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 [263][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
Figure 6.1 Multi-level software product lines in this thesis
instructional design specification is provided in Appendix §8.2. 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 [131], ASK-LDT Editor [132], LAMS [38], Learning Designer [179], COL-LAGE [130], Web-COLLAGE [35], ILDE [298] [134] 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 [15] 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, प 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
Figure 6.4 Custom instructional design specification requirements

<table>
<thead>
<tr>
<th>Aspect/ID</th>
<th>ID Specification 1</th>
<th>ID Specification 2</th>
<th>ID Specification 3</th>
<th>ID Specification 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base ID</td>
<td>IPCL</td>
<td>IPCL + ProcessPattern (ps)</td>
<td>IPCL + ProcessPattern (ps) + ContentPattern (fcrnt)</td>
<td>IPCL + Gagne’s Nine levels of learning + ABCD Technique for Goals</td>
</tr>
<tr>
<td>Goals</td>
<td>Read, write and basic arithmetic (3Rs)</td>
<td>Read, write and basic arithmetic (3Rs)</td>
<td>Organize goals using Bloom’s revised taxonomy</td>
<td>Organize goals using ABCD technique</td>
</tr>
<tr>
<td>Process</td>
<td>Eclectic method for teaching 3Rs</td>
<td>Organize instructional process as a set of plays, acts, scenes, instructions with instructions containing actual activities and tasks</td>
<td>Mainly driven by ProcessPattern (ps) but the activities should be based on Merrill’s first principles of instruction.</td>
<td>Organize process using Gagne’s nine levels of learning</td>
</tr>
<tr>
<td>Content</td>
<td>Content as per IPCL based primer</td>
<td>Content is organized as facts, cases, rules, models and theories</td>
<td>Content is organized as facts, cases, rules, models and theories and mapped to Merrill’s first principles</td>
<td>Content is organized as resources</td>
</tr>
</tbody>
</table>

Figure 6.5 A feature model for instructional design specification
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. InstructionalDesignModel has three choices MerrillModel, GagneModel and GenericActivity. 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 Play, Act, Scene and Instruction are mandatory and can go up to a maximum of 25. A detailed listing of the instructional design 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...
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 design feature model is provided in Appendix §8.1. 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
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 [42] or a comprehensive ontology is available for instructional design teaching learning theories [153]. 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 iPrimers 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
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 iPrimers. 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 iPrimer Product Line

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

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. Even though 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 iPrimers 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 iPrimers that are based on field-tested eLearning Systems \([29]\). These applications are based on puppet theater model, where syllables are shown as falling puppets, joining together to form words and so on. The iPrimers, 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 iPrimers. In the case of adult literacy in India, the iPrimer Product Line is based on a single ID Specification driven by IPCL. Appendix §8.2 provides a detailed listing of the instructional design specification used for adult literacy.
Figure 6.11 Primer and custom instructional design instance [XML from OWL] for Hindi language

Figure 6.12 iPrimer for Hindi 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 iPrimers for multiple languages and primers. Every instructional design instance leads to a varied iPrimer 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 iPrimers and discuss iPrimers 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 iPrimers and create new ones by changing processes and content. Figure §6.13 shows how an iPrimer 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 iPrimers 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 iPrimers 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 iPrimers. We wish to reiterate that the primary goal of this thesis is to facilitate customization of educational technologies for
Figure 6.13 Primer and custom instructional design instance [XML from OWL] for Telugu language

Figure 6.14 iPrimer for Telugu language - generated from instructional design instance
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 [12]. 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 [12]. 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].

- \( C_{\text{org}} \) - 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 iPrimers should incur costs for transition to product line approach.

- \( C_{\text{cab}} \) - 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.

- \( C_{\text{unique}} \) - 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.

- \( C_{\text{reuse}} \) - 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]:

\[
C_{\text{SPL}} = C_{\text{org}} + C_{\text{cab}} + \sum_{i=1}^{n} (C_{\text{unique}}(\text{product}_i) + C_{\text{reuse}}(\text{product}_i))
\]

Cost of building \( n \) stand-alone products

\[
C_{\text{stand-alone}} = \sum_{i=1}^{n} C_{\text{product}}(\text{product}_i)
\]

where \( C_{\text{product}} \) is the cost of developing an individual product.

The savings of software product lines can be estimated as:

\[
\text{Savings of product lines} = C_{\text{stand-alone}} - C_{\text{SPL}}
\]
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 [29]. We use data from our earlier experience of developing eLearning Systems [7] and TCS’ statistics on developing eLearning Systems for 9 Indian Languages [29] 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 [7]. Each existing iPrimer 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., iPrimers:

Cost of building a product line

\[ C_{SPL} = 6 \text{ person-months} + 12 \text{ person-months} + 9 \times (2 \text{ person-weeks} + 1 \text{ person-week}) \]
\[ C_{SPL} = 25 \text{ person-months} \]

Cost of building \( n \) stand-alone products

\[ C_{\text{stand-alone}} = 9 \times 6 \text{ person-months} \]
\[ C_{\text{stand-alone}} = 54 \text{ person-months} \]

where \( C_{\text{product}} \), 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 iPrimers with and without our approach. The horizontal axis shows the number of iPrimers and the vertical axis shows the number of person-months required to develop the iPrimers. The graph shows that the break-even for the initial investment in terms of core asset base is for 3 or 4 iPrimers after which as the number of iPrimers 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 iPrimer based on a newly released physical primer. This iPrimer 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

<table>
<thead>
<tr>
<th>Cost Component</th>
<th>Cost (Person-months)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{org}()$</td>
<td>6</td>
<td>In case of <em>iPrimer</em> 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.</td>
</tr>
<tr>
<td>$C_{cab}()$</td>
<td>12</td>
<td>Core assets in the case of <em>iPrimer</em> 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.</td>
</tr>
<tr>
<td>$C_{unique}()$</td>
<td>2</td>
<td>The unique parts of the <em>iPrimer</em> 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.</td>
</tr>
<tr>
<td>$C_{reuse}()$</td>
<td>1</td>
<td>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 <em>iPrimer</em>.</td>
</tr>
</tbody>
</table>

is hosted online at Google Play Store Store⁶. 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 Github⁷. The primer was also listed on Government of Telangana websites⁸. 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.

---

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 6.15 Cost Savings of iPrimer Product Line](image)

**Figure 6.15** Cost Savings of *iPrimer Product Line*

![Figure 6.16 Teachers using *iPrimer Product Line* to create *iPrimer*](image)

**Figure 6.16** Teachers using *iPrimer Product Line* to create *iPrimer*
6.6 Challenges of Software Product Lines for Educational Technologies - A future work

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 [61],

- **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?