Computing (2020) 102:1765–1792
https://doi.org/10.1007/s00607-019-00772-x

A family of software product lines in educational technologies

Sridhar Chimalakonda[1] · **Kesav V. Nori**[2]

Received: 12 December 2018 / Accepted: 28 November 2019 / Published online: 2 January 2020

Abstract

Rapid advances in education domain demand the design and customization of educational technologies for a large *scale* and *variety* of evolving requirements. Here, *scale* is the number of systems to be developed and *variety* stems from a diversified range of instructional designs such as varied goals, processes, content, teaching styles, learning styles and, also for eLearning Systems for 22 Indian Languages and variants. In this paper, we present a family of software product lines as an approach to address this challenge of modeling a family of instructional designs as well as a family of *e*Learning Systems and demonstrate it for the case of adult literacy in India (287 million learners). We present a multi-level product line that connects product lines at multiple levels of granularity in education domain. We then detail two concrete product lines (http://rice.iiit.ac.in), one that generates instructional design editors and two, which generates a family of *e*Learning Systems based on flexible instructional designs. Finally, we demonstrate our approach by generating *e*Learning Systems for Hindi and Telugu languages, which led to significant cost savings of 29 person-months for 9 *e*Learning Systems.

Keywords Software product lines · Educational technologies · Instructional design · eLearning Systems · Adult literacy

Mathematics Subject Classification 68N19 · 68N30 · 68U35 · 68T30 · 97B60

A part of this work was done during the first author's doctoral thesis at IIIT Hyderabad, India [1].

✉ Sridhar Chimalakonda
ch@iittp.ac.in

Kesav V. Nori
kesav.nori@gmail.com

[1] Department of Computer Science and Engineering, Indian Institute of Technology Tirupati, Tirupati, India

[2] Software Engineering Research Center, International Institute of Information Technology Hyderabad, Hyderabad, India

1 Motivation

The role of technology in education has undergone a massive transformation in the 21st century promising to facilitate anywhere, anytime learning to everyone [2,3]. There is also a dramatic rise in the use and design of a variety of technologies in education in the last decade or so [4–6]. With the goal of improving education, several technologies such as computer-assisted instruction [7], web-based learning [8], game-based learning [5], computer-supported collaborative learning [9], virtual learning environments [6], gesture-based learning [10] 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 cutting across the spectrum of technologies. In addition, there is also a need to constantly improve these technologies catering to emerging requirements of facilitating personalized learning [11], enhancing learner interaction experience through augmented reality [12] and gamification [13]. This further increases the complexity during the design of educational technologies and makes it an incredibly hard challenge to *customize* the *software* aspects of these technologies as per the emerging requirements.

On the other hand, there is also severe criticism on technologies for education such as huge upfront costs, lack of evidence to show positive impact on quality of education and so on [14,15]. 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 [14,16].

In essence, this scenario poses grand challenges for engineering and computing such as (i) *Advance personalized learning*[1] and (ii) *Provide a teacher for every student*.[2] Addressing these challenges requires research from multiple disciplines such as learning and social sciences, computer science, human-computer interaction and so on.

In this paper, we make an attempt to facilitate *design and customization of educational technologies*[3] *for scale and variety*, *where scale is the number of systems to be developed and variety stems from variations in different aspects of teaching and learning.*

This challenge is further exacerbated in the Indian context owing to the need to design and customize educational technologies for 22 Indian Languages and variants for a wide range of learners, teachers and in varied contexts. Essentially, any technology-based solutions should be (i) based on a strong pedagogical foundation (ii) available for multiple Indian languages and (iii) flexible to cover varied instructional designs. Most importantly, the primary concern is in terms of the effort required for creating and maintaining educational technologies for the *scale* and *variety* inherent for education in India. To this end, the core contribution of this paper is:

[1] *NAE Grand Challenges for Engineering,* https://goo.gl/U8FpKv.

[2] *Grand Research Challenges in Information Systems,* https://goo.gl/ciDVa1.

[3] 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."

A multi-level software product line approach to the design and customization of educational technologies for scale and variety in the domain of education and specifically for adult literacy in India.

Adult literacy is one of the grand challenges of India with presence of around 37% of 775 million young people and adults spread across 22 Indian official languages, who are beyond the age of schooling, speak their language, but cannot read or write [17]. The National Literacy Mission (NLM) of Government of India (GoI) has been striving to address this challenge since 1988 and has created a uniform methodology called Improved Pace and Content of Learning (IPCL)[4] for teaching adult illiterates across India [18]. Despite several technological attempts to address this problem [19,20], the following key requirements and challenges exist, setting the context for this paper:

- Facilitate the design of *e*Learning Systems for 22 Indian Languages *(scale)*.
- Facilitate the design of these *e*Learning Systems for flexible instructional designs (*varying* goals, processes and content)
- Facilitate the development of *instructional design editors* for creation of customizable instructional designs.

Even though these challenges are specific to adult literacy, design of *e*Learning Systems for other forms of education such as schooling, vocational skills, engineering; customizing them for varied contexts and delivering them in multiple languages is also similar and makes it a grand challenge. With this background, the research goal in the context of adult literacy is: *How to facilitate design and customization of eLearning Systems*[5] *to teach 287 million adult illiterates in India spread across 22 Indian Languages?"*

2 Related work

Explicit modeling of instructional design and its variants is a fundamental aspect to facilitate *scale* and *variety* inherent in the problem domain. In this paper, we consider instructional design as an underlying structure, that encompasses principles of instruction facilitate the design of educational technologies. There has been extensive research on modeling instructional design for the last several years resulting in a plethora of educational modeling languages (EMLs) [21] 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 [22]. IMS-LD emerged as a standard for learning design [23] and then focus shifted to tools such as LAMS [24] and LDSE [25] that aimed to support teachers and eventually towards an integrated learning design environment [26].

Despite this rapid progress, many researchers have pointed to several shortcomings of modeling and reusing instructional design such as complexity of authoring, lack of adequate tool support, interoperability and inability to support teachers [27]. Further,

[4] IPCL is a uniform learning methodology based on eclectic method and is used as the base for producing instructional material for all 22 Indian languages [18].

[5] We consider "eLearning Systems as a sub-class of educational technologies that are designed for improving learning and teaching in a particular context."

ontologies are used to represent different aspects of instructional design [28] and learning design using IMS-LD [29]. LOCO ontology was proposed to bridge the gap between learning objects and learning designs through context [30]. Bansal et al. have proposed a prototype ontology IMOD-Ont focusing on instructional modules [31]. 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 *e*Learning Systems for *scale* and *variety*.

Development of software components for the domain of education started way back in 1999 [32]. But the use of software engineering approaches in educational technologies has garnered significant attention with the advent of reuse of learning objects [33]. Designing reusable learning objects was extensively studied by several researchers [33–35] in educational technologies, but most of these efforts have not been fruitful due to lack of emphasis on critical aspects of instructional design [36–38]. Design principles from software engineering were borrowed to facilitate reuse of learning objects [39]. However, this emphasis itself has led to severe criticism on software engineering being misused in the context of learning objects from a learning perspective [36]. Dodero et al. proposed a model-driven approach consisting of a domain-specific language and a tool to facilitate modeling of learning designs [40] and a similar approach is used in LPCEL Editor [41]. Open Educational Resources has gained significant attention from academia and practitioners in the last decade or so, and several challenges as well [42,43]. But despite the advantages of the generative approaches, it was noted that the complexity of authoring process increases because of model development required from domain experts [44].

One area of work that is directly relevant to this paper is called as Software Product Lines (SPLs) that facilitates systematic reuse across a family of systems [45]. 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 [46]. Krueger has suggested three ways of adopting SPLs [47] (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 [45,48,49] including modeling and configuring variability using ontologies [50,51].

