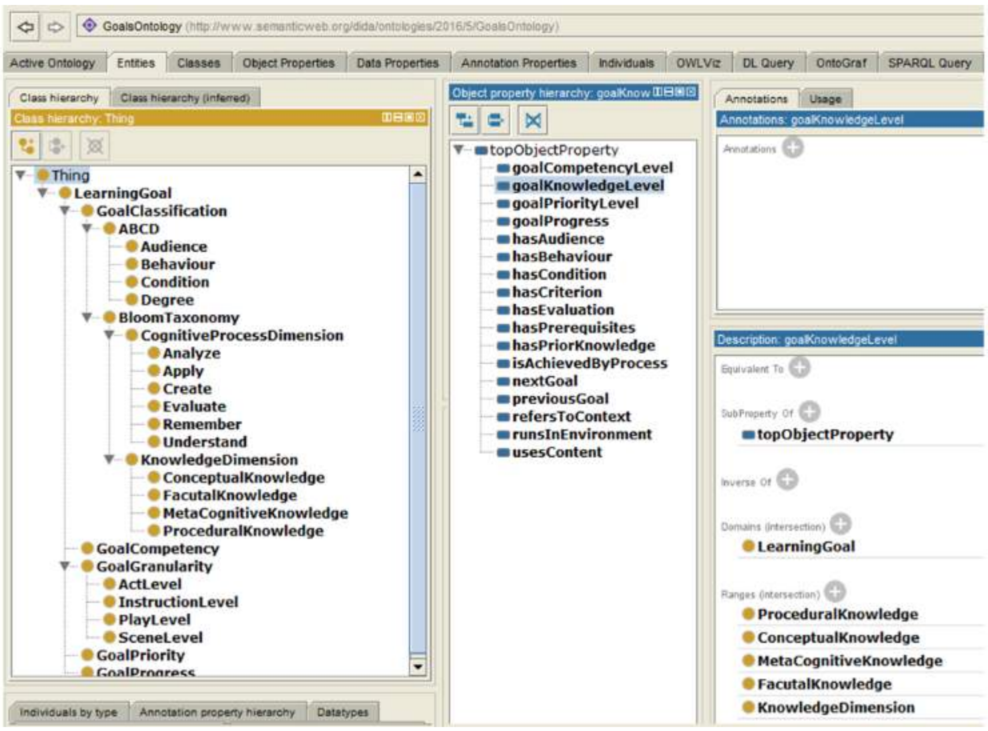


Figure 4.8 A fragment of *GoalsOntology*

ontologies for future use rather than just current needs. We also rely on the best practice of adapting existing ontologies and create new entities only if required.

4.5.1 An ontology for modeling goals

The primary goal of any instructional design is to find ways to support learners in achieving their learning goals [247]. Based on the pattern discussed for goals in Section §3.5.1, we present an ontology for representing instructional goals in this section based on revised Bloom's taxonomy [17].

Figure §4.8 shows a part of *GoalsOntology* developed using *protégé*[3] tool from Stanford. The priority of the goal is described using *GoalPriority*, progress through *Goal-Progress*, deadline through the property *goalDeadline*. An important sub-class is to classify the goal according to a taxonomy. The class *GoalClassification* is further divided into two classes *BloomTaxonomy* and *ABCD*. The *BloomTaxonomy* is further divided into *KnowledgeD-imension* and *CognitiveProcessDimension* as per revised Bloom's taxonomy. The knowledge can be classified as *FacutalKnowledge*, *ConceptualKnowledge*, *ProceduralKnowledge* and *MetaCognitiveLevelKnowledge* with increasing levels of higher order levels of thinking. This is in sync with the *ContentPattern* discussed in Section §3.5.3. The *CognitiveProcessDimen-sion* is the most commonly used way to classify goals as per Bloom's taxonomy. It has six levels *Remember*, *Understand*, *Apply*, *Analyze*, *Evaluate*, *Create* and each of them have sev-

[3]http://protege.stanford.edu/

eral verbs specifying the activities. Several object properties are shown in Figure §4.8 connecting different concepts in the ontology. Priority of the goal can be captured using *goalPriorityLevel*, competency through *goalCompetencyLevel* and *goalKnowledgeLevel* can have a range of values from the *KnowledgeDimension* and maps to the *fcrmt* pattern. Every goal should have a *goalDeadline* and its progress is monitered through *goalProgress*. A goal also has *hasPrerequisites*, *previousGoal* and *nextGoal*. This ontology is connected to *ProcessOntology* through *isAchievedByProcess*, *ContentOntology* via *usesContent*, *EvaluationOntology* through *hasEvaluation* and *runsInEnvironment*. In addition, there are several data properties that are associated with the ontology. For example, *goalDeadline* stores the deadline as *dateTime*. The goal itself can be described using *goalText*, *goalImage*, *goalAudio*, *goalVideo*, *goalMetadata*. These data properties store specific information that can be later used for (semi-)automatic generation. *GoalGranularity* is another critical class that is specific to our instructional design as we have a goals hierarchy akin to *play*, *act*, *scene*, *instruction* pattern. In addition to the standard concepts, the ontology also has concepts for *GoalPattern* consisting of properties shown in Figure §4.8. For example, *SourceOfPattern* is a data property that specifies the source of the patterns, *Trade-Offs* specifies the issues that might occur using this pattern. In our case, we realized that if specifying *goals* requires so much of effort, the entire exercise will be a burden for teachers and instructional designers making it a futile effort in the end. Hence, we have minimal mandatory properties with scope for using extended properties only if required. A detailed overview of this ontology is provided in Appendix §A.1. We will provide instances of this ontology in Chapter §6 as part of evaluation.

4.5.2 An ontology for modeling instructional processes

The *ProcessOntology* is a core ontology for specifying instructional process and is closely associated with several other ontologies. As shown in Figure §4.9, the ontology is divided into three conceptual sections at a higher level (i) *learning*, focusing on concepts that map to the underlying learning methodologies (ii) *metadata* consists of information about the process in general (iii) *user interface* to declaratively specify a few aspects of the eLearning System. The *ProcessOntology* is based on *ProcessPattern* and its primary purpose is to achieve goals specified in the *GoalsOntology* and is connected through *hasAssociatedGoal* property. These goals have to be achieved using content specified via *ContentOntology* connected through the object property *usesContent*. Similarly, *usesEvaluation*, *performedbyRole*, *runsInEnvironment* connect this ontology to *EvaluationOntology*, *RolesOntology* and *EnvironmentOntology* respectively. This ontology has several data properties like *title*, *description*, *metadata*, *noOfPlays*, *noOfScenes*, *noOfInstructions*. One important property is *hasTimeLimit* that specifies the time limit for an *activity*, *instruction*, *scene*, *act*, *play*. *Guidelines* is an important concept that we use for giving instructions to learners during their interac-

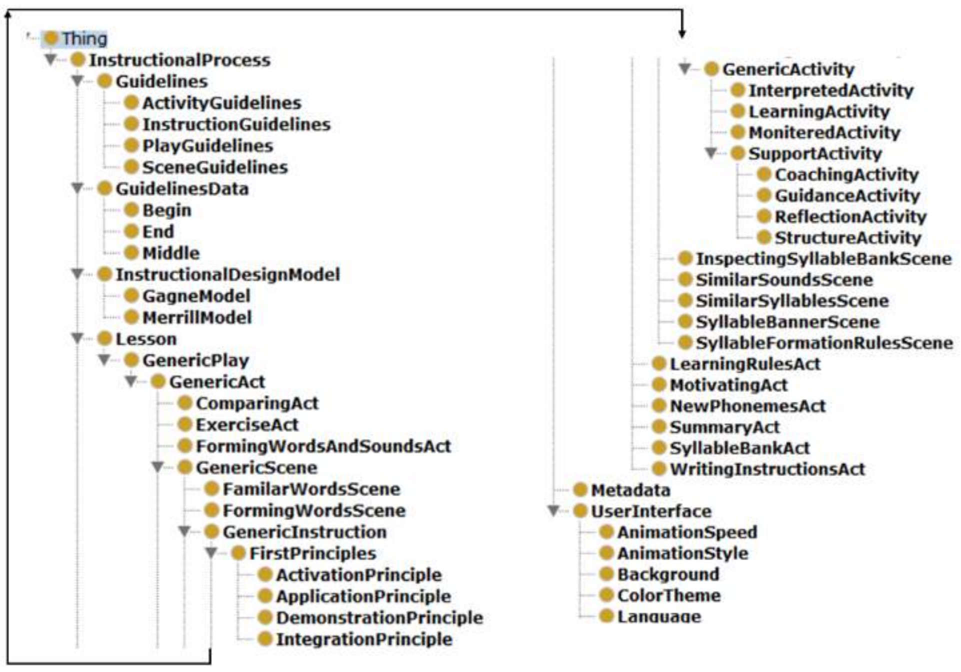


Figure 4.9 A fragment of *ProcessOntology*

tion with the eLearning System at different levels of granularity specified using *PlayGuidelines*, *ActGuidelines*, *SceneGuidelines*, *InstructionGuidelines*, *ActivityGuidelines*. For example, a guideline from a teacher might be *"Everybody look at the screen and observe how the two syllables are combined together to form a new word"*. Separating this information provides the flexibility to change guidelines. This can be specifically used to change *medium of instruction* in an eLearning System. A language like *Hindi* can be taught using *Telugu* as medium of instruction by changing the guidelines in the entire system.

The base *InstructionalDesignModel* can be specified as *MerrillModel* or any other instructional design model from the literature. We use *MerrillModel* for the reasons motivated in Section §3.5.2. Then each *lesson* is modeled using a set of plays (*GenericPlay*) that are divided into acts (*GenericAct*), which are further divided into scenes (*GenericScene*) and instructions (*GenericInstruction*). We have identified different kinds of acts for adult literacy instruction as specified in the *ProcessPattern* in Section §3.5.2. They include a *MotivatingAct*, *NewPhonemesAct*, *FormingWordsAndSoundsAct*, *SyllableBankAct*, *ComparingAct*, *LearningRulesAct*, *WritingInstructionsAct*, *ExerciseAct*, *SummaryAct*. We infered these acts from IPCL and eLearning Systems that are tested on the field. There are different kinds of scenes *SimilarSoundsScene*, *SimilarSyllablesScene*, *InspectingSyllableBankScene*, *SyllableFormationRulesScene*, *FamilarWordsScene*, *SyllableBannerScene*, *FormingWordsScene* under each act. Each scene further has instructions which have direct activities for facilitating learning. Each instruction follows one or more principles and can have one or more activities. We specify Merrill's first principles of instruction using *FirstPrinciples* that is fur-

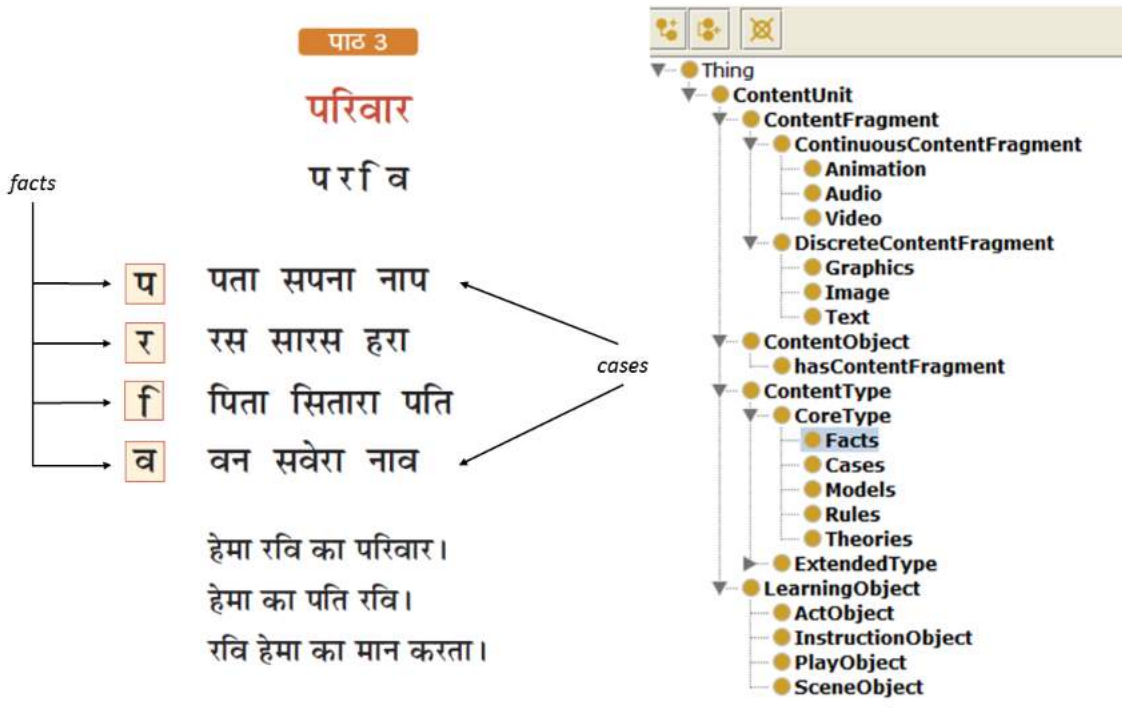


Figure 4.10 A fragment of *ContentOntology*

ther divided into *IntegrationPrinciple*, *ActivationPrinciple*, *DemonstrationPrinciple*, *ApplicationPrinciple*, *DemonstrationPrinciple*. Activity is one of the most commonly used concept in the space of instructional design and we model that using *GenericActivity*. We incorporate two kinds of activities from the literature *LearningActivity* and *SupportActivity*. But we also model four kinds of additional activities *StructureActivity*, *GuidanceActivity*, *CoachingActivity* and *ReflectionActivity* to accommodate Merrill's inner circle of structure-guidance-coaching-reflection. Modeling these activities as concepts allows us to change these activities based on learner styles or instruction styles. In addition, *InterpretedActivity* and *MoniteredActivity* help from evaluation perspective.

The current ontology also has basic concepts for *UserInterface* like *AnimationStyle*, *ColorTheme*, *AnimationSpeed*, *Language*, *Background*. The instances of these concepts will help in configuring the user interface of eLearning Systems for adult literacy based on specific requirements.

One principle behind this ontology is not to use all the classes and properties but to further filter this ontology to the specific needs and use only a fragment of the ontology for the purpose in order to reduce the burden on the teachers and instructional designers. For example, if a course has 1000 instructions in total, then specifying principles for all of these instructions might be a burden and an alternative could be to make this property optional at instructional level but mandatory at a scene or act or play level. A detailed overview of this ontology is provided in Appendix §A.2.

4.5.3 An ontology for modeling instructional material

This ontology is primarily derived from existing literature on learning objects and specifically the ALOCoM ontology [158] along with the *ContentPattern* discussed in Section §3.5.3. As shown in Figure §4.10, this ontology includes four core concepts *ContentType*, *ContentFragment*, *ContentObject*, *LearningObject*. The raw data in the form of *text*, *audio*, *animation*, *video* are concepts in *ContentFragment* and *ContentObject* is an aggregation of several content fragments. The way our ontology differs from literature is in terms of *ContentType*, which includes *Facts*, *Cases*, *Rules*, *Models*, *Theories*, which form the *CoreType*. In *ExtendedType*, there are further concepts derived from the literature [158]. Essentially, they capture learning objects at a higher level of abstraction. Another important concept is *LearningObject* which has the sub concepts of *PlayObject*, *ActObject*, *SceneObject*, *InstructionObject*. These concepts are connected to respective elements in *ProcessOntology*.

There are other ontologies for specifying *Roles*, *Evaluation*, *Environment* that are part of instructional design ontology but defining those ontologies is beyond the scope of this thesis. The *RolesOntology* is an interesting one with roles like Teacher, Mentor, TeachingAssistant, Coach and so on and can be mapped to different kinds of activities in the *ProcessOntology*. As an example, learning styles and teaching styles may be used in the role of learner and teacher in *RolesOntology*. A detailed overview of this ontology is provided in Appendix §A.3. In the next section, we will briefly discuss a domain ontology for adult literacy.

4.6 A Domain Ontology for Adult Literacy

A distinction has been made in the literature between domain and application ontologies [214]. *Domain ontologies* are aimed towards defining concepts pertaining to a specific domain like "*adult literacy for Indian languages*" whereas *application ontologies* are further refined to specific needs of an application, in our case it can be an eLearning System for a particular language. According to Census 2011, there are about 29 languages spoken by more than a million people, 60 languages by more than 100,000 people and about 122 languages by more than 10,000 people speaking it in India. Of these, there are 22 official languages. A fundamental tenet of Indian languages is that they share a common phonetic base [248]. This commonality across a family of Indian languages provides a shared domain that can be used for representing domain knowledge as an ontology. Based on this premise, NLMA has come up with IPCL, a uniform methodology as explained in Section §2.1, which is the base for creating instructional material for all Indian languages. In Indian languages, the term "aksharas" is used to refer to *alphabet*. This *akshara* refers to a sound that is formed of sounds of vowels and consonants [249]. Being invariant of position is an interesting characteristic of *akshara* that holds for all Indian languages [249].

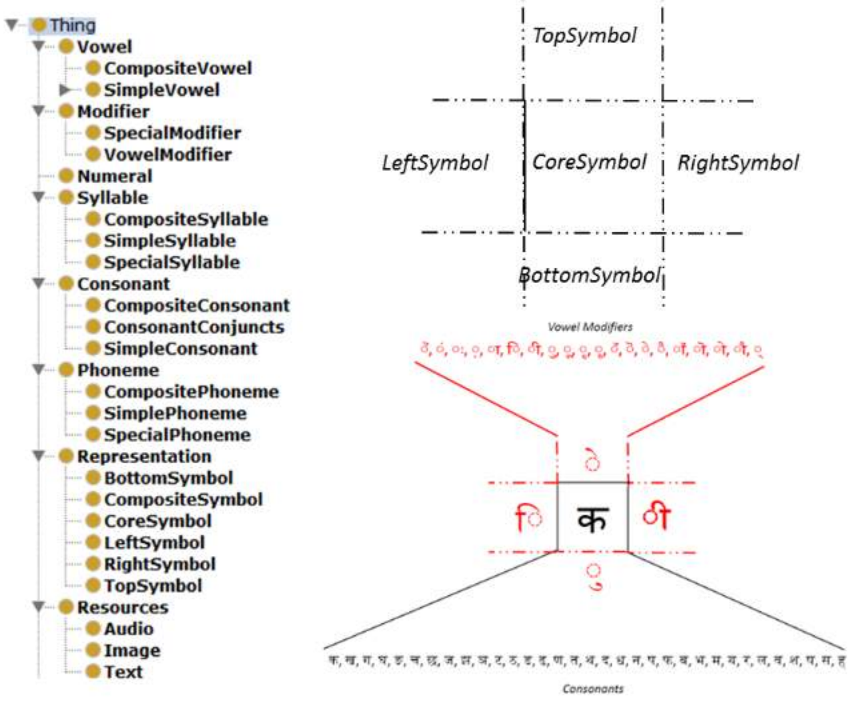


Figure 4.11 A fragment of *DomainOntology* for adult literacy

Figure §4.11 [left] shows an ontology of Indic scripts for literacy, primarily focusing on the structure of syllables in the language. The core concepts of the ontology include *Syllable* denoting the visual representation of an *akshara* or a fragment of it. This is further specialized into *SimpleSyllable*, *CompositeSyllable* and *SpecialSyllable*. Every concept in this ontology have two properties *hasRepresentation* and *hasResources* refering to *Representation* and *Resources* respectively. *Representation* has five other concepts to systematically capture a syllable. Figure §4.11[right] shows the core structure of a syllable in Indian languages. The syllable itself is composed of *CoreSymbol*, *LeftSymbol*, *RightSymbol*, *TopSymbol*, *BottomSymbol*. Each of these concepts store the respective visual fragments of the syllable at the relative positions. For example, a base consonant like क can be modified using any of the vowel modifiers from left ि to give कि, adding the right symbol ी to क results in की. Similarly, adding top symbol े gives के and bottom symbol ु results in कु. The composition of these symbols is represented as *CompositeSymbol*. The most important property of Indian languages is the broad choice of each of these symbols giving rise to an entire alphabet for a particular language. Figure §4.11[right] shows how vowel modifiers can be applied on four sides of a base consonant to give a set of composite symbols in a language. For most of the Indian languages, the number of vowel modifiers is 12 eventhough there are a few languages where the number can be less. Similarly, the number of consonants will vary from 12 to 36 for different languages.

The concept of *Resources* in the ontology helps in storing the data for respective syllables and phonemes in the form of *Text*, *Audio*, *Image*. We have observed that most of the eLearning Systems for Indian languages developed today rely on images for storing and displaying language aspects making it difficult to change the system. However, in our ontology we make an attempt to systematically separate different aspects to facilitate variety for a multitude of languages. The ontology also has the concept of *Vowel*, that is further divided into *SimpleVowel* and *CompositeVowel*. They are bound to *Syllable* through the property *hasVowel*. Similarly, *Consonant* class represents the consonants in a given language and can be either *SimpleConsonant* or *CompositeConsonant*. Most of the Indian languages have *ConsonantConjuncts* like క్క,క్ఖ,క్గ,క్ఘ,క్జ in *Telugu* language, whcih are special symbols formed with a number of syllables. *Modifier* is the core class for representing different modifiers like *VowelModifier* and *SpecialModifier*. *VowelModifier* in general consists of several signs often called as dependent vowels ◌ँ, ◌ं, ◌ः, ◌़, ◌ा, ◌ि, ◌ी, ◌ु, ◌ू, ◌ृ, ◌ॄ, ◌ॅ, ◌ॆ, ◌े, ◌ै, ◌ॉ, ◌ॊ, ◌ो, ◌ौ, ◌्. In Devanagari, there are twelve signs which when composed with consonants give rise to a number of composite syllables and is often called *Barakhadi* because of 12 modifiers. In addition to these, there are several special modifiers specific to a language. We have modeled *Phonemes* in similar lines to *Syllables* through respective concepts. *Numeral* class denotes the representation of symbols ૦ ૧ ૨ ૩ ૪ ૫ ૬ ૭ ૮ ૯ in a particular language. We did not get into writing part in detail even though we have left scope for further extension of the ontology. This entire exercise of creating detailed ontologies for adult literacy is to systematically specify different parts such that they become source of variety to facilitate semi-automatic development of eLearning Systems. A detailed overview of this ontology is provided in Appendix §A.4. In the next chapter, we will discuss how we have harnessed the common nature of families of Indian languages for design of eLearning Systems for adult literacy.

CHAPTER 5

Pattern-Oriented Software Product Lines

This chapter proposes a pattern-oriented software product lines approach for design of educational technologies. We motivate the need for software product lines in Section §5.1, discuss the ideas of product lines and software product lines in Section §5.2. We then discuss the different kinds of families that are present in the domain of instructional design in Section §5.3.1, adult literacy in Section §5.3.2 and eLearning Systems families in Section §5.3.3. Finally, in Section §5.4, we present our pattern-oriented software product lines approach, the sub-processes of this approach focusing on the unique characteristics of the approach.

5.1 Cost of Educational Technologies

There is a dramatic rise in the use of technologies in education in the last decade or so [250]. However, there is also severe criticism on several dimensions like huge upfront costs of technologies, difficulties in using them on the field, lack of evidence to show positive impact on quality of education [66]. One major challenge was an ever increasing effort required to develop and maintain a large number of educational technologies, which often ends up as an overburden on the teachers [66]. In a steering committee meeting of planning commission[1] organized by Government of India, we have proposed the need for technology-driven solutions to address several challenges in education like huge scarcity of qualified teachers and lack of quality[2] for adult literacy and in general for education in India. Several senior professors and deans in the meeting have raised three key pre-requisites for any proposed technology-driven solution. The technologies should be (i) based on strong pedagogical basis (ii) available for all Indian languages and (iii) flexi-

[1] *Planning Commission* is headed by the Prime Minster of India and is later replaced by NITI AYOG, a think-tank for guiding the Government of India

[2] Formulation of the 12th Five Year Plan- Minutes of the second meeting of the Steering Committee on Elementary Education and Literacy held on 25.08.2011-regarding

ble to cover varied teaching styles and different types of learners. Most importantly, the primary concern was in terms of the cost and effort required for creating and maintaining the technologies for the scale and variety inherent for education in India. For example, the TARA Akshar+ Literacy and Numeracy programme has made about 129,000 people literate across India at the cost of ₹5,500 (110 $) per student[3] unlike Government of India's available budget of ₹230 per student [4]. Even though the initiative seems to be successful, the entire technology has to be maintained by technical experts and updating these technologies with evolving curriculum and in multiple languages is quite expensive and infeasible. In addition, most of these technologies do not have a pedagogical basis denting quality of instruction.

It is here we wish to address some of these challenges by applying the idea of software product lines in software engineering towards design of educational technologies for adult literacy in India.

5.2 Product Lines & Software Product Lines

Samsung, one of the leading phone manufacturers in the world has released 50 smartphone models per year on an average in 2013, 2014, 2015[5]. *Toyota* releases its product line of cars like *Lexus* into the market with several variants of the base model that can be customized by the customer[6]. Established manufacturers in industries like automobiles, mobiles, laptops and so on release a base model followed by several variants of the product catering to varied market segments and customer needs. This idea of delivering customized products to customers based on individual requirements is a well-established approach in manufacturing and is generally known as *product lines*. For example, consider the case of buying a laptop from manufacturer's website, the buyer chooses the required features (*model*, *price*, *processor*, *OS*, *memory*, *hard disk* and so on) from the product catalogue and the product manufacturer assembles the product by customizing a base product in a product family and creates the customized product. The right hand side of Figure §5.1 shows laptops crawled from DELL website based on selected features from the left hand side. A product (laptop) can have several features of different types (*mandatory*, *optional*, *alternate* (exactly one of them), *or* (at least one of them)) providing various configuration options for the buyers and the manufacturer assembles them in an efficient way to produce products at lower cost and better quality.

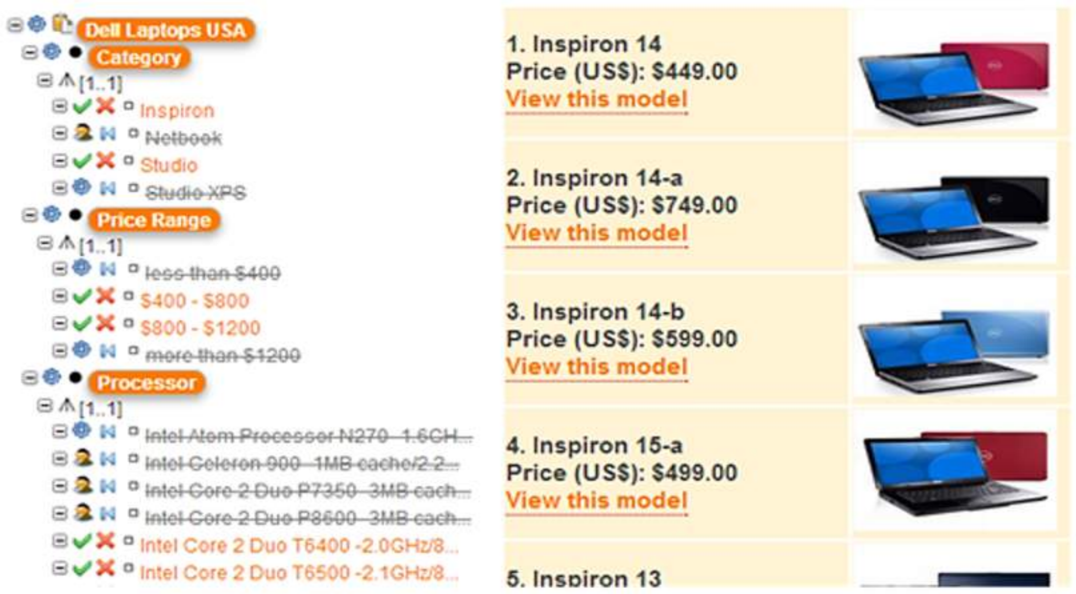
This idea of product lines applied to software is often considered as *software product lines*. However, there have been several debates on how and why software production pro-

[3]http://taraakshar.org/index.php/faqs/

[4]Personal Communication, Joint Director & Director General, National Literacy Mission Authority, India

[5]http://www.samsung.com/in/consumer/mobile-devices/smartphones/

[6]http://www.lexus.com/



Dell Laptops USA
Category
[1..1]
Inspiron
~~Netbook~~
Studio
~~Studio XPS~~
Price Range
[1..1]
~~less than $400~~
$400 - $800
$800 - $1200
~~more than $1200~~
Processor
[1..1]
~~Intel Atom Processor N270 -1.6GH...~~
~~Intel Celeron 900 -1MB cache/2.2...~~
~~Intel Core 2 Duo P7350 -3MB cach...~~
~~Intel Core 2 Duo P8600 -3MB cach...~~
Intel Core 2 Duo T6400 -2.0GHz/8...
Intel Core 2 Duo T6500 -2.1GHz/8...

1. Inspiron 14
Price (US$): $449.00
View this model

2. Inspiron 14-a
Price (US$): $749.00
View this model

3. Inspiron 14-b
Price (US$): $599.00
View this model

4. Inspiron 15-a
Price (US$): $499.00
View this model

5. Inspiron 13

Figure 5.1 Sample configuration options for product family members [*DELL* laptops]

cess is similar and different to manufacturing [251] and some authors have attributed these failures to the non-applicability of the factory concept to software [169] but Peter Wegner sums up the similarities and contradictions this way: "*Software products are in some respects like tangible products of conventional engineering disciplines such as bridges, buildings, and computers. However, there are also certain important differences that give software development a unique flavor. Because software is logical not physical, its costs are concentrated in development rather than production...*" [252]. *Mass production* and *mass customization* are two critical concepts that form the basis for most of these debates. According to the classic definition, "*Mass production is the tremendous increase in scale of the same product at low costs whereas Mass customization is a tremendous increase in variety and customization of the product without a corresponding increase in costs*" [253]. The primary focus in software is on mass customization unlike manufacturing. SPLs have matured from being a niche research area to a well established approach for software production catering to certain markets. SPLs offer several promises over one-off development like significant reduction in cost after an up-front investment on common assets (as shown in Figure §5.2), improved quality and consistency across all product family members, increased flexibility and better handling of evolution [2]. Even though not explicitly developed as SPLs, most of the software that is developed today like operating systems, databases, plug-ins for IDEs provide several configuration options [18]. For example, researchers have analyzed that linux kernel provides as many as 10,000 features for users as configuration options [254]. The *hall of fame* initiative of a 16 year conference on software product lines has published

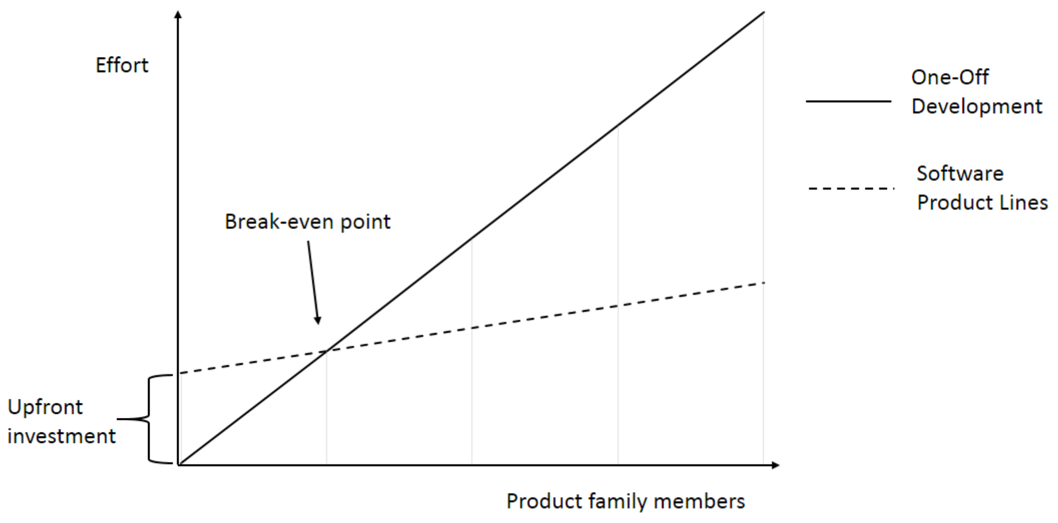


Figure 5.2 One-off development versus software product lines [2]

exemplary case studies of companies like *Boeing*, *Bosch Group*, *Siemens*, *HP* illustrating their significant gains of productivity for creating and customizing families of products in a product line [255]. Despite this success, there are several challenges associated with SPLs. While initial upfront investment is a known challenge, a less emphasized but critical challenge is increase of complexity with huge volume of features and increase in number of product family members. This has left SPL as a complex development approach that can be used by well-established organizations leading to decreased adoption of SPLs in several domains. For instance, there are not many instances of SPL applied to educational technologies [63] despite tremendous need and potential. In the next sections, we discuss the key differences between single systems and SPL development and motivate the need for SPL at three different levels of granularity in the domain of education.

