

---

# A Patterns Based Approach for Design of Educational Technologies

Sridhar Chimalakonda · Kesav V. Nori

**Abstract** Instructional design is a fundamental base for educational technologies as it lays the foundation to facilitate learning and teaching based on pedagogical underpinnings. However, most of the educational technologies today face two core challenges in this context: (i) lack of instructional design as a basis (ii) lack of support for a variety of instructional designs. In order to address these challenges, we propose a patterns based approach for design of educational technologies. This is in contrast with existing literature that focuses either on patterns in education or in software, and not both. The core idea of our approach is to leverage patterns for modeling instructional design knowledge and to connect it with patterns in software architecture. We discuss different categories of patterns in instructional design. We then present the notion of *Pattern-Oriented Instructional Design* (POID) as a way to model instructional design as a connection of patterns (*GoalPattern*, *ProcessPattern*, *ContentPattern*) and integrate it with *Pattern-Oriented Software Architecture* (POSA) based on fundamental principles in software engineering. We demonstrate our approach through adult literacy case study (287 million learners, 22 Indian Languages and a variety of instructional designs). The results of our approach (both web and mobile versions) are available at <http://rice.iiit.ac.in> and were adopted by *National Literacy Mission Authority of Government of India*.

**Keywords** patterns; modeling; architecture; instructional design; software engineering; adult literacy

---

*Submitted to Educational Technology Research and Development Journal, Springer*

This work was carried out as part of first author's doctoral thesis at IIIT Hyderabad, India and contains content from the thesis (Chimalakonda, 2017)

---

S. Chimalakonda  
Department of Computer Science & Engineering  
Indian Institute of Technology, Tirupati  
E-mail: [ch@iittp.ac.in](mailto:ch@iittp.ac.in)

K.V. Nori  
Software Engineering Research Center  
International Institute of Information Technology Hyderabad, India  
E-mail: [ch@iittp.ac.in](mailto:ch@iittp.ac.in)

## 1 Introduction

Education domain has been undergoing a major transformation in the last decade or so (Adams Becker et al., 2017; Alper & Gülbahar, 2009; Hwang & Tsai, 2011; Wu et al., 2012). On one hand, there is a significant surge on the use of educational technologies<sup>1</sup> (such as game based learning (Tobias, Fletcher, & Wind, 2014), MOOCs (Reich, 2015), gesture based learning (Sheu & Chen, 2014), augmented reality (Bower, Howe, McCredie, Robinson, & Grover, 2014) and so on) to facilitate learning and teaching. On the other hand, there is a significant fallout on the expectation of educational technologies ranging from terminological inconsistency (Bayne, 2015) to the lack of effectiveness of instructional technology in classrooms (Venkatesh, Croteau, & Rabah, 2014). Several researchers have underlined the broken promises of educational technologies (Bingimlas, 2009; Cuban & Jandrić, 2015; Spector, 2013; Spector, Merrill, Elen, & Bishop, 2014). In addition, the community has identified several grand challenges in educational technologies (Fischer, Wild, Sutherland, & Zirn, 2014; Woolf, Lane, Chaudhri, & Kolodner, 2013), computing and information systems (Association et al., 2003) in the context of education. The *National Academy of Engineering* lists “Advance personalized learning” as a grand challenge of engineering<sup>2</sup>. In this paper, we are concerned about two core challenges that underlie most of these grand challenges:

### 1.1 Challenge 1: Lack of Instructional Design as a Basis for Design of Educational Technologies

Instructional Design<sup>3</sup> has gained a significant role in the field of Technology Enhanced Learning as an underlying and complex discipline often involving multiple perspectives and connotations (Reigeluth, 2013a, 2013b; Reigeluth & Carr-Chelman, 2009). Merrill has defined instructional design as the practice of creating “*instructional experiences which make the acquisition of knowledge and skill more efficient, effective, and appealing*” (Merrill, 2012).

Berger defines instructional design as a “*systematic development of instructional specifications using learning and instructional theory to ensure the quality of instruction*” (Berger & Kam, 1996). From the definition of Berger, instructional design acts as a basis for quality of instruction. In a similar line, Carroll emphasizes that quality of instructional design leads to quality of instruction (Carroll, 1963). Bednar et al. have stated that “... effective instructional design is possible only if the developer has reflexive awareness of the theoretical basis underlying the design . . . [it] emerges from the deliberate application of some particular theory of learning” (Bednar, Cunningham, Duffy, & Perry, 1992). This strong need to have a pedagogical basis for design of educational technologies has been emphasized in the literature by several

<sup>1</sup>We consider “educational technologies as a set of processes, techniques, methods and tools that facilitate learning and teaching based on well-established instructional designs.”

<sup>2</sup><http://www.engineeringchallenges.org/challenges/learning.aspx>

<sup>3</sup>We consider instructional design as an underlying structure consisting of different aspects of instruction such as goals, process, content aimed at (i) providing a base for quality of instruction (ii) facilitating design of educational technologies

researchers (Boyle & Cook, 2001; Duffy & Jonassen, 2013; Garzotto, Retalis, & Chapter, 2009; Goodyear et al., 2004; Govindasamy, 2001). A critical and detailed analysis of the need to bridge learning theories and technology enhanced learning environments is elaborated in (Lowyck, 2014). However, most of the educational technologies today lack instructional design basis leading to poor quality of instruction (Laurillard, 2013a, 2013b; Lowyck, 2014; Reid, 2014; Spector et al., 2014; Toyama, 2011).