On the other hand, despite the success of SPL in several domains [52], there is sparse research on applying SPL in the domain of technology enhanced learning [53]. Pankratius has proposed PLANT as a product line based approach for creation and maintenance of digital information products [54]. In our prior work, we proposed TALES as an approach for automating the development of *e*Learning Systems [55]. A software product line methodology for development of e-learning system for a six sigma course was proposed in [56]. A domain engineering activity for interactive learning modules and a product line for mobile learning applications are proposed in [57] and [58]. A software product line for m-learning focusing on programming is discussed in [59] and a software product line for games is presented in [60]. Recently, researchers have expressed e-learning processes using software product lines [61].

However, none of these approaches consider instructional design domain as the basis and do not focus on *scale* and *variety* inherent in the problem domain of education. 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.

To summarize, existing approaches in the literature focus on either modeling instructional design from learning perspective or on software reuse and not both presenting a strong motivation and need for our research and approach.

3 High-level overview of proposed approach

Design of educational technologies for *scale* and *variety* while maintaining quality requires research from several disciplines. For the last decade or so, we focused on creating several technological aids to support education in India with our research spanning across educational technologies [62,63], software engineering [64] and human computer interaction [65]. Specifically, during the early stages of our research, we proposed an approach called TALES for reducing effort during creation of adult literacy *e*Learning Systems [53,55], but the focus was limited to variations in educational content rather than instructional design [1]. We also focused on modeling instructional design patterns as a basis for design of educational technologies [63]. We have also presented the challenges of applying software product lines to educational technologies [64]. A summary of our prior work is presented in [66].

On the other hand, the focus of this paper is on designing and applying software engineering approaches and principles to accelerate the design of educational technologies for *scale* and *variety* based on well-established learning methodologies and demonstrate it in the case of adult literacy. To this end, we rely on the following domain inputs (i) a strong pedagogical basis (ii) instructional material from domain experts and (iii) field-tested *e*Learning Systems.

Figure 1 shows a simple schematic of the existing approach [top] and proposed approach [bottom] for design of *i*Primers[6] 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* [67], concretely represent them using *ontologies* [68] and then apply a *software product lines* approach for semi-automatically generating *e*Learning Systems for varied instructional designs and multiple languages. The key difference is that the proposed approach can handle the *scale* and *variety* for flexible instructional designs instead of re-developing *e*Learning Systems for every new case and change in the inputs.

In the next section, we motivate the need for software product lines in the context of educational technologies.

[6] We use the term *i*Primers to mean *e*Learning Systems.

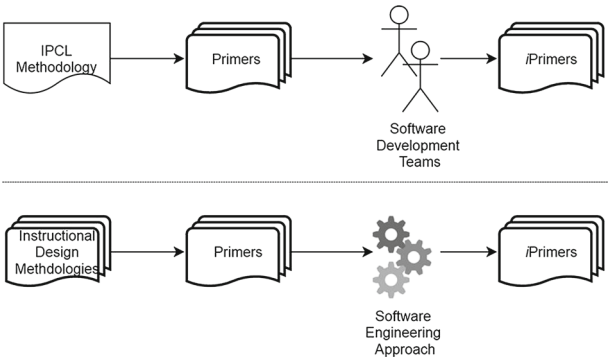


Fig. 1 Existing and proposed approach for design of educational technologies

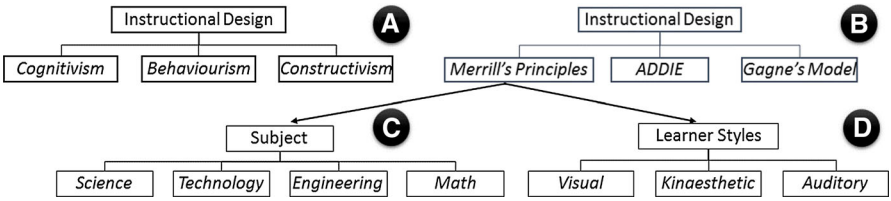


Fig. 2 Few sample instructional design families

4 Families in educational technologies

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"* [69]. A family of systems can be at different levels of granularity with multiple sub-families within families and also a hierarchy of families. In this paper, we are interested in three kinds of families.

- *Instructional Design as families* - We can classify instructional design in the form of several families and at different levels of granularity as shown in Fig. 2. For example, we can consider the three fundamental ways of *cognitivism*, *behaviourism* and *constructivism* or we can group them based on models such as *Gagne's model* [70], *Merrill's First Principles of Instruction* [71]. 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. 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 [71] and there can be several variants based on these principles giving a family of instructional designs. We use the pattern categories identified in [67] as base for modeling instructional designs and variants. However, considering the broad spectrum of all instructional designs in the literature is out of scope of this paper.

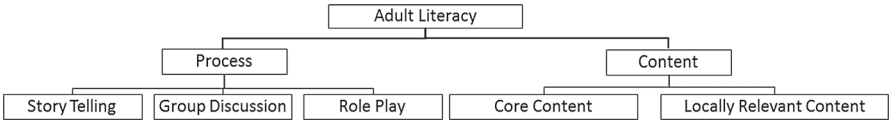


Fig. 3 Few sample families in adult literacy domain

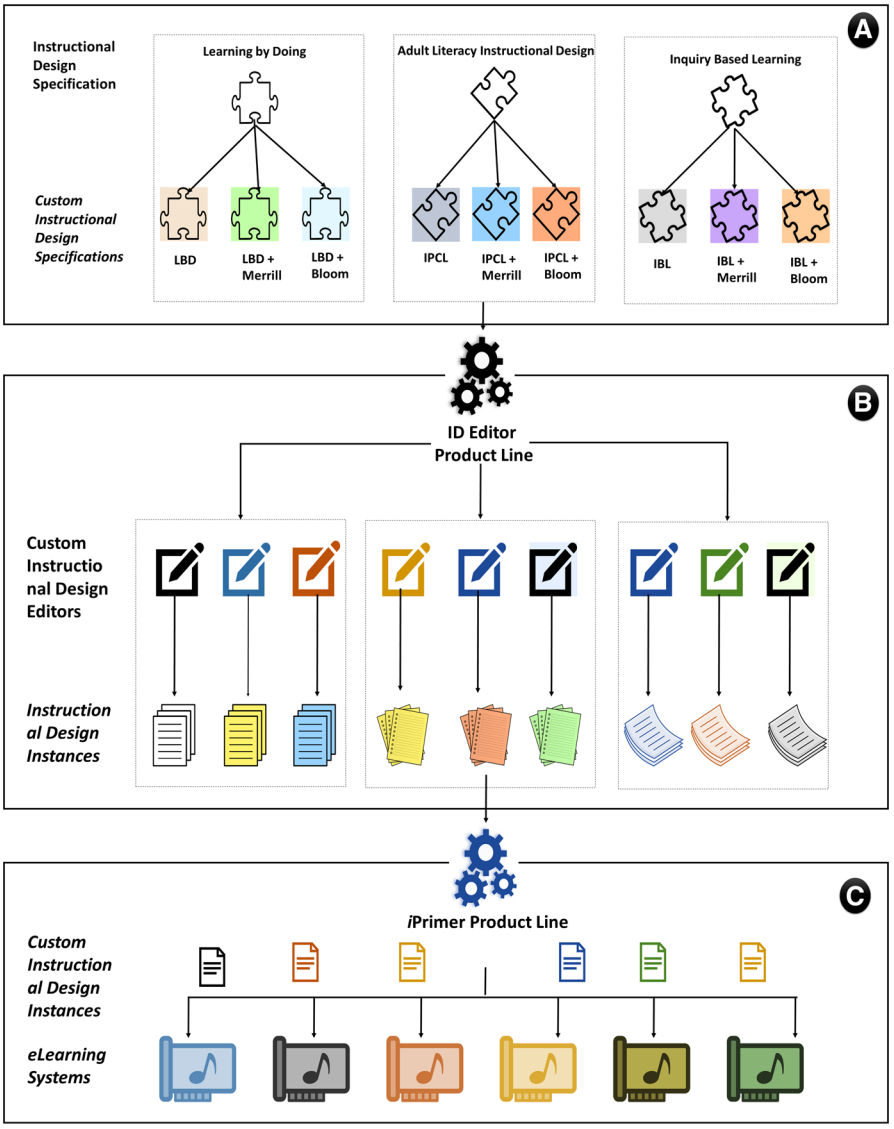
- *Adult Literacy as families* - Based on the learning methodology of IPCL proposed by NLM [18], we show a portion of family for organization of adult literacy domain in Fig. 3. Two common components of this family are *Process* and *Content*, where each one of them will have core aspects that are present in every family member and can also impose some constraints. The process itself can have many number of activities such as *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*. The basic features in the family are combined with specific needs of the particular segment of learners to deliver a specific and customized instructional design.
- *eLearning Systems as families* - User interface variations can be modeled as a hierarchy of Presentation Units that allow navigation between them. Each of the units further contain UI elements such as buttons, textboxes and so on. These UI elements have properties such as *name*, *data*, *color*, *font* and so on. The user interface can be modeled for different platforms such as *Desktop*, *Web*, *Mobile*.