5.3 From Individual Systems to Families

The predominant way of developing software today is either a generalized product like *Office*, *Gmail* or *Payroll* for a large number of customers or specific software that is designed for individual customers like in service based organizations. Consider the scenario of software for all banks? Is it same for all banks? Is it different for all banks? Similarly, if we consider software for all accounting systems? Is it same or different? The idea of looking at software systems as a family rather than completely different individual systems offers several advantages [15]. A family of systems can be at different levels of granularity with

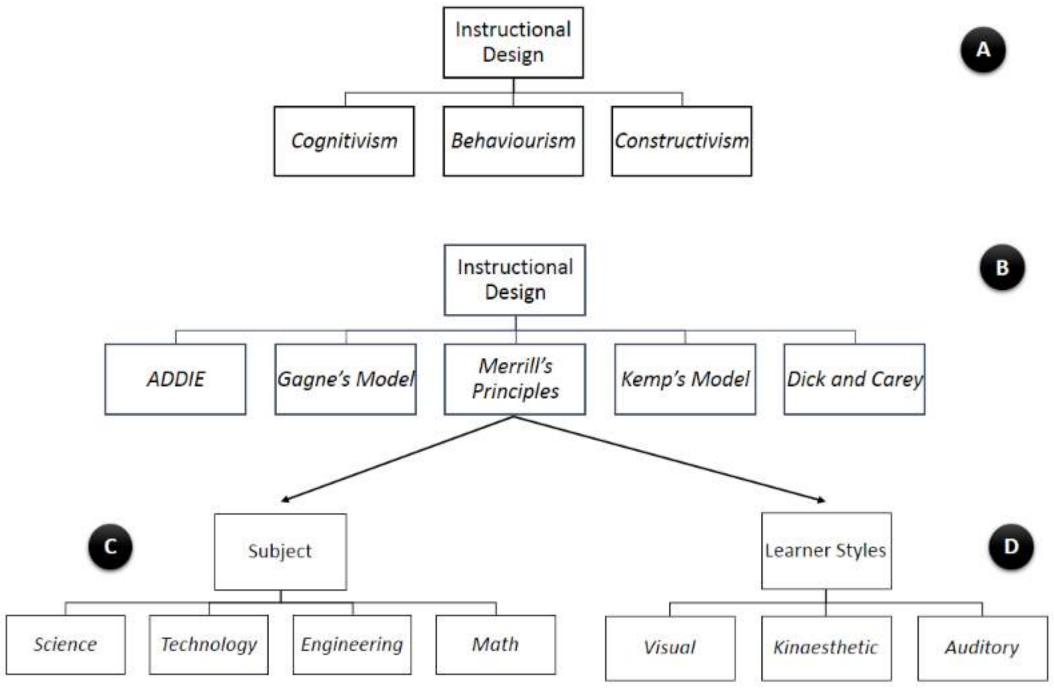


Figure 5.3 Few sample instructional design families

multiple sub-families within families and also a hierarchy of families. In this section, we discuss several families that are of interest to us in this thesis. A *family* of systems can be defined as *a set of systems that share more common properties with other members in the set than differences providing unique advantages to address the common and varying needs of specific markets* [15].

5.3.1 Instructional Design as families

In this thesis, we considered *Instructional Design* as a fundamental tenet that forms the basis for design of educational technologies. How is instructional design developed? Is it developed from scratch? or reused from existing resources? Are instructional designs common across subjects? learners? teachers? or universities? How many instructional design theories are present in the literature? What is common across them? Can instructional designs be considered as a family? Charles Reigeluth has extensively studied instructional design theories and models and documented them in three voluminous books [73][256][257]. His vision was to build a common knowledge base and a common language about instruction [73]. We can classify instructional design in the form of several families and at different levels of granularity as shown in Figure §5.3. For example, we can consider the three fundamental ways of *cognitivism*, *behaviourism* and *constructivism* or we can group them based on models like *Gagne's model* [258], *Dick and Carey's model* [259] , or the generic *ADDIE process* that is followed in most of the instructional designs. Each of these models can be grouped as a family based on subjects like STEM or K-12 or can also be formed as a family primarily based on learning styles like *Visual*, *Kinesthetic*, *Auditory*. Here [A] and [B] in Figure §5.3 are at one level of granularity whereas [C] and [D] are at the next level of granularity. This can be further refined into a hierarchy of families till the lowest level of granularity. The classification primarily depends on the goals of the specific target organization or stakeholders. For example, if a university has professors who are keen on using *"learning by doing"* approach, then a family of instructional designs can be modeled with *"learning by doing"* as the base approach and adapting it for different kinds of learners and subjects. On the other hand, if a university policy mandates accreditation with a national body, then all instructional design should use principles of accreditation as base and then adapt them for specific needs. It is mandatory that all members belonging to a family have certain common properties and vary on some aspects making them fit for the specific purposes.

The core idea is not to look at every instructional design as a unique case but as a family of similar but distinct instructional designs, to leverage the common properties of the family and facilitate flexible instructional designs. As discussed in Section §3.5.2, Merrill has distilled a large number of instructional design models and came up with five fundamental principles that are common across many instructional design models and there can be several variants based on these principles giving a family of instructional designs based on Merill's principles of instruction. We see the pattern categories identified in Chapter §3 as base for modeling instructional designs and variants. However, considering the generic scope of all instructional designs in the literature is out of scope of this thesis.

5.3.2 Adult Literacy as families

Should there be a universal technological solution for teaching all 287 million adult learners across India? or Should there be unique solution for every leaner?. This leads to a trade-off and need for a balanced solution between *one-size-fits-all* and *unique-for-every-dimension* solutions. NLMA, the highest authority for adult literacy in India has devised a uniform methodology called IPCL as the base for creating instructional designs for all languages across India [29]. The handbook of IPCL provides guidelines for customizing different aspects of instruction based on several dimensions [29]. The key goals for adult literacy primers are to create immense interest in the learners and provide functional knowledge that can add value to learners' daily life [205]. These common goals have to be customized for learners spread across India based on socio-cultural and local contexts.

Based on [205] and [29], we show a portion of family for organization of adult literacy domain in Figure §5.4. We use standard Lego blocks to show possible variations in adult literacy. Two common components of this family are *Process* and *Content*, where each one of them will have core aspects that are present in every family member and

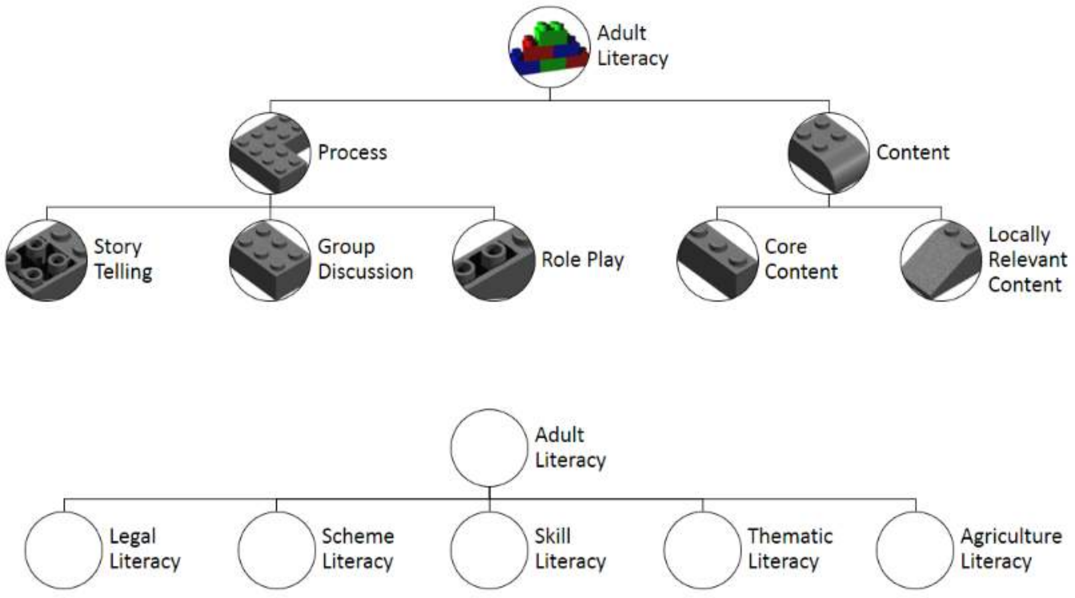


Figure 5.4 Few sample families of adult literacy domain

can also impose some constraints. The process followed for adult literacy in India is generally driven by *eclectic method* that starts teaching from known to unknown with gradual progress in learning. Now, any instructional design for adult literacy domain in India that comes under this family must follow *eclectic method* unlike *synthetic method* that teaches from alphabets to words. The process itself can have many number of activities like *StoryTelling*, *GroupDiscussion*, *RolePlay* that can be customized based on specific needs. This family also says that *Content* should be present and can be further divided into *CoreContent* and *LocallyRelevantContent*. *CoreContent* mandates topics like national integration, secularism, democracy, scientific temper, communal harmony, women's equality, population education and development, etc. whereas *LocallyRelevantContent* will be customized by specific states and stakeholders based on learner's livelihood, their socio-cultural realities, special issue-based and thematic aspects such as gender parity, health and hygiene, agricultural, animal husbandry, self-help groups, local self-government, livelihood programmes, etc. Then same family can be classified based on the primary *topic/knowledge* that can be used to teach knowledge that will be useful for learners in their daily life. *Legal Literacy* can focus on teaching learners laws pertaining to them, how to seek help from law whereas *Scheme Literacy* can provide knowledge of Government schemes. *Skill Literacy* can help them in gaining knowledge pertaining to a particular skill like tailoring, plumbing and so on. *Thematic Literacy* is a generic way to accommodate themes that can be local to the specific audience and *Agriculture Literacy* can provide farming knowledge. Each of these families can mandate that any adult literacy instructional design based on the corresponding family should be within the scope of the defined goals and constraints. These common themes are combined with specific needs of the particular segment of learners to deliver a specific and customized instructional design. These commonalities and variabilities among family members are generally modeled using feature diagrams in SPL [14][18].

5.3.3 eLearning Systems as families

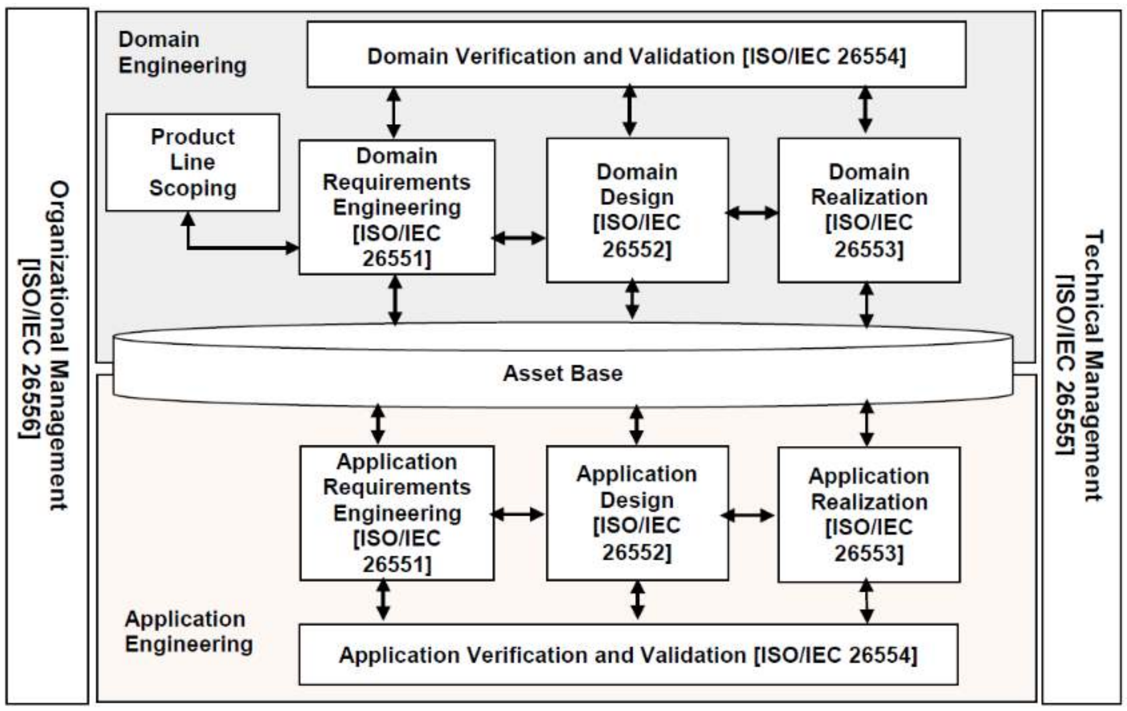
Eventhough we do not focus on modeling variants of user interface in this thesis, it can be a critical source of variability for eLearning Systems. There exists certain ways to model user interfaces [260] as a hierarchy of Presentation Units that allow navigation between them and each of the units further contain UI elements like buttons, textboxes and so on. These UI elements have properties like *name*, *data* and so on. Each of these elements also have properties to describe their visual appearance like *color*, *font* and so on. The user interface can be modeled for different platforms like *Desktop*, *Web*, *Mobile* or the user interface can also be modeled using *structural* and *behavioral* elements. These elements can be organized at a higher level of abstraction in different ways giving several user interface variants. For example, there can be several *Views* of Model-View-Controller pattern corresponding to different variants of user interfaces. In the case of adult literacy eLearning Systems, we are interested to model the user interface elements primarily with three resources *text*, *image*, *audio*.

5.4 Pattern-Oriented Software Product Lines

Over the last two decades, there has been an extensive research on SPL as evidenced through a series of focused conferences like Software Product Lines Conference (SPLC), workshops like Variability Modelling of Software-intensive Systems (VaMoS), Product Line Approaches in Software Engineering (PLEASE). An analysis of the literature on SPL reveals that are two major terminologies to discuss the idea of developing a family of software-intensive systems. Software Engineering Institute (SEI) has steered the research and development on software product lines (sometimes called as software product family) and has published several technical reports and case studies [261]. On the other hand, several researchers and organizations also used the term “software product line engineering” for their work in the area of SPL [2]. Several organizations, universities and research institutes performed collaborative research on SPL, which supported the systematic building of a community of software product line engineering research and practice. Some of those projects include ARES (1995-1998), PRAISE (1998-2001), ESAPS (1999-2001), CAFÉ (2001-2003), and FAMILIES (2003-2005). In 2014, Metzger and Pohl have done an extensive study of 600 articles published in the area of SPL and noted that there has been impressive quantitative and qualitative progress in the field with key challenges for industrial adoption [262]. Krueger has suggested three ways of adopting SPLs [263] (i) *proactive*, in which the entire product line is planned and developed from scratch (ii) *extractive*, that focuses on analyzing a set of existing products and moving towards an SPL (iii) *reactive*, that starts with one product and extends into an SPL. Depending on the product line strategy an organization can choose the appropriate product line adoption approach. There are several approaches for development and analysis of SPLs in the literature [16][2][264][265][266]. One related approach proposed for the domain of flight control focuses on using architecture and design patterns for SPL [267] but largely confines to modeling variability. Researchers have also used ontologies for modeling and configuring variability during SPL [268][269] [270]. A software product line is developed based on ontologies for developing knowledge-driven semantic web applications [271] to facilitate interoperability between semantic services and intelligent agents. A more detailed account of research in SPL can be found in [262] and a recent bibliographic analysis of research in SPL over 20 years is provided in [272].

On the other hand, many of the SPL approaches have been applied in the last couple of decades in practice across several domains with successful results [255]. However, there is sparse research on applying SPL in the domain of technology enhanced learning [273]. Pankratius has proposed PLANT as a product line based approach for creation and maintenance of digital information products [59]. In our prior work, we proposed TALES as an approach for automating the development of eLearning Systems [11]. A software product line methodology for development of e-learning system for a six sigma course was proposed in [274]. A domain engineering activity for interactive learning modules is proposed in [275]. [276]. However, none of these approaches consider instructional design as the basis and focus on adult literacy which is the goal of this thesis. After a critical analysis of literature, we find that SPL is largely undermined in technology enhancing learning community despite their significant potential and hence motivating our approach.

A standard reference model is created for a set of software and systems product line standards [4] based on the seminal work of software product line engineering [2]. Figure §5.5 shows the set of product line standards mapped to reference model in [4]. We briefly explained this architecture and the role of patterns in these product line standards in [3]. In this thesis, we extend this reference model and propose a pattern-oriented software product line approach (in short *PoPL*) as shown in Figure §5.6. Our approach primarily differs from existing literature in terms of using *patterns* and *ontologies* as a central knowledge base for driving several sub-processes of a traditional SPL. This emphasis on patterns has been used as one of the critical foundations for several software and systems product line standards devised by ISO/IEC JTC/SC7 [3]. In essence, SPL consists of two distinct life cycles of domain engineering and application engineering unlike single-systems engineering [2]. Domain engineering and application engineering further consist of requirements engineering, design and realization. Each of these processes and sub-processes can be



Organizational Management [ISO/IEC 26556]
Domain Engineering
Domain Verification and Validation [ISO/IEC 26554]
Product Line Scoping
Domain Requirements Engineering [ISO/IEC 26551]
Domain Design [ISO/IEC 26552]
Domain Realization [ISO/IEC 26553]
Asset Base
Application Requirements Engineering [ISO/IEC 26551]
Application Design [ISO/IEC 26552]
Application Realization [ISO/IEC 26553]
Application Engineering
Application Verification and Validation [ISO/IEC 26554]
Technical Management [ISO/IEC 26555]

Figure 5.5 Reference model for software and systems product lines from [3][4]

considered as high-level patterns (akin to software architecture patterns) that emerged from recurring and standardized practice from SPL communities, researchers and practitioners. These processes are further refined to explicitly model subprocesses, activities and further detailed as tasks to deliver a product line. Also, a lot of effort goes in creating and maintaining variability in the life cycle of an SPL.

We discuss the core distinctions of our approach when and where applicable from traditional SPL rather than repeating well-established processes in SPL. We also briefly discuss some aspects of an SPL for modeling a family of instructional designs as an example along with the approach. The concrete SPLs are detailed in Section §6.3 and Section §6.4 respectively.

A. Product Line Organization

The need for a product line usually emerges from the business need of an organization[s] which provides the rationale for why an SPL can be considered at all, the economics and feasibility of SPL. We have observed that in the case of adult literacy in India, the need emerged from a great societal challenge and has several organizations like NLMA of Government of India, researchers in IIIT-Hyderabad, Corporate Social Responsibility of TCS, NGOs and so on. This led to several challenges [63] and is unlike traditional SPL where the organization that is motivating SPL has a strong business case driving the entire initiative.

B. Product Line Analysis & Scoping

An important prerequisite for SPL is a deeper understanding and analysis of the commonal-

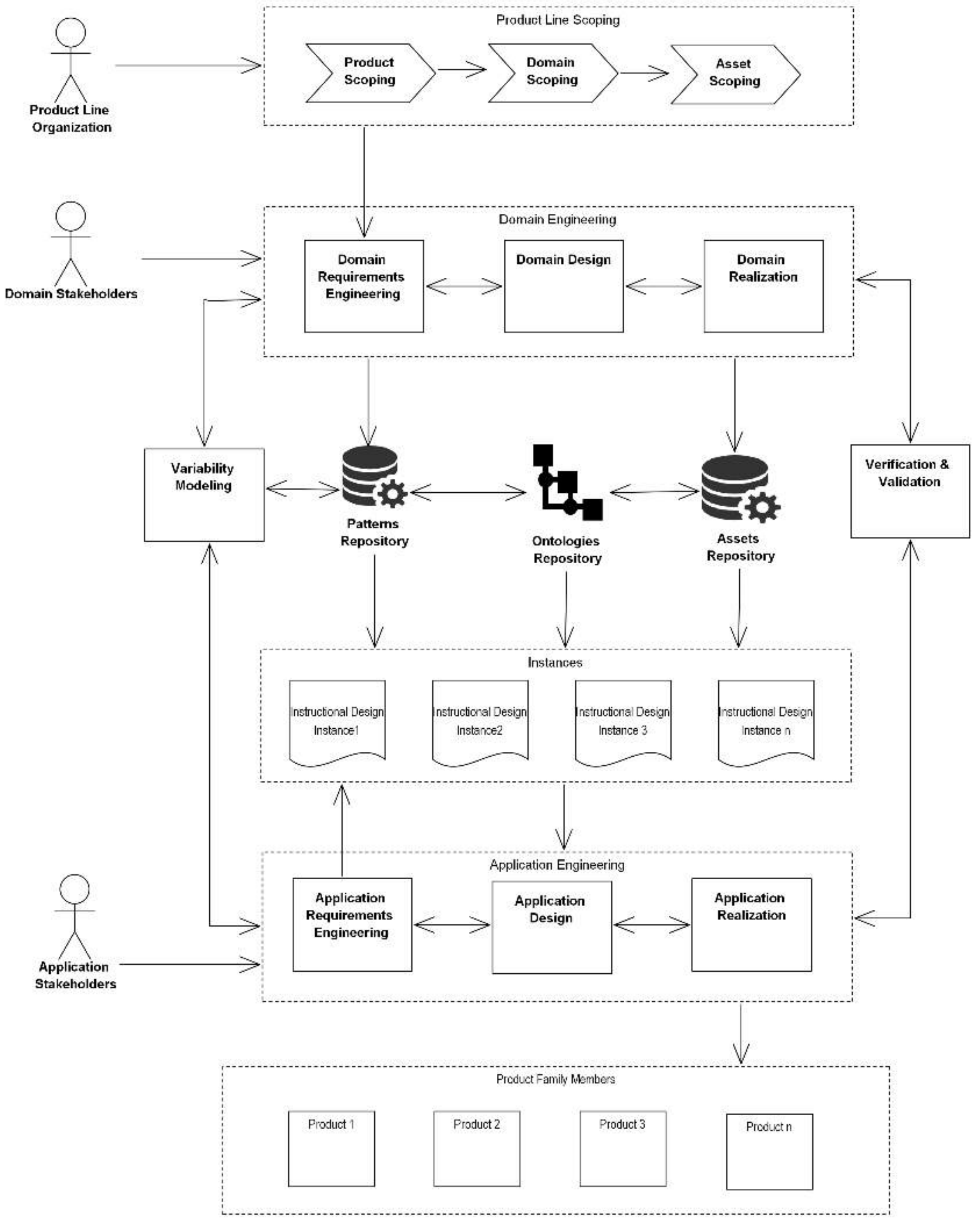


Figure 5.6 Overview of Pattern Oriented Software Product Lines

ity and variability in a particular domain. In our approach, we consider scoping as a critical activity that is performed prior to domain engineering and provides guidelines and directions for the rest of the approach. We consider a constrained view of product line scoping from the ISO/IEC 26551 standard [277] on product line requirements engineering. The primary goal of product line scoping is to identify the member products of a product line along with their major (externally visible) common and variable features and analyze the products from economic and development perspectives. This sub-process further consists of (i) *product scoping*, that determines target markets, product categories, common and variable features and estimated timelines for development of product line members (ii) *domain scoping* focuses on understanding the potential of the targeted product line market and refines product scoping (iii) *asset scoping* identifies existing assets and their potential for reusability from a development perspective.

C. Domain Stakeholders

Designing and implementing an SPL requires involvement of stakeholders from various backgrounds. Domain stakeholders are concerned with not one product member but the entire product line and they provide inputs for the entire domain engineering life cycle. The main activities of domain stakeholders include market analysis, trade-offs between various product members, strategizing the feasibility of a product line and so on. For adult literacy, we had a diversified range of stakeholders like learning science experts, software engineering researchers, Government officials, NGOs and so on. However, based on goals of the product line organization, priority of stakeholders and their requirements is taken into consideration. Most importantly, each of the stakeholders can have different viewpoints. For example, Instructional Designers focus on designing the process of instruction and learning materials using appropriate learning theories, Language Experts are interested in standardizing instructional material for all Indian Languages and their dialects, Government Authorities are involved at multiple levels for monitoring the quality of instruction and timely delivery, Software Developers and Maintainers focus on developing and maintaining eLearning Systems, Voluntary Organizations help in delivery of systems on the field. Naturally, *instructors* and *learners* are two primary stakeholders for instructional design product line. The main output of this process is to provide key inputs for domain engineering subprocess of SPL.

D. Feature Modeling

Feature modeling is the most widely used notation to capture the commonalities and variabilities of an SPL [14][2][18]. Essentially, feature models represent all the common and variable features of an SPL. Feature models are extended with different notations like [278][189][279], formalisms [280][281][282] and so on. Several researchers have focused on formalizing feature models [283][284] and verification of product lines using formal semantics [285]. The increasing number of large-scale features models motivated researchers towards a domain specific language for managing features [286]. A detailed

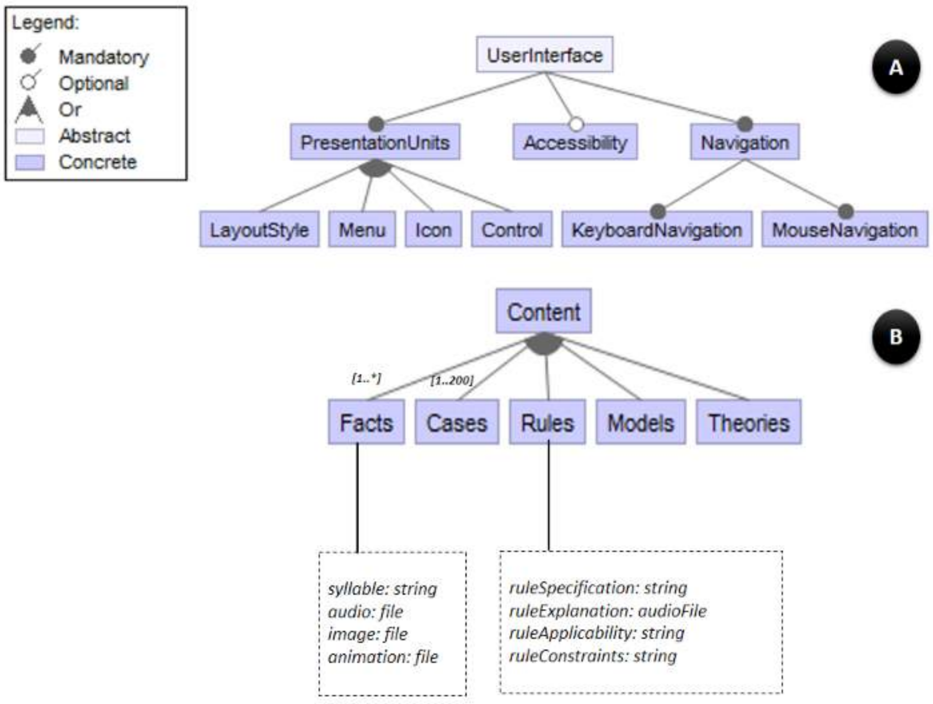


Figure 5.7 Some feature modeling notations

overview of feature oriented product lines research is presented in [18]. One particular goal that was of interest to us was to model *knowledge* with features. For example, how to add metadata to features such that this data can be later used during application realization for creating customized product family members? *Feature attributes* are a way to associate a feature to a type like integer or string or to a group of sub-features [287]. This was later extended with cardinalities, feature diagram references, and user-defined annotations [288] to make feature diagrams more descriptive and to facilitate staged configuration of features [289]. A notable work in this direction is the mapping of feature models as views on ontologies [290]. Peng et al. have proposed an ontology-based feature modeling approach as a way to formally represent feature models [291] and Wang et al. used OWL for verifying feature models [292]. Within ontology-based feature modeling, an empirical study conducted in 2015 showed that using individuals or instances of ontologies can provide further flexibility for reconfiguration of products [293]. A formal language is proposed to deal with non-boolean features focusing on feature attributes and multi-features using formal methods [294]. Despite this progress, annotating knowledge to feature models is still an open problem especially in the domain of education. We are not aware of existing methods that address the basic requirement of annotating features with images, audio and domain knowledge in an effective way. For example, every *fact* that is part of *ContentPattern* in instructional design can be associated with a syllable and sound, with provision for customization. While we do not attempt at addressing this problem in this thesis, we point this as an important future direction. To the best of our knowledge, we are not aware of existing approaches that use domain patterns as a base for feature modeling. Figure § 5.7[A] shows a fragment of user interface feature model using basic notation and Figure §5.7 [B] shows features annotated with cardinalities [1..*], [1..200], in this case there can be a number of *facts* as part of content but *cases* are limited to a maximum of 200. Cardinalities are initially criticized for adding clutter to feature models but were shown to be useful in specific domains like embedded software [287]. We use cardinalities in the domain of education as they can help in specifying several attributes of fetures. For example, for a feature like a *lesson* in an instructional design can have a maximum of 3 learning goals and can be modeled with cardinalities. Most importantly, we see feature attributes as a key extension of feature models that can immensely help product lines in this thesis. *Facts* can be associated with the attributes of *syllable* of type unicode *string*, a vocal rendering of the syllable using an *audio* file attribute, an optional attribute for storing *image*. Each of these attributes can be used for realizing variability. A feature like *Rules* can have *ruleSpecification*, *ruleExplanation*, *ruleApplicability*, *ruleConstraints* as some key attributes for storing knowledge about the corresponding feature. A detailed instructional design feature model is listed in Appendix §B.1.

E. Domain Engineering (DE)

Domain Engineering is one of the fundamental pillars of SPL that is concerned with encapsulating past experiences of building systems in a particular domain by creating reusable assets and providing means to develop new systems by utilizing and reusing these assets [2]. In short, domain engineering is concerned with defining and realizing the commonality and variability of an SPL, thus establishing a common platform for developing high-quality applications rapidly within the scope of the product line. Domain engineering involves analyzing common and variable features of product line members and provides a common platform for rapidly creating customizable product members. Domain engineering has three sub-processes which handle three key aspects (i) Domain requirements engineering (ii) Domain design (iii) Domain realization. Domain requirements engineering is concerned with determining the common and variable features of a product line with inputs from product line scoping. The output of this sub-process is used by domain design to produce a reference product line architecture for all product line members addressing variability at an architecture level. We advocate the use of pattern oriented software architectures at this stage of processes and map it with patterns in the domain. Commonality and variability analysis with respect to architecture design is also done as part of this sub-process. The reusable assets required to implement product line architecture are developed as part of domain realization focusing on variability mechanisms. Common assets are usually built as components and services with exposed interfaces. These assets should be built with provisions for implementing variabilities by using different mechanisms like patterns, aspects, pre-processors and so on. Domain engineering does not actually deliver concrete assets but produces a set of reusable assets at different levels of granularity, which can be used for further realization.

F. Verification & Validation

Verification and validation is a sub-process that has to be performed in relation to every other sub-processes in the entire product line. In domain engineering, verification and validation is concerned with creating the right common assets and verifying whether those assets would satisfy variability needs of the product line. However, verification and validation is more important in application engineering as most of the assets are utilized for concrete implementation of the product family member unlike abstract and partial assets in domain engineering. This sub-process includes testing methods and assets to verify the quality of product members in the family and in turn the product line.