*How to facilitate design of educational technologies with an instructional design basis?*

## **1.2 Challenge 2: A Scale & Variety of Instructional Designs and Educational Technologies**

Instructional Design is used as an umbrella term that can refer to an entire discipline or as a process, art, science or technology (Brown & Green, 2015). *One size does not fit all* is a core principle that suits well for teaching and learning as every context is different because of aspects such as varied learning styles, varied instructional designs, varied learning environments and so on. There are over 100 instructional design models in the literature (Reigeluth & Carr-Chelman, 2009) and several perspectives of instructional design. In one of the early works, a continuum of instructional strategies was presented with one end focusing on instructor-centered to the other end focusing on student-centered with a wide range of activities ranging from *drill and practice* to *projects* and *inquiry* (Gustafson & Branch, 1997).

The ADDIE model involving *analysis*, *design*, *development*, *implementation* and *evaluation* phases is one of the most widely used instructional design model that is applicable in several contexts (Kruse, 2002). Gagne's series of nine learning events (Gagne & Briggs, 1974) has laid foundation for several instructional design models such as Dick and Carey model (Dick, Carey, Carey, et al., 2001) and Merrill's first principles of instruction (Merrill, 2012). Millwood summarizes over 25 learning theories in a concept map connecting the different facets of instructional design (Millwood, 2014). Gibbons (Gibbons, 2013) presents eight views of instructional design (i) organizational view (ii) systems approach view (iii) design language view (iv) instructional systems design view (v) functional-modular view (vi) architectural view (vii) team process view (ix) operational principle view. Mizoguchi et al. have done an comprehensive survey of learning theories from instructional technology perspective and modeled them using ontologies (Mizoguchi, Hayashi, & Bourdeau, 2007). The synthesis of literature on instructional design reveals that there is a strong need to support a variety of instructional designs during 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) (Botturi, Derntl, Boot, & Figl, 2006; Botturi, Stubbs, & Global, 2008; Martínez-Ortiz, Moreno-Ger, Sierra, & Fernandez-Manjon, 2007) such as po-EML (Caeiro, Llamas, & Anido, 2014), PALO (Rodríguez-Artacho & Maillo,

2004), Web COLLAGE (Villasclaras-Fernández, Hernández-Leo, Asensio-Pérez, & Dimitriadis, 2013) 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 (Sampson & Zervas, 2014). IMS-LD emerged as a standard for learning design (Consortium et al., 2003) and then focus shifted to tools such as LAMS (Dalziel, 2003) and LDSE (Laurillard et al., 2013a) that aim to support teachers. A vision paper aimed to create an approach that integrates most of these tools towards an integrated learning design environment (Hernández-Leo, Chacón, Prieto, Asensio-Pérez, & Derntl, 2013). 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 (Neumann et al., 2009).

*How to facilitate design of educational technologies to support a variety of instructional designs?*

*Section §2 of the paper discusses core guiding principles in computing towards a patterns based approach. We then present an overview of the source of patterns, their evolution and how to document them in Section §3. How patterns can help scale and variety is discussed in Section §4 followed by a patterns based approach for design of educational technologies in Section §5. Section §6 presents the idea of pattern-oriented instructional design with its sub sections proposing patterns for goals in Section §6.1, instructional process in Section §6.2 and instructional material in Section §6.3. We present Pattern-Oriented Software Architecture in Section §7 and discuss implementation of our approach in Section §8. We end the paper with conclusions and future work in Section §9.*

### 1.3 Patterns as a Solution

One interesting approach that has been undermined and largely unexploited in technology enhanced learning is the use of patterns and pattern languages for modeling instructional designs. The core idea of *patterns* and *pattern languages* is the encapsulation, modeling and delivery of expert’s knowledge and best practices to novices in a discipline. Essentially, patterns are derived from experiences and provide abstract representations of recurring solutions to recurring problems in a given context (Alexander, Ishikawa, & Silverstein, 1977). The roots of patterns are claimed to be in the field of architecture (Alexander et al., 1977) and extensively practiced in software engineering mainly for improving quality of software design and facilitating reuse (Buschmann, Henney, & Schimdt, 2007; Gamma, Helm, Johnson, & Vlissides, 1994).

There is also extensive work on patterns and pattern languages for different aspects of teaching and learning (Cristea & Garzotto, 2004; Goodyear et al., 2004). The Pedagogy Patterns Project was a major effort to capture best practices in the area of teaching and learning as a way to document best advices for teachers and support quality of instruction (Sharp, Manns, & Eckstein, 2000) (Bergin et al., 2012). The E-LEN project is another initiative aimed at

providing pedagogically-informed technology and experiences as pattern languages for new institutions mainly focusing on learning management systems (Avgeriou, Papasalouros, Retalis, & Skordalakis, 2003). A pattern language for creative learning is presented in (Iba & Miyake, 2010) and for adaptive learning in (Midgley, 2014). Laurillard has created pedagogical patterns using a design science approach (Laurillard, 2012). Patterns and pattern repositories for person centered e-learning were proposed in (Derntl & Motschnig-Pitrik, 2004). Another direction was to mine patterns derived from practitioner workshops were documented in (Mor, 2014).