In the next section, we present a *multi-level software product line* that connects multiple software product lines in this paper.

5 A family of software product lines

Figure 4 succinctly summarizes different levels of product lines that we considered in this paper. We reiterate our notion of instructional design as a set of *goals*, *process*, *content*, context, evaluation, environment. Figure 4a 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 are 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 requirements. The scope of this meta product line is to create custom instructional design specifications based on a core instructional design. Similarly, there can be a number of sub-product families within this product line for different types of instructional designs.

How to create instances of the custom instructional design specifications? Fig. 4b shows a product line at the next level whose product family members are custom



Legend – *Different colors indicate variants of Instructional Designs, Editors, Instances and iPrimers*

Fig. 4 Multi-level software product lines

instructional design editors that take an instructional design schema.[7] We designed a prototype platform 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

[7] A detailed listing of the instructional deign specification is available at https://git.io/vdxJG.

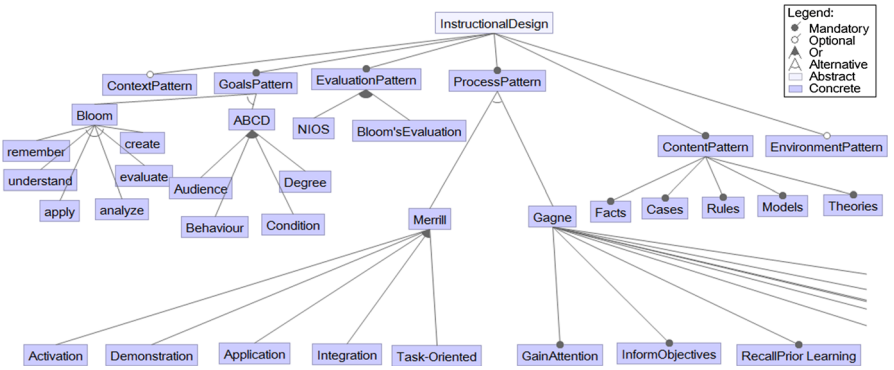


Fig. 5 A fragment of instructional design feature model

with literature from educational technologies, these editors are similar in principle to learning design editors such as *ReLoad* and *ReCourse Editor* [72], *ASK-LDT Editor* [73], *LAMS* [24], *Learning Designer* [25], *Web-COLLAGE* [74], *ILDE* [75] 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.

While creation of this SPL was done with inputs from domain experts and expert teachers in adult literacy, the primary audience of this product line is instructional designers or teachers who can customize the instructional designs as per their specific requirements. This task becomes non-trivial if the specific needs require modification of instructional designs beyond configuration of provided commonalities and variabilities. The detailed customization process is discussed in Sect. 6.

Figure 4c shows the next level of product line that is specific to a custom instructional design specification, in this case it is based on IPCL and adult literacy instructional design. We designed a prototype that takes a specific instance of adult literacy instructional design and generates *e*Learning Systems, which are the product family members. 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 [76], (ii) feature modeling plugin from University of Waterloo is a solution specifically useful for cardinal features and feature cloning and feature attributes [77]. For example, a *fact* in *ContentPattern* as a feature should be cloned for various instances. The primary audience of this product line is novice teachers who can customize the eLearning Systems for specific requirements and the customization process is detailed in Sect. 7. In the next sections, we succinctly describe the two product lines for instructional design and *e*Learning Systems.

6 A software product line for a family of instructional designs

6.1 A basic feature model

Based on patterns [67] and ontologies [68], we present a *feature model* for modeling a family of instructional designs. Here, we consider standard definitions from SPL literature [78] where a *feature* is a characteristic or end-user-visible behavior of a software system, and 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 5 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. We use feature modeling plugin to annotate features with additional knowledge such as *syllables*, *sounds* and so on.

6.2 Product family members, feature model and configurations

Table 1 shows a brief description of requirements of four different kinds of instructional design specifications for adult literacy. 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. Table 1 shows three concepts namely *Goals*, *Process* and *Content* for different instructional design specifications. The primary goals are 3Rs 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, which can be modeled using the product line.

ID Specification 2 family uses instructional design patterns in [67] to describe the Process and Content aspects of the instructional design whereas *ID Specification 3* family uses Bloom's revised 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 paper but uses ABCD technique, Gagne's nine events of instruction for goals and process, and resources for content. Each of these instructional design specifications can be used to create *ID Editors*[8] 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 Fig. 6a,b, we show a feature model[9] encompassing key features of this product line. This is essentially based on the ontologies for different aspects of instructional

[8] We use the term *ID Editors* to mean instructional design editors.

[9] A detailed instructional deign feature model is available at https://git.io/vdxJG.

Table 1 Custom instructional design specification requirements

Aspect	ID Spec 1	ID Spec 2	ID Spec 3	ID Spec 4
Base ID	IPCL	IPCL + *ProcessPattern (pasi)* + *ContentPattern (fcrmt)*	IPCL + *pasi* + *fcrmt* + Merrill's First Principles of Instruction + Bloom's Revised Taxonomy	IPCL + Gagne's Nine levels of learning + ABCD Technique for Goals
Goals	3Rs	3Rs	Bloom's Revised Taxonomy	ABCD technique
Process	Eclectic method for teaching 3Rs	*pasi* with *instructions* containing actual *activities* and *tasks*	*pasi* with activities based on Merrill's First Principles	Gagne's nine levels of learning
Content	IPCL primer	*fcrmt*	*fcrmt* + Merrill's First Principles	Resources

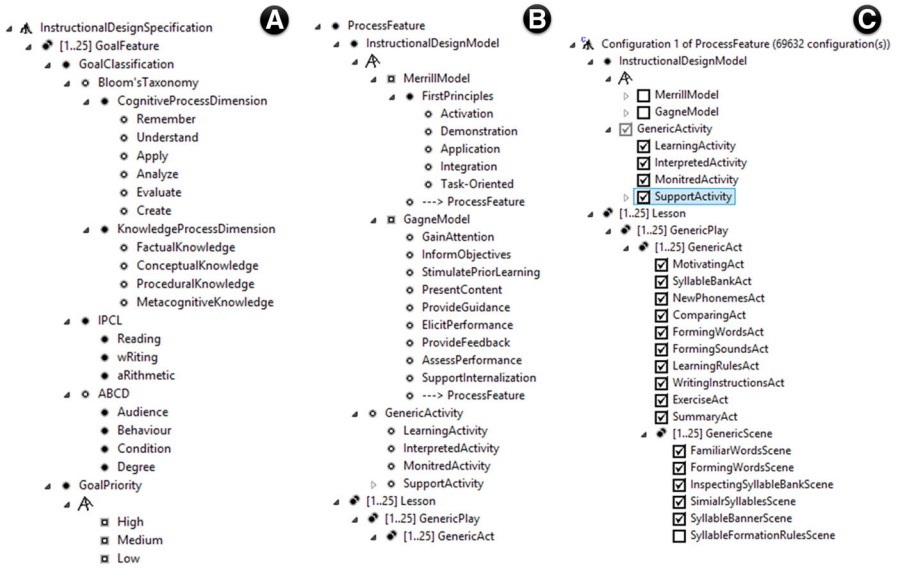


Fig. 6 A feature model for ID specification (**a**, **b**) and feature configurations (**c**)

design comprising of mandatory features such as *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 at least one of *Play*, *Act*, *Scene* and *Instruction*, and up to a maximum of 25 *Plays* in a given instructional design.

