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

5.1 Cost of Educational Technologies

There is a dramatic rise in the use of technologies in education in the last decade or so [248]. However, there is also severe criticism on several dimensions like huge upfront costs of technologies, difficulties in using them on the field, lack of evidence to show positive impact on quality of education [64]. 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 [64]. In a steering committee meeting of planning commission\(^1\) organized by Government of India, we have proposed the need for technology-driven solutions to address several challenges in education like huge scarcity of qualified teachers and lack of quality\(^2\) for adult literacy and in general for education in India. Several senior professors and deans in the meeting have raised three key pre-requisites for any proposed technology-driven solution. The technologies should be (i) based on strong pedagogical basis (ii) available for all Indian languages and (iii) flexi-

\(^1\)Planning Commission is headed by the Prime Minister of India and is later replaced by NITI AYOG, a think-tank for guiding the Government of India

\(^2\)Formulation of the 12th Five Year Plan- Minutes of the second meeting of the Steering Committee on Elementary Education and Literacy held on 25.08.2011-regarding
ble to cover varied teaching styles and different types of learners. Most importantly, the primary concern was in terms of the cost and effort required for creating and maintaining the technologies for the scale and variety inherent for education in India. For example, the TARA Akshar+ Literacy and Numeracy programme has made about 129,000 people literate across India at the cost of ₹5,500 (110 $) per student\(^3\) unlike Government of India’s available budget of ₹230 per student \(^4\). Even though the initiative seems to be successful, the entire technology has to be maintained by technical experts and updating these technologies with evolving curriculum and in multiple languages is quite expensive and infeasible. In addition, most of these technologies do not have a pedagogical basis denting quality of instruction.

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

### 5.2 Product Lines & Software Product Lines

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

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

\(^3\)http://taraakshar.org/index.php/faq/

\(^4\)Personal Communication, Joint Director & Director General, National Literacy Mission Authority, India

\(^5\)http://www.samsung.com/in/consumer/mobile-devices/smartphones/

\(^6\)http://www.lexus.com/
cess is similar and different to manufacturing [249] and some authors have attributed these failures to the non-applicability of the factory concept to software [167] but Peter Wegner sums up the similarities and contradictions this way: “Software products are in some respects like tangible products of conventional engineering disciplines such as bridges, buildings, and computers. However, there are also certain important differences that give software development a unique flavor. Because software is logical not physical, its costs are concentrated in development rather than production...” [250]. Mass production and mass customization are two critical concepts that form the basis for most of these debates. According to the classic definition, “Mass production is the tremendous increase in scale of the same product at low costs whereas Mass customization is a tremendous increase in variety and customization of the product without a corresponding increase in costs” [251]. The primary focus in software is on mass customization unlike manufacturing. SPLs have matured from being a niche research area to a well established approach for software production catering to certain markets. SPLs offer several promises over one-off development like significant reduction in cost after an up-front investment on common assets (as shown in Figure §5.2), improved quality and consistency across all product family members, increased flexibility and better handling of evolution [12]. Even though not explicitly developed as SPLs, most of the software that is developed today like operating systems, databases, plug-ins for IDEs provide several configuration options [15]. For example, researchers have analyzed that Linux kernel provides as many as 10,000 features for users as configuration options [252]. The hall of fame initiative of a 16 year conference on software product lines has published

Figure 5.1 Sample configuration options for product family members [DELL laptops]
exemplary case studies of companies like Boeing, Bosch Group, Siemens, HP illustrating their significant gains of productivity for creating and customizing families of products in a product line [253]. Despite this success, there are several challenges associated with SPLs. While initial upfront investment is a known challenge, a less emphasized but critical challenge is increase of complexity with huge volume of features and increase in number of product family members. This has left SPL as a complex development approach that can be used by well-established organizations leading to decreased adoption of SPLs in several domains. For instance, there are not many instances of SPL applied to educational technologies [61] despite tremendous need and potential. In the next sections, we discuss the key differences between single systems and SPL development and motivate the need for SPL at three different levels of granularity in the domain of education.