While existing literature on patterns emphasizes the need for capturing best practices in teaching and learning, the focus has been mostly on pedagogy and technology aspects are largely ignored. Even in research that considered technology, there is a huge gap between domain (teaching and learning) and technology patterns motivating further research. *The key focus in this paper is to leverage the potential of patterns for modeling a scale and variety of instructional designs and use them as a base for design of educational technologies.*

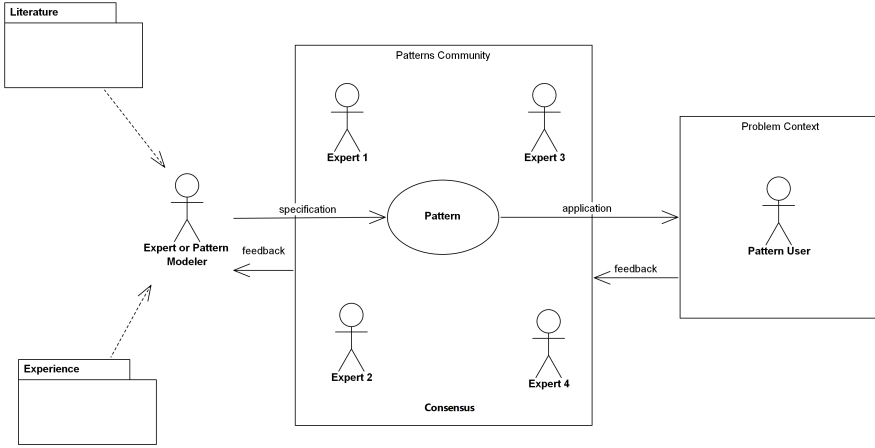
## 2 Guiding Principles

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

It is a dire necessity to improve flexibility and re-usability while reducing complexity during the design of educational technologies and some of these concerns are tackled by the software engineering community through a set of fundamental principles (Ghezzi, Jazayeri, & Mandrioli, 2002). One principle that is of interest to this paper is the notion of ***separation of concerns*** that helps in handling different dimensions of a system while improving re-usability and reducing complexity (Dijkstra, 1982; Greenfield, Short, Cook, Kent, & Crupi, 2004). This principle can be used to separate concerns in the form of layers either horizontally or vertically; views (Kruchten, 1995); modules (Parnas, 1972); aspects (Kiczales et al., 1997); patterns (Greenfield et al., 2004); features (Kang, Cohen, Hess, Novak, & Peterson, 1990) and so on. ***Modularity*** is a specialization of this principle that deals with separating software into components (Parnas, 1972) while ***Abstraction*** is another specialization that hides complexity of the system enabling designers to focus on specific concerns (Ghezzi et al., 2002). Domain-driven design is another fundamental principle that tries to address complexity by emphasizing domain as the basis for software design (Evans, 2004). We apply these principles throughout the paper for systematically modeling instructional design in the form of patterns. We discuss these principles further in Section §6. In the next section, we will summarize some key aspects related to patterns and get into the crux of our pattern-oriented design approach.

## 3 Source, Evolution and Structure of Patterns

**Source and Evolution of Patterns** - During the professional journey of an expert in a field, the expert encounters several recurring problems and different ways of solving those problems. When a group of experts in a particular field



**Fig. 1** Overview of patterns life cycle

communicate, discuss, debate and document their experience, they realize that certain solutions work and do not work in particular contexts. This leads to the idea that the primary source of patterns is literature or experience, either own or documented experience in the form of best practices and guidelines. Granularity of a pattern is another critical aspect that has to be considered during the design of patterns. For example, in the field of software engineering, there are coding idioms (a kind of patterns), design patterns, architectural patterns, each representing an increasing granularity of abstraction. Pattern languages help in connecting these patterns by describing the relationships between them and how they can be integrated to address specific problems. The agreement on whether a particular knowledge is pattern or not usually comes through a consensus among a group of experts in the particular field. The Hillside group has been sponsoring a series of conferences along with workshops named Pattern Languages of Programs (PLoP) since 1994 (Berczuk, 1994). These venues provide a forum for pattern authors to gather, discuss, learn and document patterns. This series has led to development of communities of patterns such as *AsianPLoP*, *EuroPLoP*, *ScrumPLoP* in order to create patterns with a consensus from local communities. A pattern typically goes through the phases of *discovery*, *specification*, *validation*, *application* and *maintenance* and is continuously revised based on updated pattern knowledge (Derntl, 2005). Figure §1 shows a typical process during pattern development. In the domain of education also, patterns have mainly emerged from participatory pattern workshops (Mor, Warburton, & Winters, 2012).

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

Alexandrian Form	Gang of Four Form	Pedagogical Patterns
<ul style="list-style-type: none"><li>• A picture</li><li>• **** denoting start</li><li>• A headline in bold type (essence of the problem)</li><li>• The body - background, motivation, variations</li><li>• The solution, in bold type</li><li>• A diagram for the solution</li><li>• *** as termination</li><li>• A paragraph that ties the pattern to all the <i>smaller</i> related patterns</li></ul>	<ul style="list-style-type: none"><li>• Name</li><li>• Classification</li><li>• Also Known As (optional)</li><li>• Motivation</li><li>• Applicability</li><li>• Structure</li><li>• Participants</li><li>• Collaborations</li><li>• Consequences</li><li>• Implementation</li><li>• Known Uses</li><li>• Related Patterns</li></ul>	<ul style="list-style-type: none"><li>• Name</li><li>• Thumbnail</li><li>• Audience/context</li><li>• Forces</li><li>• Solution</li><li>• Discussion/ consequences/ implementation</li><li>• Special resources</li><li>• Related patterns</li><li>• Example instances</li><li>• Contraindications</li><li>• References</li></ul>