As shown in Fig. 6c, the *ProcessFeature* can be configured in multiple ways such as *ProcessPattern*, *MerrillModel*, *GagneModel*. We can also select or deselect required *lessons*, *plays*, *acts*, *scenes* as per specific requirements. 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 *ID Editors* for creating instances of the specific instructional design. We present the reference implementation architecture in the next section.

6.3 A reference implementation 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. Figure 7 shows a reference architecture for the product lines in this paper. This architecture can be implemented in multiple ways but we discuss our current implementation here. An instructional designer or teacher creates the patterns

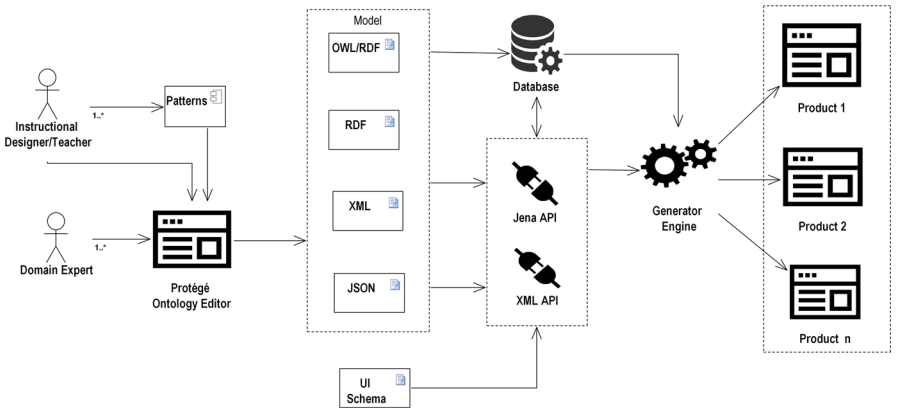


Fig. 7 A reference implementation architecture for product lines in this paper

as document/text and uses that to create an ontology with an editor such as protégé or reuses an existing ontology such as IMS-LD ontology [29] or the comprehensive ontology of instructional design teaching learning theories [79]. We store this ontology as OWL, XML and JSON for further processing by tools. This data is part of Model in Model-View-Controller pattern. We have used Jena API for processing OWL/RDF files and generate 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 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 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 primarily based on files and stores all resources in a single package and is largely implemented using *Javascript*, *jquery*, *Jena API*, XML parser and custom animations.

The concrete process of creating *ID Editors* is shown in Fig. 8. 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 such as *goals*, *process*, *content*. The *ID Editor Product Line* is an engine written in *JavaScript*[10] 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 Sect. 5. We have implemented this using multiple technologies such as *Java*[11] and *Python*.[12] However, the need is to create multiple instances of instructional

[10] https://git.io/fjNJC.

[11] https://git.io/fjNJW.

[12] *Semantic Web Forms*, https://bit.ly/2ZhsmfO by undergrads at IIIT Sri City, India.

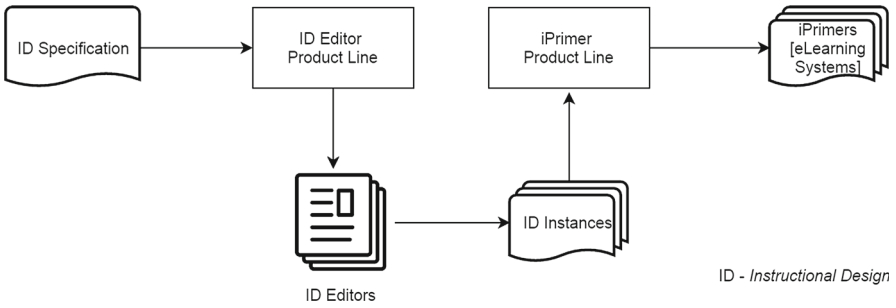


Fig. 8 Flow of ID Editor Product Line and *i*Primer Product Line

designs that form the basis for several *i*Primers. We discuss the product line for creating a family of *e*Learning Systems in the next section.

7 A software product line for a family of *e*Learning systems

The primary goal of this product line is to create a family of *e*Learning Systems based on specific instructional designs tailored to the needs of teaching functional literacy for all Indian languages. The 32 State Resource Centers (SRCs) 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 National Literacy Mission Authority (NLMA):

- Basic literacy primer [22+], Post literacy primer [22+]
- Life long literacy primers [22+]
- Primers for teaching skills such as tailoring, vocational skills, [*n*+, where *n* is in the order of hundreds]
- Exclusive primers such as legal literacy, election literacy, agriculture literacy [*n*+, where *n* is in the order of tens]

Teaching 3Rs is the core commonality among these primers but with varied instructional processes and content and generally available in 22 languages. It is estimated that currently there are around 1000 primers available with SRCs in print format. Even though the primer is fixed till the next version, officers at different levels (mandal, village, school and teachers) customize the *process*, *content* and adapt it to the local context. For example, a simple way could be to use local *names* and teach syllables using them. This is a great source of variability for *i*Primers of our product line, and not possible using print form. Technically, we are interested in *i*Primers that are based on field-tested *e*Learning Systems [19]. These systems 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 facilitate varied instructional processes, use locally relevant content and present a multimedia application with animations for the learners.

Figure 8 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

Fig. 9 Primer and custom instructional design instance [OWL/XML] for *Hindi* language

India, the *iPrimer Product Line* is based on a single *ID Specification* driven by IPCL. This *ID Editor* is used to create several instances of the instructional design specification for varied processes, content, 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 8 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 9 shows a fragment of primer of *Hindi* language. This primer has around 180 pages with 24 lessons and each lesson focusing on 3Rs. This primer is available in both print as well as digitized format (pdf). This digitized form is given 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[13] contains all the information related to a specific instructional design and serves as the base for creating variations based on this instructional design. Figure 10 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, content and generates animations accordingly. Everything that is shown in Fig. 10 can be varied as per the feature model configurations discussed in earlier sections. This allows to rapidly customize the *i*Primers and create new ones by changing processes and content. Similarly, we have generated the *i*Primer for *Telugu*. Figure 11 shows how an *i*Primer has been generated for *Telugu* language based on a specific instructional design instance. Here, the processes, content, user interface that are relevant for that specific instructional design have been generated. Figure 12 shows some variations that are possible for *Telugu* language.

[13] More details at https://git.io/vdxJV.

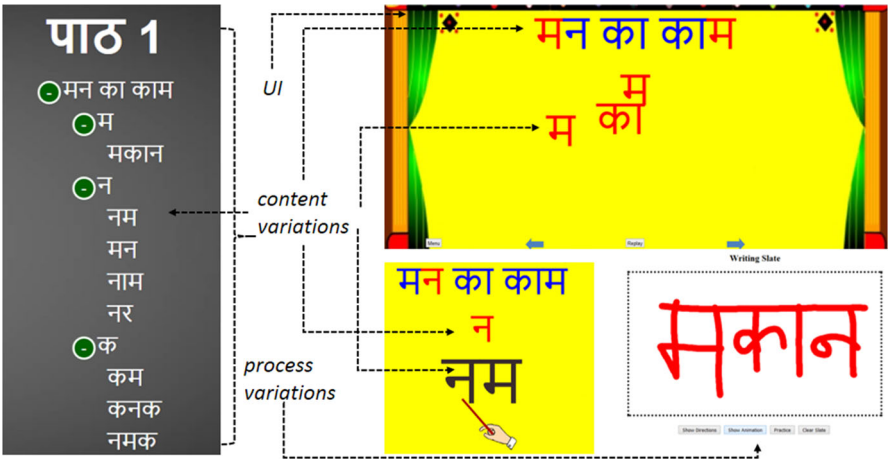


Fig. 10 *i*Primer for *Hindi* language-generated from instructional design instance