5.3 From Individual Systems to Families

The predominant way of developing software today is either a generalized product like Office, Gmail or Payroll for a large number of customers or specific software that is designed for individual customers like in service based organizations. Consider the scenario of software for all banks? Is it same for all banks? Is it different for all banks? Similarly, if we consider software for all accounting systems? Is it same or different? The idea of looking at software systems as a family rather than completely different individual systems offers several advantages [11]. A family of systems can be at different levels of granularity with
Figure 5.3 Few sample instructional design families

multiple sub-families within families and also a hierarchy of families. In this section, we discuss several families that are of interest to us in this thesis. A family of systems can be defined as a set of systems that share more common properties with other members in the set than differences providing unique advantages to address the common and varying needs of specific markets [11].

5.3.1 Instructional Design as families

In this thesis, we considered Instructional Design as a fundamental tenet that forms the basis for design of educational technologies. How is instructional design developed? Is it developed from scratch? or reused from existing resources? Are instructional designs common across subjects? learners? teachers? or universities? How many instructional design theories are present in the literature? What is common across them? Can instructional designs be considered as a family? Charles Reigeluth has extensively studied instructional design theories and models and documented them in three voluminous books [71][254][255]. His vision was to build a common knowledge base and a common language about instruction [71]. We can classify instructional design in the form of several families and at different levels of granularity as shown in Figure §5.3. For example, we can consider the three fundamental ways of cognitivism, behaviourism and constructivism or we can group them based on models like Gagne’s model [256], Dick and Carey’s model [257] , or the generic ADDIE process that is followed in most of the instructional designs. Each of these models
can be grouped as a family based on subjects like STEM or K-12 or can also be formed as a family primarily based on learning styles like Visual, Kinesthetic, Auditory. Here [A] and [B] in Figure §5.3 are at one level of granularity whereas [C] and [D] are at the next level of granularity. This can be further refined into a hierarchy of families till the lowest level of granularity. The classification primarily depends on the goals of the specific target organization or stakeholders. For example, if a university has professors who are keen on using “learning by doing” approach, then a family of instructional designs can be modeled with “learning by doing” as the base approach and adapting it for different kinds of learners and subjects. On the other hand, if a university policy mandates accreditation with a national body, then all instructional design should use principles of accreditation as base and then adapt them for specific needs. It is mandatory that all members belonging to a family have certain common properties and vary on some aspects making them fit for the specific purposes.

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

### 5.3.2 Adult Literacy as families

Should there be a universal technological solution for teaching all 287 million adult learners across India? or Should there be unique solution for every leaner?. This leads to a trade-off and need for a balanced solution between one-size-fits-all and unique-for-every-dimension solutions. NLMA, the highest authority for adult literacy in India has devised a uniform methodology called IPCL as the base for creating instructional designs for all languages across India [26]. The handbook of IPCL provides guidelines for customizing different aspects of instruction based on several dimensions [26]. The key goals for adult literacy primers are to create immense interest in the learners and provide functional knowledge that can add value to learners’ daily life [203]. These common goals have to be customized for learners spread across India based on socio-cultural and local contexts.