Fig. 2 Diversity of structure of patterns

**Structure of Patterns** - There are patterns everywhere and at different levels of granularity expressed in different ways by different experts in different communities (Meszaros & Doble, 1998). This presents the challenge of what constitutes pattern knowledge and how to capture it. Traditionally, patterns are captured as a description of *context*, *problem*, *solution*. However, several researchers have proposed different structures for capturing pattern knowledge. Figure §2 shows examples of commonly used pattern structures from the domains of software engineering (Gamma et al., 1994) and pedagogy (*The Pedagogical Patterns Project*, 2014). Alexandrian form is the most commonly used format for representing patterns (Alexander et al., 1977). Using this form, Alexander has documented a pattern language comprising of 253 patterns for the domain of towns, buildings and construction (Alexander et al., 1977). The 23 Gang of Four (GoF) patterns are the most commonly used patterns in the domain of software design (Gamma et al., 1994) and use the meta data as shown in Figure §2. The Pedagogical Patterns project uses the structure shown in Figure §2. These are just a few examples of pattern structures used by different communities. A pattern language for representing patterns itself has emerged from the community (Meszaros & Doble, 1998).

4 Patterns for Scale and Variety

Figure §3 shows a detailed pattern structure derived from existing pattern representations. The key idea of this pattern structure is to capture as much information as possible such that this metadata could be used for searching and managing patterns and pattern repositories. In addition to the textual information as shown in Figure §3, patterns could be represented visually focusing on *structure* as well as *behaviour*. However, one critical requirement for patterns in this paper is to facilitate domain experts to model pattern knowledge without overburden.

In his seminal work on patterns, Alexander emphasizes that a solution in pattern can be used “a million times over without ever doing it the same time



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

**Fig. 3** A detailed pattern structure

*twice*". We extend this notion to include not just the solution but also the problem, its variations, representations of the pattern, different aspects of the solution and its variants to facilitate scale and variety.

There are several possibilities of creating variants of patterns. The simplest case being that a pattern can be instantiated multiple times as shown on the left hand side of Figure §4 or instantiated with simple variations as shown on the right hand side. In Figure §4, we show a simple example of different kinds of possible relationships between two patterns in a domain. If we change these relationships, we get a different view of the pattern in context leading to a variation. Similarly, patterns can be composed using different operators as shown in Figure §4. Changing the operators results in new composed patterns and variations. If we move beyond just two patterns and consider a large number of patterns, then different ways of composition with multiple operators leads



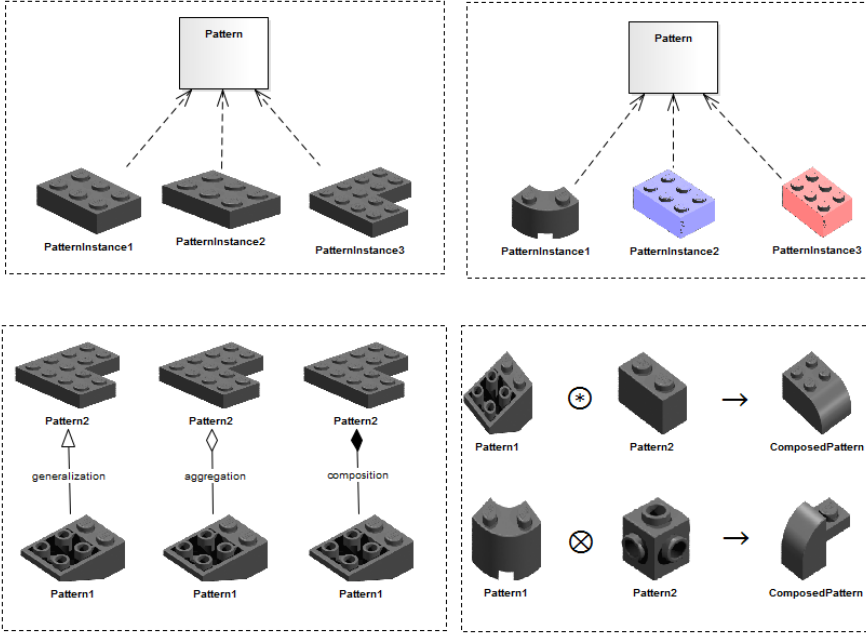


Fig. 4 Some pattern variations, relationships and compositions

to different ways of modeling the domain. Consider the case of an instructional design with 10 patterns, then different compositions of these patterns can lead to several instructional design variants. This leads to creation of varied instructional designs catering to the needs of specific requirements. The essence of this discussion is to emphasize that modeling domain in terms of *patterns* facilitates systematic creation of several variants, which is one of the primary goals of this paper. We will elaborate more on these patterns with examples throughout the paper. But formalizing the representation and composition of patterns is beyond the scope of this paper and is outlined as future work.