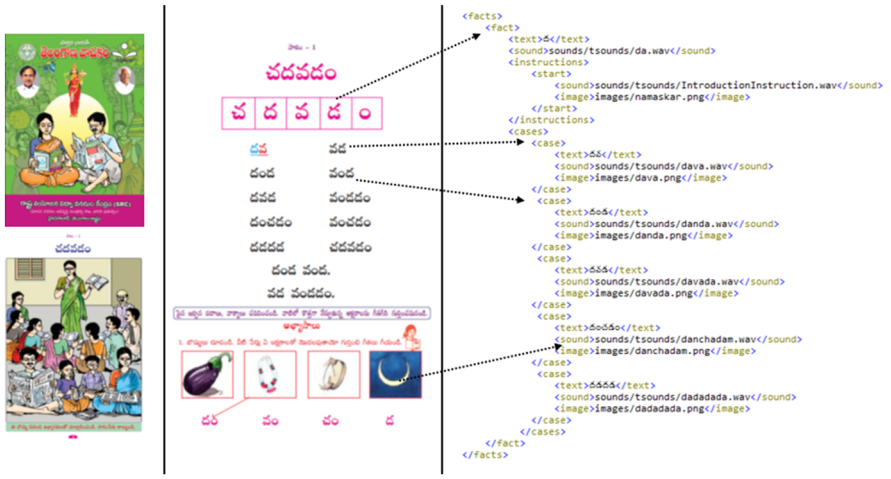


Fig. 11 Primer and custom instructional design instance [OWL/XML] for *Telugu* language

The core idea here is to be able to generate as many *i*Primers as possible with minimum effort by using SPLs. We have observed that this product line can be configured easily to create *i*Primers but one practical limitation is the manual process of creating sound elements (around 2500) for every *i*Primer. However, new technologies could be used to address this concern in the future. We present the experimental results of our approach in the next section.

8 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

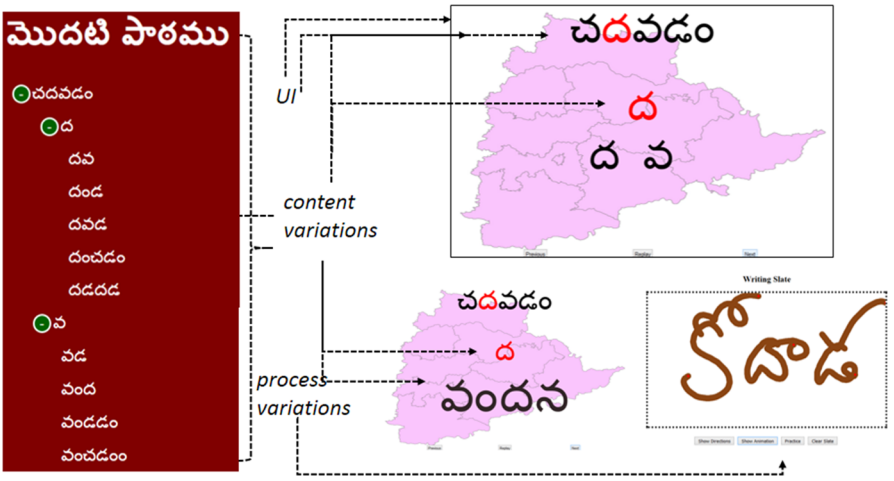


Fig. 12 *i*Primer for *Telugu* language - generated from instructional design instance

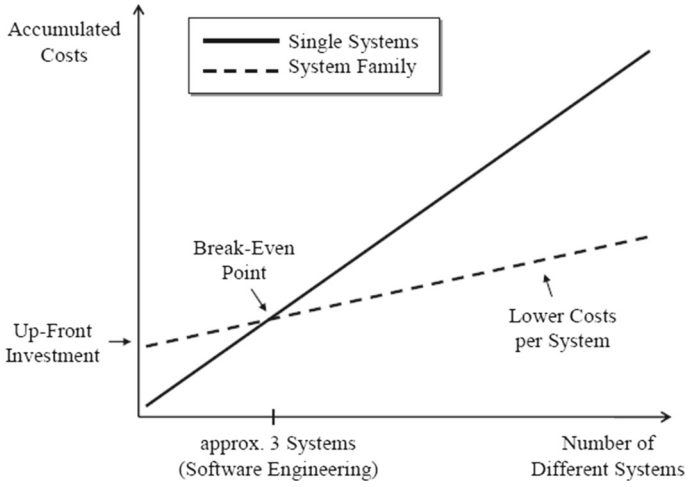


Fig. 13 Break-even point in cost savings for single systems versus product lines from [48]

that our primary goal in this paper is to facilitate customization of educational technologies for scale and variety and demonstrate it for 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 [48]. The literature has a number of measures to calculate the cost and return on investment on software product lines [80] and a comparison of multiple models is given in [81]. A typical cost savings curve for SPLs shows that break-even point for achieving cost benefits of SPL over single systems occurs at 3rd product [48,82]. Tüzün and Tekinerdogan have analyzed experience curve of return on investment in SPLs from the literature [83] and found that it is similar to the classical one as shown in Fig. 13.

In this paper, we use the Structured Intuitive Model for Product Line Economics (SIMPLE) model as it is considered as one of the commonly used approaches to measure the effectiveness of product lines in the literature [48,81,84]. 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 paper 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 [80].

- C_{org} - The cost to an organization for adopting product line approach instead of single system development. In this paper, the product lines are developed by researchers and hence no direct organization costs, however this cost should be included in the long run.
- 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.
- 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.

Boehm suggests calculation of cost functions using different estimation techniques such as, judgement-based, algorithmic model, or analogies [85]. In this paper, non-SPL development costs for *i*Primers come from TCS' experience of developing adult literacy systems [19]; and for *ID Editors* through our initial experience of manually developing multiple *ID Editors*.

The costs of developing a software product line for n distinct products using the SIMPLE model can be calculated as follows [82,86]:

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

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

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

The savings of software product lines can be estimated as:

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

Tata Consultancy Services (TCS), an Indian software services organization has been involved with development of *e*Learning Systems for adult literacy in India

for more than 15 years [19]. We use data from our earlier experience of developing *e*Learning Systems [55] and TCS' statistics on developing *e*Learning Systems for 9 Indian Languages [19] as the initial base for calculating cost savings of *iPrimer Product Line*. The effort for creating an *e*Learning System was around 5–6 person years and in our earlier work, we have applied software reuse techniques and reduced the effort for creating *e*Learning Systems to 5 to 6 person-months [55]. Each existing *i*Primer approximately consists of 2000 visual elements; 2500 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 *e*Learning System for teaching 3Rs. The *iPrimer Product Line* essentially generates these 24 lessons as shown in Fig. 10 for *Hindi* language with manual inputs for words and sounds. We have also developed similar *i*Primers for *Telugu* language. 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 $+ 9 * 3$ person-weeks

$C_{SPL} = 25$ person-months

Cost of building n stand-alone products

$C_{stand\text{-}alone} = 54$ person-months $(9 * 6$ person-months$)$

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

$$\text{Hence, } C_{savings} = 29 \text{ person-months } (C_{SPL} - C_{stand\text{-}alone})$$

Table 2 shows the individual cost components for *iPrimer Product Line* and Fig. 14 compares the cost of creating *i*Primers (green color) and *ID Editors* (blue color) with and without our approach. $C_{stand-alone}$ denotes standalone costs and $C_{spl(iPrimer)}$ and $C_{spl(ID)}$ denote cost for *i*Primers and *ID Editors* respectively. The horizontal axis shows the number of *i*Primers/*ID Editors* and the vertical axis shows the number of person-months required to develop the *i*Primers. From the figure, it can be noted that the development cost for the first *i*Primer is 6 person-months using single systems approach and 18.75 person-months using SPL; 12 and 19.5 person-months for two *i*Primers; 18 and 20.25 person-months for three *i*Primers; and 24 and 21 person-months for four iPrimers. This shows that the break-even for the initial investment in terms of core asset base is 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. We have used the *iPrimer Product Line* to create preliminary 9 *i*Primers for 9 varied specific requirements. The effort for each of the *i*Primers is shown in Fig. 14. 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 is released as an android app on *Google Play Store*.[14] A

[14] https://bit.ly/2RVh2Su.