G. Variability Modeling

SPLs are built for the long run, unlike single-systems engineering and hence would require management of several aspects. One of the most critical aspects for the success of SPL is the explicit modeling of variability during different sub-processes in the product line. There is lot of research in the literature for modeling variabilities in an SPL depending on the specific requirements of the product line [265]. Variability is generally categorized into *essential variability*, from users' perspective and *technical variability*, from realization perspective [2]. A detailed overview of variability modeling approaches is surveyed in [265].

H. Patterns, Ontologies & Assets Repository

The core distinction of our approach from existing literature stems from the use of patterns and ontologies for modeling domain knowledge and using them as the base for semi-automatic generation of family members. The patterns repository is essentially a collection of common patterns with scope for implementing variations derived from domain engineering. We consider both domain patterns and software patterns as key inputs to application engineering for creating specific product family members. Patterns provide an opportunity to create variants as discussed in Section §3.3. For example, the *ContentPattern* can be instantiated for several Indian languages with varying *facts*, *cases*, *rules*, *models* and *theories*. We consider patterns as common assets that can be further customized and refined at later stages in the product line to support variability at a high level of abstraction. For instance, *ContentPattern* can be adapted to support Open Educational Resources based on semantic annotations. We use this patterns repository as the base for creating ontologies corresponding to patterns. We then use these patterns to create an ontology repository consisting of specific ontologies. In our approach, we use *patterns* to model both problem space and solution space of the domain, features to model problem space from a user perspective and ontologies to concretely represent both of them. Assets repository is a critical resource for both domain and application assets. Assets may include patterns, features, models, requirements specifications, architectures, components, test cases and other resources. The primary outcome of these repositories is to create different kinds of *reference product lines* that serve as baseline for product family members. This idea of multi-level product lines is discussed in the next chapter in Section §6.2.

I. Application Stakeholders

Most of the processes and tasks in SPL are partial and incomplete till the application requirements are specified by application stakeholders. These stakeholders consider the requirements of domain stakeholders, variabilities that are provided by the SPL and specify application specific requirements. All domain stakeholders can be application stakeholders but in a limited capacity based on specific goals of the application in context.

J. Application Engineering

Application Engineering is a process that is concerned with deriving specific product family members by strategically reusing domain assets and by exploiting the variability built into the domain platform. Based on common requirements and common features that are built into the platform during domain engineering, the main goal of application engineering is to configure and create the specific product family member from the given set of features. Like domain engineering, application engineering consists of three core sub-processes (i) application requirements engineering involves developing application-specific requirements reusing common and variable requirements (ii) application design derives a specific architecture from reference architecture (iii) application realization implements the concrete features of the product family member. Eventually, product family members result as an outcome of application engineering. Feedback received during application realization is traced back to different processes in SPL and is further used to improve the design of product line.

Both domain engineering and application engineering have an iterative life cycle involving requirements, design, realization, testing [2]. In addition to these, there are also two key sub-processes of technical and organizational management in the SPL life cycle from an organization perspective [4]. Considering the vast number of processes and sub-processes in SPL and the effort required for performing these tasks, a common practice is to implement core or mandatory processes during development of SPL [263].

We will discuss the two product lines developed as part of this thesis in the next chapter.

CHAPTER 6

Evaluation & Implementation

In this chapter, we discuss how the patterns and ontologies presented in Chapter §3 and Chapter §4 form the basis for creating software product lines for instructional design and eLearning Systems. In Section §6.1, we briefly discuss the strategy for evaluating the work in this thesis and in Section §6.2, we present a multi-level software product line covering the scope of product family members in this thesis. We present a software product line for a family of instructional designs in Section §6.3 along with feature models, feature attributes, feature configurations in subsequent sections. We then discuss a reference architecture in Section §6.3.4 that is used as the base for designing prototype platforms for instructional design and eLearning Systems. In Section §6.4, we give two concretely examples of eLearning Systems for Hindi and Telugu language generated from our prototype platforms. The experimental results and cost savings of using product lines in this thesis are presented in Section §6.5. We finally end the chapter with challenges of applying software product lines for educational technologies.

6.1 Overview

This thesis is set at the intersection of educational technologies and software engineering for addressing the challenge of scale and variety emerged in the context of designing technologies for adult literacy in India. As discussed in Section §1.5, several aspects related to learning methodologies, field assessments are beyond the scope of this thesis, even though they are natural extensions for future work. We demonstrate our approach for design of educational technologies for scale and variety using two case studies of product lines, one for instructional designs and another for adult literacy eLearning Systems. In the last decade or so, SPL community has witnessed a voluminous number of tools from academia as well as industry to support the entire software product line development life cycle [265][281].

We have primarily used two tool suites for modeling features in our SPL (i) *FeatureIDE* is developed on top of *Eclipse* and is quite useful as it supports multiple feature modeling techniques and also for generating code in several programming languages [295]. (ii) feature modeling plugin from University of Waterloo is a dated solution specifically useful for cardinal features and feature cloning and feature attributes [296]. For example, a *fact* in *ContentPattern* as a feature should be cloned for various instances. In their research, the same group has produced a minimalistic modeling language called *Clafer* and a set of tools as part of SPL platform [297]. The primary goal of *Clafer* is to address long standing concerns (merging feature and class models, mapping features to component configurations) in feature modeling by integrating feature modeling and meta modeling with rich semantics [297]. However, we realized that owing to the specific requirements from educational technologies domain, there is a strong need to extend the idea of feature attributes such that data pertaining to aspects in instructional designs can be annotated with feature models. For demonstration purposes, we have manually annotated features with concrete data for further processing by tools. In the next sections, we discuss our on-going work on developing product lines as part of demonstrating the approach proposed in this thesis.

6.2 Multi-Level Software Product Lines

Figure §6.1[A] succinctly summarizes different levels of product lines that we considered in this thesis. We reiterate our notion of instructional design as a set of *goals*, *process*, *content*, context, evaluation, environment and so on towards facilitating learning. Figure §6.1 [A] is a meta-level product line that deals with creating specific instructional designs from a chosen base instructional design. Here, there could be several sub-product lines focusing on a particular instructional design. For example, an instructional design like *learning by doing* [LBD] might be chosen as the base for all instructional designs in a particular university. Then, the derivations of LBD customized as per specific requirements of the courses in the university form a product family. Here, the input is a specification or schema of an instructional design and can consist of all features [including pre-requisites, activities, assessment and so on]. All product family members might not require all the features of LBD and hence only a subset of this instructional design specification is required for specific instructional design requirements. The scope of this meta product line is to create custom instructional design specifications based on a given instructional design specification. Similarly, there can be a number of sub-product families within this product line pertaining to a type of instructional design inquiry-based learning, IPCL for adult literacy and so on.

How to create instances of the custom instructional design specifications? Figure §6.1[B] shows a product line at the next level whose product family members are custom instructional design editors that take an instructional design scheme. A detailed listing of the

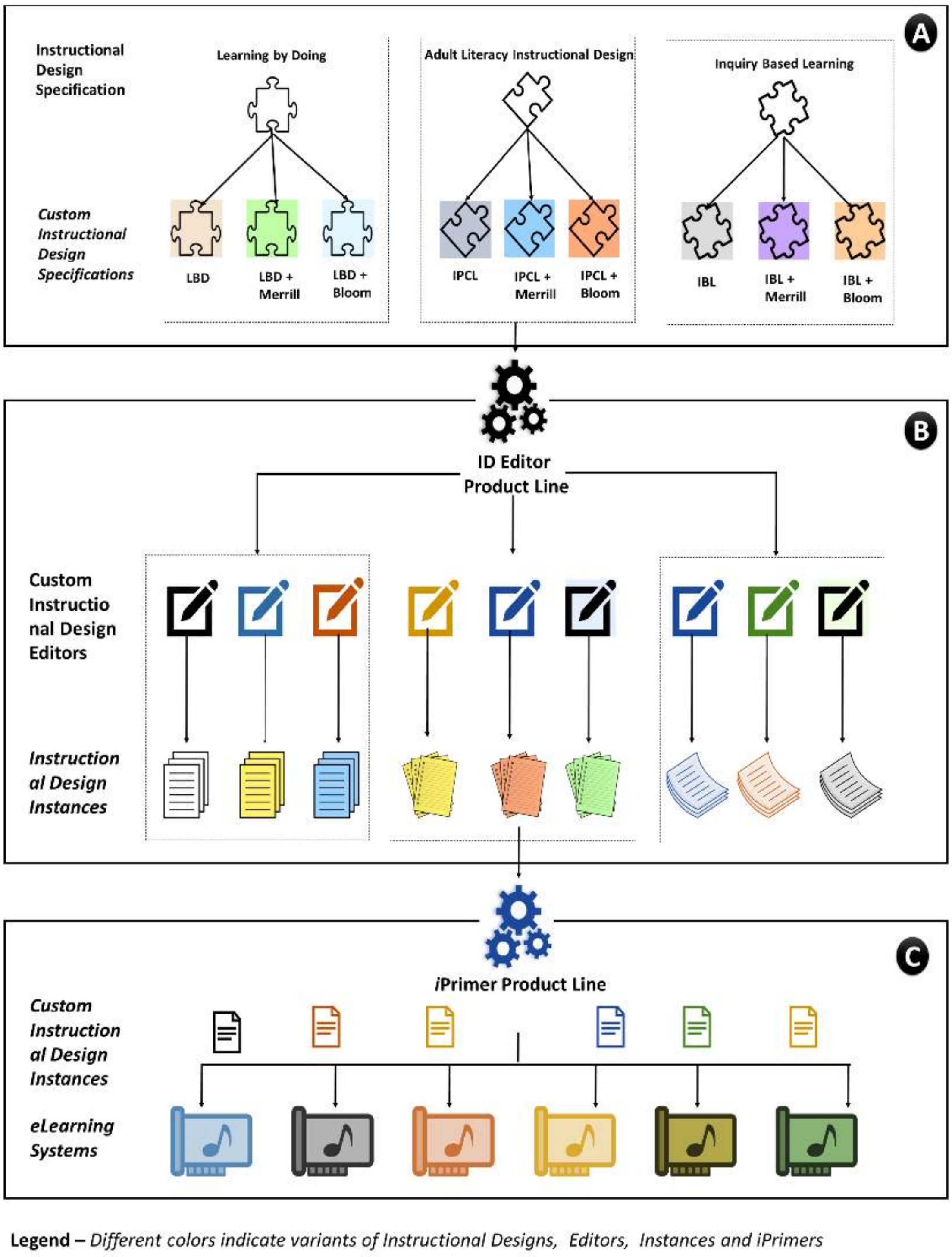

A
Instructional Design Specification
Learning by Doing
Adult Literacy Instructional Design
Inquiry Based Learning
Custom Instructional Design Specifications
LBD
LBD + Merrill
LBD + Bloom
IPCL
IPCL + Merrill
IPCL + Bloom
IBL
IBL + Merrill
IBL + Bloom
B
ID Editor Product Line
Custom Instructional Design Editors
Instructional Design Instances
C
iPrimer Product Line
Custom Instructional Design Instances
eLearning Systems
Legend – Different colors indicate variants of Instructional Designs, Editors, Instances and iPrimers

Figure 6.1 Multi-level software product lines in this thesis

instructional deign specification is provided in Appendix §B.2. We designed a prototype to generate these custom editors based on the specific instructional design specifications (instances). Each of these editors can be used to generate the concrete instructional designs with data. Even though motivated by adult literacy, these two product lines are in the context of generic instructional design. To co-relate with literature from educational technologies, these editors are similar in principle to learning design editors like *ReLoad* and *ReCourse Editor* [133], *ASK-LDT Editor* [134], *LAMS* [40], *Learning Designer* [181], *COLLAGE* [132], *Web-COLLAGE* [37], *ILDE* [298] [136] and so on, where each of these editors are single system development initiatives as part of EU funded projects unlike the proposed product line approach.

Figure §6.1[C] shows the next level of product line that is specific to a custom instructional design specification, in this case one based on IPCL and adult literacy instructional design. We designed a prototype that takes a specific instance of adult literacy instructional design and generates eLearning Systems, which are the product family members for this product line. In the next sections, we succinctly describe the two product lines for instructional design and eLearning Systems.

6.3 A Software Product Line for a family of Instructional Designs

6.3.1 A Basic Feature Model

How to model the mammoth number of instructional designs in a systematic way? Based on patterns discussed in Chapter §3 and ontologies in Chapter §4, we present a feature model for modeling a family of instructional designs. Here, we consider standard definitions from SPL literature [18] where a *feature* is a characteristic or end-user-visible behavior of a software system, a *feature model* essentially consists of all the features of a product line and their relationships. A *product* member of a product line is specified by a valid feature selection. Figure §6.2 shows a generic feature model created using *FeatureIDE* and consists of mandatory features *GoalsPattern*, *ProcessPattern*, *ContentPattern*, *EvaluationPattern*, and optional features *ContextPattern*, *EnvironmentPattern*, which means that any instructional design created from this model must specify these aspects as per the constraints posed in the feature model. For example, the instructional designer has a choice between two ways of specifying goals namely *Bloom* or *ABCD* technique. Figure §6.3 shows few more details of a feature model for instructional process based on *ProcessPattern* and *ProcessOntology*. However, as specified in Section §6.3.2, we are interested in feature models with cardinalities, feature attributes and hence we use feature modeling plugin.

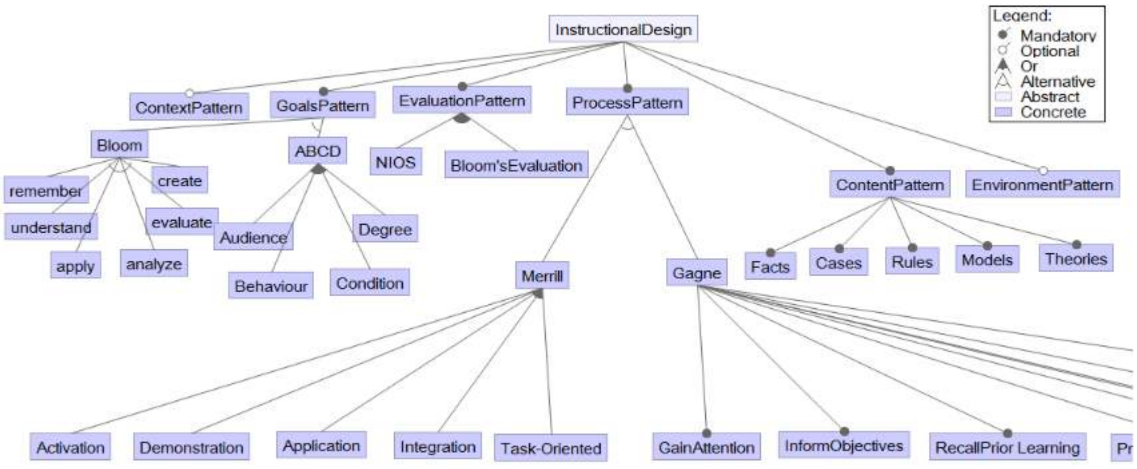


Figure 6.2 A fragment of instructional design feature model

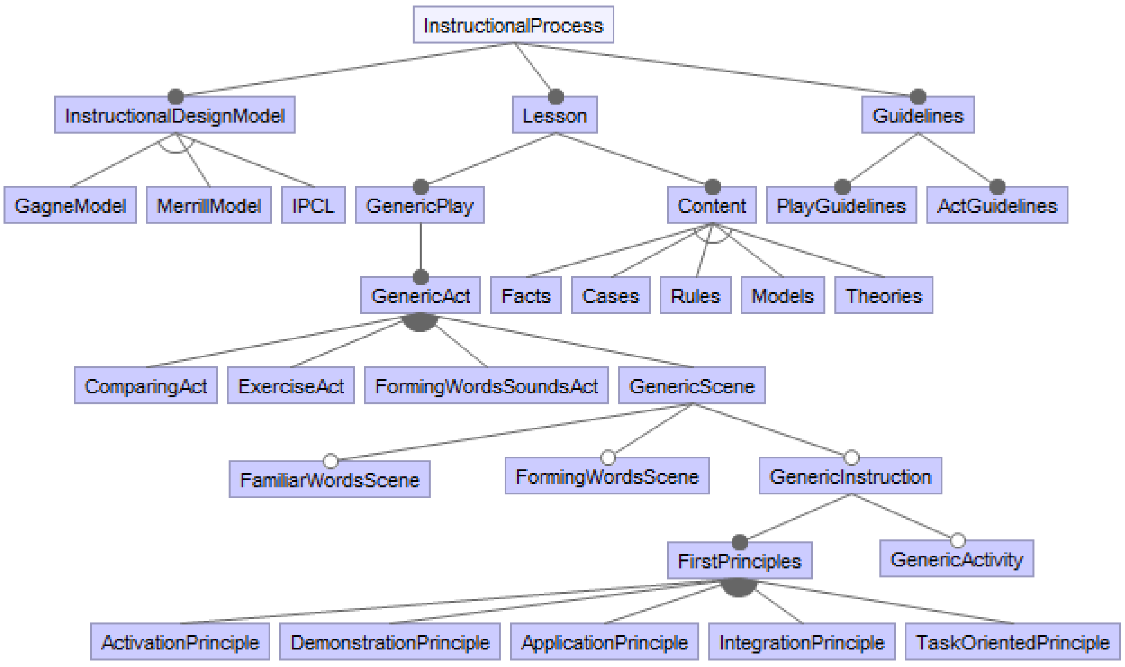


Figure 6.3 A fragment of instructional process feature model

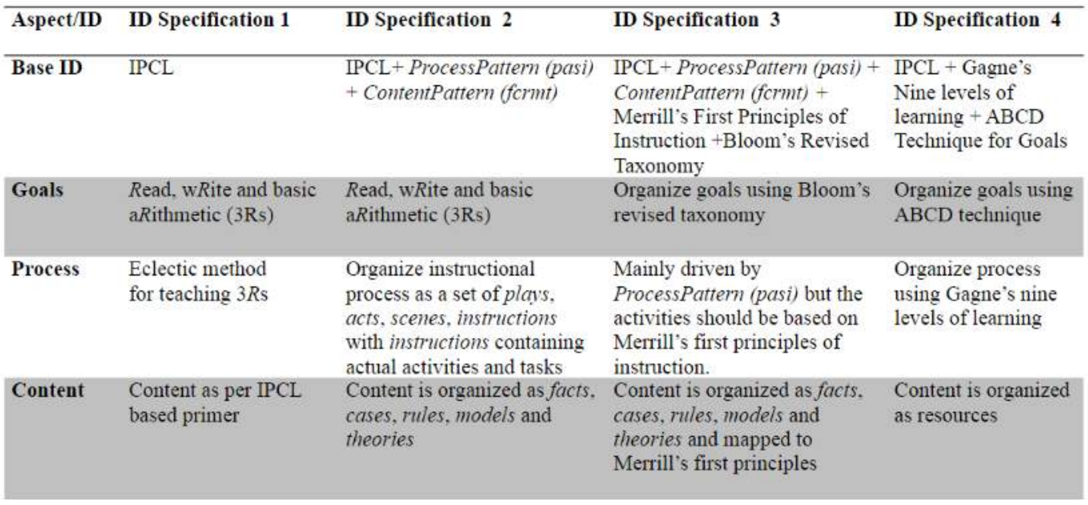
6.3.2 Feature Attributes

Feature models primarily specify the features of all product members in a product line primarily from a user perspective. However, if feature models have to be used for (semi-)automatically generating product members or in providing a partial implementation from domain engineering, then feature description alone might not be sufficient and features have to be extended with additional knowledge. For example, to represent *syllables* like इ, पि in adult literacy, a text in *unicode* should be associated with every feature of that type. Similarly, a goal might have a priority and can be *High*, *Medium*, *Low*. This data can be used by tools during application engineering. However, it was studied that cardinalities and feature attributes make it difficult for verification of valid feature configurations and hence could be useful in only specific domains [287]. In our case, we use cardinalities to impose constraints on the product member and annotate features with data to facilitate further processing by tools. While a feature modeling plugin for *Eclipse* supports feature attributes [296], it is a preliminary prototype developed way back in 2004 and was moved towards the direction of formal verification of features through *Clafer* platform[297]. This need from educational technologies domain requires features to be more powerful and expressive than current notations. This is a future direction beyond this thesis and we restrict ourselves to manually annotate features with attributes related to instructional design for our purposes.

6.3.3 Product Family Members, Feature Model and Feature Configurations

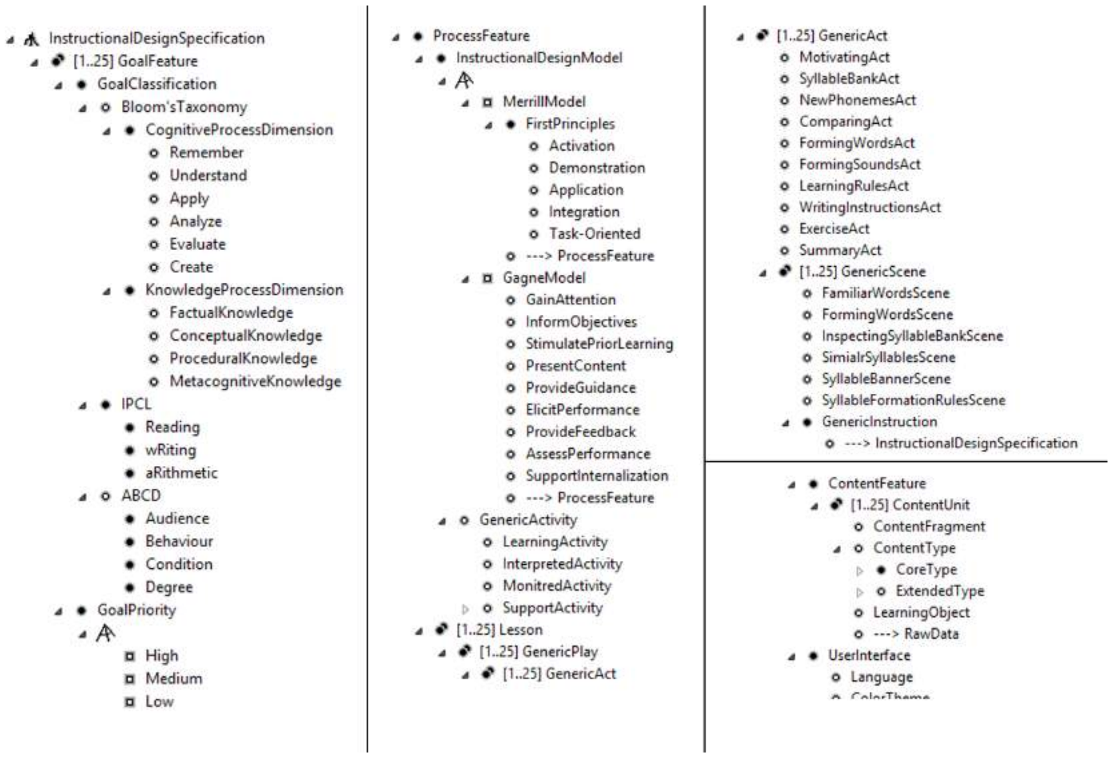
Figure §6.4 shows a brief description of requirements of four different kinds of instructional design specifications for adult literacy. As discussed throughout the thesis, IPCL is the base instructional design for all instructional designs for adult literacy in India. For ID Specification 1, the base ID is provided by IPCL consisting of a set of guidelines for creating primers for all Indian languages based on a core structure, process and content. The essence of IPCL concept is to teach by creating relevant content for learners. Figure §6.4 shows three concepts namely *Goals*, *Process* and *Content* for different instructional design specifications. The primary goals are Reading, wRiting and aRithmetic at three levels as per the progress of the learners. IPCL describes that an instructional process can be based on synthetic, analytic or eclectic method but suggests use of *eclectic method*. Content is organized as instructional material in the printed primer. There are several primers that are prepared based on this specification and the instructional designer should be able to model them using the product line.

ID Specification 2 family uses patterns in Chapter §3 to describe the Process and Content aspects of the instructional design whereas ID Specification 3 family uses Bloom's revised



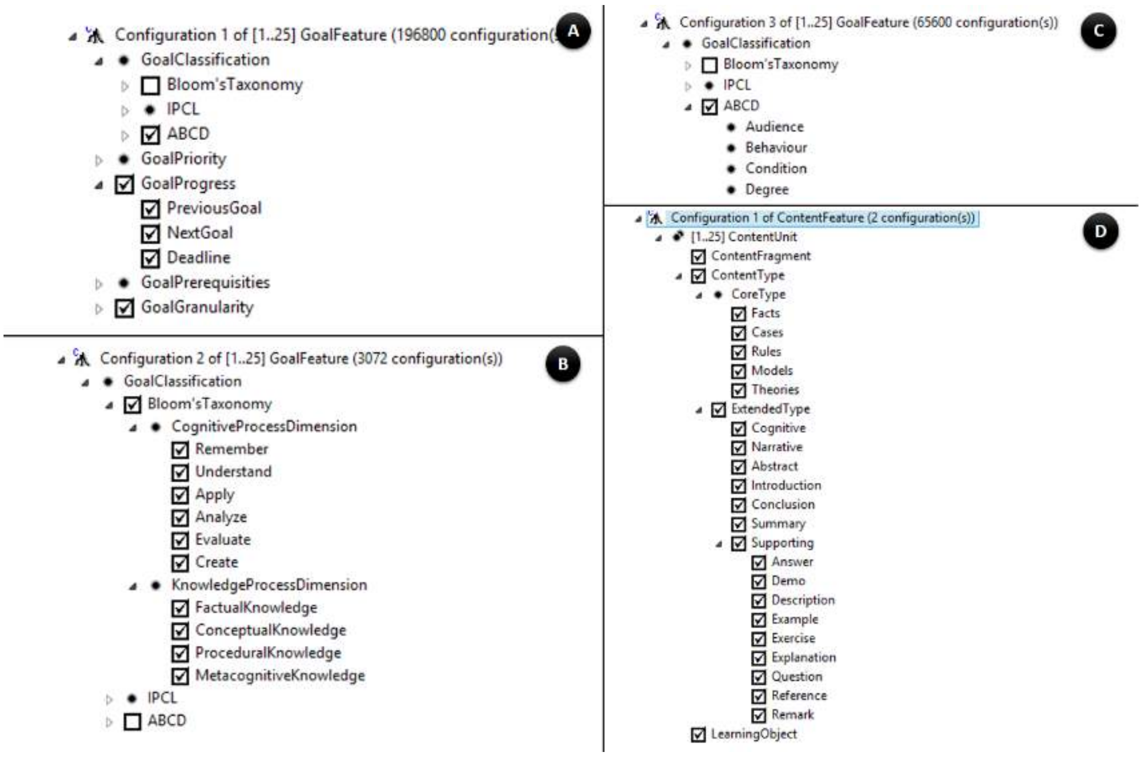
Aspect/ID	ID Specification 1	ID Specification 2	ID Specification 3	ID Specification 4
Base ID	IPCL	IPCL+ *ProcessPattern (pasi)* + *ContentPattern (fcrmt)*	IPCL+ *ProcessPattern (pasi)* + *ContentPattern (fcrmt)* + Merrill's First Principles of Instruction +Bloom's Revised Taxonomy	IPCL + Gagne's Nine levels of learning + ABCD Technique for Goals
Goals	*R*ead, w*R*ite and basic a*R*ithmetic (3Rs)	*R*ead, w*R*ite and basic a*R*ithmetic (3Rs)	Organize goals using Bloom's revised taxonomy	Organize goals using ABCD technique
Process	Eclectic method for teaching 3*R*s	Organize instructional process as a set of *plays, acts, scenes, instructions* with *instructions* containing actual activities and tasks	Mainly driven by *ProcessPattern (pasi)* but the activities should be based on Merrill's first principles of instruction.	Organize process using Gagne's nine levels of learning
Content	Content as per IPCL based primer	Content is organized as *facts, cases, rules, models* and *theories*	Content is organized as *facts, cases, rules, models* and *theories* and mapped to Merrill's first principles	Content is organized as resources

Figure 6.4 Custom instructional design specification requirements



```
InstructionalDesignSpecification
  [1..25] GoalFeature
    GoalClassification
      Bloom'sTaxonomy
        CognitiveProcessDimension
          Remember
          Understand
          Apply
          Analyze
          Evaluate
          Create
        KnowledgeProcessDimension
          FactualKnowledge
          ConceptualKnowledge
          ProceduralKnowledge
          MetacognitiveKnowledge
      IPCL
        Reading
        wRiting
        aRithmetic
      ABCD
        Audience
        Behaviour
        Condition
        Degree
    GoalPriority

        High
        Medium
        Low

ProcessFeature
  InstructionalDesignModel

      MerrillModel
        FirstPrinciples
          Activation
          Demonstration
          Application
          Integration
          Task-Oriented
        ---> ProcessFeature
      GagneModel
        GainAttention
        InformObjectives
        StimulatePriorLearning
        PresentContent
        ProvideGuidance
        ElicitPerformance
        ProvideFeedback
        AssessPerformance
        SupportInternalization
        ---> ProcessFeature
    GenericActivity
      LearningActivity
      InterpretedActivity
      MonitredActivity
      SupportActivity
  [1..25] Lesson
    [1..25] GenericPlay
      [1..25] GenericAct

[1..25] GenericAct
  MotivatingAct
  SyllableBankAct
  NewPhonemesAct
  ComparingAct
  FormingWordsAct
  FormingSoundsAct
  LearningRulesAct
  WritingInstructionsAct
  ExerciseAct
  SummaryAct
  [1..25] GenericScene
    FamiliarWordsScene
    FormingWordsScene
    InspectingSyllableBankScene
    SimialrSyllablesScene
    SyllableBannerScene
    SyllableFormationRulesScene
    GenericInstruction
      ---> InstructionalDesignSpecification

ContentFeature
  [1..25] ContentUnit
    ContentFragment
    ContentType
      CoreType
      ExtendedType
    LearningObject
    ---> RawData
UserInterface
  Language
```

Figure 6.5 A feature model for instructional design specification



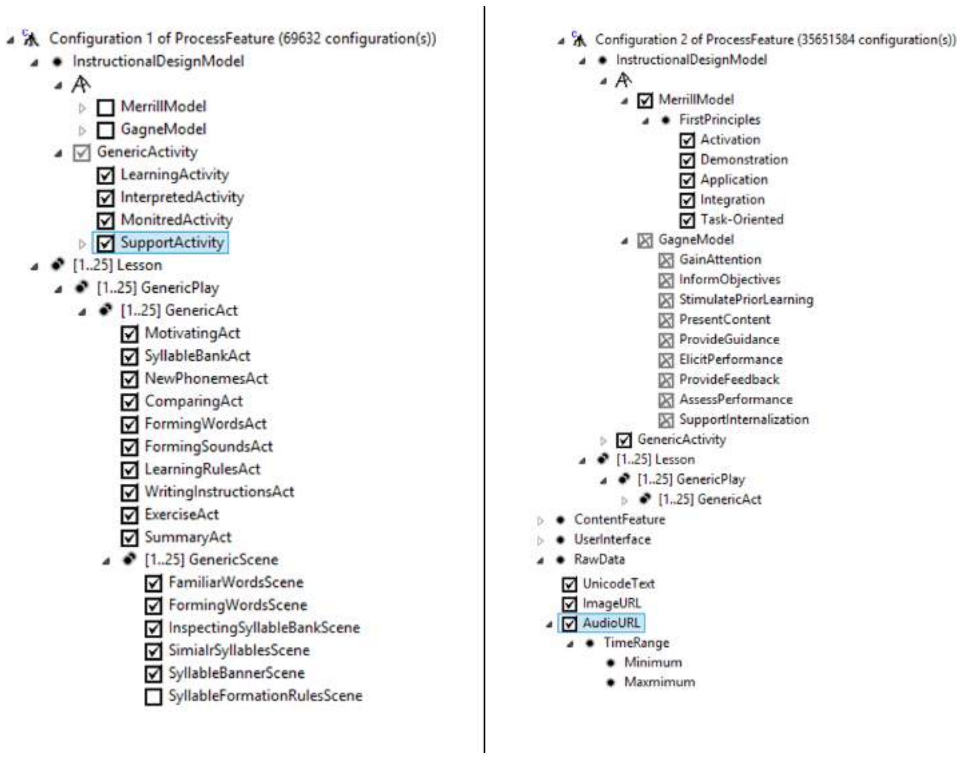
```
A
Configuration 1 of [1..25] GoalFeature (196800 configuration(
  ● GoalClassification
    ☐ Bloom'sTaxonomy
    ● IPCL
    ☑ ABCD
  ● GoalPriority
  ☑ GoalProgress
    ☑ PreviousGoal
    ☑ NextGoal
    ☑ Deadline
  ● GoalPrerequisities
  ☑ GoalGranularity

B
Configuration 2 of [1..25] GoalFeature (3072 configuration(s))
  ● GoalClassification
    ☑ Bloom'sTaxonomy
      ● CognitiveProcessDimension
        ☑ Remember
        ☑ Understand
        ☑ Apply
        ☑ Analyze
        ☑ Evaluate
        ☑ Create
      ● KnowledgeProcessDimension
        ☑ FactualKnowledge
        ☑ ConceptualKnowledge
        ☑ ProceduralKnowledge
        ☑ MetacognitiveKnowledge
    ● IPCL
    ☐ ABCD

C
Configuration 3 of [1..25] GoalFeature (65600 configuration(s))
  ● GoalClassification
    ☐ Bloom'sTaxonomy
    ● IPCL
    ☑ ABCD
      ● Audience
      ● Behaviour
      ● Condition
      ● Degree

D
Configuration 1 of ContentFeature (2 configuration(s))
  ● [1..25] ContentUnit
    ☑ ContentFragment
    ☑ ContentType
      ● CoreType
        ☑ Facts
        ☑ Cases
        ☑ Rules
        ☑ Models
        ☑ Theories
      ☑ ExtendedType
        ☑ Cognitive
        ☑ Narrative
        ☑ Abstract
        ☑ Introduction
        ☑ Conclusion
        ☑ Summary
        ☑ Supporting
          ☑ Answer
          ☑ Demo
          ☑ Description
          ☑ Example
          ☑ Exercise
          ☑ Explanation
          ☑ Question
          ☑ Reference
          ☑ Remark
    ☑ LearningObject
```

Figure 6.6 Feature configurations for varied goals and content

taxonomy for modeling goals, maps Process and Content patterns to Merrill's principles of instruction. ID Specification 4 does not use the patterns proposed in this thesis but uses ABCD technique, Gagne's nine events of instruction for goals and process, and core resources for content. Each of these instructional design specifications can be used to create instructional design editors specific to the family such that instructional designers can create several concrete instructional designs by changing the variabilities in terms of goals, process and content.