## 5 A Patterns Based Approach

The notion of patterns has its roots in the field of Architecture (Alexander et al., 1977) but was adopted in other disciplines such as software engineering (Gamma et al., 1994) and interaction design (Borchers, 2001) among others. Whilst there exists several definitions and views, *patterns are primarily concerned with the idea of finding recurring solutions to recurring problems in a certain context*. According to Alexander, the emphasis has to be on pattern languages that facilitate the assembly of patterns in order to create numerous possible solutions, rather than patterns themselves (Alexander et al., 1977). Buschmann et al. have distilled existing literature on patterns and proposed the following uses (Buschmann et al., 2007; Buschmann, Meunier, Rohnert, Sommerlad, & Stal, 1996):

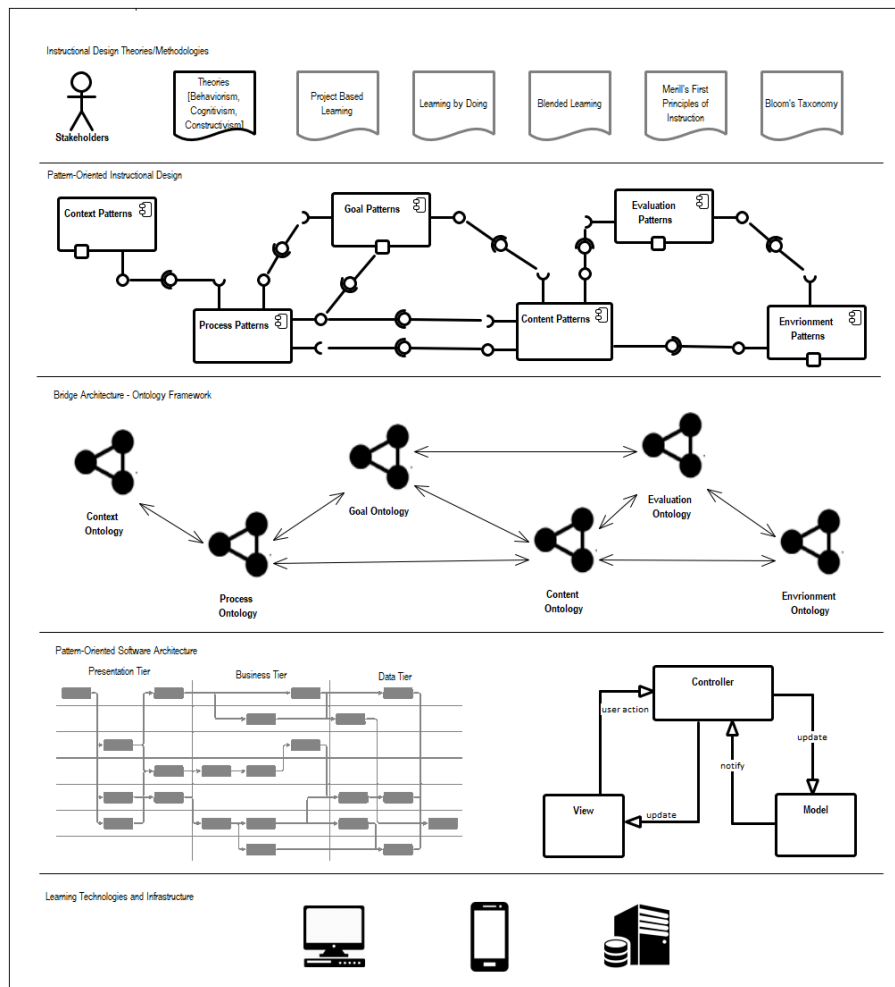
- *Capturing, Documenting and Communicating Experience*
  - Patterns document existing best practices built on tried and tested design experience
  - Patterns identify and specify abstractions that are above the level of single objects, classes, and components
  - Patterns provide a common vocabulary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

In this paper, we apply the idea of patterns primarily for (i) *capturing experience* (ii) *providing instructional design as basis for educational technologies* (iii) *facilitating reuse during design of educational technologies*. Specifically, we use patterns for modeling domain (instructional design) and software. We also look at patterns as a central way to encapsulate commonly understood knowledge of experts (instructional designers, software architects) and facilitate use of this experience by naïve professionals. This is extremely important in a discipline like Technology Enhanced Learning (TEL) with huge scarcity of expert teachers at all levels of education. Figure §5 presents the core architecture of our proposed patterns based approach to design of educational technologies. This architecture stems from fundamental principles in software engineering and integrates multiple architecture styles (Layered, Domain-Driven and Component-Architecture). This architecture shows a holistic perspective (top-down and bottom-up) and tries to integrate patterns in both domain as well as software through five layers.

We briefly explain the core design principles and different layers of our approach in the next sections.

## Core Design Principles

**Separation of Concerns (SoC) Principle** - SoC principle was applied in (Caeiro-Rodríguez, Llamas-Nistal, & Anido-Rifón, 2006; Rodríguez-Artacho & Maillo, 2004) for modeling instructional design but an analysis of this work reveals that this principle was applied either in modeling instructional design or in creating technology and not both, which we aim to address through our approach. As shown in Figure §5, we continuously applied SoC principle at different levels (low-level, high-level and pattern approaches) in instructional design and software as a way to address complexity, evolution and reusability of instructional design. We also applied the notion of abstraction as an under-



**Fig. 5** Architecture of patterns based approach

lying principle of SoC to separate technology specific aspects from technology independent aspects through bridge architecture as shown in Figure §5.

**Design for Reuse and Customization** - It is extremely effort intensive to design educational technologies for scale and variety. Learning from the negative impact of ad hoc reuse in software engineering, our approach in Figure §5 is explicitly designed for systematic reuse emphasizing a patterns-based approach. Every aspect of this architecture is modeled with explicit interfaces using required dependencies, provided services, open ports, assemblers and connectors. This ensures that each pattern makes its assumptions and capabilities explicit enabling systematic reuse.