Table 2 Cost components of *iPrimer Product Line*

C_{aspect}	Cost	Description
$C_{org}()$	6 person-months	For *iPrimer Product Line*, there is no direct cost to adopt the product line approach as it developed as part of our research rather than a single organization. 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 *iPrimer Product Line* include ontologies in RDF/OWL for instructional design based on patterns, *JavaScript* files, a parser that reads XML configuration files and generates instances, UI components such as 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 *process* steps and *content* i.e., words, syllables, either extracted from digital primer or entered manually and also special syllables or words that are specific to the particular language. The cost to create sound files for new words and instructions 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.

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*.[15] The *i*Primer was also listed on Government of Telangana websites[16]. In addition, a workshop was conducted in November 2016 for 24 voluntary teachers of adult literacy on the use of *iPrimer Product Line*. Figure 14 also shows the cost of developing *ID Editors* with and without SPL. The cost functions are exactly the same as that of *iPrimer Product Line* except the cost of manual effort for customizing the generated *ID Editor*, which is one person-month instead of three person-weeks. Figure 15 shows a glimpse of the session where teachers used *iPrimer Product Line* to create partial lessons based on dynamic instructional design data.

Even though we discussed the experimental results in terms of cost, both the *ID Editor Product Line* and *iPrimer Product Line* cater to the needs of scale and variety. In the case of *ID Editor Product Line*, scale and variety is the number and variations of instructional design editors that can be created and further customized. For example, by changing the patterns for *goals*, *process* and *content*, the instructional design can be changed. In the case of *iPrimer Product Line*, the scale and variety are with respect to the number of *e*Learning Systems that can be created and the variations that can be

[15] https://git.io/vdxkd.

[16] http://tslma.nic.in/, http://srctelangana.com/.

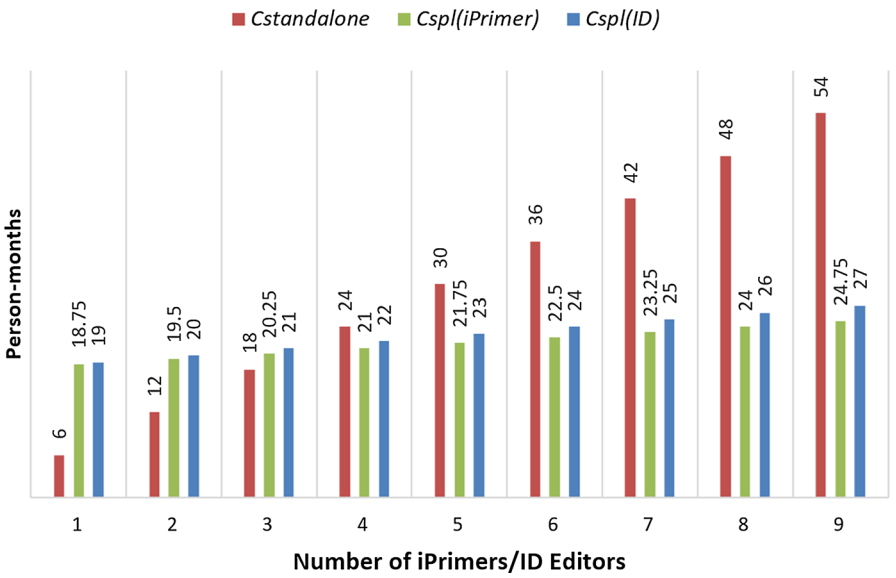


Fig. 14 Cost Savings of *iPrimer Product Line* and *ID Editor Product Line*

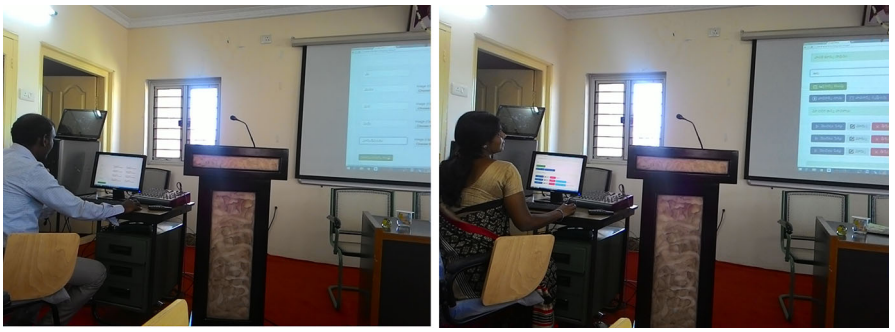


Fig. 15 Teachers using *iPrimer Product Line* to create *i*Primers

supported. This SPL supports variations in *goals*, *process* and *content* including some animations and UI elements. Most importantly, the SPL is agnostic of language and can work for the spectrum of Indian Languages. We have used both the product lines to create multiple *ID Editors* based on variations of IPCL and 9 *e*Learning Systems. Also, for *Telugu* language, the content variations are done in association with State Resource Center, Telangana.

9 Discussion and limitations

The proposed work has several limitations at multiple levels. The first and foremost drawback is in terms of effort required for creating and maintaining SPL for adult literacy domain for new requirements other than those defined in the scope of SPL.

As proposed in Sect. 6, the SPL is based on patterns and ontologies for instructional designs within the scope of the proposed SPL. However, if new pedagogies or improved patterns in instructional design emerge, then existing patterns and ontologies have to be modified as per new requirements. This exercise has to be done in consultation with domain experts in both instructional design and adult literacy.

Secondly, even though the *ID Editor Product Line* in Sect. 7 can be used to generate custom instructional designs as it supports different aspects of instructional design such as *process*, *content* and so on, these can be processed further only if there is an *iPrimer Product Line* in the next stage, which essentially is creating another SPL. In its current form, the next stage SPL is available for adult literacy *e*Learning Systems based on IPCL and its variations as presented in Sect. 7. This SPL for *e*Learning Systems currently supports the requirements of adult literacy. However, there is immense scope to improve the SPL by further developing the platform to support different kinds of variations in animations and user interface elements.

A critical question that we did not directly address in this paper is the learning effectiveness of the *e*Learning Systems that are generated from the SPL. As mentioned in Sect. 3, we used field-tested *e*Learning Systems [19] as the basis for the *iPrimer Product Line*. As the *e*Learning Systems that are generated are similar to the ones that are field-tested and are reviewed by domain experts, we did not explicitly conduct studies to understand learning effectiveness of the generated systems.

Furthermore, the *e*Learning System generation process has some manual steps in terms of providing raw material such as content and then customizing the generated product making it necessary to have software engineers or technical staff to support the instructional designers or teachers.

Despite the existence of several economic models in the literature for SPLs, evaluation of SPLs has many open challenges for the community as it involves qualitative data rather than quantitative and empirical data [81]. The SIMPLE cost model that is used in Sect. 8 to evaluate our approach also suffers from similar problems. For example, as per the SIMPLE model, integration costs are included in C_{reuse} and based on our experience with 9 *i*Primers, we determined the cost as 1 person-week. However, if the input *content* for new *i*Primer could not be represented as XML, then integrating this data with rest of the tools in *iPrimer Product Line* takes more effort than the estimated 1 person-week, which is a threat to the validity of the cost model. While we had access to 24 teachers who used our platform to create *i*Primers as a way to validate our approach, the practical challenge was to get them use our platform again for any further experimentation as the approval should come from NLMA of Government of India. In the case of *ID Editors*, we did not have access to expert teachers who can create instructional design variations beyond IPCL and hence we could not validate further beyond the case of adult literacy.

10 Future work

10.1 Educational technologies beyond adult literacy

First and foremost, we see that the proposed approach in this paper could be applied for design of educational technologies beyond adult literacy. For example, National Skill Development Corporation of Govt. of India has set up 38 Sector Skill Councils for different industrial skills such as automotive, healthcare and so on, devised model curriculum[17] and further 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. Our approach could be used in this domain as it is extremely effort-intensive to manually develop educational technologies and for scale and variety.