In Figure §6.5, we show a feature model encompassing key features of this product line. This is essentially based on the ontologies for different aspects of instructional design comprising of mandatory features like *GoalClassification*, *IPCL*, several optional features, selective features and so on. *InstructionaDesignModel* has three choices *MerillModel*, *GagneModel* and *GenericActivty*. Once a teacher chooses *MerrillModel*, then *FirstPrinciples* are mandated by default. In case of *GoalPriority*, only one priority out of *High*, *Medium*, *Low* can be chosen. The feature model also mandates that atleast one *Play*, *Act*, *Scene* and *Instruction* are mandatory and can go upto a maximum of 25. A detailed listing of the instructional deign feature model is provided in Appendix §B.1.

Figure §6.6 shows different feature configurations for the different product family members. A *GoalFeature* was configured in three ways as in Figure §6.6[A,B,C]. Similarly, *ContentFeature* was configured based on specific instructional design requirements. The *ProcessFeature* was configured in two ways one using the *ProcessPattern* and the other using



```
Configuration 1 of ProcessFeature (69632 configuration(s))
  ● InstructionalDesignModel
      ☐ MerrillModel
      ☐ GagneModel
    ☑ GenericActivity
      ☑ LearningActivity
      ☑ InterpretedActivity
      ☑ MonitredActivity
      ☑ SupportActivity
  ● [1..25] Lesson
    ● [1..25] GenericPlay
      ● [1..25] GenericAct
          ☑ MotivatingAct
          ☑ SyllableBankAct
          ☑ NewPhonemesAct
          ☑ ComparingAct
          ☑ FormingWordsAct
          ☑ FormingSoundsAct
          ☑ LearningRulesAct
          ☑ WritingInstructionsAct
          ☑ ExerciseAct
          ☑ SummaryAct
        ● [1..25] GenericScene
            ☑ FamiliarWordsScene
            ☑ FormingWordsScene
            ☑ InspectingSyllableBankScene
            ☑ SimialrSyllablesScene
            ☑ SyllableBannerScene
            ☐ SyllableFormationRulesScene
```

```
Configuration 2 of ProcessFeature (35651584 configuration(s))
  ● InstructionalDesignModel
      ☑ MerrillModel
        ● FirstPrinciples
            ☑ Activation
            ☑ Demonstration
            ☑ Application
            ☑ Integration
            ☑ Task-Oriented
      ☒ GagneModel
          ☒ GainAttention
          ☒ InformObjectives
          ☒ StimulatePriorLearning
          ☒ PresentContent
          ☒ ProvideGuidance
          ☒ ElicitPerformance
          ☒ ProvideFeedback
          ☒ AssessPerformance
          ☒ SupportInternalization
    ☑ GenericActivity
  ● [1..25] Lesson
    ● [1..25] GenericPlay
      ● [1..25] GenericAct
● ContentFeature
● UserInterface
● RawData
  ☑ UnicodeText
  ☑ ImageURL
  ☑ AudioURL
    ● TimeRange
      ● Minimum
      ● Maxmimum
```

Figure 6.7 Feature configurations for varied instructional processes

MerrillModel as shown in Figure §6.7. These possible variations could run into thousands but valid configurations provide different instructional design models with varied goals, processes, content and so on. These custom instructional design specifications are used to create custom authoring tools (editors) for creating instances of the specific instructional design. A detailed listing of the instructional deign feature model is provided in Appendix §B.1. We first present the reference architecture in the next section followed by a software product line for eLearning Systems in §6.4.

6.3.4 A Reference Architecture

The next step is to take these feature configurations and generate custom instructional design authoring tools (editors) based on specific requirements. One of the key architecture requirements for this product line is that the product family members or web applications should run on limited technical capabilities considering their deployment environment. Internet connectivity cannot be presumed as most of the systems would be in rural villages of India and most of the teachers are either non-technical people or low-computer proficiency teachers. With this constraints, Figure §6.8 shows a reference architecture for the product lines in this thesis. This reference architecture can be implemented in multiple ways but we discuss our current implementation here. An instructional designer/teacher creates the patterns as document/text and uses that to create an ontology through an ontology editor. We

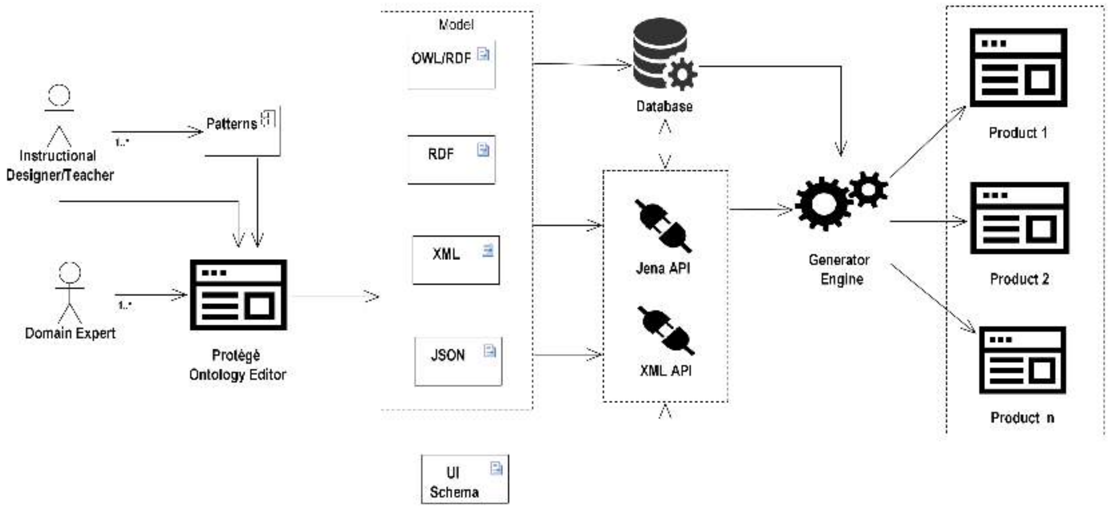


Figure 6.8 A reference architecture for product lines in this thesis

used protégé for creating ontologies in this thesis. It can also be the case that an existing ontology be taken. For example, IMS-LD ontology is available in public domain [44] or a comprehensive ontology is available for instructional design teaching learning theories [155]. This ontology can be stored as OWL or RDF file. In addition, we also store ontology as an OWL/XML schema as the current version of platform uses XML for storing knowledge. We also use JSON to store some parts of the OWL or XML for further processing by tools. This data is part of Model in Model-View-Controller pattern. We are currently using Jena API for processing OWL/RDF files and generating a basic web application based on the data in the OWL file. This web application uses the UI schema as input for the generator. We are currently generating two families of applications (product family members) using this reference architecture. The first set of members are ID editors for selected OWL/XML schema and the generator engine parses the OWL/XML and creates a web application that can be used to create specific instances of instructional design. The other set of applications are *i*Primers or eLearning Systems for adult literacy and the generator creates animations based on the specific instructional design described using OWL/XML. This product line is explained in the next section. The current implementation of reference architecture is primary based on files, does not use server but stores all resources in a single package and is implemented mostly using *Javascript*, *jquery*, *Nodejs*, *Jena API*, XML parser, custom animations among others.

The concrete process of creating *ID Editors*[1] is shown in Figure §6.9. The core input for this process comes in the form of *ID Specifications*, which are created by domain experts. These *ID Specifications* consist of different aspects of instructional design like *goals*, *process*, *content* based on patterns described in Chapter §3 and ontologies in Chapter §4. The *ID*

[1] We will use the term ID Editors to mean Instructional Design Editors

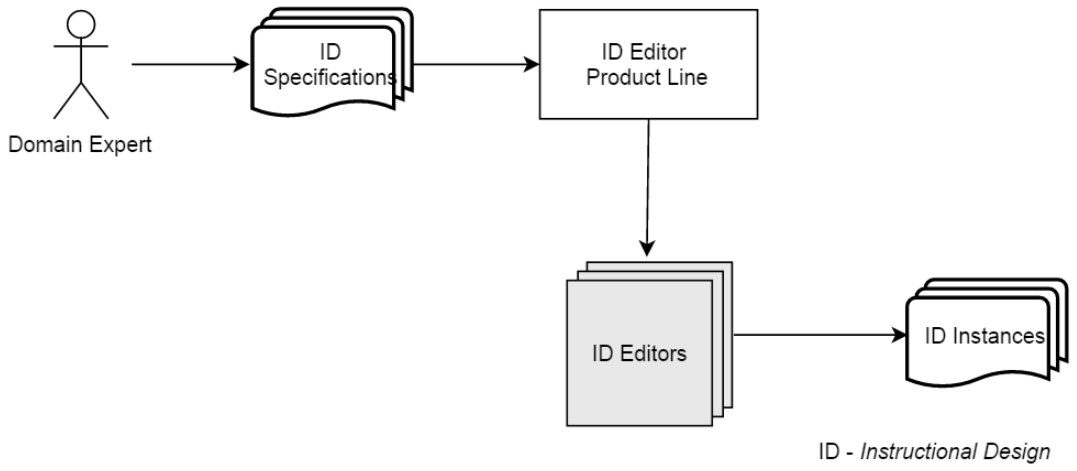


Figure 6.9 Flow of Instructional Design Product Line

Editor Product Line is an engine written in *JavaScript*[2] that parses the *ID Specification* stored in the form of RDF/XML and generates *ID Editors*. This *ID Editor* is a simple form editor consisting of selected aspects of instructional design that are applicable for all instructional design instances based on this concrete specification. This is unlike the current approach of manually creating instructional design editors for every instructional design specification as discussed in Section §6.2. We have implemented this using multiple technologies like *Java*[3], *Python*[4]. However, the need has been to create multiple instances of instructional designs which form the basis for several *i*Primers. We discuss the product line for creating a family of eLearning Systems in the next section.

6.4 A Software Product Line for a family of eLearning Systems

The primary goal of this product line is to create a family of eLearning Systems based on specific instructional designs tailored to the needs of teaching functional literacy for all Indian languages. The 32 State Resource Centers across all states in India are responsible for producing the following primers based on IPCL (first three are mandatory and the rest depend on specific needs) under the aegis of NLMA:

- Basic literacy primer [22+]
- Post literacy primer [22+]
- Life long literacy primers [22+]

[2]https://github.com/enthusiastic2learn/ID-Editor
[3]https://github.com/enthusiastic2learn/IDPlatform
[4]An implementation named *Semantic Web Forms* was done by undergrad students as part of Software Engineering course at IIIT-Sri City, India and is available at https://github.com/chrizandr/semantic_web/

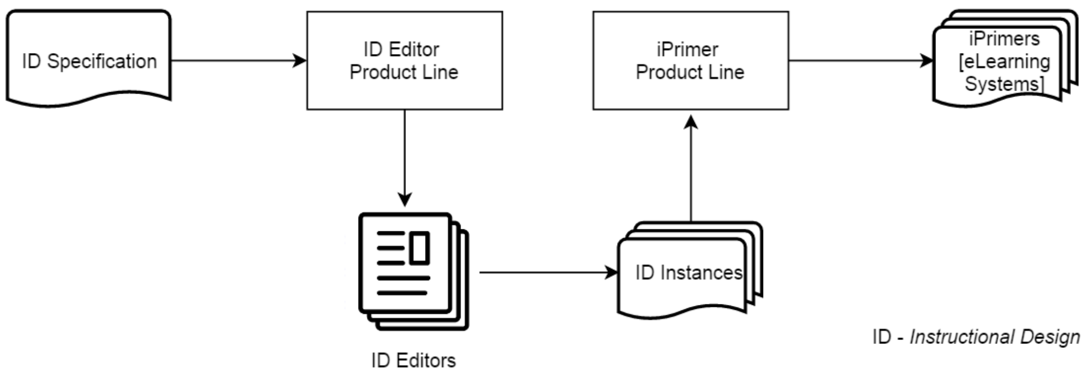


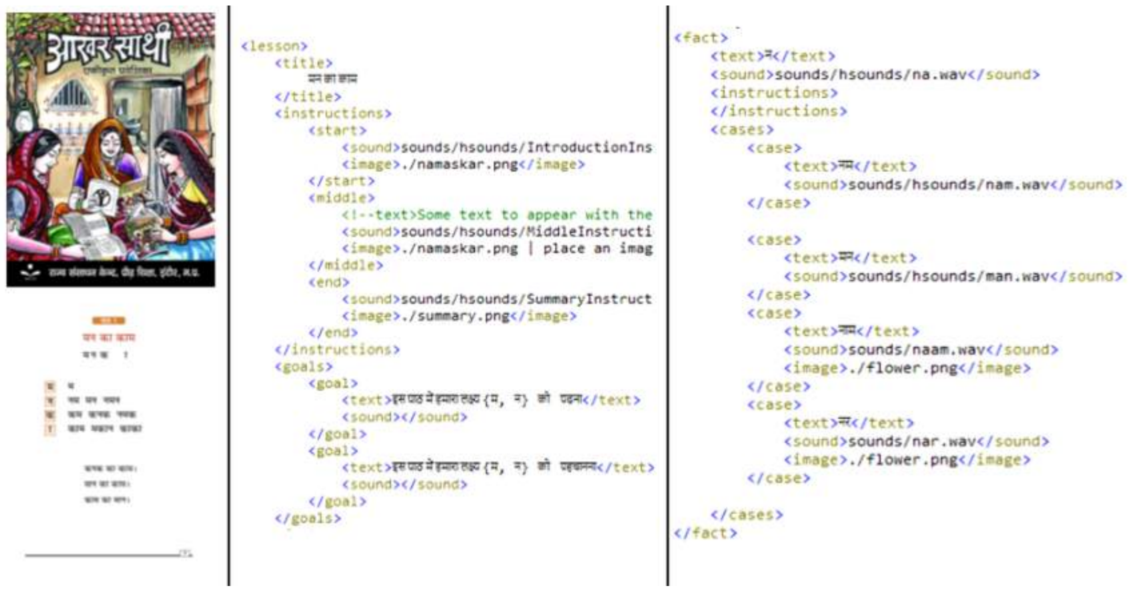
Figure 6.10 Flow of *i*Primer Product Line

- Primers for teaching skills like or tailoring, vocational skills (Jan Shikshan Sansthan (JSS), Life Enrichment Education and so on along with literacy [*n*+, where *n* is in the order of hundreds]
- Exclusive primers were specifically made for legal literacy, election literacy, agriculture literacy, environment literacy among many others [*n*+, where *n* is in the order of tens]

An important commonality among these primers is that they teach 3Rs but using varied instructional processes and different themes. Each of these primers are generally available in 22 languages. It is estimated that currently there are atleast 1000 primers available with SRCs in print format. Eventhough the primer is fixed till the next version is developed, officers at different levels (mandal, village, school and teachers) attempt to customize the process, content and adapt it to the local context. For example, a simple way could be to ask the name of learners and find if they know how it looks like? and what are the syllables in it? However, this is not supported in traditional print form, but is a great source of variability for *i*Primers of our product line. How to support immigrants at a given place who want to learn a local language but using their mother tongue as medium of instruction? This leads to another set of variations in the primers with medium of instruction being different for 22 Indian languages?

Technically, we are interested in *i*Primers that are based on field-tested eLearning Systems [1]. These applications are based on puppet theater model, where syllables are shown as falling puppets, joining together to form words and so on. The *i*Primers, product members of this family should essentially follow instructional processes, use locally relevant content and present a multimedia application with animations for the learners.

Figure §6.10 shows the flow of *iPrimer Product Line*. This product line essentially parses the instructional design instances to generate *i*Primers. In the case of adult literacy in India, the *iPrimer Product Line* is based on a single *ID Specification* driven by IPCL. Appendix §B.2 provides a detailed listing of the instructional design specification used for adult literacy.



```
<lesson>
    <title>
        मन का काम
    </title>
    <instructions>
        <start>
            <sound>sounds/hsounds/IntroductionIns
            <image>./namaskar.png</image>
        </start>
        <middle>
            <!--text>Some text to appear with the
            <sound>sounds/hsounds/MiddleInstructi
            <image>./namaskar.png | place an imag
        </middle>
        <end>
            <sound>sounds/hsounds/SummaryInstruct
            <image>./summary.png</image>
        </end>
    </instructions>
    <goals>
        <goal>
            <text>इस पाठ में हमारा लक्ष्य {म, न} को पढ़ना</text>
            <sound></sound>
        </goal>
        <goal>
            <text>इस पाठ में हमारा लक्ष्य {म, न} को पहचानना</text>
            <sound></sound>
        </goal>
    </goals>
```

```
<fact>
    <text>न</text>
    <sound>sounds/hsounds/na.wav</sound>
    <instructions>
    </instructions>
    <cases>
        <case>
            <text>नम</text>
            <sound>sounds/hsounds/nam.wav</sound>
        </case>

        <case>
            <text>मन</text>
            <sound>sounds/hsounds/man.wav</sound>
        </case>
        <case>
            <text>नाम</text>
            <sound>sounds/naam.wav</sound>
            <image>./flower.png</image>
        </case>
        <case>
            <text>नर</text>
            <sound>sounds/nar.wav</sound>
            <image>./flower.png</image>
        </case>

    </cases>
</fact>
```

Figure 6.11 Primer and custom instructional design instance [XML from OWL] for *Hindi* language

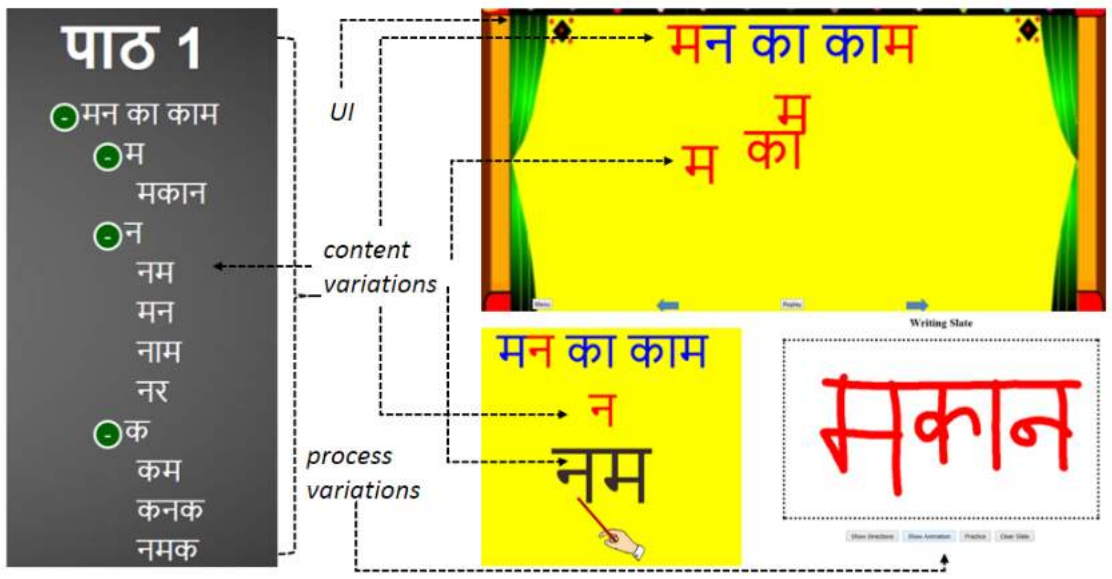
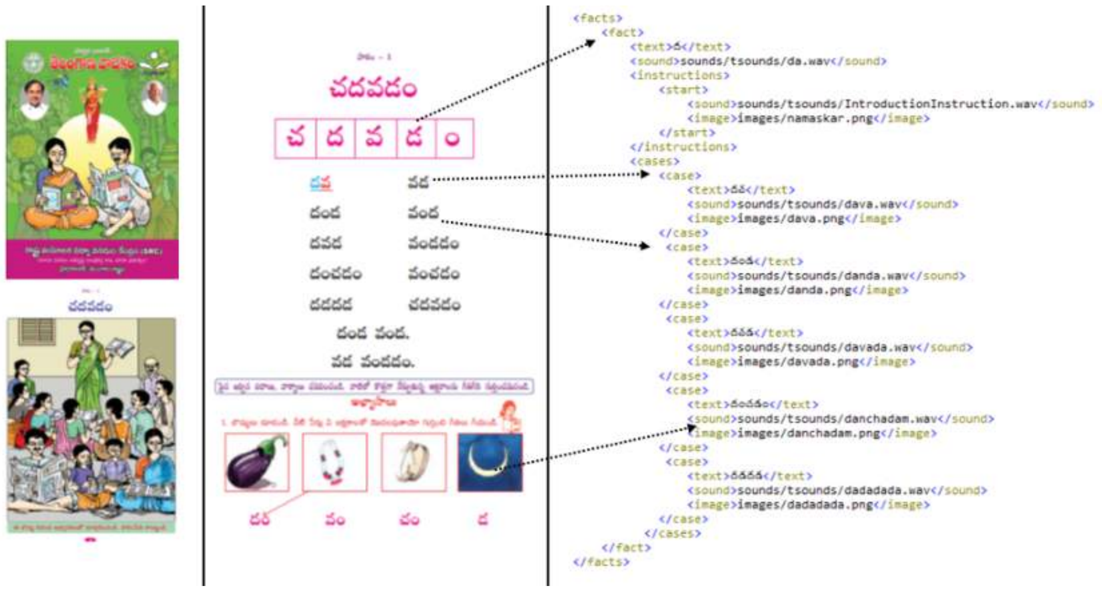


Figure 6.12 *i*Primer for *Hindi* language - generated from instructional design instance



```xml
<facts>
    <fact>
        <text>ద</text>
        <sound>sounds/tsounds/da.wav</sound>
        <instructions>
            <start>
                <sound>sounds/tsounds/IntroductionInstruction.wav</sound>
                <image>images/namaskar.png</image>
            </start>
        </instructions>
        <cases>
            <case>
                <text>దవ</text>
                <sound>sounds/tsounds/dava.wav</sound>
                <image>images/dava.png</image>
            </case>
             <case>
                <text>దండ</text>
                <sound>sounds/tsounds/danda.wav</sound>
                <image>images/danda.png</image>
            </case>
             <case>
                <text>దవడ</text>
                <sound>sounds/tsounds/davada.wav</sound>
                <image>images/davada.png</image>
            </case>
             <case>
                <text>దంచడం</text>
                <sound>sounds/tsounds/danchadam.wav</sound>
                <image>images/danchadam.png</image>
            </case>
             <case>
                <text>దదదద</text>
                <sound>sounds/tsounds/dadadada.wav</sound>
                <image>images/dadadada.png</image>
            </case>
          </cases>
    </fact>
</facts>
```

Figure 6.13 Primer and custom instructional design instance [XML from OWL] for *Telugu* language

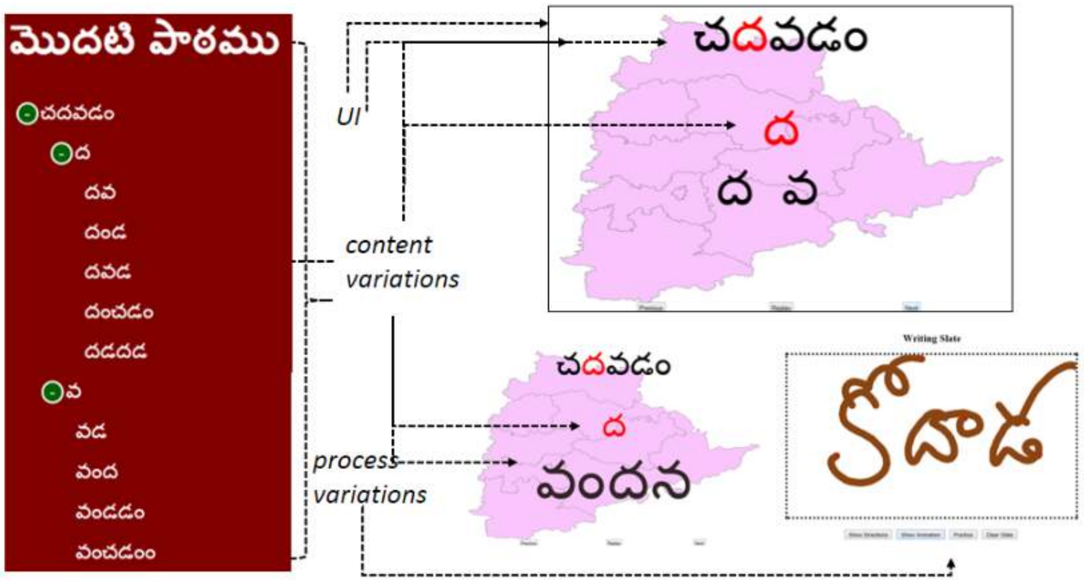


Figure 6.14 *i*Primer for *Telugu* language - generated from instructional design instance

This *ID Editor* is used to create several instances of the instructional design specification for varied processes, content and visual and audio elements. These instances are parsed by *iPrimer Product Line* to eventually create *i*Primers for multiple languages and primers. Every instructional design instance leads to a varied *i*Primer of the product line. The *iPrimer Product Line* has to be customized if the base *ID Specification* is changed to other than pre-defined *ID Specification* as the RDF/XML parser has to be re-written and it would take about a person-week to re-write the parser for ID specifications beyond adult literacy. Section §6.5 presents the results of cost savings of *ID Editor Product Line* and *iPrimer Product Line*. We used the *iPrimer Product Line* to generate several *i*Primers and discuss *i*Primers for *Hindi* and *Telugu* Language in this section.

Figure §6.11 shows a fragment of primer of *Hindi* language. This primer has around 180 pages with 24 lessons and each lesson teaching 3Rs. This primer is available in both print as well as digitized format (pdf). This digitized form is used as an input to a custom instructional design editor for creating a custom instructional design instance as shown on the right hand side. This OWL/XML file contains all the information related to a specific instructional design and serves as the base for creating variations based on this instructional design. A detailed listing of this instructional design instance is presented in Appendix §B.3. Figure §6.12 shows how some variations can be created using the *iPrimer Product Line*. The *iPrimer Product Line* primarily reads the OWL/XML file for instructional process consisting of activities, their order, and content that has to be used in the process and generates animations accordingly. Everything that is shown in Figure §6.12 can be varied as per the feature model configurations discussed earlier in this chapter. This allows to rapidly customize the *i*Primers and create new ones by changing processes and content. Figure §6.13 shows how an *i*Primer has been generated for *Telugu* language based on a specific instructional design instance in Appendix §B.4. Here, the processes, content, user interface that are relevant for that specific instructional design have been generated. Figure §6.14 shows some variations that are possible for *Telugu* language. The core idea here is to be able to generate as many *i*Primers as possible with minimum effort by applying the idea of software product lines. We have observed that this product line can be configured easily to create *i*Primers but one major obstacle is with respect to sound, which has to be created manually in the current version. However, we are thinking of using teachers'/learners' voice to record instructions and content at a personalized level as part of our future work.

6.5 Experimental Results

In this section, we discuss the experimental results of using our approach and technologies for (semi-)automatic creation of *ID Editors* and *i*Primers. We wish to reiterate that the primary goal of this thesis is to facilitate customization of educational technologies for scale and variety and demonstrate it in the context of adult literacy in India. One of the core claims of software product lines is that product lines facilitate creation of product variants at reduced cost [2]. The literature has a number of measures to calculate the cost and return on investment on software product lines [299]. In this thesis, we consider the commonly used model of Structured Intuitive Model for Product Line Economics (SIMPLE) to measure the effectiveness of product lines [2]. The SIMPLE model describes seven scenarios for creation of SPLs that may typically occur in an organization. The generic scenario is concerned with creation of SPLs and stand alone products from existing products and resources. Specifically, the SPLs in this thesis fall into the category of *Scenario 2*, where the organization plans to develop a set of products as a product line based on common core assets. The SIMPLE model consists of four cost components to calculate the total cost of SPLs [299].

- C_{org} - The cost to an organization for adopting product line approach instead of single system development. In this thesis, the product lines are developed by researchers and hence no direct organization costs. However, in the long run, the organization that develops software for all *i*Primers should incur costs for transition to product line approach.
- C_{cab} - The cost to develop core assets that are reusable across the product line. This cost includes the patterns discovered, ontologies created along with traditional SPL activities introduced in Section §5.4.
- C_{unique} - The cost to develop unique features of the product beyond the product line. This generally involves manual effort to customize the generated product from the product line.
- C_{reuse} - The cost to reuse core assets, adapt them for the needs of developing new products in the product line.

The costs of developing a software product line for *n* distinct products can be calculated as follows [300][301]:

Cost of building a product line

$$C_{SPL} = C_{org}() + C_{cab}() + \sum_{i=1}^{n}(C_{unique}(product_i) + C_{reuse}(product_i))$$

Cost of building *n* stand-alone products

$$C_{stand-alone} = \sum_{i=1}^{n} C_{product}(product_i)$$

where $C_{product}$ is the cost of developing an individual product.