**Domain Driven Architecture Design** - According to this architecture style, the design of software systems is essentially a realization of the under-

lying domain that is modeled by experts in that domain. In Figure §5, POSA is driven by POID emerging from domain. By design, POID and POSA are closely mapped via patterns in instructional design and software engineering. In addition, this architecture follows a layered style with several layers from the top dealing with domain and layers from the bottom dealing with technology. Even though the layers shown in the diagram seem to be fixed, the number of layers can be increased or decreased based on the application type, quality attributes of desired system, and technology constraints among others. Bridge architecture (message bus architecture style) defined via well defined interfaces allows better communication between the layers.

- A *Instructional Design/Methodologies* - lays a pedagogical foundation for design of educational technologies - Educational experts have long emphasized that developing educational technologies without strong instructional basis is futile and can lead to poor quality of instruction (Govindasamy, 2001; Laurillard, 2013a). Most of the educational technologies today require huge effort from teachers in configuring the technologies rather than on focusing instruction (Laurillard et al., 2013b). This is further aggravated with a huge dearth of qualified teachers. The first layer from the top in Figure §5 focuses on providing a pedagogical foundation for design of educational technologies. We rely on well-established principles and practices from a pedagogical perspective in this paper and specifically focus on how we can structure these practices towards systematic design of educational technologies. For the case of adult literacy in this paper, we rely on Improved Pace and Contents of Learning (IPCL) approach (*Handbook for Developing IPCL Material*, 2003) as a pedagogical foundation. This pedagogy is further integrated with other commonly accepted approaches such as Merrill's principles of instruction from teaching or instructional process perspective (Merrill, 2012) and Bloom's taxonomy from learning perspective. The rationale for these principles is presented in Section §6.1 and Section §6.2 of patterns.
- B *Pattern-Oriented Instructional Design* - The goal of this layer to model instructional design using patterns (Section §6)
- C *An Ontology Based Modeling Framework* - acts a bridge between domain and software platforms - The key purpose of this bridge layer is to connect from instructional design (domain) to educational technologies (software) through a common interface. We used ontologies as the primary mechanism to represent knowledge from patterns and to connect with the rest of the software architecture (Chimalakonda & Nori, 2013b). We extended the existing literature on instructional design, and proposed an ontology based framework called IDont for systematically modeling instructional design (Chimalakonda & Nori, 2013b).
- D *Pattern-Oriented Software Architecture* - models the software architecture of educational technologies using patterns - In their seminal work that inspired successful use of patterns in software engineering, Buschmann et al. emphasize that the primary purpose of architectural patterns is to cre-

ate a fundamental structural organization schema for software systems (Buschmann et al., 1996). In our approach, instructional architecture patterns derived from POID and software architecture patterns that drive POSA provide this structure for instructional design and educational technologies. We use common interfaces to bridge these different types of patterns at different levels of abstraction (*instructional architecture patterns*  $\rightarrow$  *instructional design patterns*  $\rightarrow$  *software architecture patterns*  $\rightarrow$  *design patterns*). Essentially, POSA represents the architecture of educational technologies. In the fourth layer and on the left hand side of Figure §5, we have a three-tier architecture of a TEL system that implements two important architectural patterns from (Buschmann et al., 1996) (i) From Mud to Structure (Layers) – allows controlled decomposition of overall system (ii) MVC and several design patterns (*Strategy*, *Composite*, *Factory* and so on). On the right hand side is an MVC architectural pattern that is derived from instructional architecture patterns and instructional design patterns in POID.

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

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

## 6 Pattern-Oriented Instructional Design

The changing landscape of educational technologies requires a diversified range of instructional designs catering to multiple perspectives and views from a diversified range of stakeholders (Reigeluth, 2013b). This is primarily because there is no “one-size-fits-all” solution for all needs. In fact, even the problem itself varies depending on who is viewing it, where does it come from, how critical is it? what are the short term and long term needs? how to address them? their capabilities and resources available at that point of time. This presents a definite need for systematic instructional design such that it can be flexibly modified as per changing requirements. Several researchers have tried to address these concerns through a number of Educational modeling languages (EMLs) (Botturi et al., 2008; Caeiro et al., 2014; Dalziel, 2003; Hernández-Leo et al., 2013; Hernández-Leo, Moreno, Chacón, & Blat, 2014; Koper, 2005; Rodríguez-Artacho & Maillo, 2004; Villasclaras-Fernández et al., 2013). Despite significant research, modeling instructional design remained an open research problem with several challenges (Burgos, 2015; Goddard, Griffiths, & Mi, 2015; Hernández-Leo et al., 2013; Neumann et al., 2009). Our analysis of the state-of-the-art in learning design also revealed that the goal of end-to-end automation (from pedagogy to technology) through tools has resulted in too generic, too complex specifications leading to slow progress in this field and is in line with existing research (Laurillard, 2013a; Laurillard et al., 2013b).