Similarly, the effort required to design and customize educational technologies for schooling in India is massive, and so is the case of effort required to support teaching of engineering subjects across all disciplines. 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. We see that our approach could be used as the baseline for addressing some of these challenges towards accelerating the development of educational technologies.

10.2 Future directions for SPL

Despite the emergence of SPL four decades ago in software engineering, there are only few cases of SPL applied to the domain of education. Our work is an attempt towards that direction but there are many open challenges that require further research for application of SPL in this domain [64].

- *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? How can a business case be established for an SPL?
- *Dealing Non-Technical Stakeholders* - How to 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?
- *Process Diversity & Version Management* - How to map the diversified processes and versions during the development and maintenance of SPL?

In addition, within SPL, there are several open challenges [87]:

- Using current feature modeling notations in SPL, features can only be selected for product configuration but the need in educational technologies is to have features that have additional information associated with them for different aspects of instructional design such as goals, process steps and content, which is not possible with current notations. For example, expressing goals using Bloom's taxonomy

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

could be a feature but specifying an exact learning goal requires more than just features.
- Design of light-weight SPL approaches for educational technologies domain is a definite need as instructional design itself is a complex activity.
- Educational technologies domain is essentially socio-technical in nature and presents the need for a family of product lines catering to the needs for variety at multiple levels across 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.

11 Conclusions

In this paper, we aimed at creating a multi-level software product line approach for the design and customization of educational technologies to address *scale* and *variety* in education. Specifically, we addressed the need to support creation of *e*Learning Systems for flexible instructional designs and multiple Indian Languages for adult literacy in India. We explained the development of software product lines for a family of (i) instructional designs (ii) *e*Learning Systems and discussed how these are connected to each other through a reference architecture. We demonstrated our approach through *ID Editor Product Line* and *iPrimer Product Line* and further semi-automatically generated *e*Learning Systems for adult literacy case study. The work in this article is one of the first attempts of large scale application of software engineering approaches for design educational technologies and can lead to a significant line of research on integrating software reuse approaches and personalized learning; and for multiple domains in education such as schools, vocational skills and different forms of engineering and medical education.

Acknowledgements We thank TCS for its initial inputs, *National Literacy Mission* for recommending our work at national level, Government of Telangana for being one of the first adoptors of our technologies and all funding agencies for supporting our research travels.

References

1. Chimalakonda S (2017) A software engineering approach for design of educational technologies. Ph.D. dissertation, International Institute of Information Technology Hyderabad
2. Beetham H, Sharpe R (2013) Rethinking pedagogy for a digital age: designing for 21st century learning. Routledge, Abingdon
3. Laurillard D (2013) Rethinking university teaching: a conversational framework for the effective use of learning technologies. Routledge, Abingdon
4. Kirkwood A, Price L (2014) Technology-enhanced learning and teaching in higher education: What is 'enhanced' and how do we know? A critical literature review. Learn Media Technol 39(1):6–36
5. Qian M, Clark KR (2016) Game-based learning and 21st century skills: a review of recent research. Comput Hum Behav 63:50–58

6. Weller M (2007) Virtual learning environments: using, choosing and developing your VLE. Routledge, Abingdon
7. Carbonell JR (1970) Ai in cai: an artificial-intelligence approach to computer-assisted instruction. IEEE Trans Man Mach Syst 11(4):190–202
8. Khan BH (1997) Web-based instruction. Educational Technology, Englewood Cliffs
9. Lipponen L (2002) Exploring foundations for computer-supported collaborative learning. In: Proceedings of the conference on computer support for collaborative learning: foundations for a CSCL community. International Society of the Learning Sciences, pp. 72–81
10. Sheu F-R, Chen N-S (2014) Taking a signal: a review of gesture-based computing research in education. Comput Educ 78:268–277
11. Sampson D, Karagiannidis C (2010) Personalised learning: educational, technological and standardisation perspective. Interact Educ Multimed 4:24–39
12. Wu H-K, Lee SW-Y, Chang H-Y, Liang J-C (2013) Current status, opportunities and challenges of augmented reality in education. Comput Educ 62:41–49
13. Kapp KM (2012) The gamification of learning and instruction: game-based methods and strategies for training and education. Wiley, London
14. Toyama K (2011) There are no technology shortcuts to good education. Educational technology debate: exploring ICT and learning in development countries. Accessed at http://edutechdebate.org/ict-in-schools/there-are-no-technology-shortcuts-to-good-education/
15. Flavin M (2017) Disruptive technology enhanced learning: the use and misuse of digital technologies in higher education. Springer, Berlin
16. Howard SK, Mozejko A (2015) Teachers: technology, change and resistance. In: Teaching and digital technologies: big issues and critical questions, pp 307–317
17. UNESCO (2014) Education for all global monitoring report 2013/4: teaching and learning—achieving quality for all. United Nations Educational and Scientific and Cultural Organization
18. DAE (2003) Handbook for developing IPCL material. Directorate of Adult Education, India
19. TCS (2019) CSR case study, computer based functional literacy. Tata Consultancy Services. [Online]. http://www.tcs.com
20. Patel I (2002) Information and communication technology and distance adult literacy education in India. Institute of Rural Management Anand
21. Botturi L, Stubbs ST, Global I (2008) Handbook of visual languages for instructional design: theories and practices. Inf Sci Ref Hershey 7:226
22. Sampson DG, Zervas P (2014) A hierarchical framework for open access to education and learning. Int J Web Based Commun 10(1):25–51
23. Consortium IGL et al (2003) IMS learning design specification
24. Dalziel J (2003) Implementing learning design: the learning activity management system (LAMS)
25. Laurillard D, Charlton P, Craft B, Dimakopoulos D, Ljubojevic D, Magoulas G, Masterman E, Pujadas R, Whitley EA, Whittlestone K (2013) A constructionist learning environment for teachers to model learning designs. J Comput Assist Learn 29(1):15–30
26. Hernández-Leo D, Asensio-Pérez JI, Derntl M, Pozzi F, Chacón J, Prieto LP, Persico D (2018) An integrated environment for learning design. Front ICT 5:9
27. Neumann S, Klebl M, Griffiths D, Leo DH, de la Fuente Valentín L, Hummel HGK, Brouns F, Derntl M, Oberhuemer P (2010) Report of the results of an ims learning design expert workshop. iJET 5:58–72
28. Rodríguez-Artacho M, Maillo MFV (2004) Modeling educational content: the cognitive approach of the PALo language. Educ Technol Soc 7(3):124–137
29. Amorim R, Lama M, Sánchez E, Riera A, Vila X (2006) A learning design ontology based on the IMS specification. J Educ Technol Soc 9(1):38
30. Knight C, Gasevic D, Richards G (2006) An ontology-based framework for bridging learning design and learning content. J Educ Technol Soc 9(1):23
31. Bansal SK, Dalrymple O (2016) Imod-ont: towards an ontology for instructional module design. In: 2016 IEEE tenth international conference on semantic computing (ICSC). IEEE, pp 354–357
32. Roschelle J, DiGiano C, Koutlis M, Repenning A, Phillips J, Jackiw N, Suthers D (1999) Developing educational software components. Computer 32(9):50–58
33. Douglas I (2001) 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, pp F4E–1