The savings of software product lines can be estimated as:

Savings of product lines = $C_{stand-alone}$ - C_{SPL}

Tata Consultancy Services, an Indian software services organization has been involved with development of eLearning Systems for adult literacy in India for more than 15 years [1]. We use data from our earlier experience of developing eLearning Systems [11] and TCS' statistics on developing eLearning Systems for 9 Indian Languages[1] as the initial base for calculating cost savings of *iPrimer Product Line*. The effort for creating an eLearning System was around 5 to 6 person years and in our earlier work, we have applied software reuse techniques and reduced the effort for creating eLearning Systems to 5 to 6 person months [11]. Each existing *i*Primer approximately consists of 20,000 visual elements; 2,500 sound elements with 500 words based on a physical primer for a language. These elements are organized in the form of approximately 24 lessons constituting an eLearning System for teaching 3Rs. The *iPrimer Product Line* essentially generates these 24 lessons as shown in Figure §6.12 and Figure §6.14 for *Hindi* and *Telugu* languages with manual inputs for words and sounds. Based on this existing data, we evaluate the cost savings of *iPrimer Product Line* as follows:

Here, we present the costs for building 9 products i.e., *i*Primers:

Cost of building a product line

C_{SPL} = 6 person-months + 12 person-months + 9 * (2 person-weeks + 1 person-week)

C_{SPL} = 25 person-months

Cost of building *n* stand-alone products

$C_{stand-alone}$ = 9 * 6 person-months

$C_{stand-alone}$ = 54 person-months

where $C_{product}$, the cost of developing an individual product is 6 person-months.

The savings of software product lines can be estimated as:

Savings of product lines = 54 person-months - 25 person-months i.e., 29 person-months

Table §6.1 shows the individual cost components for *iPrimer Product Line* and Figure §6.15 shows the cost of creating *i*Primers with and without our approach. The horizontal axis shows the number of *i*Primers and the vertical axis shows the number of person-months required to develop the *i*Primers. The graph shows that the break-even for the initial investment in terms of core asset base is for 3 or 4 *i*Primers after which as the number of *i*Primers to be developed increases, the cost required for developing them in a stand-alone fashion increases rapidly whereas it is steady in the case of SPL. The *iPrimer Product Line* hosted at `http://rice..iiit.ac.in` was used by a low-computer proficiency teacher at State Resource Center, Telangana, India to create 10 lessons of *i*Primer based a newly released physical primer. This *i*Primer was packaged as an android app using *Apache Cordova*[5] and

[5]https://cordova.apache.org

Table 6.1 Cost components of *iPrimer Product Line*

Cost Component	**Cost (Person-months)**	**Description**
$C_{org}()$	6 person-months	In case of *iPrimer Product Line*, we do not have a single organization but we have developed the product line as part of this thesis essentially meaning no direct cost for an organization to adopt the product line approach. However, based on our experience and collaboration with TCS, we consider a time of 6 person-months as an organizational cost.
$C_{cab}()$	12 person-months	Core assets in the case of *iPrimer Product Line* are ontologies of instructional design that were developed based on patterns, which are represented in RDF/OWL format, *JavaScript* files, a parser that reads configuration files as an XML and generates instances, UI components like animation generator and so on. We have spent around 12 person-months to create this core asset base which is part of the reusable infrastructure of this product line.
$C_{unique}()$	2 person-weeks	The unique parts of the *i*Primers are primarily *process* steps and *content* in terms of words, syllables, which have to be extracted from a soft copy of the primer or to be entered manually. In addition, the software has to be adapted to handle special syllables or words that are specific to the particular language. The cost to create sound files for new words is a major source of manual effort as text-to-speech tools for Indian Languages are not yet acceptable for purposes of literacy teaching.
$C_{reuse}()$	1 person-week	The cost to modify existing resources i.e., instructional design instance with data or raw XML aspects for user interface elements pertaining to a specific *i*Primer.

is hosted online at Google Play Store Store[6]. The low-computer proficiency teacher was able to create these lessons in about a day but without audio and the instructional design instance created using the *iPrimer Product Line* is available on *Github* [7]. The primer was also listed on Government of Telangana websites[8]. In addition, a workshop was conducted in November 2016 for 24 preraks of adult literacy on the use of *iPrimer Product Line*. Figure §6.16 shows a glimpse of the session where teachers used *iPrimer Product Line* to create partial lessons based on dynamic words given by the audience.

[6]https://play.google.com/store/apps/details?id=iiit.rice.al.telugu&hl=en

[7]https://github.com/enthusiastic2learn/ThesisAppendices/blob/master/Telangana%20Vachakam-%20Standard%20iPrimer%2010%20Lessons.xml

[8]`http://tslma.nic.in/` and State Resource Center, Government of Telangana at `http://srctelangana.com/`

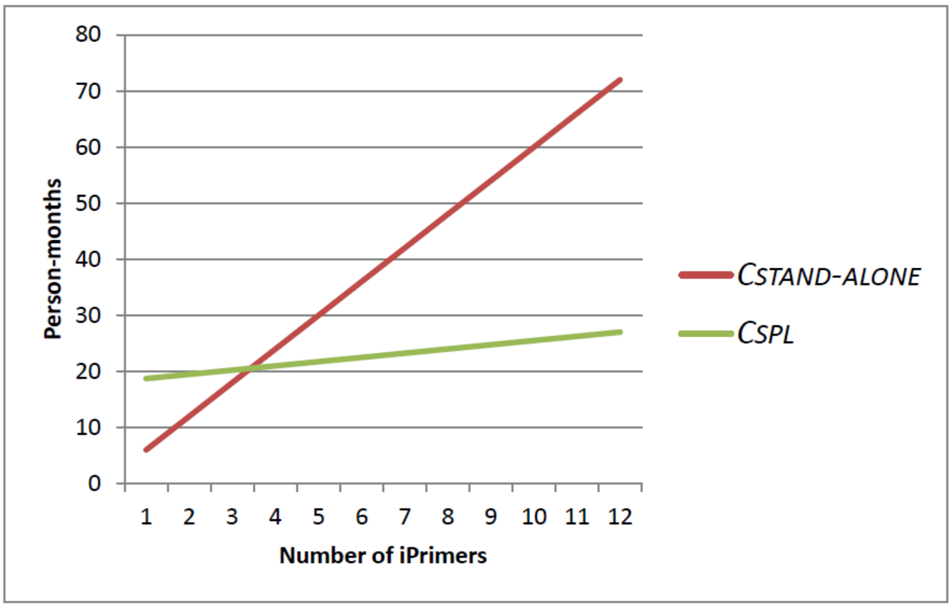


Figure 6.15 Cost Savings of *iPrimer Product Line*

We have also populated the cost of developing *ID Editors* with and without SPL in Figure §6.17. Here, the cost of manual effort for customizing the generated *ID Editor* is one person-month instead of three person-weeks as in *iPrimer Product Line*.

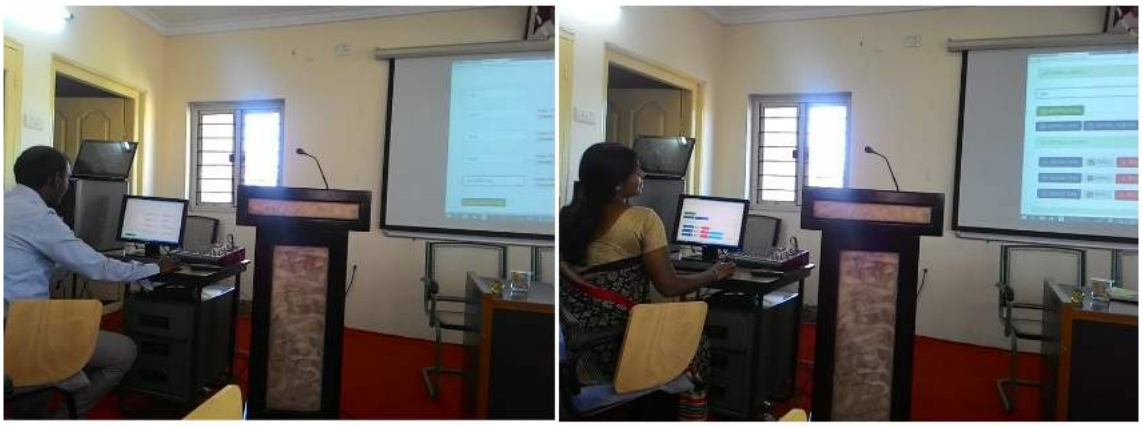


Figure 6.16 Teachers using *iPrimer Product Line* to create iPrimers

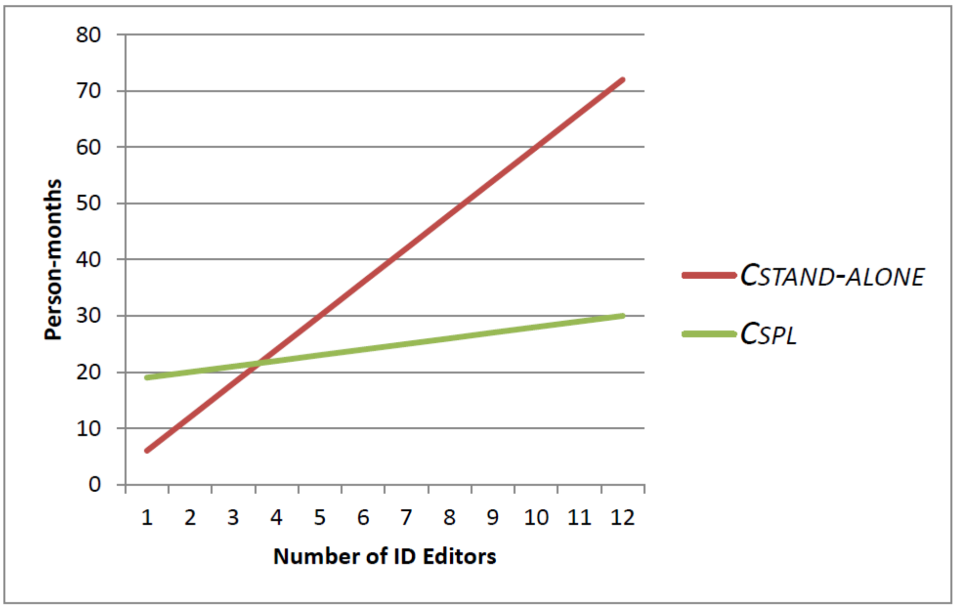


Figure 6.17 Cost Savings of *ID Editor Product Line*

6.6 Challenges of Software Product Lines for Educational Technologies

Despite the emergence of SPL four decades ago in software engineering, there are minimal cases of SPL applied to the domain of education. We see the following as the major challenges for hindrance of SPL application in this domain [63], .

- *Societal Context Vs Business Context* - How does the notion of SPL change in a societal context (like adult literacy in India) rather than a business context (an organization like Boeing or Bosch Group)? What is the motivation and how can a business case be established?
- *Dealing Non-Technical Stakeholders* - How can we deal with non-technical and diversified stakeholders during design and development of SPL?
- *Cross Organizational SPL* - How to design an SPL that spans across different organizations from different domains?
- *Globally Distributed Software Product Lines* - How to develop SPL in a globally distributed environment?
- *Process Diversity & Version Management* - How can we map the diversified processes during the development and maintenance of SPL?
- *Lean Software Product Lines* - How to make SPL as a light-weight approach?

CHAPTER 7

Conclusions & Next Steps

7.1 Conclusions

In this thesis, we aimed at creating an approach for design of educational technologies to address *scale* and *variety* in education. Specifically, we addressed the need to support creation of eLearning Systems for flexible instructional designs and multiple Indian Languages in the context of adult literacy in India. The fundamental basis of our approach is to design educational technologies based on instructional design for quality of instruction. To this end, we relied on commonly accepted teaching/learning methodologies and instructional material based on these methodologies.

Based on principles from computing, we explored the possibility of extending and applying software engineering approaches like *patterns*, *ontologies* and *software product lines* to address the challenges of scale and variety. The first step in the proposed approach is to systematically model different aspects of instructional design using *patterns*. We have discovered two patterns: *ProcessPattern*, that organizes instructional process in the form of *plays*, *acts*, *scenes* and *instructions* and *ContentPattern* that organizes instructional material as *facts*, *cases*, *rules*, *models* and *theories*. To adhere to quality of instruction, the *ProcessPattern* and *ContentPattern* are mapped to Merrill's First Principles of Instruction and Bloom's taxonomy respectively. Based on the patterns that encapsulate practices in instructional design, we proposed an ontology based modeling framework with a set of ontologies (context, *goals*, *process*, *content*, evaluation, environment) to concretely represent the patterns and facilitate automation. This ontology framework is based on standard ontologies for learning design (process) and instructional material in the literature. After modeling the domain using patterns and representing them using ontologies, we presented a pattern-oriented software product line approach for modeling families of instructional designs and families of eLearning Systems for adult literacy. To demonstrate our approach, we have created a prototype platform for modeling instructional design variants and a prototype platform for semi-automatically generating eLearning Systems for adult literacy.

In the next section, we briefly outline the next steps as well as some future directions of the thesis.

7.2 Future Research Directions - Beyond this thesis

We see the following potential extensions of this thesis:

7.2.1 Pattern Modeling Languages and Pattern Composition

While using patterns facilitates reuse and is based on instructional design, we see the following future research directions:

- Modeling notations for representing patterns and supporting techniques and tools for handling patterns throughout life cycle.
- Formal approaches for composing patterns either within the domain or software or between domain and software, validating assembly of patterns and their integration.

7.2.2 Ontology and Pattern Modeling Life Cycle

We have extended and created several ontologies in our ontology framework proposed in Chapter §4. However, by definition, every domain can have several perspectives and hence several ontologies. In our ontology framework, we have introduced the notion of meta-ontologies for representing high level aspects of instructional design like process and content, which are then customized for specific cases of Merill's First Principles of Instruction and Bloom's taxonomy. With this context, the following are some potential directions for further research:

- Creating ontologies for different aspects of instructional design catering to the diversified needs of a wide variety of subjects at different levels of education to be delivered in multiple languages and on a wide variety of different platforms requires massive collaborative effort from multiple disciplines.
- Creating collaborative, distributed and agile environments for domain and subject matter experts to create, share and disseminate their patterns and ontologies is a critical step towards design of educational technologies for scale and variety.
- In addition, it becomes increasingly difficult to manage the magnanimous number of patterns and ontologies motivating the need for life cycle management tools.

7.2.3 Software Product Lines for Personalized Learning

Even though the software product line approach outlined in Chapter § 5 is a natural way to address scale and variety of educational technologies, personalized learning is a grand challenge for computing requiring further research from software product lines community.

- Using current feature modeling notations, features can only be selected for product configuration but the need in educational technologies is to have features that have knowledge associated with them for different aspects of instructional design like goals, process steps and content, which is not possible with current notations. For example, expressing goals using Bloom's taxonomy or ABCD technique could be a feature but specifying an exact learning goal requires more than just features.
- Design of light-weight approaches for SPL for educational technologies domain is a definite need as instructional design itself is a complex activity.
- Educational technologies domain presents the need for a family of product lines catering to the needs for variety at multiple levels.
- The socio-technical nature of education domain motivates the need for SPLs that are spread across domains like learning methodologies, software engineering and human-computer interaction.
- Design of SPLs that span across different organizations from different domains.
- Facilitating assembly of educational technologies from open educational resources and further customizing them for personalized learning requires research in every aspect of software product lines from scoping to all aspects of domain and application engineering.
- In addition, lean and globally distributed software product lines could be two potential research directions for addressing the challenges of designing educational technologies for personalized learning.

7.2.4 Educational Technologies Beyond Adult Literacy

The approach presented in this thesis can be applied for design of educational technologies beyond adult literacy. Skill education is undergoing a major revamp in India extensively supported by Government of India through National Skill Development Corporation (NSDC). NSDC has set up 38 Sector Skill Councils pertaining to different industrial skills like automotive, healthcare and so on. Model curriculum[1] for different sectors and

[1] NSDC, http://www.nsdcindia.org/model-curriculum

roles has been designed. Further NSDC has partnered with 267 training partners in 25 sectors and 2500+ fixed and mobile centers. The partners are advised to customize the model curriculum as per local context, requirements, students and training methodology. However, most of the training today is done manually except using a few online videos. Can computing help? We see that manually developing educational technologies for the scale and variety present in this domain is a herculean task and our approach can be experimented in this sector.

How much effort is required to provide customized educational technologies for schooling in India? (KG1 to KG12)? How much effort is required for supporting teaching of engineering subjects (approximately around 800 courses?)? The problems become compounded due to the number of subjects to be learnt, each subject having many topics, the number of languages used as media of instruction, and the number of students as well as teachers. There is a serious debt of ways to accelerate educational technologies even if we borrow ideas of software reuse from software engineering. We see that these challenges can push the limits of software engineering and computing.

In this thesis, we made an attempt to address a research problem driven by societal challenges. We hope that this research lays down a foundation for this line of research and adds significant value to computing as well as society.

Related Publications

Research

Journal Papers

- Chimalakonda, S., & Nori, K. V. (2017, January). A Pattern for Modeling Instructional Process for Design of eLearning Systems- Quality, Scale and Variety, Vol. 78.1 (2017), *Indian Journal of Adult Education*.
- Chimalakonda, S., & Nori, K. V., An Ontology Based Framework for Modeling Instructional Design, *To be submitted to Educational Technology Research and Development (ETRD, Springer)*
- Chimalakonda, S., & Nori, K. V., Modeling Domain and Software Patterns using Ontologies: Approach, Tool and Case Study, *To be submitted to Journal of Systems and Software*
- Chimalakonda, S., & Nori, K. V., A Software Product Line Approach for Modeling a family of Educational Technologies, *To be submitted to IEEE Transactions on Learning Technologies*
- Chimalakonda, S., & Nori, K. V., Design of Educational Technologies – Quality, Scale and Variety, *To be submitted to Springer Briefs [Monographs of 50 to 120 pages]*

Full Papers

- Chimalakonda, S., & Nori, K. V. (2012, July). Towards a Model Driven eLearning Framework to Improve Quality of Teaching. In *Technology for Education (T4E), 2012 IEEE Fourth International Conference on* (pp. 138-143). IEEE.
- Nori, K. V., & Chimalakonda, S. (2011). Technological Aids To Improve Quality of Teaching. In *Proceedings of ANQ Congress* (Vol. 7). (*Best Paper Award*)
- Chimalakonda, S., & Nori, K. V. (2012, January). Accelerating educational technologies using software product lines. In *Technology Enhanced Education (ICTEE), 2012 IEEE International Conference on* (pp. 1-4). IEEE.

Short Papers

- Chimalakonda, S; Nori, K. V., "A Patterns-Based Approach for Modeling Instructional Design and TEL Systems," *Advanced Learning Technologies (ICALT), 2014 IEEE 14th International Conference on*

- Chimalakonda, S; Nori, K. V., ”IDont: An Ontology Based Educational Modeling Framework for Instructional Design,” *Advanced Learning Technologies (ICALT), 2013 IEEE 13th International Conference on* , vol., no., pp.253,255, 15-18 July 2013
- Chimalakonda, S; Nori, K. V., ”What makes it hard to teach software engineering to end users? some directions from adaptive and personalized learning,” *Software Engineering Education and Training (CSEE&T), 2013 IEEE 26th Conference on* , vol., no., pp.324,328, 19-21 May 2013

Doctoral Consortium

- Chimalakonda, S. (2011, July). GAMBLE: Towards Ensuring Quality of Education Using Goal Driven Model Based Learning Environments: Automating a Family of eLearning Systems by Integrating Lean and Software Product Lines. In *Advanced Learning Technologies (ICALT), 2011 11th IEEE International Conference on* (pp. 648-649). IEEE. (Early Stage)
- Chimalakonda, S., & Nori, K. V. (2012, July). Towards a Synthesis of Learning Methodologies, Learning Technologies and Software Product Lines. In *Advanced Learning Technologies (ICALT), 2012 IEEE 12th International Conference on* (pp. 732-733). IEEE. (Middle Stage)

Tutorial

- Chimalakonda, S., & Nori, K. V. (2012, July). A Software Engineering Perspective for Accelerating Educational Technologies. In *Advanced Learning Technologies (ICALT), 2012 IEEE 12th International Conference on* (pp. 754-755). IEEE.

Work-In-Progress

- Chimalakonda, S., & Nori, K. V. (2013, April). EasyAuthor: supporting low computer proficiency teachers in the design of educational content for adult illiterates. In *CHI’13 Extended Abstracts on Human Factors in Computing Systems* (pp. 649-654). ACM.
- Chimalakonda, S; Nori, K. V., ”GURU: An Experimental Interactive Environment for Teachers/Learners,” *Advanced Learning Technologies (ICALT), 2013 IEEE 13th International Conference on* , vol., no., pp.248,249, 15-18 July 2013 doi: 10.1109/ICALT.2013.76

Peer-Reviewed Workshop Papers

- Chimalakonda, S; Nori, K. V, ”What makes it hard to apply software product lines to educational technologies?,” *Product Line Approaches in Software Engineering (PLEASE), 2013 4th International Workshop on* , vol., no., pp.17,20, 20-20 May 2013.
- Chimalakonda, S., & Nori, K. V. (2012, June). What makes it hard to design instructional software? Towards a collaborative platform for stakeholders of instructional software. In *Cooperative and Human Aspects of Software Engineering (CHASE), 2012 5th International Workshop on* (pp. 15-19). IEEE.

Peer-Reviewed Posters

- Chimalakonda, S; Nori, K. V., ”Designing Technology for 287 Million Learners,” Advanced Learning Technologies (ICALT), 2013 IEEE 13th International Conference on , vol., no., pp.197,198, 15-18 July 2013

- Chimalakonda, S., & Nori, K. V. (2012, July). Software Engineering Aids to Accelerate Educational Technologies. In *Advanced Learning Technologies (ICALT), 2012 IEEE 12th International Conference on* (pp. 495-496). IEEE.

Presentations/Miscellaneous

- Chimalakonda, S., & Nori, K. V. , Fusion of Software Product Lines and Educational Technologies - Experiences, Challenges and Future Directions, *Presented at Product LinE Approaches in Software Engineering (PLEASE 2012) @ ICSE 2012.*
- Chimalakonda, S., & Nori, K. V., Why Software Product Lines for a family of eLearning Systems {9 Existing + 13 New + X Variants} has worked? and Why is it not enough?, *Accepted for presentation at HICSS-45 and PLEASE 2011 @ ICSE 2011.*
- Chimalakonda, S., & Nori, K. V., Towards Improving Quality of Instruction & Accelerating Educational Technologies using Software Product Lines (Poster), *Presented at Microsoft TechVista 2012, Kolkata, India*
- Chimalakonda, S., & Nori, K. V., Designing Technology for 287 million Adult Learners –Experiences & Challenges, *Presented at Microsoft TechVista 2013, Coimbatore, India*

Industry/Practice

Contributions to *International Standards* through Working group 4 (WG4) of the seventh subcommittee (SC7) titled “Software and Systems Engineering” of the Joint ISO/IEC Technical Committee (JTC1) of the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC)

- Chimalakonda, S. Dan Lee. 2016. *On the Evolution of Software and Systems Product Line Standards.* SIGSOFT Softw. Eng. Notes 41, 3 (June 2016), 27-30. DOI: http://dx.doi.org/10.1145/2934240.2934248
- ISO/IEC 26550:2015, *Software and systems engineering – Reference model for product line engineering and management*, Technical contribution through comments and comment resolution

  The work on patterns and software product lines of this thesis has impacted the following standards in software and systems product lines
- *Co-Editor*, ISO/IEC 26551,*Software and systems engineering – Tools and methods for product line requirements engineering*, Published as International Standard, 2016
- *Co-Editor*, ISO/IEC 26552, Software and systems engineering – Tools and methods for product line architecture design, Co-Editor, work-in-progress since 2014, WD stage
- *Co-Editor*, ISO/IEC 26553, *Software and systems engineering – Tools and methods for product line realization*, work-in-progress from 2012 onwards
- *Co-Editor*, ISO/IEC 26554, *Software and systems engineering – Tools and methods for product line verification and validation*, work-in-progress from 2012 onwards

- Co-Editor, ISO/IEC 26555, *Software and systems engineering – Tools and methods for product line technical management*, Published as International Standard, 2016

Society

This thesis was motivated from the grand challenge of adult literacy in India and as such is proposed to make solid contributions to society. During the progress of this thesis, we made technical suggestions to a steering committee meeting on adult education and literacy.

- Chimalakonda, S., & Nori, K. V. „ Towards Addressing the Societal Challenge of Quality Education using Technology, Challenges and Research Directions, *A Preliminary Note on Improving Quality of Education circulated to Planning Commission, August 25th, 2011*

  Research Suggestions to *Planning Commission*[2]

  - The 12th Five Year plan [2012-2017] should focus on "Quality of Education" improving teaching-learning processes, and technology-driven solutions
  - Qualitative Assessment is also more important while assessing learning outcomes rather than objective tests and quantitative measures. This approach can help teachers understand the process of deriving an answer rather than testing answers alone
  - "how can we empower inexperienced teachers through technology such that they can deliver same quality of teaching as an experienced teacher? Which would be possible only if technology-driven solutions are embedded with strong teaching-learning processes?
  - Technology solutions have to be looked cautiously and should be in in sync with learning methodologies
- The work in this thesis was presented to Shri Y.S.K. Seshu Kumar, Joint Secretary & DG - National Literacy Mission Authority and Chairman of CBSE board (then) in July 2016 and later to the 32 SRCs for further proliferation. The SRCs are formally instructed to use the approach and technologies developed as part of the thesis. Eventhough not part of this thesis, some experiments are specifically planned with SRC Hyderabad in the coming days.

[2]**Formulation of the 12th Five Year Plan- Minutes of the second meeting of the Steering Committee on Elementary Education and Literacy held on 25.08.2011-regarding**

Bibliography

[1] T. CSR. (2016) CSR Case Study, Computer Based Functional Literacy. Tata Consultancy Services. [Online]. Available: http://www.tcs.com

[2] K. Pohl, G. Böckle, and F. J. van Der Linden, *Software product line engineering: foundations, principles and techniques.* Springer Science & Business Media, 2005.

[3] S. Chimalakonda and D. H. Lee, "On the Evolution of Software and Systems Product Line Standards," *SIGSOFT Softw. Eng. Notes*, vol. 41, no. 3, pp. 27–30, Jun. 2016. [Online]. Available: http://doi.acm.org/10.1145/2934240.2934248

[4] I. J. WG4, "ISO/IEC 26550:2015 - Software and systems engineering – Reference model for product line engineering and management," ISO/IEC, Tech. Rep., 2015.

[5] A. H. Brown and T. D. Green, *The essentials of instructional design: Connecting fundamental principles with process and practice.* Routledge, 2015.

[6] M. D. Merrill, *First principles of instruction.* John Wiley & Sons, 2012.

[7] C. Berger and R. Kam, "Definitions of instructional design," *Retrieved January*, vol. 30, p. 2006, 1996.

[8] J. B. Carroll, "A model of school learning." *Teachers college record*, 1963.

[9] R. Luppicini, "A systems definition of educational technology in society," *Educational Technology & Society*, vol. 8, no. 3, pp. 103–109, 2005.

[10] A. Januszewski and M. Molenda, *Educational technology: A definition with commentary.* Routledge, 2013.

[11] S. Chimalakonda, "Towards Automating the Development of a family of eLearning Systems,", MS Thesis, 2010.

[12] C. Alexander, S. Ishikawa, and M. Silverstein, "A Pattern Language: Towns, Buildings, Construction (Center for Environmental Structure Series)," 1977.

[13] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[14] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," DTIC Document, Tech. Rep., 1990.

[15] D. L. Parnas, "On the design and development of program families," *Software Engineering, IEEE Transactions on*, no. 1, pp. 1–9, 1976.

[16] P. Clements and L. Northrop, *Software product lines: practices and patterns.* Addison-Wesley Reading, 2002, vol. 59.

[17] L. W. Anderson, D. R. Krathwohl, P. W. Airasian, K. A. Cruikshank, R. E. Mayer, P. R. Pintrich, J. Raths, and M. C. Wittrock, "A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives, abridged edition," *White Plains, NY: Longman*, 2001.

[18] S. Apel, D. Batory, C. Kästner, and G. Saake, *Feature-oriented software product lines: concepts and implementation.* Springer Science & Business Media, 2013.

[19] H. Beetham and R. Sharpe, *Rethinking pedagogy for a digital age: Designing for 21st century learning.* routledge, 2013.

[20] R. S. Nickerson and P. P. Zodhiates, *Technology in education: Looking toward 2020.* Routledge, 2013.

[21] K. M. Kapp, *The gamification of learning and instruction: game-based methods and strategies for training and education.* John Wiley & Sons, 2012.

[22] J. Daniel, "Making sense of MOOCs: Musings in a maze of myth, paradox and possibility," *Journal of Interactive Media in Education*, vol. 3, 2012.

[23] P. Goodyear, "Psychological foundations for networked learning," in *Networked learning: Perspectives and issues.* Springer, 2002, pp. 49–75.

[24] T. Govindasamy, "Successful implementation of e-learning: Pedagogical considerations," *The Internet and Higher Education*, vol. 4, no. 3, pp. 287–299, 2001.

[25] P. Goodyear and D. Yang, "Patterns, pattern languages and educational design," in *Beyond the comfort zone: Proceedings of the 21st ASCILITE Conference*, 2004, pp. 339–347.

[26] (2014) STEM Grand Challenge. Office of Naval Research. [Online]. Available: http://www.onr.navy.mil/Conference-Event-ONR/STEM-Forum/Grand-STEM-Challenge.aspx

[27] B. Sanou, “The World in 2014: ICT Facts and Figures,” *International Telecommunications Union*, 2014.

[28] *Education for All Global Monitoring Report 2013/4: Teaching and learning: Achieving quality for all.* United Nations Educational and Scientific and Cultural Organization, 2014.

[29] *Handbook for Developing IPCL Material.* Directorate of Adult Education, India, 2003.

[30] I. Patel, *Information and Communication Technology and Distance Adult Literacy Education in India.* Institute of Rural Management Anand, 2002.

[31] D. A. Wagner and R. B. Kozma, *New technologies for literacy and adult education: A global perspective.* Unesco, 2005.

[32] I. Martínez-Ortiz, P. Moreno-Ger, J. L. Sierra, and B. Fernandez-Manjon, “Educational modeling languages,” in *Computers and Education.* Springer, 2007, pp. 27–40.

[33] L. Botturi, M. Derntl, E. Boot, and K. Figl, “A classification framework for educational modeling languages in instructional design,” in *6th IEEE International Conference on Advanced Learning Technologies (ICALT 2006), Kerkrade (The Netherlands)*, 2006, pp. 1216–1220.

[34] L. Botturi, S. T. Stubbs, and I. Global, *Handbook of visual languages for instructional design: Theories and practices.* Information Science Reference Hershey, 2008.

[35] M. Caeiro, M. Llamas, and L. Anido, “PoEML: Modeling learning units through perspectives,” *Computer Standards & Interfaces*, vol. 36, no. 2, pp. 380–396, 2014.

[36] M. Rodríguez-Artacho and M. F. V. Maillo, “Modeling educational content: the cognitive approach of the PALO language,” *Educational Technology & Society*, vol. 7, no. 3, pp. 124–137, 2004.

[37] E. Villasclaras-FernáNdez, D. HernáNdez-Leo, J. I. Asensio-PéRez, and Y. Dimitriadis, “Web Collage: An implementation of support for assessment design in CSCL macro-scripts,” *Computers & Education*, vol. 67, pp. 79–97, 2013.

[38] D. G. Sampson and P. Zervas, “A hierarchical framework for open access to education and learning,” *International Journal of Web Based Communities*, vol. 10, no. 1, pp. 25–51, 2014.

[39] I. G. L. Consortium *et al.*, “IMS learning design specification,” 2003.

[40] J. Dalziel, "Implementing learning design: The learning activity management system (LAMS)," 2003.

[41] D. Laurillard, P. Charlton, B. Craft, D. Dimakopoulos, D. Ljubojevic, G. Magoulas, E. Masterman, R. Pujadas, E. A. Whitley, and K. Whittlestone, "A constructionist learning environment for teachers to model learning designs," *Journal of Computer Assisted Learning*, vol. 29, no. 1, pp. 15–30, 2013.