Instead, we learn from the community and attempt to present a pointed approach towards design of educational technologies for a family of instruc-

tional designs in the context of adult literacy in India. *Our work fundamentally deviates from the state-of-the-art as we focus on not one instructional design but on a family of similar but distinct instructional designs.*

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

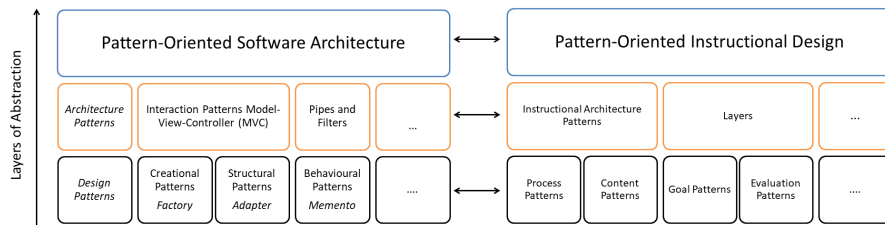
- Existing approaches focus on patterns and pattern languages either for communication or engineering purposes and not both
- Current approaches focus on patterns mainly from a pedagogical perspective rather than their structure and provide minimal support towards design of educational technologies
-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

To summarize, existing approaches focus on patterns either in domain or in software and not both. Most importantly, none of the existing works focus on *scale* and *variety*, which is the main goal of this paper. Paquette has summarized the extensive work and progress in the field of instructional design and concluded the strong need for researching and applying ontological engineering and software methods for instructional design and engineering (Paquette, 2014). On the other hand, from a software engineering perspective, domain engineering is a critical activity to address complexity, reuse and evolution needs of software systems (Taylor, Medvidovic, & Dashofy, 2009). In this paper, we take a cue from Apel et al. (Apel, Batory, Kästner, & Saake, 2013) and Czarnecki et al. (Czarnecki & Eisenecker, 2000) and consider domain as an area of knowledge:

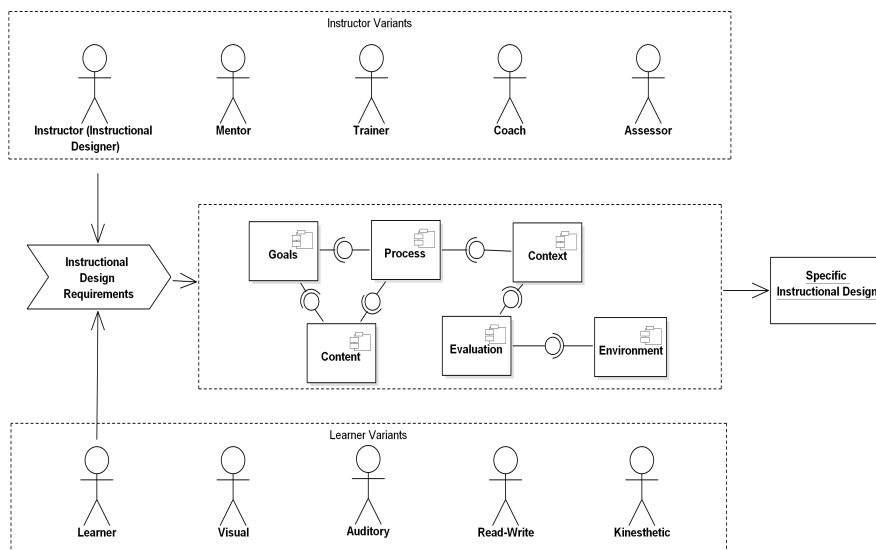
- that covers the desired requirements of the systems in that area
- includes a set of concepts and terminology understood by practitioners in that area
- and includes the knowledge of how to build software systems (or parts of software systems) in that area

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

We base the design of POID on Pattern-Oriented Software Architecture (POSA) (Buschmann et al., 2007) to model instructional design aspects as patterns. Figure §6 shows a high level diagram of how POID corresponds to



**Fig. 6** Comparison of Pattern-Oriented Software Architecture and Pattern-Oriented Instructional Design

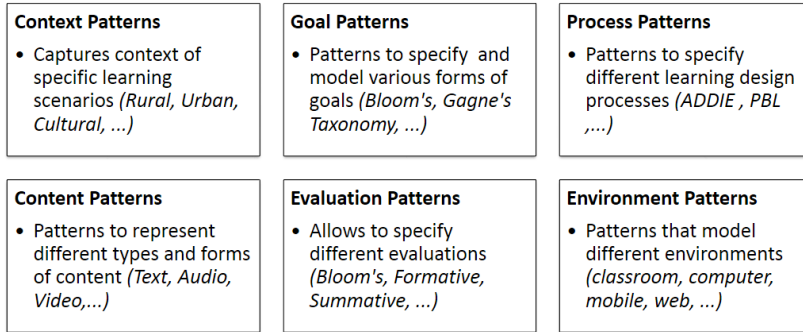


**Fig. 7** An overview of Pattern-Oriented Instructional Design

POSA with different levels of granularity. For example, on the left hand side, we have design patterns at the bottom and then architecture patterns. On similar lines, we have instructional design patterns at the bottom and instructional architecture patterns on top of them.

Figure §7 shows an overview of Pattern-Oriented Instructional Design. The core instructional design requirements stem from different kinds of instructors and learners, which are then used as input to compose different patterns such as *goals*, *process*, *content* and so on to create a specific instructional design. A *Pattern-Oriented Instructional Design* is an integration of patterns at instructional architecture level and instructional design patterns towards designing a specific instructional design for a specific set of educational requirements. Here, we consider instructional architectural patterns as high-level organizing structures that address recurring high-level problems in instructional design. For example, Can we have a pattern that allows different evaluations for the





**Fig. 8** A classification of patterns in instructional design

same instructional goals? These patterns are essentially an integration of instructional design patterns, which focus on a specific aspect such as goals or process and so on. One popular example of an architectural pattern for interactive systems in software engineering is the Model-View-Controller (MVC) that allows flexible design of user interfaces (Krasner, Pope, et al., 1988). Can we have these kinds of patterns for instructional design? It is here our POID approach is a direction integrating patterns at a higher level of abstraction.