Based on [203] and [26], we show a portion of family for organization of adult literacy domain in Figure §5.4. We use standard Lego blocks to show possible variations in adult literacy. Two common components of this family are Process and Content, where each one of them will have core aspects that are present in every family member and
can also impose some constraints. The process followed for adult literacy in India is generally driven by *eclectic method* that starts teaching from known to unknown with gradual progress in learning. Now, any instructional design for adult literacy domain in India that comes under this family must follow *eclectic method* unlike *synthetic method* that teaches from alphabets to words. The process itself can have many number of activities like StoryTelling, GroupDiscussion, RolePlay that can be customized based on specific needs. This family also says that Content should be present and can be further divided into CoreContent and LocallyRelevantContent. CoreContent mandates topics like national integration, secularism, democracy, scientific temper, communal harmony, women’s equality, population education and development, etc. whereas LocallyRelevantContent will be customized by specific states and stakeholders based on learner’s livelihood, their socio-cultural realities, special issue-based and thematic aspects such as gender parity, health and hygiene, agricultural, animal husbandry, self-help groups, local self-government, livelihood programmes, etc. Then same family can be classified based on the primary topic/knowledge that can be used to teach knowledge that will be useful for learners in their daily life. Legal Literacy can focus on teaching learners laws pertaining to them, how to seek help from law whereas Scheme Literacy can provide knowledge of Government schemes. Skill Literacy can help them in gaining knowledge pertaining to a particular skill like tailoring, plumbing and so on. Thematic Literacy is a generic way to accommodate themes that can be local to the specific audience and Agriculture Literacy can provide farming knowledge. Each of these families can mandate that any adult literacy instructional design based on the corresponding family
should be within the scope of the defined goals and constraints. These common themes are combined with specific needs of the particular segment of learners to deliver a specific and customized instructional design. These commonalities and variabilities among family members are generally modeled using feature diagrams in SPL [10][15].

5.3.3 eLearning Systems as families

Even though we do not focus on modeling variants of user interface in this thesis, it can be a critical source of variability for eLearning Systems. There exists certain ways to model user interfaces [258] as a hierarchy of Presentation Units that allow navigation between them and each of the units further contain UI elements like buttons, textboxes and so on. These UI elements have properties like name, data and so on. Each of these elements also have properties to describe their visual appearance like color, font and so on. The user interface can be modeled for different platforms like Desktop, Web, Mobile or the user interface can also be modeled using structural and behavioral elements. These elements can be organized at a higher level of abstraction in different ways giving several user interface variants. For example, there can be several Views of Model-View-Controller pattern corresponding to different variants of user interfaces. In the case of adult literacy eLearning Systems, we are interested to model the user interface elements primarily with three resources text, image, audio.

5.4 Pattern-Oriented Software Product Lines

Over the last two decades, there has been an extensive research on SPL as evidenced through a series of focused conferences like Software Product Lines Conference (SPLC), workshops like Variability Modelling of Software-intensive Systems (VaMoS), Product Line Approaches in Software Engineering (PLEASE). An analysis of the literature on SPL reveals that are two major terminologies to discuss the idea of developing a family of software-intensive systems. Software Engineering Institute (SEI) has steered the research and development on software product lines (sometimes called as software product family) and has published several technical reports and case studies [259]. On the other hand, several researchers and organizations also used the term “software product line engineering” for their work in the area of SPL [12]. Several organizations, universities and research institutes performed collaborative research on SPL, which supported the systematic building of a community of software product line engineering research and practice. Some of those projects include ARES (1995-1998), PRAISE (1998-2001), ESAPS (1999-2001), CAFÉ (2001-2003), and FAMILIES (2003-2005). In 2014, Metzger and Pohl have done an extensive study of 600 articles published in the area of SPL and noted that there has been impressive quantitative and qualitative progress in the field with key challenges for
industrial adoption [260]. Krueger has suggested three ways of adopting SPLs [261] (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 [13][12][262][263][264]. One related approach proposed for the domain of flight control focuses on using architecture and design patterns for SPL [265] but largely confines to modeling variability. Researchers have also used ontologies for modeling and configuring variability during SPL [266][267][268]. A software product line is developed based on ontologies for developing knowledge-driven semantic web applications [269] to facilitate interoperability between semantic services and intelligent agents. A more detailed account of research in SPL can be found in [260] and a recent bibliographic analysis of research in SPL over 20 years is provided in [270].

On the other hand, many of the SPL approaches have been applied in the last couple of decades in practice across several domains with successful results [253]. However, there is sparse research on applying SPL in the domain of technology enhanced learning [271]. Pankratis has proposed PLANT as a product line based approach for creation and maintenance of digital information products [57]. In our prior work, we proposed TALES as an approach for automating the development of eLearning Systems [7]. A software product line methodology for development of e-learning system for a six sigma course was proposed in [272]. A domain engineering activity for interactive learning modules is proposed in [273]. [274]. However, none of these approaches consider instructional design as the basis and focus on adult literacy which is the goal of this thesis. After a critical analysis of literature, we find that SPL is largely undermined in technology enhancing learning community despite their significant potential and hence motivating our approach.