[42] D. Hernández-Leo, J. Chacón, L. P. Prieto, J. I. Asensio-Pérez, and M. Derntl, "Towards an integrated learning design environment," in *Scaling up Learning for Sustained Impact*. Springer, 2013, pp. 448–453.

[43] S. Neumann, M. Klebl, D. Griffiths, D. Hernández-Leo, L. De la Fuente-Valentin, H. Hummel, F. Brouns, M. Derntl, P. Oberhuemer *et al.*, "Report of the results of an IMS learning design expert workshop," 2009.

[44] R. Amorim, M. Lama, E. Sánchez, A. Riera, and X. Vila, "A learning design ontology based on the IMS specification," *Journal of Eductational Technology and Society*, vol. 9, no. 1, p. 38, 2006.

[45] C. Knight, D. Gasevic, and G. Richards, "An ontology-based framework for bridging learning design and learning content," *Journal of Eductational Technology and Society*, vol. 9, no. 1, p. 23, 2006.

[46] D. Stokić, K. Pata, V. Devedžić, J. Jovanović, L. Urošević, D. Gašević, B. Kieslinger, and J. Wild, "Intelligent learning extended organizations," *Proceedings of TELearn2008*, 2008.

[47] I. Douglas, "Instructional design based on reusable learning objects: Applying lessons of object-oriented software engineering to learning systems design," in *Frontiers in Education Conference, 2001. 31st Annual*, vol. 3. IEEE, 2001, pp. F4E–1.

[48] F. Neven and E. Duval, "Reusable learning objects: a survey of LOM-based repositories," in *Proceedings of the tenth ACM international conference on Multimedia*. ACM, 2002, pp. 291–294.

[49] T. Boyle, "Design principles for authoring dynamic, reusable learning objects," *Australian Journal of Educational Technology*, vol. 19, no. 1, pp. 46–58, 2003.

[50] D. A. Wiley, *Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy*, 2003.

[51] R. Koper and R. van Es, "Modelling units of learning from a pedagogical perspective," *Online Educ. Using Learn. Objects*, vol. 40, pp. 43–58, 2004.

[52] C. Pahl and E. Holohan, "Applications of semantic web technology to support learning content development," *Interdisciplinary Journal of E-Learning and Learning Objects*, vol. 5, 2009.

[53] D. G. Sampson and P. Zervas, "A Workflow for Learning Objects Lifecycle and Reuse: Towards Evaluating Cost Effective Reuse." *Educational Technology & Society*, vol. 14, no. 4, pp. 64–76, 2011.

[54] N. Friesen, "Three objections to learning objects," *Online education using learning objects*, pp. 59–70, 2004.

[55] P. E. Parrish, "The trouble with learning objects," *Educational Technology Research and Development*, vol. 52, no. 1, pp. 49–67, 2004.

[56] P. R. Polsani, "Use and abuse of reusable learning objects," *Journal of Digital information*, vol. 3, no. 4, 2006.

[57] S. Nurmi and T. Jaakkola, "Promises and pitfalls of learning objects," *Learning, Media and Technology*, vol. 31, no. 3, pp. 269–285, 2006.

[58] J. Sinclair, M. Joy, J.-K. Yau, and S. Hagan, "A practice-oriented review of learning objects," *Learning Technologies, IEEE Transactions on*, vol. 6, no. 2, pp. 177–192, 2013.

[59] V. Pankratius, *Product lines for digital information products.* KIT Scientific Publishing, 2007.

[60] D. Zhou, Z. Zhang, S. Zhong, and P. Xie, "The design of software architecture for e-learning platforms," in *Technologies for E-Learning and Digital Entertainment.* Springer, 2008, pp. 32–40.

[61] S. Chimalakonda and K. V. Nori, "Towards a Synthesis of Learning Methodologies, Learning Technologies and Software Product Lines," in *Advanced Learning Technologies (ICALT), 2012 IEEE 12th International Conference on.* IEEE, 2012, pp. 732–733.

[62] S. Chimalakonda and K. V. Nori, "Towards a Model Driven eLearning Framework to Improve Quality of Teaching," in *Technology for Education (T4E), 2012 IEEE Fourth International Conference on.* IEEE, 2012, pp. 138–143.

[63] S. Chimalakonda and K. V. Nori, "What makes it hard to apply software product lines to educational technologies?" in *Product Line Approaches in Software Engineering (PLEASE), 2013 4th International Workshop on.* IEEE, 2013, pp. 17–20.

[64] S. Chimalakonda and K. V. Nori, "EasyAuthor: supporting low computer proficiency teachers in the design of educational content for adult illiterates," in *CHI'13 Extended Abstracts on Human Factors in Computing Systems.* ACM, 2013, pp. 649–654.

[65] S. Chimalakonda and K. V. Nori, "Designing Technology for 287 Million Learners," in *Advanced Learning Technologies (ICALT), 2013 IEEE 13th International Conference on.* IEEE, 2013, pp. 197–198.

[66] K. Toyama, "There are no technology shortcuts to good education," *Educational Technology Debate*, vol. 8, 2011.

[67] (2014, May) The Pedagogical Patterns Project. http://www.pedagogicalpatterns.org/.

[68] P. Avgeriou, A. Papasalouros, S. Retalis, and M. Skordalakis, "Towards a pattern language for learning management systems," *Educational Technology & Society*, vol. 6, no. 2, pp. 11–24, 2003.

[69] R. Mizoguchi, J. Vanwelkenhuysen, and M. Ikeda, "Task ontology for reuse of problem solving knowledge," *Towards Very Large Knowledge Bases: Knowledge Building & Knowledge Sharing*, pp. 46–59, 1995.

[70] D. Dicheva, "Ontologies and semantic web for e-learning," in *Handbook on information technologies for education and training.* Springer, 2008, pp. 47–65.

[71] K. Verbert, J. Jovanovic, E. Duval, D. Gasevic, and M. Meire, "Ontology-based learning content repurposing: the ALOCoM framework," *International Journal on E-Learning*, vol. 5, no. 1, pp. 67–74, 2006.

[72] R. Koper and Y. Miao, "Using the IMS LD standard to describe learning designs," 2007.

[73] C. M. Reigeluth, *Instructional design theories and models: An overview of their current status.* Routledge, 2013.

[74] (2016) National Literacy Mission. [Online]. Available: http://www.nlm.ac.in

[75] S. B. Merriam, "Andragogy and self-directed learning: Pillars of adult learning theory," *New directions for adult and continuing education*, vol. 2001, no. 89, pp. 3–14, 2001.

[76] U. I. for Lifelong Learning (UIL), "Belem Framework for Action," 2010.

[77] U. Hanemann, “Harnessing the potential of ICTs for literacy teaching and learning: effective literacy and numeracy programmes using radio, tv, mobile phones, tablets, and computers. vols.” *Hamburg: UNESCO Institute for Lifelong Learning*, 2014.

[78] *Confintea VI: sixth international conference on adult education: final report.* UNESCO Institute for Lifelong Learning, Germany, 2010.

[79] G. Farrell, “ICT and literacy: who benefits?: experience from Zambia and India,” 2004.

[80] A. Dighe, “Use of ICTs in Literacy.” *Indian Journal of Adult Education*, 2010.

[81] T. Akshar+. (2016, July) Literacy Programmes in India, Digital Literacy in India, TARA Akshar+. [Online]. Available: http://taraakshar.org/

[82] B. Kothari, “Let a billion readers bloom: Same language subtitling (SLS) on television for mass literacy,” *International review of education*, vol. 54, no. 5-6, pp. 773–780, 2008.

[83] B. Kothari and T. Bandyopadhyay, “Can India’s “literate” read?” *International Review of Education*, vol. 56, no. 5-6, pp. 705–728, 2010.

[84] D. A. Wagner, C. Daswani, and R. Karnati, “Technology and mother-tongue literacy in southern India: Impact studies among young children and out-of-school youth,” *Information Technologies & International Development*, vol. 6, no. 4, pp. pp–23, 2010.

[85] S. Mitra and V. Rana, “Children and the Internet: Experiments with minimally invasive education in India,” *British Journal of Educational Technology*, vol. 32, no. 2, pp. 221–232, 2001.

[86] D. A. Wagner, “M4R: A landscape research review of mobiles for reading,” Draft technical report. Philadelphia: University of Pennsylvania, Tech. Rep., 2013.

[87] D. A. Wagner, N. M. Castillo, K. M. Murphy, M. Crofton, and F. T. Zahra, “Mobiles for literacy in developing countries: An effectiveness framework,” *Prospects*, vol. 44, no. 1, pp. 119–132, 2014.

[88] A. Chudgar, “The promise and challenges of using mobile phones for adult literacy training: Data from one Indian state,” *International Journal of Educational Development*, vol. 34, pp. 20–29, 2014.

[89] R. K. Megalingam, A. P. Rajendran, A. T. Solamon, and D. Dileep, “EduPad—A tablet based educational system for improving adult literacy in rural India,” in *Technology Enhanced Education (ICTEE), 2012 IEEE International Conference on.* IEEE, 2012, pp. 1–5.

[90] K. Browne, C. Anand, and E. Gosse, "Gamification and serious game approaches for adult literacy tablet software," *Entertainment Computing*, vol. 5, no. 3, pp. 135–146, 2014.

[91] V. Motkuri, "When Will India Achieve Universal Adult Literacy: Status and Prospects," 2013.

[92] D. A. Wagner, "Learning and literacy: A research agenda for post-2015," *International Review of Education*, pp. 1–15, 2014.

[93] A. Dighe, "Pedagogical Approaches to Literacy Acquisition," *Education for All: Literacy for Life*, 2005.

[94] K. M. Culp, M. Honey, and E. Mandinach, "A retrospective on twenty years of education technology policy," *Journal of Educational Computing Research*, vol. 32, no. 3, pp. 279–307, 2005.

[95] S. Grineski, "Questioning the role of technology in higher education: Why is this the road less traveled?" *The Internet and higher education*, vol. 2, no. 1, pp. 45–54, 1999.

[96] J. K. Maney, "The role of technology in education: Reality, pitfalls, and potential," *Handbook of educational policy*, pp. 387–415, 1999.

[97] C. Dede, "Reshaping the Role of Technology in Education," in *Breakthrough Teaching and Learning*. Springer, 2011, pp. 1–3.

[98] S. Livingstone, "Critical reflections on the benefits of ICT in education," *Oxford Review of Education*, vol. 38, no. 1, pp. 9–24, 2012.

[99] M. Burns, "Success, Failure or no Significant Difference: Charting a Course for Successful Educational Technology Integration." *International Journal of Emerging Technologies in Learning*, vol. 8, no. 1, pp. 38–45, 2013.

[100] R. A. Reiser and J. V. Dempsey, *Trends and issues in instructional design and technology*. Merrill/Prentice Hall, 2011.

[101] B. Zimmermann, S. Bergsträßer, C. Rensing, and R. Steinmetz, "A Requirements Analysis of Adaptations of Re-Usable Content," in *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications*, 2006, pp. 2096–2010.

[102] A. K. Bednar, D. Cunningham, T. M. Duffy, and J. D. Perry, "Theory into practice: How do we link," *Constructivism and the technology of instruction: A conversation*, pp. 17–34, 1992.

[103] T. Boyle and J. Cook, “Towards a pedagogically sound basis for learning object portability and re-use,” in *Meeting at the Crossroads. Proceedings of the 18th Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education. The University of Melbourne*, vol. 101, 2001, p. 109.

[104] P. Goodyear, P. Avgeriou, R. Baggetun, S. Bartoluzzi, S. Retalis, F. Ronteltap, and E. Rusman, “Towards a pattern language for networked learning,” in *proceedings of networked learning*, 2004, pp. 449–455.

[105] F. Garzotto, S. Retalis, and V. Chapter, “A critical perspective on design patterns for e-learning,” *Learning Objects: Is-sues, Applications and Technologies*, pp. 346–372, 2009.

[106] T. M. Duffy and D. H. Jonassen, *Constructivism and the technology of instruction: A conversation.* Routledge, 2013.

[107] J. Lowyck, “Bridging learning theories and technology-enhanced environments: A critical appraisal of its history,” in *Handbook of research on educational communications and technology.* Springer, 2014, pp. 3–20.

[108] H. Sharp, M. L. Manns, and J. Eckstein, “The pedagogical patterns project (poster session),” in *Addendum to the 2000 proceedings of the conference on Object-oriented programming, systems, languages, and applications (Addendum).* ACM, 2000, pp. 139–140.

[109] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software.* Pearson Education, 1994.

[110] F. Buschmann, K. Henney, and D. Schimdt, *Pattern-oriented Software Architecture: On Patterns and Pattern Language.* John Wiley & Sons, 2007, vol. 5.

[111] J. Bergin, J. Eckstein, M. Volter, M. Sipos, E. Wallingford, K. Marquardt, J. Chandler, H. Sharp, and M. L. Manns, *Pedagogical patterns: advice for educators.* Joseph Bergin Software Tools, 2012.

[112] A. Cristea and F. Garzotto, “Designing patterns for adaptive or adaptable educational hypermedia: a taxonomy,” in *World Conference on Educational Multimedia, Hypermedia and Telecommunications*, vol. 2004, no. 1, 2004, pp. 808–813.

[113] T. Iba and T. Miyake, “Learning patterns: A pattern language for creative learning ii,” in *Proceedings of the 1st Asian Conference on Pattern Languages of Programs.* ACM, 2010, p. 4.

[114] C. Midgley, *Goals, goal structures, and patterns of adaptive learning.* Routledge, 2014.

[115] D. Laurillard, *Teaching as a design science: Building pedagogical patterns for learning and technology.* Routledge, 2012.

[116] M. Derntl and R. Motschnig-Pitrik, "Patterns for blended, person-centered learning: Strategy, concepts, experiences, and evaluation," in *Proceedings of the 2004 ACM symposium on Applied computing.* ACM, 2004, pp. 916–923.

[117] Y. Mor, *Practical Design Patterns for Teaching and Learning with Technology.* Springer, 2014.

[118] R. McGreal, "Learning objects: A practical definition," *International Journal of Instructional Technology and Distance Learning (IJITDL)*, vol. 9, no. 1, 2004.

[119] D. Churchill, "Towards a useful classification of learning objects," *Educational Technology Research and Development*, vol. 55, no. 5, pp. 479–497, 2007.

[120] C. M. Reigeluth and L. M. Nelson, "A new paradigm of ISD?" *Educational media and technology yearbook*, vol. 22, 1997.

[121] L. T. S. Committee *et al.*, "IEEE standard for learning object metadata," *IEEE Standard*, vol. 1484, no. 1, pp. 2007–04, 2002.

[122] K. Bennett and P. McGee, "Transformative power of the learning object debate," *Open Learning: The Journal of Open, Distance and e-Learning*, vol. 20, no. 1, pp. 15–30, 2005.

[123] D. D. Williams, "Evaluation of learning objects and instruction using learning objects," *The instructional use of learning objects. Available from http://www. reusability. org/read/chapters/williams. doc*, 2000.

[124] R. H. Kay and L. Knaack, "Evaluating the learning in learning objects," *Open Learning*, vol. 22, no. 1, pp. 5–28, 2007.

[125] D. Wiley, "Getting axiomatic about learning objects," 2000.

[126] X. Ochoa and E. Duval, "Quantitative analysis of learning object repositories," *Learning Technologies, IEEE Transactions on*, vol. 2, no. 3, pp. 226–238, 2009.

[127] R. McGreal, "A typology of learning object repositories," in *Handbook on information technologies for education and training.* Springer, 2008, pp. 5–28.

[128] R. Koper, "Modeling units of study from a pedagogical perspective: the pedagogical meta-model behind EML," 2001.

[129] C. Süß and B. Freitag, "LMML-the learning material markup language framework," in *Proceedings of the International Workshop Interactive Computer Aided Learning, Villach, Austria*, 2002.

[130] L. Botturi, "E2ML: A visual language for the design of instruction," *Educational Technology Research and Development*, vol. 54, no. 3, pp. 265–293, 2006.

[131] D. Griffiths, J. Blat, R. Garcia, H. Vogten, and K. Kwong, "Learning design tools," in *Learning design*. Springer, 2005, pp. 109–135.

[132] D. Hernández-Leo, E. Villasclaras-Fernández, I. Jorrín-Abellán, J. Asensio-Pérez, Y. Dimitriadis, I. Ruiz-Requies, and B. Rubia-Avi, "COLLAGE, a collaborative learning design editor based on patterns," 2006.

[133] D. Griffiths, P. Beauvoir, O. Liber, and M. Barrett-Baxendale, "From reload to ReCourse: learning from IMS learning design implementations," *Distance Education*, vol. 30, no. 2, pp. 201–222, 2009.

[134] D. Sampson, P. Karampiperis, and P. Zervas, "ASK-LDT: A Web-based learning scenarios authoring environment based on IMS Learning Design," *International Journal on Advanced Technology for Learning (ATL)*, vol. 2, no. 4, pp. 207–215, 2005.

[135] P. Charlton and G. D. Magoulas, "Context-aware framework for supporting personalisation and adaptation in creation of learning designs," *S. Graf, F. Lin, Kinshuk, & R. McGreal (Eds.), Intelligent and adaptive learning systems: Technology enhanced support for learners and teachers*, pp. 229–248, 2011.

[136] D. Hernández-Leo, P. Moreno, J. Chacón, and J. Blat, "LdShake support for team-based learning design," *Computers in Human Behavior*, vol. 37, pp. 402–412, 2014.

[137] T. Boyle, "Layered learning design: Towards an integration of learning design and learning object perspectives," *Computers & Education*, vol. 54, no. 3, pp. 661–668, 2010.

[138] G. Paquette, "Technology-Based Instructional Design: Evolution and Major Trends," in *Handbook of Research on Educational Communications and Technology*. Springer, 2014, pp. 661–671.

[139] M. Caeiro, L. Anido, and M. Llamas, "A critical analysis of IMS Learning Design," in *Designing for Change in Networked Learning Environments*. Springer, 2003, pp. 363–367.

[140] M. Caeiro-Rodríguez, L. Anido-Rifón, and M. Llamas-Nistal, "Challenges in Educational Modelling: Expressiveness of IMS Learning Design," *Subscription Prices and Ordering Information*, p. 215, 2010.

[141] G. Durand and S. Downes, "Toward simple learning design 2.0," in *Computer Science & Education, 2009. ICCSE'09. 4th International Conference on.* IEEE, 2009, pp. 894–897.

[142] F. König and A. Paramythis, "Towards improved support for adaptive collaboration scripting in IMS LD," in *Sustaining TEL: From Innovation to Learning and Practice.* Springer, 2010, pp. 197–212.

[143] D. Griffiths and J. Blat, "The role of teachers in editing and authoring units of learning using IMS Learning Design," *International Journal on Advanced Technology for Learning, Special Session on" Designing Learning Activities: From Content-based to Context-based Learning Services*, vol. 2, no. 4, 2005.

[144] L. P. Prieto, Y. Dimitriadis, B. Craft, M. Derntl, V. Émin, M. Katsamani, D. Laurillard, E. Masterman, S. Retalis, and E. Villasclaras, "Learning design Rashomon II: exploring one lesson through multiple tools," *Research in Learning Technology*, vol. 21, 2013.

[145] M. Uschold and M. Gruninger, "Ontologies: Principles, methods and applications," *The knowledge engineering review*, vol. 11, no. 02, pp. 93–136, 1996.

[146] B. Chandrasekaran, J. R. Josephson, and V. R. Benjamins, "What are ontologies, and why do we need them?" *IEEE Intelligent systems*, vol. 14, no. 1, pp. 20–26, 1999.

[147] H.-J. Happel and S. Seedorf, "Applications of ontologies in software engineering," in *Proc. of Workshop on Sematic Web Enabled Software Engineering"(SWESE) on the ISWC.* Citeseer, 2006, pp. 5–9.

[148] V. A. Pinto, C. L. de Rezende Rohlfs, and F. S. Parreiras, "Applications of Ontologies in Enterprise Modelling: A Systematic Mapping Study," in *Advances in Conceptual Modeling.* Springer, 2014, pp. 23–32.

[149] D. Dermeval, J. Vilela, I. I. Bittencourt, J. Castro, S. Isotani, P. Brito, and A. Silva, "Applications of ontologies in requirements engineering: a systematic review of the literature," *Requirements Engineering*, pp. 1–33, 2015.

[150] F. Giunchiglia and I. Zaihrayeu, "Lightweight ontologies," in *Encyclopedia of Database Systems.* Springer, 2009, pp. 1613–1619.

[151] H.-J. Happel, W. Maalej, and S. Seedorf, "Applications of ontologies in collaborative software development," in *Collaborative Software Engineering.* Springer, 2010, pp. 109–129.

[152] D. G. Sampson, M. D. Lytras, G. Wagner, and P. Diaz, "Ontologies and the Semantic Web for E-learning," *Educational Technology & Society*, vol. 7, no. 4, pp. 26–28, 2004.

[153] R. Mizoguchi and J. Bourdeau, “Using ontological engineering to overcome common AI-ED problems,” *Journal of Artificial Intelligence and Education*, vol. 11, pp. 107–121, 2000.

[154] V. Psyché, J. Bourdeau, R. Nkambou, and R. Mizoguchi, “Making learning design standards work with an ontology of educational theories,” in *12th Artificial Intelligence in Education (AIED2005)*. IOS Press, 2005, pp. 539–546.

[155] R. Mizoguchi, Y. Hayashi, and J. Bourdeau, “Inside theory-aware and standards-compliant authoring system,” in *SW-EL’07*, 2007, pp. 18–pages.

[156] T. Kasai, K. Nagano, and R. Mizoguchi, “Instructional Design Support System Based on Both Theory and Practice and Its Evaluation,” *Proceedings of ICCE2011*, pp. 1–8, 2011.

[157] V. Devedzic, “Ontologies: borrowing from software patterns,” *intelligence*, vol. 10, no. 3, pp. 14–24, 1999.

[158] K. Verbert, J. Klerkx, M. Meire, J. Najjar, and E. Duval, “Towards a global component architecture for learning objects: An ontology based approach,” in *On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops*. Springer, 2004, pp. 713–722.

[159] C. Vidal-Castro, M.-Á. Sicilia, and M. Prieto, “Representing instructional design methods using ontologies and rules,” *Knowledge-Based Systems*, vol. 33, pp. 180–194, 2012.

[160] G. Paquette, “An ontology and a software framework for competency modeling and management,” *Educational Technology & Society*, vol. 10, no. 3, pp. 1–21, 2007.

[161] J. Roschelle, C. DiGiano, M. Koutlis, A. Repenning, J. Phillips, N. Jackiw, and D. Suthers, “Developing educational software components,” *Computer*, vol. 32, no. 9, pp. 50–58, 1999.

[162] J. M. Dodero and D. Díez, “Model-driven instructional engineering to generate adaptable learning materials,” 2006.

[163] J. M. Dodero, I. Ruiz-Rube, M. Palomo-Duarte, J. Cabot *et al.*, “Model-driven learning design,” *Journal of Research and Practice in Information Technology*, vol. 44, no. 3, p. 267, 2012.

[164] J. Torres, J. Resendiz, I. Aedo, and J. M. Dodero, “A model-driven development approach for learning design using the LPCEL Editor,” *Journal of King Saud University-Computer and Information Sciences*, vol. 26, no. 1, pp. 17–27, 2014.

[165] J.-M. Dodero, F.-J. Garcia-Penalvo, C. Gonzalez, P. Moreno-Ger, M.-A. Redondo, A. Sarasa, and J.-L. Sierra, "Points of view on software engineering for eLearning (panel session)," in *Computers in Education (SIIE), 2012 International Symposium on.* IEEE, 2012, pp. 1–4.

[166] T. Xie, N. Tillmann, and J. De Halleux, "Educational software engineering: Where software engineering, education, and gaming meet," in *Proceedings of the 3rd International Workshop on Games and Software Engineering: Engineering Computer Games to Enable Positive, Progressive Change.* IEEE Press, 2013, pp. 36–39.

[167] C. Ghezzi, M. Jazayeri, and D. Mandrioli, *Fundamentals of software engineering.* Prentice Hall PTR, 2002.

[168] E. W. Dijkstra, "On the role of scientific thought," in *Selected writings on computing: a personal perspective.* Springer, 1982, pp. 60–66.

[169] J. Greenfield, K. Short, S. Cook, S. Kent, and J. Crupi, *Software factories: assembling applications with patterns, models, frameworks, and tools.* Wiley Pub., 2004.

[170] P. B. Kruchten, "The 4+ 1 view model of architecture," *Software, IEEE*, vol. 12, no. 6, pp. 42–50, 1995.

[171] D. L. Parnas, "On the criteria to be used in decomposing systems into modules," *Communications of the ACM*, vol. 15, no. 12, pp. 1053–1058, 1972.

[172] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin, "Aspect-oriented programming," in *European conference on object-oriented programming.* Springer, 1997, pp. 220–242.

[173] E. Evans, *Domain-driven design: tackling complexity in the heart of software.* Addison-Wesley Professional, 2004.

[174] S. Berczuk, "Finding solutions through pattern languages," *Computer*, vol. 27, no. 12, pp. 75–76, 1994.

[175] M. Derntl, *Patterns for person centered e-learning.* Citeseer, 2005.

[176] Y. Mor, S. Warburton, and N. Winters, "Participatory pattern workshops: a methodology for open learning design inquiry," *Research in Learning Technology*, vol. 20, 2012.

[177] G. Meszaros and J. Doble, "A pattern language for pattern writing," *Pattern languages of program design*, vol. 3, pp. 529–574, 1998.

[178] J. O. Borchers, “A pattern approach to interaction design,” *Ai & Society*, vol. 15, no. 4, pp. 359–376, 2001.

[179] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, “{Pattern-oriented Software Architecture Volume 1},” 1996.

[180] D. Laurillard, *Rethinking university teaching: A conversational framework for the effective use of learning technologies.* Routledge, 2013.

[181] D. Laurillard, P. Charlton, B. Craft, D. Dimakopoulos, D. Ljubojevic, G. Magoulas, E. Masterman, R. Pujadas, E. A. Whitley, and K. Whittlestone, “A constructionist learning environment for teachers to model learning designs,” *Journal of Computer Assisted Learning*, vol. 29, no. 1, pp. 15–30, 2013.

[182] S. Chimalakonda and K. V. Nori, “IDont: An Ontology Based Educational Modeling Framework for Instructional Design,” in *Advanced Learning Technologies (ICALT), 2013 IEEE 13th International Conference on.* IEEE, 2013, pp. 253–255.

[183] R. Koper, *An introduction to learning design.* Springer, 2005.

[184] T. Goddard, D. Griffiths, and W. Mi, “Why has Ims Learning Design not Led to the Advances which were Hoped for?” in *The Art & Science of Learning Design.* Springer, 2015, pp. 121–136.

[185] D. Burgos, “A Critical Review of Ims Learning Design,” in *The Art & Science of Learning Design.* Springer, 2015, pp. 137–153.

[186] P. Goodyear and S. Retalis, *Technology-enhanced learning: Design patterns and pattern languages.* Sense Publishers, 2010.

[187] Y. Mor, “SNaP! Re-using, sharing and communicating designs and design knowledge using scenarios, narratives and patterns,” *Handbook of design in educational technology*, pp. 189–200, 2013.

[188] R. N. Taylor, N. Medvidovic, and E. M. Dashofy, *Software architecture: foundations, theory, and practice.* Wiley Publishing, 2009.

[189] K. Czarnecki and U. W. Eisenecker, “Generative programming,” *Edited by G. Goos, J. Hartmanis, and J. van Leeuwen*, p. 15, 2000.

[190] G. E. Krasner, S. T. Pope *et al.*, “A description of the model-view-controller user interface paradigm in the smalltalk-80 system,” *Journal of object oriented programming*, vol. 1, no. 3, pp. 26–49, 1988.

[191] R. F. Mager, “Preparing instructional objectives.” 1962.

[192] J. R. Josephson and S. G. Josephson, *Abductive inference: Computation, philosophy, technology*. Cambridge University Press, 1996.

[193] A. Aamodt and E. Plaza, "Case-based reasoning: Foundational issues, methodological variations, and system approaches," *AI communications*, vol. 7, no. 1, pp. 39–59, 1994.

[194] S. Papert, *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc., 1980.

[195] L. Magnani, N. Nersessian, and P. Thagard, *Model-based reasoning in scientific discovery*. Springer Science & Business Media, 2012.

[196] A. Mørch, "Three levels of end-user tailoring: Customization, integration, and extension," *Computers and design in context*, pp. 51–76, 1997.

[197] C. Alario-Hoyos, M. L. Bote-Lorenzo, E. GóMez-SáNchez, J. I. Asensio-PéRez, G. Vega-Gorgojo, and A. Ruiz-Calleja, "GLUE!: An architecture for the integration of external tools in Virtual Learning Environments," *Computers & Education*, vol. 60, no. 1, pp. 122–137, 2013.

[198] A. Karagkasidis, "Developing GUI Applications: Architectural Patterns Revisited." in *EuroPLoP*, 2008.

[199] W. Zimmer *et al.*, "Relationships between design patterns," *Pattern languages of program design*, vol. 57, 1995.

[200] J. McCarthy, *Programs with common sense*. Defense Technical Information Center, 1963.

[201] A. Newell, "The knowledge level," *Artificial intelligence*, vol. 18, no. 1, pp. 87–127, 1982.

[202] D. B. Lenat and E. A. Feigenbaum, "On the thresholds of knowledge," *Artificial intelligence*, vol. 47, no. 1-3, pp. 185–250, 1991.

[203] J. F. Sowa, "Knowledge representation: logical, philosophical, and computational foundations," 1999.

[204] C. Baral and G. De Giacomo, "Knowledge Representation and Reasoning: What's Hot." in *AAAI*, 2015, pp. 4316–4317.

[205] M. of Human Resource Development of India. (2010, June) Country Paper: Status and Major Challenges of Literacy in India. UNESCO. [Online]. Available: http://www.unesco.org/fileadmin/MULTIMEDIA/HQ/ED/ED/pdf/Abuja_programme.pdf

[206] S. Staab and R. Studer, *Handbook on ontologies.* Springer Science & Business Media, 2013.

[207] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge engineering: principles and methods," *Data & knowledge engineering*, vol. 25, no. 1, pp. 161–197, 1998.

[208] W. Wong, W. Liu, and M. Bennamoun, "Ontology learning from text: A look back and into the future," *ACM Computing Surveys (CSUR)*, vol. 44, no. 4, p. 20, 2012.

[209] D. Fensel, "Ontologies: Dynamic networks of formally represented meaning," *Vrije University: Amsterdam*, 2001.

[210] N. F. Noy, "Semantic integration: a survey of ontology-based approaches," *ACM Sigmod Record*, vol. 33, no. 4, pp. 65–70, 2004.

[211] A. Gomez-Perez, M. Fernández-López, and O. Corcho, *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web.* Springer Science & Business Media, 2006.

[212] R. Mizoguchi and J. Bourdeau, "Using Ontological Engineering to Overcome AI-ED Problems: Contribution, Impact and Perspectives," *International Journal of Artificial Intelligence in Education*, pp. 1–16, 2015.

[213] M. Uschold and M. Gruninger, "Ontologies and semantics for seamless connectivity," *ACM SIGMod Record*, vol. 33, no. 4, pp. 58–64, 2004.

[214] N. Guarino, *Formal ontology in information systems: Proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy.* IOS press, 1998, vol. 46.

[215] W. O. W. Group *et al.*, "{OWL} 2 Web Ontology Language Document Overview," 2009.

[216] M. Krötzsch, *OWL 2 Profiles: An introduction to lightweight ontology languages.* Springer, 2012.

[217] M. Fernández-López and A. Gómez-Pérez, "Overview and analysis of methodologies for building ontologies," *The Knowledge Engineering Review*, vol. 17, no. 02, pp. 129–156, 2002.