34. Koper R, van Es R (2004) Modelling units of learning from a pedagogical perspective. Online Educ Using Learn Objects 40:43–58
35. Sampson DG, Zervas P (2011) A workflow for learning objects lifecycle and reuse: towards evaluating cost effective reuse. Educ Technol Soc 14(4):64–76
36. Polsani PR (2006) Use and abuse of reusable learning objects. J Digit Inf 3(4):164
37. Nurmi S, Jaakkola T (2006) Promises and pitfalls of learning objects. Learn Media Technol 31(3):269–285
38. Sinclair J, Joy M, Yau J-K, Hagan S (2013) A practice-oriented review of learning objects. IEEE Trans Learn Technol 6(2):177–192
39. Boyle T (2003) Design principles for authoring dynamic, reusable learning objects. Aust J Educ Technol 19(1):46–58
40. Dodero JM, Ruiz-Rube I, Palomo-Duarte M, Cabot J et al (2012) Model-driven learning design. J Res Pract Inf Technol 44(3):267
41. Torres J, Resendiz J, Aedo I, Dodero JM (2014) A model-driven development approach for learning design using the lpcel editor. J King Saud Univ Comput Inf Sci 26(1):17–27
42. McGreal R, Kinuthia W, Marshall S, McNamara T (2013) Open educational resources: innovation, research and practice. Commonwealth of Learning (COL), Vancouver
43. Santos-Hermosa G, Ferran-Ferrer N, Abadal E (2017) Repositories of open educational resources: an assessment of reuse and educational aspects. Int Rev Res Open Distance Learn 18(5):84
44. Dodero J-M, Garcia-Penalvo F-J, Gonzalez C, Moreno-Ger P, Redondo M-A, Sarasa A, Sierra J-L (2012) Points of view on software engineering for elearning (panel session). In: 2012 International symposium on computers in education (SIIE). IEEE, pp 1–4
45. Clements P, Northrop L (2002) Software product lines: practices and patterns, vol 59. Addison-Wesley, Reading
46. Metzger A, Pohl K (2014) Software product line engineering and variability management: achievements and challenges. In: Proceedings of the on future of software engineering. ACM, pp 70–84
47. Krueger C (2001) Easing the transition to software mass customization. In: International workshop on software product-family engineering. Springer, pp 282–293
48. Pohl K, Böckle G, van Der Linden FJ (2005) Software product line engineering: foundations, principles and techniques. Springer, Berlin
49. Thüm T, Apel S, Kästner C, Schaefer I, Saake G (2014) A classification and survey of analysis strategies for software product lines. ACM Comput Surv (CSUR) 47(1):6
50. Asikainen T, Männistö T, Soininen T (2007) Kumbang: a domain ontology for modelling variability in software product families. Adv Eng Inf 21(1):23–40
51. Lee S-B, Kim J-W, Song C-Y, Baik D-K (2007) 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 and applications (SERA 2007). IEEE, pp 727–734
52. SPLC (2019, July) Software product lines hall of fame. [Online]. http://splc.net/fame.html
53. Chimalakonda S, Nori KV (2012) A software engineering perspective for accelerating educational technologies. In: 2012 IEEE 12th international conference on advanced learning technologies (ICALT). IEEE, pp 754–755
54. Pankratius V (2007) Product lines for digital information products. KIT Scientific Publishing, Karlsruhe
55. Chimalakonda S (2010) Towards automating the development of a family of elearning systems. International Institute of Information Technology Hyderabad, India, Technical Reports
56. Ahmed F, Zualkernan IA (2011) A software product line methodology for development of e-learning system. Int J Comput Sci Emerg Technol 2:285–295
57. Dalmon DL, Brandão LO, Brandão AA, Isotani S et al (2012) A domain engineering for interactive learning modules. J Res Pract Inf Technol 44(3):309
58. Júnior VF, Duarte Filho NF, de Oliveira Junior EA, Barbosa EF (2014) Towards the establishment of a software product line for mobile learning applications. In: SEKE, pp 678–683
59. Marcolino AS, Barbosa EF (2017) Towards a software product line architecture to build m-learning applications for the teaching of programming. In: Proceedings of the 50th Hawaii international conference on system sciences
60. Lessa Filho CAC, Domínguez AH (2018) A software product line for development of educational games. Braz J Comput Educ 26(01):1

61. Azouzi S, Ghannouchi SA, Brahmi Z (2017) Software product line to express variability in e-learning process, In: European, mediterranean, and middle eastern conference on information systems. Springer, pp 173–185
62. Chimalakonda S, Nori KV (2012) Towards a synthesis of learning methodologies, learning technologies and software product lines. In: 2012 IEEE 12th international conference on advanced learning technologies (ICALT). IEEE, pp 732–733
63. Chimalakonda S, Nori KV (2012) Towards a model driven elearning framework to improve quality of teaching. In: 2012 IEEE fourth international conference on technology for education (T4E). IEEE, pp 138–143
64. Chimalakonda S, Nori KV, (2013) What makes it hard to apply software product lines to educational technologies? In: 2013 4th international workshop on product line approaches in software engineering (PLEASE). IEEE, pp 17–20
65. Chimalakonda S, Nori KV (2013) 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, pp 649–654
66. Chimalakonda S, Nori KV (2013) Designing technology for 287 million learners. In: 2013 IEEE 13th international conference on advanced learning technologies (ICALT). IEEE, pp 197–198
67. Chimalakonda S, Nori KV (2014) A patterns-based approach for modeling instructional design and tel systems. In: 2014 IEEE 14th international conference on advanced learning technologies (ICALT). IEEE, pp 54–56
68. Chimalakonda S, Nori KV (2013) Idont: an ontology based educational modeling framework for instructional design. In: 2013 IEEE 13th international conference on advanced learning technologies (ICALT). IEEE, pp 253–255
69. Parnas DL (1976) On the design and development of program families. IEEE Trans Softw Eng 1:1–9
70. Gagne RM, Briggs LJ (1974) Principles of instructional design. Rinehart & Winston, Holt
71. Merrill MD (2012) First principles of instruction. Wiley, London
72. Griffiths D, Beauvoir P, Liber O, Barrett-Baxendale M (2009) From reload to recourse: learning from ims learning design implementations. Distance Educ 30(2):201–222
73. Sampson D, Karampiperis P, Zervas P (2005) Ask-ldt: a web-based learning scenarios authoring environment based on IMS learning design. Int J Adv Technol Learn 2(4):207–215
74. Villasclaras-FernáNdez E, HernáNdez-Leo D, Asensio-PéRez JI, Dimitriadis Y (2013) Web collage: an implementation of support for assessment design in cscl macro-scripts. Comput Educ 67:79–97
75. Hernández-Leo D, Asensio-Pérez JI, Derntl M, Prieto LP, Chacón J (2014) Ilde: community environment for conceptualizing, authoring and deploying learning activities. In: European conference on technology enhanced learning. Springer, pp 490–493
76. Thüm T, Kästner C, Benduhn F, Meinicke J, Saake G, Leich T (2014) Featureide: an extensible framework for feature-oriented software development. Sci Comput Progr 79:70–85
77. Antkiewicz M, Czarnecki K (2004) Featureplugin: feature modeling plug-in for eclipse. In: Proceedings of the 2004 OOPSLA workshop on eclipse technology eXchange. ACM, pp 67–72
78. Apel S, Batory D, Kästner C, Saake G (2013) Feature-oriented software product lines: concepts and implementation. Springer, Berlin
79. Mizoguchi R, Hayashi Y, Bourdeau J (2007) Inside theory-aware and standards-compliant authoring system. In: SW-EL'07, p 18
80. Bockle G, Clements P, McGregor JD, Muthig D, Schmid K (2004) Calculating roi for software product lines. IEEE Softw 21(3):23–31
81. Ali MS, Babar MA, Schmid K (2009) A comparative survey of economic models for software product lines, In: 2009 35th euromicro conference on software engineering and advanced applications. IEEE, pp 275–278
82. Krüger J (2016) A cost estimation model for the extractive software-product-line approach, Ph.D. dissertation, Otto-von-Guericke-University Magdeburg
83. Tüzün E, Tekinerdogan B (2015) Analyzing impact of experience curve on roi in the software product line adoption process. Inf Softw Technol 59:136–148
84. Weiss DM (2008) The product line hall of fame. In: 2008 12th international software product line conference. IEEE, pp 395–395
85. Boehm BW (1984) Software engineering economics. IEEE Trans Softw Eng 1:4–21
86. Clements PC, McGregor JD, Cohen SG (2005) The structured intuitive model for product line economics (simple), DTIC Document, Technical Reports

87. Nori KV, Reddy YR, Chimalakonda S (2014) Challenges for software engineering in educational technologies. In: 2014 International conference on contemporary computing and informatics (IC3I). IEEE, pp 267–272