A standard reference model is created for a set of software and systems product line standards [276] based on the seminal work of software product line engineering [12]. Figure §5.5 shows the set of product line standards mapped to reference model in [276]. We briefly explained this architecture and the role of patterns in these product line standards in [275]. In this thesis, we extend this reference model and propose a pattern-oriented software product line approach (in short PoPL) as shown in Figure §5.6. Our approach primarily differs from existing literature in terms of using patterns and ontologies as a central knowledge base for driving several sub-processes of a traditional SPL. This emphasis on patterns has been used as one of the critical foundations for several software and systems product line standards devised by ISO/IEC JTC/SC7 [275]. In essence, SPL consists of two distinct life cycles of domain engineering and application engineering unlike single-systems engineering [12]. Domain engineering and application engineering further consist of requirements engineering, design and realization. Each of these processes and sub-
processes can be considered as high-level patterns (akin to software architecture patterns) that emerged from recurring and standardized practice from SPL communities, researchers and practitioners. These processes are further refined to explicitly model subprocesses, activities and further detailed as tasks to deliver a product line. Also, a lot of effort goes in creating and maintaining variability in the life cycle of an SPL.

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

A. Product Line Organization
The need for a product line usually emerges from the business need of an organization[s] which provides the rationale for why an SPL can be considered at all, the economics and feasibility of SPL. We have observed that in the case of adult literacy in India, the need emerged from a great societal challenge and has several organizations like NLMA of Government of India, researchers in IIIT-Hyderabad, Corporate Social Responsibility of TCS, NGOs and so on. This led to several challenges [61] and is unlike traditional SPL where the organization that is motivating SPL has a strong business case driving the entire initiative.

B. Product Line Analysis & Scoping
An important prerequisite for SPL is a deeper understanding and analysis of the commonal-
Figure 5.6 Overview of Pattern Oriented Software Product Lines
ity and variability in a particular domain. In our approach, we consider scoping as a critical activity that is performed prior to domain engineering and provides guidelines and directions for the rest of the approach. We consider a constrained view of product line scoping from the ISO/IEC 26551 standard [277] on product line requirements engineering. The primary goal of product line scoping is to identify the member products of a product line along with their major (externally visible) common and variable features and analyze the products from economic and development perspectives. This sub-process further consists of (i) **product scoping**, that determines target markets, product categories, common and variable features and estimated timelines for development of product line members (ii) **domain scoping** focuses on understanding the potential of the targeted product line market and refines product scoping (iii) **asset scoping** identifies existing assets and their potential for reusability from a development perspective.