[218] O. Corcho, M. Fernández-López, and A. Gómez-Pérez, "Methodologies, tools and languages for building ontologies. Where is their meeting point?" *Data & knowledge engineering*, vol. 46, no. 1, pp. 41–64, 2003.

[219] M. Cristani and R. Cuel, “A survey on ontology creation methodologies,” *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 1, no. 2, pp. 49–69, 2005.

[220] E. P. B. Simperl and C. Tempich, “Ontology engineering: a reality check,” in *On the move to meaningful internet systems 2006: CoopIS, DOA, GADA, and ODBASE*. Springer, 2006, pp. 836–854.

[221] J. Brank, M. Grobelnik, and D. Mladenic, “A survey of ontology evaluation techniques,” in *Proceedings of the conference on data mining and data warehouses (SiKDD 2005)*, 2005, pp. 166–170.

[222] S. C. J. Lim, Y. Liu, and W. B. Lee, “A methodology for building a semantically annotated multi-faceted ontology for product family modelling,” *Advanced Engineering Informatics*, vol. 25, no. 2, pp. 147–161, 2011.

[223] A. De Nicola, M. Missikoff, and R. Navigli, “A software engineering approach to ontology building,” *Information systems*, vol. 34, no. 2, pp. 258–275, 2009.

[224] S. Ahmed, S. Kim, and K. M. Wallace, “A methodology for creating ontologies for engineering design,” *Journal of computing and information science in engineering*, vol. 7, no. 2, pp. 132–140, 2007.

[225] N. Noy, D. L. McGuinness *et al.*, “Ontology development 101,” *Knowledge Systems Laboratory, Stanford University*, 2001.

[226] S. Boyce and C. Pahl, “Developing domain ontologies for course content,” *Educational Technology & Society*, vol. 10, no. 3, pp. 275–288, 2007.

[227] N. Henze, P. Dolog, and W. Nejdl, “Reasoning and Ontologies for Personalized E-Learning in the Semantic Web.” *Educational Technology & Society*, vol. 7, no. 4, pp. 82–97, 2004.

[228] P. Brusilovsky and E. Millán, “User models for adaptive hypermedia and adaptive educational systems,” in *The adaptive web*. Springer-Verlag, 2007, pp. 3–53.

[229] R. Cheung, C. Wan, and C. Cheng, *An ontology-based framework for personalized adaptive learning*. Springer, 2010.

[230] R. Mizoguchi, K. Sinitsa, and M. Ikeda, “Task ontology design for intelligent educational/training systems,” in *Position Paper for ITS□96 Workshop on Architectures and Methods for Designing Cost-Effective and Reusable ITSs, Montreal*, 1996.

[231] V. Aleven, B. M. Mclaren, J. Sewall, and K. R. Koedinger, "A new paradigm for intelligent tutoring systems: Example-tracing tutors," *International Journal of Artificial Intelligence in Education*, vol. 19, no. 2, pp. 105–154, 2009.

[232] M. C. Desmarais and R. S. d Baker, "A review of recent advances in learner and skill modeling in intelligent learning environments," *User Modeling and User-Adapted Interaction*, vol. 22, no. 1-2, pp. 9–38, 2012.

[233] A. Papasalouros, K. Kanaris, and K. Kotis, "Automatic Generation Of Multiple Choice Questions From Domain Ontologies." in *e-Learning*. Citeseer, 2008, pp. 427–434.

[234] M. Carvalho and H. Pain, "An Ontology for a Literacy teaching ITS," in *AI-ED99 Workshop*, 1999.

[235] BBC. (2016) Curriculum Ontology. BBC. [Online]. Available: http://www.bbc.co.uk/ontologies/curriculum

[236] J. Bourdeau, R. Mizoguchi, Y. Hayashi, V. Psyche, and R. Nkambou, "When the Domain of the Ontology is Education," *Proc. of I2LOR'07*, 2007.

[237] S. R. Heiyanthuduwage, R. Schwitter, and M. A. Orgun, "OWL 2 learn profile: an ontology sublanguage for the learning domain," *SpringerPlus*, vol. 5, no. 1, p. 1, 2016.

[238] S. Ahmed and D. Parsons, "Abductive science inquiry using mobile devices in the classroom," *Computers & Education*, vol. 63, pp. 62–72, 2013.

[239] S. Isotani, R. Mizoguchi, S. Isotani, O. M. Capeli, N. Isotani, A. R. De Albuquerque, I. I. Bittencourt, and P. Jaques, "A Semantic Web-based authoring tool to facilitate the planning of collaborative learning scenarios compliant with learning theories," *Computers & Education*, vol. 63, pp. 267–284, 2013.

[240] G. C. Challco, D. Moreira, R. Mizoguchi, and S. Isotani, "Towards an ontology for gamifying collaborative learning scenarios," in *Intelligent Tutoring Systems*. Springer, 2014, pp. 404–409.

[241] A. Lalingkar, C. Ramanathan, and S. Ramani, "MONTO: a machine-readable ontology for teaching word problems in mathematics," *Journal of Educational Technology and Society*, 2015.

[242] A. Abran, J. J. Cuadrado, E. García-Barriocanal, O. Mendes, S. Sánchez-Alonso, and M. A. Sicilia, "Engineering the ontology for the SWEBOK: Issues and techniques," in *Ontologies for software engineering and software technology*. Springer, 2006, pp. 103–121.

[243] J. Jovanović, C. Knight, D. Gašević, and G. Richards, “Learning object context on the semantic web,” in *Advanced Learning Technologies, 2006. Sixth International Conference on.* IEEE, 2006, pp. 669–673.

[244] S. Wang and A. Koohang, “Ontology of learning objects repository for pedagogical knowledge sharing,” *International Journal of Doctoral Studies*, vol. 4, pp. 1–12, 2009.

[245] G. Paquette, “A Competency-Based Ontology for Learning Design Repositories,” *International Journal of Advanced Computer Science and Applications*, vol. 5, no. 1, pp. 55–62, 2014.

[246] O. Bohl, J. Scheuhase, R. Sengler, and U. Winand, “The sharable content object reference model (SCORM)-a critical review,” in *Computers in education, 2002. proceedings. international conference on.* IEEE, 2002, pp. 950–951.

[247] A. Ram and D. B. Leake, *Goal-driven learning.* MIT press, 1995.

[248] S. P. Mudur, N. Nayak, S. Shanbhag, and R. Joshi, “An architecture for the shaping of Indic texts,” *Computers & Graphics*, vol. 23, no. 1, pp. 7–24, 1999.

[249] A. K. Singh, “A computational phonetic model for indian language scripts,” in *Constraints on Spelling Changes: Fifth International Workshop on Writing Systems.* Nijmegen, The Netherlands, 2006.

[250] F. Wild, P. Lefrere, and P. Scott, “Advances in Technology Enhanced Learning,” 2013.

[251] B. J. Cox, “Planning the software industrial revolution,” *IEEE software*, vol. 7, no. 6, pp. 25–33, 1990.

[252] P. Wegner, *Research directions in software technology.* MIT Press, 1979, vol. 1.

[253] B. J. Pine, *Mass customization: the new frontier in business competition.* Harvard Business Press, 1993.

[254] R. Lotufo, S. She, T. Berger, K. Czarnecki, and A. Wąsowski, “Evolution of the linux kernel variability model,” in *International Conference on Software Product Lines.* Springer, 2010, pp. 136–150.

[255] SPLC. (2016, July) Software Product Lines Hall of Fame. [Online]. Available: http://splc.net/fame.html

[256] C. M. Reigeluth, *Instructional-design theories and models: A new paradigm of instructional theory.* Routledge, 2013, vol. 2.

[257] C. M. Reigeluth and A. A. Carr-Chelman, "Instructional-Design Theories and Models: Building a Common Knowledge Base. Volume III." *Education Review//Reseñas Educativas*, 2009.

[258] R. M. Gagne and L. J. Briggs, *Principles of instructional design.* Holt, Rinehart & Winston, 1974.

[259] W. Dick, L. Carey, J. O. Carey *et al.*, *The systematic design of instruction.* Longman New York, 2001, vol. 5.

[260] H. Hussmann, G. Meixner, and D. Zuehlke, *Model-driven development of advanced user interfaces.* Springer, 2011, vol. 340.

[261] L. M. Northrop, "SEI's software product line tenets," *IEEE software*, vol. 19, no. 4, p. 32, 2002.

[262] A. Metzger and K. Pohl, "Software product line engineering and variability management: achievements and challenges," in *Proceedings of the on Future of Software Engineering.* ACM, 2014, pp. 70–84.

[263] C. Krueger, "Easing the transition to software mass customization," in *International Workshop on Software Product-Family Engineering.* Springer, 2001, pp. 282–293.

[264] M. Matinlassi, "Comparison of software product line architecture design methods: COPA, FAST, FORM, KobrA and QADA," in *Proceedings of the 26th International Conference on Software Engineering.* IEEE Computer Society, 2004, pp. 127–136.

[265] L. Chen and M. A. Babar, "A systematic review of evaluation of variability management approaches in software product lines," *Information and Software Technology*, vol. 53, no. 4, pp. 344–362, 2011.

[266] T. Thüm, S. Apel, C. Kästner, I. Schaefer, and G. Saake, "A classification and survey of analysis strategies for software product lines," *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, p. 6, 2014.

[267] J. S. Fant, H. Gomaa, and R. G. Pettit, "A pattern-based modeling approach for software product line engineering," in *System Sciences (HICSS), 2013 46th Hawaii International Conference on.* IEEE, 2013, pp. 4985–4994.

[268] K. Mohan and B. Ramesh, "Ontology-based support for variability management in product and families," in *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on.* IEEE, 2003, pp. 9–pp.

[269] T. Asikainen, T. Männistö, and T. Soininen, "Kumbang: A domain ontology for modelling variability in software product families," *Advanced Engineering Informatics*, vol. 21, no. 1, pp. 23–40, 2007.

[270] S.-B. Lee, J.-W. Kim, C.-Y. Song, and D.-K. Baik, "An approach to analyzing commonality and variability of features using ontology in a software product line engineering," in *5th ACIS International Conference on Software Engineering Research, Management & Applications (SERA 2007)*. IEEE, 2007, pp. 727–734.

[271] A. P. d. Silva, E. Costa, I. I. Bittencourt, P. H. Brito, O. Holanda, and J. Melo, "Ontology-based software product line for building semantic web applications," in *Proceedings of the 2010 Workshop on Knowledge-Oriented Product Line Engineering*. ACM, 2010, p. 1.

[272] R. Heradio, H. Perez-Morago, D. Fernandez-Amoros, F. J. Cabrerizo, and E. Herrera-Viedma, "A bibliometric analysis of 20 years of research on software product lines," *Information and Software Technology*, vol. 72, pp. 1–15, 2016.

[273] S. Chimalakonda and K. V. Nori, "A Software Engineering Perspective for Accelerating Educational Technologies," in *Advanced Learning Technologies (ICALT), 2012 IEEE 12th International Conference on*. IEEE, 2012, pp. 754–755.

[274] F. Ahmed and I. A. Zualkernan, "A software product line methodology for development of e-Learning system," *International Journal of Computer Science and Emerging Technologies*, vol. 2, pp. 285–295, 2011.

[275] D. L. Dalmon, L. O. Brandão, A. A. Brandão, S. Isotani *et al.*, "A domain engineering for interactive learning modules," *Journal of Research and Practice in Information Technology*, vol. 44, no. 3, p. 309, 2012.

[276] V. F. Júnior, N. F. Duarte Filho, E. A. de Oliveira Junior, and E. F. Barbosa, "Towards the Establishment of a Software Product Line for Mobile Learning Applications." in *SEKE*, 2014, pp. 678–683.

[277] I. J. WG4, "ISO/IEC 26551:2016, Software and systems engineering □ Tools and methods for product line requirements engineering," ISO/IEC, Tech. Rep., 2016.

[278] M. Eriksson, J. Börstler, and K. Borg, "The PLUSS approach–domain modeling with features, use cases and use case realizations," in *International Conference on Software Product Lines*. Springer, 2005, pp. 33–44.

[279] D. Batory, "Feature-oriented programming and the AHEAD tool suite," in *Proceedings of the 26th International Conference on Software Engineering*. IEEE Computer Society, 2004, pp. 702–703.

[280] P.-Y. Schobbens, P. Heymans, and J.-C. Trigaux, “Feature diagrams: A survey and a formal semantics,” in *14th IEEE International Requirements Engineering Conference (RE’06)*. IEEE, 2006, pp. 139–148.

[281] D. Benavides, S. Segura, and A. Ruiz-Cortés, “Automated analysis of feature models 20 years later: A literature review,” *Information Systems*, vol. 35, no. 6, pp. 615–636, 2010.

[282] K. Czarnecki, P. Grünbacher, R. Rabiser, K. Schmid, and A. Wąsowski, “Cool features and tough decisions: a comparison of variability modeling approaches,” in *Proceedings of the sixth international workshop on variability modeling of software-intensive systems*. ACM, 2012, pp. 173–182.

[283] D. Batory, “Feature models, grammars, and propositional formulas,” in *International Conference on Software Product Lines*. Springer, 2005, pp. 7–20.

[284] K. Czarnecki and A. Wasowski, “Feature diagrams and logics: There and back again,” in *Software Product Line Conference, 2007. SPLC 2007. 11th International*. IEEE, 2007, pp. 23–34.

[285] A. Classen, M. Cordy, P. Heymans, A. Legay, and P.-Y. Schobbens, “Formal semantics, modular specification, and symbolic verification of product-line behaviour,” *Science of Computer Programming*, vol. 80, pp. 416–439, 2014.

[286] M. Acher, P. Collet, P. Lahire, and R. B. France, “Familiar: A domain-specific language for large scale management of feature models,” *Science of Computer Programming*, vol. 78, no. 6, pp. 657–681, 2013.

[287] K. Czarnecki, T. Bednasch, P. Unger, and U. Eisenecker, “Generative programming for embedded software: An industrial experience report,” in *International Conference on Generative Programming and Component Engineering*. Springer, 2002, pp. 156–172.

[288] K. Czarnecki, S. Helsen, and U. Eisenecker, “Formalizing cardinality-based feature models and their specialization,” *Software process: Improvement and practice*, vol. 10, no. 1, pp. 7–29, 2005.

[289] K. Czarnecki, S. Helsen, and U. Eisenecker, “Staged configuration through specialization and multilevel configuration of feature models,” *Software Process: Improvement and Practice*, vol. 10, no. 2, pp. 143–169, 2005.

[290] K. Czarnecki, C. Hwan, P. Kim, and K. Kalleberg, “Feature models are views on ontologies,” in *10th International Software Product Line Conference (SPLC’06)*. IEEE, 2006, pp. 41–51.

[291] X. Peng, W. Zhao, Y. Xue, and Y. Wu, "Ontology-based feature modeling and application-oriented tailoring," in *International Conference on Software Reuse*. Springer, 2006, pp. 87–100.

[292] H. H. Wang, Y. F. Li, J. Sun, H. Zhang, and J. Pan, "Verifying feature models using OWL," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 5, no. 2, pp. 117–129, 2007.

[293] D. Dermeval, T. Tenório, I. I. Bittencourt, A. Silva, S. Isotani, and M. Ribeiro, "Ontology-based feature modeling: An empirical study in changing scenarios," *Expert Systems with Applications*, vol. 42, no. 11, pp. 4950–4964, 2015.

[294] M. Cordy, P.-Y. Schobbens, P. Heymans, and A. Legay, "Beyond boolean product-line model checking: dealing with feature attributes and multi-features," in *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013, pp. 472–481.

[295] T. Thüm, C. Kästner, F. Benduhn, J. Meinicke, G. Saake, and T. Leich, "FeatureIDE: An extensible framework for feature-oriented software development," *Science of Computer Programming*, vol. 79, pp. 70–85, 2014.

[296] M. Antkiewicz and K. Czarnecki, "FeaturePlugin: feature modeling plug-in for Eclipse," in *Proceedings of the 2004 OOPSLA workshop on eclipse technology eX-change*. ACM, 2004, pp. 67–72.

[297] K. Bąk, Z. Diskin, M. Antkiewicz, K. Czarnecki, and A. Wąsowski, "Clafer: unifying class and feature modeling," *Software & Systems Modeling*, pp. 1–35, 2014.

[298] D. Hernández-Leo, J. I. Asensio-Pérez, M. Derntl, L. P. Prieto, and J. Chacón, "ILDE: community environment for conceptualizing, authoring and deploying learning activities," in *European Conference on Technology Enhanced Learning*. Springer, 2014, pp. 490–493.

[299] G. Bockle, P. Clements, J. D. McGregor, D. Muthig, and K. Schmid, "Calculating ROI for software product lines," *IEEE software*, vol. 21, no. 3, pp. 23–31, 2004.

[300] P. C. Clements, J. D. McGregor, and S. G. Cohen, "The structured intuitive model for product line economics (SIMPLE)," DTIC Document, Tech. Rep., 2005.

[301] J. Krüger, "A Cost Estimation Model for the Extractive Software-Product-Line Approach," Ph.D. dissertation, Otto-von-Guericke-University Magdeburg, 2016.

Appendices

APPENDIX A

Ontologies for Instructional Design

Several ontologies for instructional design are created during this thesis. These ontologies are based on patterns and are primarily designed to support extensibility. As such, these ontologies have several placeholders for different aspects of instructional design which can be extended based on future needs. These ontologies are also designed to be modular such that they can be replaced with suitable ontologies. The core elements of the ontologies are explained in Chapter §4. In this appendix, we present a visualization of these ontologies through SOVA plugin[1] on protégé. We also made a detailed RDF/XML version of these ontologies available on Github [2].

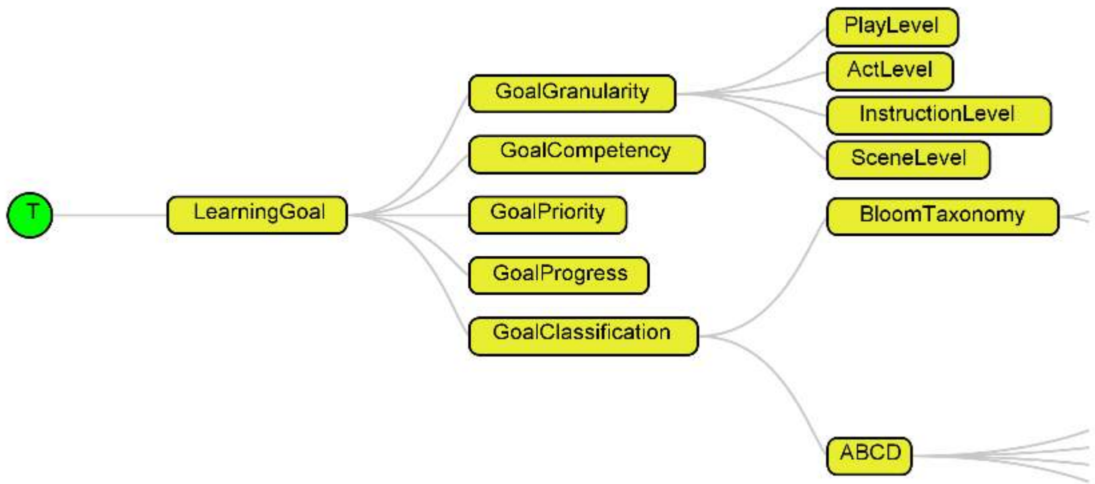
The list of ontologies we have created are as follows:

- Goals Ontology
- Process Ontology
- Content Ontology

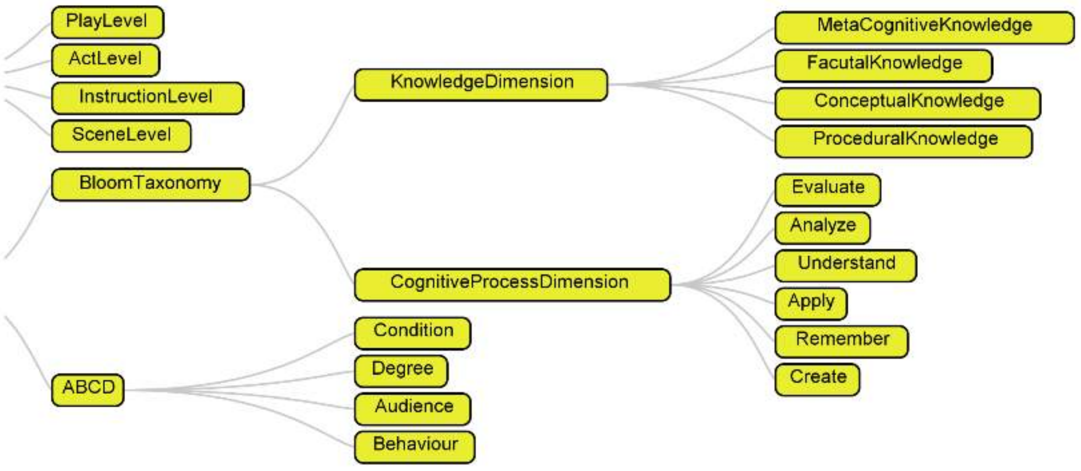
[1] *Simple Ontology Visualization, https://kask.eti.pg.gda.pl/redmine/projects/sova*
[2] *https://github.com/enthusiastic2learn/ThesisAppendices*

A.1 Goals Ontology

We described *Goals Ontology* in Section §4.5.1 along with core concepts in the ontology. Here, we provide a few screenshots of the *Goals Ontology*.



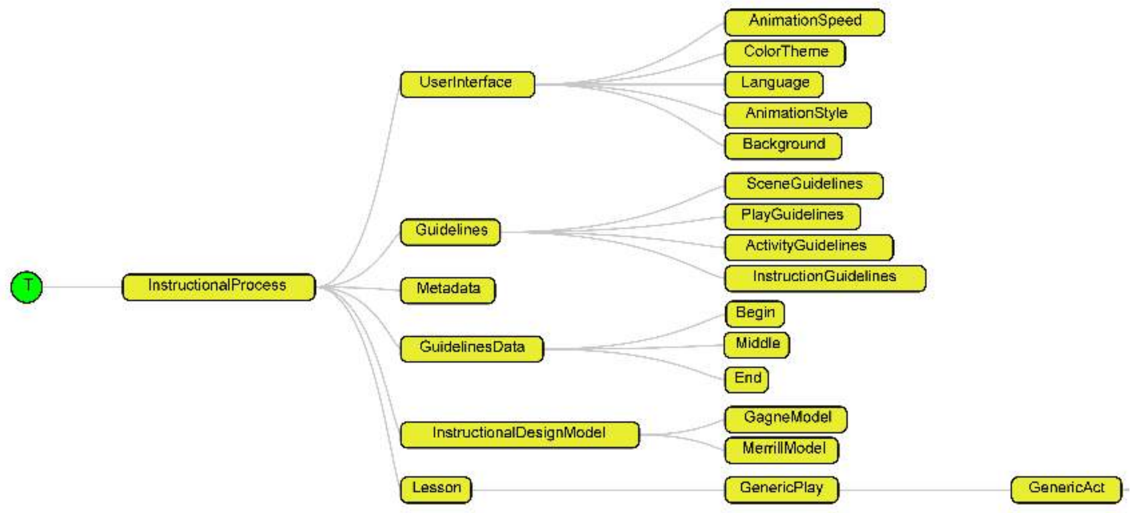
Goals Ontology - Part 1



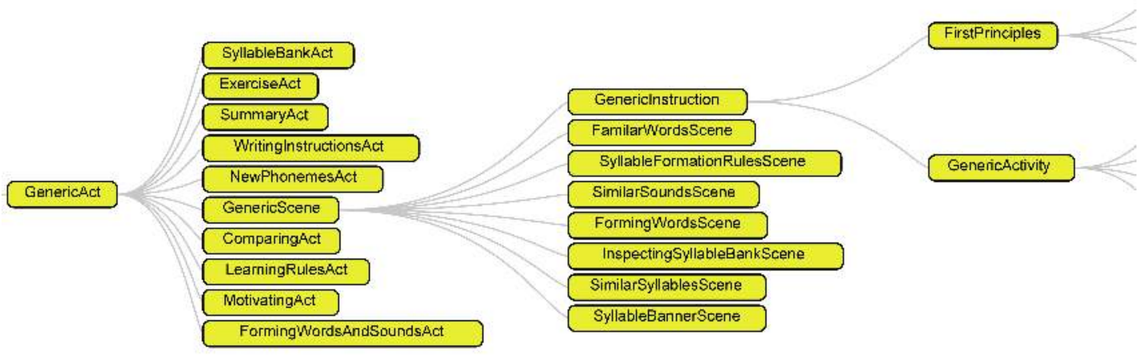
Goals Ontology - Part 2

A.2 Process Ontology

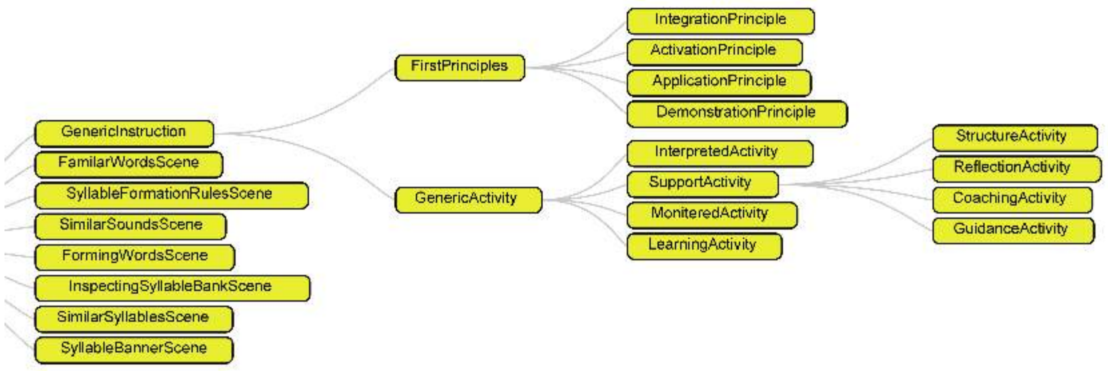
The *Process Ontology* described in Section §4.5.2 is a critical ontology for instructional design. Here, we provide an overview of *Process Ontology*.



Process Ontology - Part 1



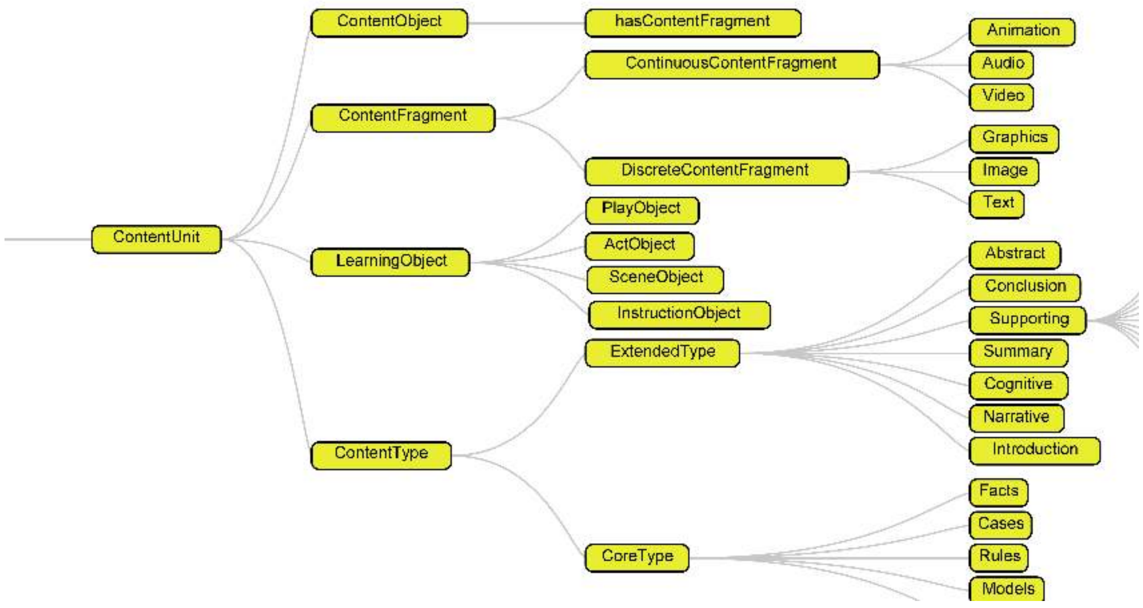
Process Ontology - Part 2



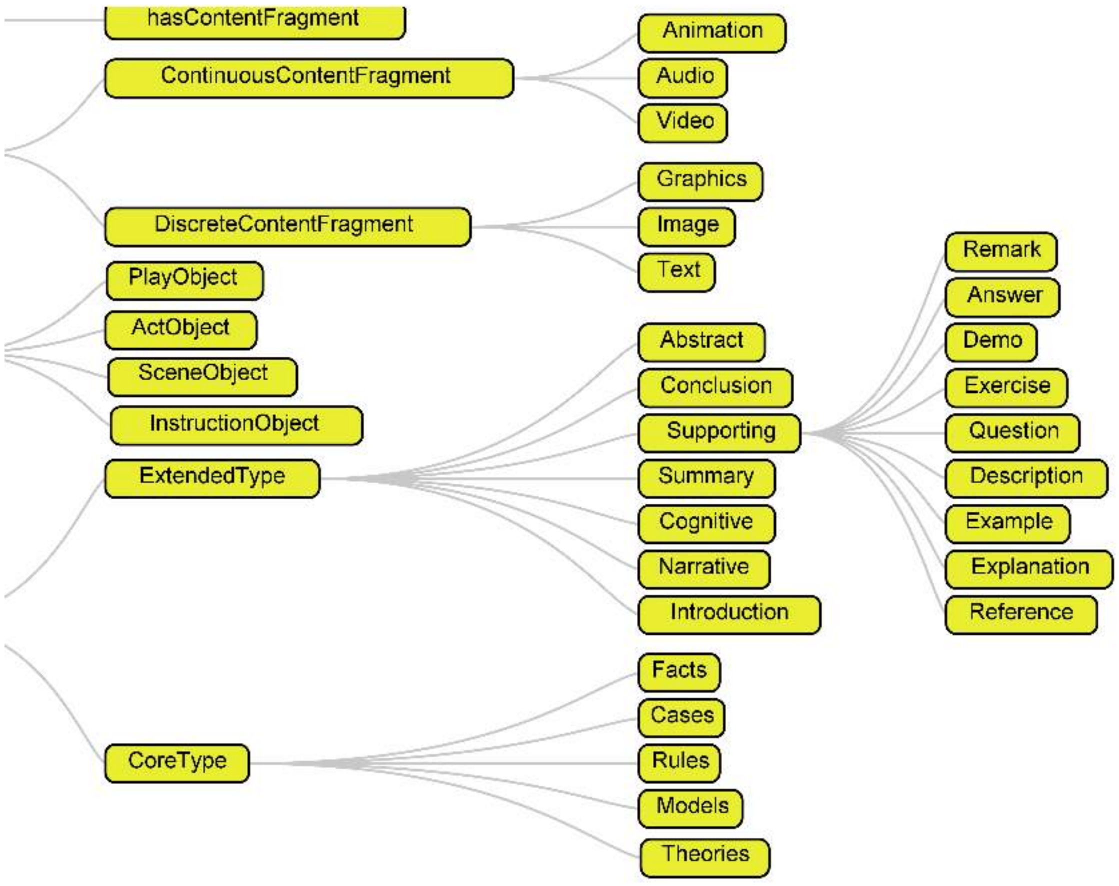
Process Ontology - Part 3

A.3 Content Ontology

The *Content Ontology* contains the list of all concepts to model different aspects of content. We detailed the *Content Ontology* in Section §4.5.3 and discussed how various concepts of the ontology emerge from *Content Pattern* discussed in Section §3.5.3. Here, we provide a visualization of the *Content Ontology*.