**C. Domain Stakeholders**

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

**D. Feature Modeling**

Feature modeling is the most widely used notation to capture the commonalities and variabilities of an SPL [10][12][15]. Essentially, feature models represent all the common and variable features of an SPL. Feature models are extended with different notations like [278][187][279], formalisms [280][281][282] and so on. Several researchers have focused on formalizing feature models [283][284] and verification of product lines using formal semantics [285]. The increasing number of large-scale features models motivated researchers towards a domain specific language for managing features [286]. A detailed
overview of feature oriented product lines research is presented in [15]. One particular
goal that was of interest to us was to model knowledge with features. For example, how
to add metadata to features such that this data can be later used during application rea-
ralization for creating customized product family members? Feature attributes are a way
to associate a feature to a type like integer or string or to a group of sub-features [287].
This was later extended with cardinalities, feature diagram references, and user-defined
annotations [288] to make feature diagrams more descriptive and to facilitate staged con-
figuration of features [289]. A notable work in this direction is the mapping of feature
models as views on ontologies [290]. Peng et al. have proposed an ontology-based fea-
ture modeling approach as a way to formally represent feature models [291] and Wang
et al. used OWL for verifying feature models [292]. Within ontology-based feature mod-
eling, an empirical study conducted in 2015 showed that using individuals or instances of
ontologies can provide further flexibility for reconfiguration of products [293]. A formal
language is proposed to deal with non-boolean features focusing on feature attributes and
multi-features using formal methods [294]. Despite this progress, annotating knowledge to
feature models is still an open problem especially in the domain of education. We are not
aware of existing methods that address the basic requirement of annotating features with
images, audio and domain knowledge in an effective way. For example, every fact that is
part of ContentPattern in instructional design can be associated with a syllable and sound,
with provision for customization. While we do not attempt at addressing this problem in
this thesis, we point this as an important future direction. To the best of our knowledge,
we are not aware of existing approaches that use domain patterns as a base for feature modeling. Figure § 5.7[A] shows a fragment of user interface feature model using basic notation and Figure §5.7 [B] shows features annotated with cardinalities [1..*], [1..200], in this case there can be a number of facts as part of content but cases are limited to a maximum of 200. Cardinalities are initially criticized for adding clutter to feature models but were shown to be useful in specific domains like embedded software [287]. We use cardinalities in the domain of education as they can help in specifying several attributes of features. For example, for a feature like a lesson in an instructional design can have a maximum of 3 learning goals and can be modeled with cardinalities. Most importantly, we see feature attributes as a key extension of feature models that can immensely help product lines in this thesis. Facts can be associated with the attributes of syllable of type unicode string, a vocal rendering of the syllable using an audio file attribute, an optional attribute for storing image. Each of these attributes can be used for realizing variability. A feature like Rules can have ruleSpecification, ruleExplanation, ruleApplicability, ruleConstraints as some key attributes for storing knowledge about the corresponding feature. A detailed instructional design feature model is listed in Appendix §B.1.

E. Domain Engineering (DE)

Domain Engineering is one of the fundamental pillars of SPL that is concerned with encapsulating past experiences of building systems in a particular domain by creating reusable assets and providing means to develop new systems by utilizing and reusing these assets [12]. In short, domain engineering is concerned with defining and realizing the commonality and variability of an SPL, thus establishing a common platform for developing high-quality applications rapidly within the scope of the product line. Domain engineering involves analyzing common and variable features of product line members and provides a common platform for rapidly creating customizable product members. Domain engineering has three sub-processes which handle three key aspects (i) Domain requirements engineering (ii) Domain design (iii) Domain realization. Domain requirements engineering is concerned with determining the common and variable features of a product line with inputs from product line scoping. The output of this sub-process is used by domain design to produce a reference product line architecture for all product line members addressing variability at an architecture level. We advocate the use of pattern oriented software architectures at this stage of processes and map it with patterns in the domain. Commonality and variability analysis with respect to architecture design is also done as part of this sub-process. The reusable assets required to implement product line architecture are developed as part of domain realization focusing on variability mechanisms. Common assets are usually built as components and services with exposed interfaces. These assets should be built with provisions for implementing variabilities by using different mechanisms like patterns, aspects, pre-processors and so on. Domain engineering does not actually deliver concrete assets
but produces a set of reusable assets at different levels of granularity, which can be used for further realization.

**F. Verification & Validation**

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

**G. Variability Modeling**

SPLs are built for the long run, unlike single-systems engineering and hence would require management of several aspects. One of the most critical aspects for the success of SPL is the explicit modeling of variability during different sub-processes in the product line. There is lot of research in the literature for modeling variabilities in an SPL depending on the specific requirements of the product line [263]. Variability is generally categorized into essential variability, from users' perspective and technical variability, from realization perspective [12]. A detailed overview of variability modeling approaches is surveyed in [263].

**H. Patterns, Ontologies & Assets Repository**

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

I. Application Stakeholders

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

J. Application Engineering

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

Both domain engineering and application engineering have an iterative life cycle involving requirements, design, realization, testing [12]. In addition to these, there are also two key sub-processes of technical and organizational management in the SPL life cycle from an organization perspective [276]. Considering the vast number of processes and sub-processes in SPL and the effort required for performing these tasks, a common practice is to implement core or mandatory processes during development of SPL [261].

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