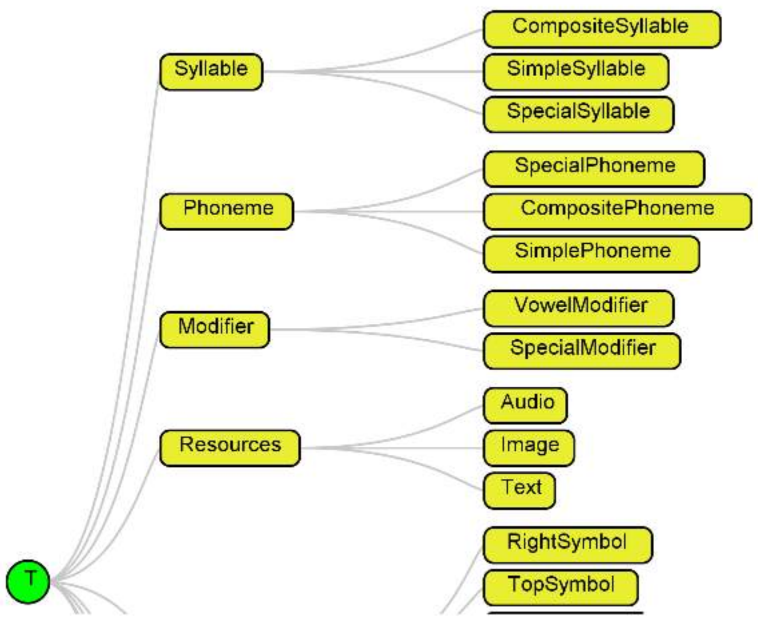
Content Ontology - Part 1



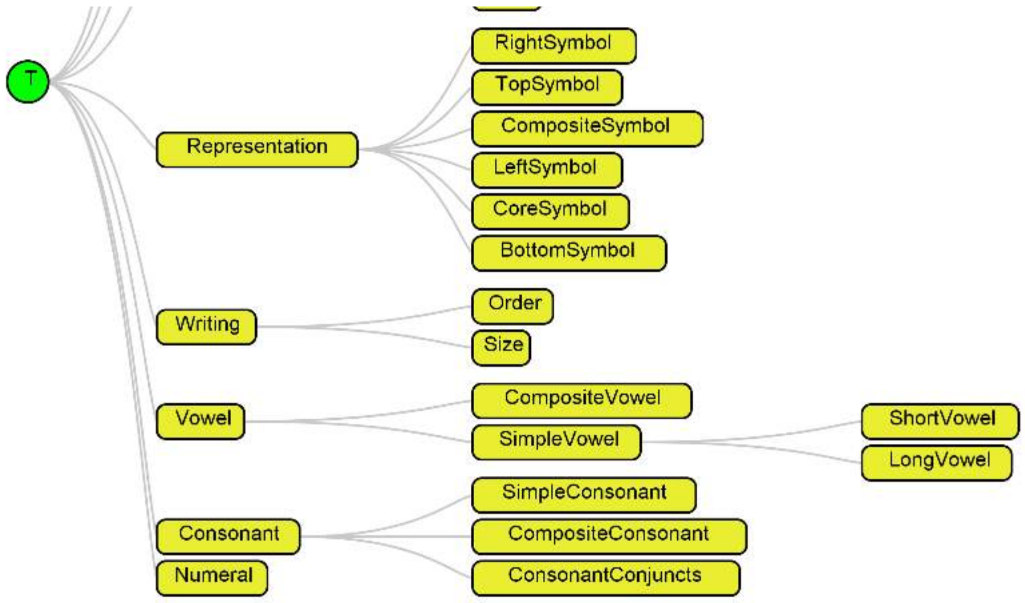
Content Ontology - Part 2

A.4 Domain Ontology for Adult Literacy

The approach and technologies in this thesis are demonstrated through adult literacy case study in India. As such *Domain Ontology* helps in modeling different aspects of domain to facilitate scale & variety. Here we present the domain ontology for adult literacy as explained in Section §4.6.



Domain Ontology - Part 1



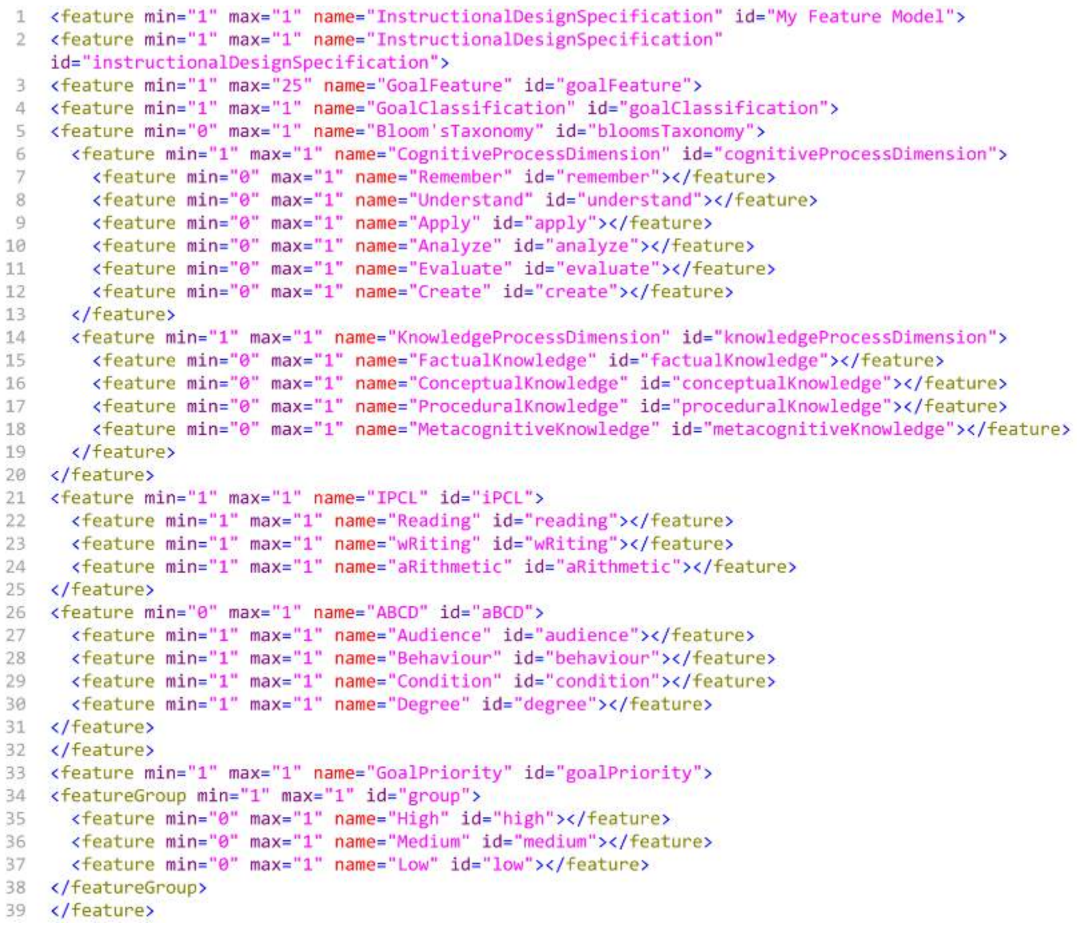
Domain Ontology - Part 2

APPENDIX B

Feature Models, Schemas & Instructional Design Instances

In this appendix, we list instructional design feature model, schema of instructional design specification introduced in Chapter §6 and two concrete instances of instructional design.

B.1 Instructional Design Feature Model



```
<feature min="1" max="1" name="InstructionalDesignSpecification" id="My Feature Model">
<feature min="1" max="1" name="InstructionalDesignSpecification"
id="instructionalDesignSpecification">
<feature min="1" max="25" name="GoalFeature" id="goalFeature">
<feature min="1" max="1" name="GoalClassification" id="goalClassification">
<feature min="0" max="1" name="Bloom'sTaxonomy" id="bloomsTaxonomy">
  <feature min="1" max="1" name="CognitiveProcessDimension" id="cognitiveProcessDimension">
    <feature min="0" max="1" name="Remember" id="remember"></feature>
    <feature min="0" max="1" name="Understand" id="understand"></feature>
    <feature min="0" max="1" name="Apply" id="apply"></feature>
    <feature min="0" max="1" name="Analyze" id="analyze"></feature>
    <feature min="0" max="1" name="Evaluate" id="evaluate"></feature>
    <feature min="0" max="1" name="Create" id="create"></feature>
  </feature>
  <feature min="1" max="1" name="KnowledgeProcessDimension" id="knowledgeProcessDimension">
    <feature min="0" max="1" name="FactualKnowledge" id="factualKnowledge"></feature>
    <feature min="0" max="1" name="ConceptualKnowledge" id="conceptualKnowledge"></feature>
    <feature min="0" max="1" name="ProceduralKnowledge" id="proceduralKnowledge"></feature>
    <feature min="0" max="1" name="MetacognitiveKnowledge" id="metacognitiveKnowledge"></feature>
  </feature>
</feature>
<feature min="1" max="1" name="IPCL" id="iPCL">
  <feature min="1" max="1" name="Reading" id="reading"></feature>
  <feature min="1" max="1" name="wRiting" id="wRiting"></feature>
  <feature min="1" max="1" name="aRithmetic" id="aRithmetic"></feature>
</feature>
<feature min="0" max="1" name="ABCD" id="aBCD">
  <feature min="1" max="1" name="Audience" id="audience"></feature>
  <feature min="1" max="1" name="Behaviour" id="behaviour"></feature>
  <feature min="1" max="1" name="Condition" id="condition"></feature>
  <feature min="1" max="1" name="Degree" id="degree"></feature>
</feature>
</feature>
<feature min="1" max="1" name="GoalPriority" id="goalPriority">
<featureGroup min="1" max="1" id="group">
  <feature min="0" max="1" name="High" id="high"></feature>
  <feature min="0" max="1" name="Medium" id="medium"></feature>
  <feature min="0" max="1" name="Low" id="low"></feature>
</featureGroup>
</feature>
```

```xml
<feature min="0" max="1" name="GoalProgress" id="goalProgress">
<feature min="0" max="1" name="PreviousGoal" id="previousGoal"></feature>
<feature min="0" max="1" name="NextGoal" id="nextGoal"></feature>
<feature min="0" max="1" name="Deadline" id="deadline"></feature>
</feature>
<feature min="1" max="1" name="GoalPrerequisities" id="goalPrerequisities">
<reference min="0" max="1" id="reference1"></reference>
</feature>
<feature min="0" max="1" name="GoalGranularity" id="goalGranularity">
<feature min="0" max="1" name="PlayLevel" id="playLevel"></feature>
<feature min="0" max="1" name="ActLevel" id="actLevel"></feature>
<feature min="0" max="1" name="SceneLevel" id="sceneLevel"></feature>
<feature min="0" max="1" name="InstructionalLevel" id="instructionalLevel"></feature>
</feature>
<reference min="0" max="1" id="reference0"></reference>
</feature>
<feature min="1" max="1" name="ProcessFeature" id="processFeature">
<feature min="1" max="1" name="InstructionalDesignModel" id="instructionalDesignModel">
<featureGroup min="1" max="1" id="group3">
  <feature min="0" max="1" name="MerrillModel" id="merrillModel">
    <feature min="1" max="1" name="FirstPrinciples" id="firstPrinciples">
      <feature min="0" max="1" name="Activation" id="activation"></feature>
      <feature min="0" max="1" name="Demonstration" id="demonstration"></feature>
      <feature min="0" max="1" name="Application" id="application"></feature>
      <feature min="0" max="1" name="Integration" id="integration"></feature>
      <feature min="0" max="1" name="Task-Oriented" id="taskOriented"></feature>
     </feature>
    <reference min="0" max="1" id="reference3"></reference>
  </feature>
  <feature min="0" max="1" name="GagneModel" id="gagneModel">
    <feature min="0" max="1" name="GainAttention" id="gainAttention"></feature>
    <feature min="0" max="1" name="InformObjectives" id="informObjectives"></feature>
    <feature min="0" max="1" name="StimulatePriorLearning" id="stimulatePriorLearning"></feature>
    <feature min="0" max="1" name="PresentContent" id="presentContent"></feature>
    <feature min="0" max="1" name="ProvideGuidance" id="provideGuidance"></feature>
    <feature min="0" max="1" name="ElicitPerformance" id="elicitPerformance"></feature>
    <feature min="0" max="1" name="ProvideFeedback" id="provideFeedback"></feature>
    <feature min="0" max="1" name="AssessPerformance" id="assessPerformance"></feature>
    <feature min="0" max="1" name="SupportInternalization" id="supportInternalization"></feature>
    <reference min="0" max="1" id="reference2"></reference>
  </feature>
</featureGroup>
<feature min="0" max="1" name="GenericActivity" id="genericActivity">
  <feature min="0" max="1" name="LearningActivity" id="learningActivity"></feature>
  <feature min="0" max="1" name="InterpretedActivity" id="interpretedActivity"></feature>
  <feature min="0" max="1" name="MonitredActivity" id="monitredActivity"> </feature>
  <feature min="0" max="1" name="SupportActivity" id="supportActivity">
    <feature min="0" max="1" name="StructureActivity" id="structureActivity"></feature>
    <feature min="0" max="1" name="GuidanceActivity" id="guidanceActivity"></feature>
    <feature min="0" max="1" name="CoachingActivity" id="coachingActivity"></feature>
    <feature min="0" max="1" name="ReflectionActivity" id="reflectionActivity"></feature>
  </feature>
</feature>
</feature>
<feature min="1" max="25" name="Lesson" id="lesson">
<feature min="1" max="25" name="GenericPlay" id="genericPlay">
  <feature min="1" max="25" name="GenericAct" id="genericAct">
    <feature min="0" max="1" name="MotivatingAct" id="motivatingAct"></feature>
    <feature min="0" max="1" name="SyllableBankAct" id="syllableBankAct">
    </feature>
    <feature min="0" max="1" name="NewPhonemesAct" id="newPhonemesAct"></feature>
    <feature min="0" max="1" name="ComparingAct" id="comparingAct"></feature>
    <feature min="0" max="1" name="FormingWordsAct" id="formingWordsAndSoundsAct"></feature>
    <feature min="0" max="1" name="FormingSoundsAct" id="formingSoundsAct"></feature>
    <feature min="0" max="1" name="LearningRulesAct" id="learningRulesAct"></feature>
    <feature min="0" max="1" name="WritingInstructionsAct" id="writingInstructionsAct"></feature>
    <feature min="0" max="1" name="ExerciseAct" id="exerciseAct"></feature>
    <feature min="0" max="1" name="SummaryAct" id="summaryAct"></feature>
    <feature min="1" max="25" name="GenericScene" id="genericScene">
      <feature min="0" max="1" name="FamiliarWordsScene" id="familiarWordsScene"></feature>
      <feature min="0" max="1" name="FormingWordsScene" id="formingWordsScene"></feature>
      <feature min="0" max="1" name="InspectingSyllableBankScene" id="inspectingSyllableBankScene"></feature>
      <feature min="0" max="1" name="SimialrSyllablesScene" id="simialrSyllablesScene"> </feature>
      <feature min="0" max="1" name="SyllableBannerScene" id="syllableBannerScene"></feature>
      <feature min="0" max="1" name="SyllableFormationRulesScene" id="syllableFormationRulesScene"></feature>
      <feature min="1" max="1" name="GenericInstruction" id="genericInstruction">
        <reference min="0" max="1" id="reference4"></reference>
      </feature></feature>
  </feature>
</feature>
</feature>
</feature>
<feature min="1" max="1" name="ContentFeature" id="contentFeature">
<feature min="1" max="25" name="ContentUnit" id="contentUnit">
<feature min="0" max="1" name="ContentFragment" id="contentFragment">
</feature>
<feature min="0" max="1" name="ContentType" id="contentType">
  <feature min="1" max="1" name="CoreType" id="coreType">
    <feature min="0" max="1" name="Facts" id="facts"></feature>
    <feature min="0" max="1" name="Cases" id="cases"></feature>
    <feature min="0" max="1" name="Rules" id="rules"></feature>
    <feature min="0" max="1" name="Models" id="models"></feature>
    <feature min="0" max="1" name="Theories" id="theories"></feature>
  </feature>
  <feature min="0" max="1" name="ExtendedType" id="extendedType">
    <feature min="0" max="1" name="Cognitive" id="cognitive"></feature>
    <feature min="0" max="1" name="Narrative" id="narrative"></feature>
    <feature min="0" max="1" name="Abstract" id="abstract"></feature>
    <feature min="0" max="1" name="Introduction" id="introduction"></feature>
    <feature min="0" max="1" name="Conclusion" id="conclusion"></feature>
    <feature min="0" max="1" name="Summary" id="summary"></feature>
    <feature min="0" max="1" name="Supporting" id="supporting">
    <feature min="0" max="1" name="Answer" id="answer"></feature>
      <feature min="0" max="1" name="Demo" id="demo"></feature>
      <feature min="0" max="1" name="Description" id="description"></feature>
      <feature min="0" max="1" name="Example" id="example"></feature>
      <feature min="0" max="1" name="Exercise" id="exercise"></feature>
      <feature min="0" max="1" name="Explanation" id="explanation"></feature>
      <feature min="0" max="1" name="Question" id="question"></feature>
      <feature min="0" max="1" name="Reference" id="reference"></feature>
      <feature min="0" max="1" name="Remark" id="remark"></feature>
      </feature>
  </feature>
</feature>
<feature min="0" max="1" name="LearningObject" id="learningObject"></feature>
<reference min="0" max="1" id="reference5"></reference>
</feature>
</feature>
<feature min="1" max="1" name="UserInterface" id="userInterface">
<feature min="0" max="1" name="Language" id="languge"></feature>
<feature min="0" max="1" name="ColorTheme" id="colorTheme"></feature>
<feature min="0" max="1" name="AnimationStyle" id="animationStyle"></feature>
<feature min="0" max="1" name="AnimationSpeed" id="animationSpeed"></feature>
<feature min="0" max="1" name="Background" id="background"></feature>
</feature>
<feature min="1" max="1" name="RawData" id="rawData">
<feature min="0" max="1" name="UnicodeText" id="text"></feature>
<feature min="0" max="1" name="ImageURL" id="imageURL"></feature>
<feature min="0" max="1" name="AudioURL" id="audioURL">
<feature min="1" max="1" name="TimeRange" id="timeRange">
  <feature min="1" max="1" name="Minimum" id="minimum"></feature>
  <feature min="1" max="1" name="Maxmimum" id="maxmimum"></feature>
</feature>
</feature>
<reference min="0" max="1" id="reference6">
</reference>
</feature>
</feature>
```

B.2 Instructional Design Specification Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
<xs:element name="design">
<xs:complexType>
<xs:sequence>
    <xs:element name="language" type="xs:string"></xs:element>
    <xs:element name="backgroundimage" type="xs:string"></xs:element>
    <xs:element name="backgroundcolor" type="xs:string"></xs:element>
    <xs:element name="colortheme" type="xs:string"></xs:element>
    <xs:element name="animationstyle" type="xs:string"></xs:element>
    <xs:element name="animationspeed" type="xs:string"></xs:element>
    <xs:element name="sizeofletters" type="xs:string"></xs:element>
    <xs:element name="menu">
<xs:complexType>
<xs:sequence>
    <xs:element name="title" type="xs:string"></xs:element>
    <xs:element name="backgroundimage" type="xs:string"></xs:element>
    <xs:element name="backgroundcolor" type="xs:string"></xs:element>
    <xs:element name="colortheme" type="xs:string"></xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="imagefornextbutton" type="xs:string"></xs:element>
<xs:element name="imageforpreviousbutton" type="xs:string"></xs:element>
<xs:element name="imageformenubutton"></xs:element>
<xs:element name="imageforhandandstick" type="xs:string"></xs:element>
<xs:element name="lesson">
<xs:complexType>
<xs:sequence>
<xs:element name="title" type="xs:string"></xs:element>
<xs:element name="instructions">
   <xs:complexType>
         <xs:sequence>
               <xs:element name="start">
                     <xs:complexType>
                           <xs:sequence>
                                 <xs:element name="sound" type="xs:string"></xs:element>
                                 <xs:element name="image" type="xs:string"></xs:element>
                           </xs:sequence>
                     </xs:complexType>
               </xs:element>
               <xs:element name="middle">
                     <xs:complexType>
                           <xs:sequence>
                                 <xs:element name="text" type="xs:string"></xs:element>
                                 <xs:element name="sound" type="xs:string"></xs:element>
                                 <xs:element name="image" type="xs:string"></xs:element>
                           </xs:sequence>
                     </xs:complexType>
               </xs:element>
               <xs:element name="end">
                     <xs:complexType>
                           <xs:sequence>
                                 <xs:element name="sound" type="xs:string"></xs:element>
                                 <xs:element name="image" type="xs:string"></xs:element>
                           </xs:sequence>
                     </xs:complexType>
               </xs:element>
         </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:element name="goals">
   <xs:complexType>
         <xs:sequence>
               <xs:element name="goal" maxOccurs="unbounded">
                     <xs:complexType>
                           <xs:sequence>
                                 <xs:element name="text" type="xs:string"></xs:element>
                                 <xs:element name="sound"></xs:element>
                           </xs:sequence>
                     </xs:complexType>
               </xs:element>
         </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:element name="letters">
<xs:complexType>
<xs:sequence>
<xs:element name="letter" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
     <xs:element name="text" type="xs:string"></xs:element>
     <xs:element name="sound" type="xs:string"></xs:element>
     <xs:element name="instructions">
           <xs:complexType>
                 <xs:sequence>
                       <xs:element name="start">
                             <xs:complexType>
                               <xs:sequence>
                                     <xs:element name="sound" type="xs:string"></xs:element>
                                     <xs:element name="image" type="xs:string"></xs:element>
                               </xs:sequence>
                             </xs:complexType>
                       </xs:element>
                       <xs:element name="middle">
                             <xs:complexType>
                                   <xs:sequence>
                                         <xs:element name="sound"></xs:element>
                                         <xs:element name="image"></xs:element>
                                   </xs:sequence>
                             </xs:complexType>
                       </xs:element>
                       <xs:element name="end">
                             <xs:complexType>
                                   <xs:sequence>
                                         <xs:element name="sound"></xs:element>
                                         <xs:element name="image"></xs:element>
                                   </xs:sequence>
                             </xs:complexType>
                       </xs:element>
                 </xs:sequence>
           </xs:complexType>
     </xs:element>
     <xs:element name="words">
           <xs:complexType>
             <xs:sequence>
                   <xs:element name="word">
                         <xs:complexType>
                               <xs:sequence>
                                     <xs:element name="text" type="xs:string"></xs:element>
                                     <xs:element name="sound" type="xs:string"></xs:element>
                                     <xs:element name="image" type="xs:string"></xs:element>
                               </xs:sequence>
                         </xs:complexType>
                   </xs:element>
             </xs:sequence>
       </xs:complexType>
     </xs:element>
</xs:sequence></xs:complexType>
</xs:element> </xs:sequence> </xs:complexType>
</xs:element> </xs:sequence> </xs:complexType>
</xs:element> </xs:sequence> </xs:complexType>
</xs:element>
</xs:schema>
```

B.3 Instructional Design Instance for Hindi Language

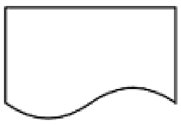
```
<?xml version='1.0' encoding='UTF-8' ?>
<design>
    <language>Hindi</language>
    <backgroundimage></backgroundimage>
    <backgroundcolor>white</backgroundcolor>
    <colortheme>{blue, red}</colortheme>
    <animationstyle>fallingletters</animationstyle>
    <animationspeed>slow</animationspeed>
    <sizeofletters>large</sizeofletters>
    <menu>
        <title>पाठ 1</title>
        <backgroundimage>menu.jpg</backgroundimage>
        <backgroundcolor>black</backgroundcolor>
        <colortheme>white</colortheme>
    </menu>
    <imagefornextbutton>right.png</imagefornextbutton>
    <imageforpreviousbutton>left.png</imageforpreviousbutton>
    <imageformenubutton></imageformenubutton>
    <imageforhandandstick>handandstick.png</imageforhandandstick>
    <lesson>
        <title>मन का काम</title>
        <instructions>
            <start>
                <sound>sounds/hsounds/instruction1.wav</sound>
                <image>./handandstick.png</image>
            </start>
        </instructions>
        <goals>
            <goal>
                <text>इस पाठ में हमारा लक्ष्य {म,  न}  को पढना</text>
                <sound></sound>
            </goal>
            <goal>
                <text>इस पाठ में हमारा लक्ष्य {म,  न}  को पहचानना</text>
                <sound></sound>
            </goal>
        </goals>
        <letters>
            <letter>
                <text>म</text>
                <sound>sounds/hsounds/ma.wav</sound>
                <words>
                    <word>
                        <text>मन</text>
                        <sound>sounds/hsounds/makaan.wav</sound>
                        <image>./house.jpg</image>
                    </word>
                </words>
            </letter>
            <letter>
                <text>न</text>
                <sound>sounds/hsounds/na.wav</sound>
                <words>
                    <word>
                        <text>नम</text>
                        <sound>sounds/hsounds/nam.mp3</sound>
                    </word>
                    <word>
                        <text>मन</text>
                        <sound>sounds/hsounds/man.mp3</sound>
                    </word>
                    <word>
                        <text>नाम</text>
                        <sound>sounds/naam.wav</sound>
                        <image>./naam.png</image>
                    </word>
                </words>
            </letter>
                <letter>
                <text>क</text>
                <sound>sounds/hsounds/ka.wav</sound>
                <words>
                    <word>
                        <text>कम</text>
                        <sound>sounds/kam.wav</sound>
                    </word>
                    <word>
                        <text>कनक</text>
                        <sound>sounds/kanak.wav</sound>
                    </word>
                    <word>
                        <text>कथा</text>
                        <sound>sounds/katha.wav</sound>
                        <image>./katha.png</image>
                    </word>
                </words>
            </letter>
    </letters>
    </lesson>
</design>
```

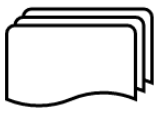
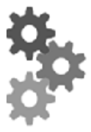
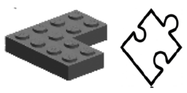
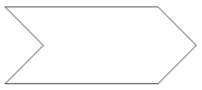
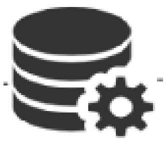
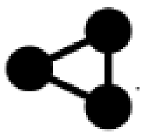
B.4 Instructional Design Instance for Telugu Language

```
<?xml version='1.0' encoding='UTF-8' ?>
<design>
    <language>Telugu</language>
    <backgroundimage>back.png</backgroundimage>
    <backgroundcolor>orange</backgroundcolor>
    <colortheme>{black, red}</colortheme>
    <animationstyle>fallingletters</animationstyle>
    <animationspeed>slow</animationspeed>
    <sizeofletters>large</sizeofletters>
    <menu>
        <title>మొదటి పాఠము </title>
        <backgroundimage>menu.jpg</backgroundimage>
        <backgroundcolor>black</backgroundcolor>
        <colortheme>white</colortheme>
    </menu>
    <imagefornextbutton>right.png</imagefornextbutton>
    <imageforpreviousbutton>left.png</imageforpreviousbutton>
    <imageformenubutton></imageformenubutton>
    <imageforhandandstick>handandstick.png</imageforhandandstick>
    <lesson>
        <title>
            అవసరాలు
        </title>
        <instructions>
            <start>
                <text></text>
                <sound>sounds/tsounds/instruction1.wav</sound>
                <image>./namaskar.png</image>
            </start>
        </instructions>
        <goals>
            <goal>
                <text>ఈ పాఠములో మనము ఈ అక్షరాలను గుర్తుపెట్టుకోవటం నేర్చుకుంటాము {
                }</text>
                <sound>sounds/goal.wav</sound>
            </goal>
         </goals>
        <letters>
            <letter>
                <text>అ</text>
                <sound>sounds/tsounds/a.wav</sound>
                <instructions>
                    <middle>
                        <sound>sounds/tsounds/instruction2.wav</sound>
                        <image>./namaskar.png</image>
                    </middle>
                 </instructions>
                <words>
                    <word>
                        <text>అర</text>
                        <sound>sounds/word.wav</sound>
                        <image>./ara.jpg</image>
                    </word>
               </words>
            </letter>
            <letter>
                <text>వ</text>
                <sound>sounds/va.wav</sound>
                 <words>
                    <word>
                        <text>వర</text>
                        <sound>sounds/vara.wav</sound>
                    </word>
                    <word>
                        <text>అరువు</text>
                        <sound>sounds/aravu.wav</sound>
                    </word>
            </words>
            </letter>
                <letter>
                <text>స</text>
                <sound>sounds/sa.wav</sound>
                <instructions>
                 </instructions>
                <words>
                    <word>
                        <text>నస</text>
                        <sound>sounds/nasa.wav</sound>
                    </word>
                    <word>
                        <text>వరస</text>
                        <sound>sounds/varasa.wav</sound>
                    </word>
                    <word>
                        <text>ముసురు</text>
                        <sound>sounds/musur.wav</sound>
                        <image>./musuru.png</image>
                    </word>
                </words>
            </letter>
                <letter>
                <text>ల</text>
                <sound>sounds/la.wav</sound>
                <words>
                    <word>
                        <text>వల</text>
                        <sound>sounds/vala.wav</sound>
                    </word>
                    <word>
                        <text>అసలు</text>
                        <sound>sounds/asalu.wav</sound>
                    </word>
                    <word>
                        <text>వలస</text>
                        <sound>sounds/valasa.wav</sound>
                    </word>
                </words>
            </letter>
    </letters>
    </lesson>
</design>
```

APPENDIX C

Notation

Symbol	Description
	Software development teams
	Primer i.e., aa book for teaching literacy and any other instructional materials
	*i*Primer i.e, eLearning System as defined in key definitions
	To represent documents like instructional design specification and *i*Primer

	To represent multiple documents such as instructional design specifications and *i*Primers
	To represent automation processes and tools such as generating instructional design editors and *i*Primers
	We use several lego blocks to represent an artifact that can be composed to create product variants. Lego blocks with varied colours are used standard notation in software product lines
	To show a specific task in a process mainly for progression. We used this in ontology development process and in *Content Pattern*
	We used this icon to represent a *configurable repository* and *database*. For example, we used this icon to represent patterns repository and assets repository as part of the core asset base
	This icon represents an ontology in this thesis. This is used to represent ontologies such as *Goals*, *Process*, *Content* and so on

	Domain ontology is represented using this icon
	The complete instructional design ontology is represented using this icon. This symbol is also used for representing a repository of ontologies
	These icons represent an editor and specifically instructional design editors
	Several instructional design instances are created during this thesis and we use these symbols to represent multiple instructional design instances
	This icon is used to represent an eLearning System specifically to emphasize multimedia elements present in the *i*Primer
	To represent APIs for ontologies and XML

"Om!, this syllable is this whole world", Mandukya Upanishad

Everything in this universe is a *pattern*

A prolific and complex *pattern* is the recitation of *Vedas* and *Sanskrit* mantras using *ghana-pāṭha* (literally "dense recitation"), in which words are split and repeated back and forth in a particular order to alleviate inner energies.

word1word2, word2word1, word1word2word3, word3word2word1, word1word2word3; word2word3, word3word2, word2word3word4... - Unknown

"…knowledge is of two types - empirical and conceptual. Empirical knowledge can be taught, described and discussed. Conceptual axiomatic knowledge cannot" - Kena Upanishad

"Sir, what is that through which, if it is known, everything else becomes known?" Mundaka Upanishad

As by knowing one lump of clay, dear one,
We come to know all things made out of clay –
That they differ only in name and form,
While the stuff of which all are made is clay;

As by knowing one gold nugget, dear one,
We come to know all things made out of gold –
That they differ only in name and form,
While the stuff of which all are made is gold;

As by knowing one tool of iron, dear one,
We come to know all things made out of iron –
That they differ only in name and form,
While the stuff of which all are made is iron –

Chandogya Upanishad

Patterns, Ontologies and Software Product Lines from the world of Epics, Vedas and Upanishads – Special thanks to my dad for inspiration