The POID process starts by understanding instructional design and then progresses towards a detailed analysis of the instructional design to identify patterns in the domain. Then these patterns are continuously refined and relationships between them are identified and established. Figure §8 shows a classification of patterns in POID into categories of *Context*, *Goals*, *Process*, *Content*, *Evaluation* and *Environment*. This list is primitive and is designed such that it can be adapted and extended by instructional designers to support evolution. Each of these categories have patterns that aim to address a particular aspect of instructional design. For example, *Goals* in instructional design can be represented in various forms such as Bloom's or Gagne's taxonomy or ABCD technique, and each of them can have their associated patterns, from which the teacher or instructional designer chooses patterns for their particular context. In our approach, every pattern provides and requires a set of interfaces that clearly define the boundaries of that pattern and how that pattern communicates with the rest of the patterns. The focus in this paper is not just on patterns or pattern categories, but on integration of these patterns towards specific instructional designs.

Figure §5 shows two possible instructional architecture patterns; one that integrates goals and evaluations (goal provides an interface  $\leftrightarrow$  evaluation requires an interface). The core idea of this pattern is to provide a flexible architecture that allows changing of goals and evaluations in an instructional design with less effort. Another possible pattern could be an integration of processes and content. This kind of abstraction leads to POID, which in itself

is a technology independent solution in the language of problem domain. However, it is important to note that patterns were never proposed to be specific and complete solutions rather provide a basic structure of a generic solution to a family of problems (Buschmann et al., 2007), which have to be further adapted and implemented for a specific context. Finally, Alexander hints that the problem of scale and variety can be addressed using patterns in his seminal work (Alexander et al., 1977) as:

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

This emphasizes that patterns provide an opportunity to find a generic solution to a problem and then find several customized solutions to a family of similar problems. Even though patterns are generic in nature, they have to be discovered from literature and experience for a particular context. In this paper, we use our decade-long experience in the case of adult literacy in India. *We conducted a workshop in collaboration with Tata Consultancy Services and in association with NLMA to discover our patterns with the community towards a consensus*<sup>4</sup>. The workshop consisted of directors of State Resource Centers representing officers who are responsible for creating instructional process and material for teaching adult illiterates based on local requirements. Specifically, we introduced our patterns for instructional process and content and incorporated their feedback. We also held interviews with the directors of SRCs to figure out their use of technology for adult literacy especially to gather what kinds of technologies have been accepted and what are the hindrances to the use of technologies. One key learning that came out of that workshop is a strong requirement that the technology has to be customized as per local needs. One director specifically noted that “the instructional process that you follow in a state like *Bihar* and the one in a state like *Tamilnadu* differ and the technology has to be adapted as per varied needs”. In addition to our experience, we also rely on commonly accepted approaches such as Bloom’s taxonomy (Anderson et al., 2001), Merrill’s first principles of instruction (Merrill, 2012) as a basis for the patterns proposed in this paper.

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

## 6.1 A pattern for modeling goals

We assume that any instruction is *goal-driven*; making it critical to explicitly state instructional goals. Several terminologies such as “learning goals”, “learning outcomes”, “learning objectives” have been in use to discuss goals. But the crux is to explicitly state and represent these goals such that they become clear to different stakeholders such as teachers, learners, policy makers and so on. In our case, for design of educational technologies while adhering

<sup>4</sup>at TCS, Hyderabad, India in 2011

to instructional design principles, they can also help in tracking the progress of learners, evaluation and in helping teachers to improve instructional strategies. We advocate a *goal-driven approach* throughout this paper as it is critical and essential for any instructional design.

For example, if we consider the representation of goals in instructional design, several variations are possible. In the simplest case, a goal can be represented using Bloom's taxonomy and several instances of this goal can be

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

**Fig. 9** A sample structure and partial instance of *GoalsPattern*

created. If we extend the scenario and think of two goals as part of instructional design, the two goals can be associated in several ways.

The crux of most commonly used Bloom's taxonomy is to provide a classification for organizing educational objectives based on thinking models and to facilitate better communication among stakeholders in education (Anderson et al., 2001). This taxonomy has evolved since its inception but was originally divided into three domains: *cognitive*, *affective*, *psychomotor* with the first two domains focusing on acquiring knowledge and attitude and the third domain on skills to put that knowledge to constructive use (Anderson et al., 2001). Each of these domains are further divided into levels that indicate progress "*from simple to complex and concrete to abstract*" (Anderson et al., 2001). In revised Bloom's taxonomy, goals are organized in the increasing order of complexity in six levels: *remember*, *understand*, *apply*, *analyze*, *evaluate*, and *create* from a cognitive domain perspective. ABCD model proposed by R. Mager is another commonly used framework for writing learning objectives (Mager, 1962). In this model, a learning objective consists of four components: [*audience*- who] will be able to [*behavior*-perform] [*condition*- constraints] [*degree* - level of quality]. The point of this discussion is to emphasize that there are several ways of modeling goals motivating the need for several patterns for goals in different contexts. In the context of this paper, based on these inputs and our experience of designing eLearning Systems for adult literacy in India, we

	<i>Variation 1</i>	<i>Variation 2</i>	<i>Variation 3</i>
<i>Reading</i>	Read and write words and sentences having most frequent letters and vowels	Read and write words and sentences having almost all letters, vowel signs	Read and write words and sentences having any letters, vowel signs
<i>Writing</i>	Write one's name	Write family members names and address	Comprehend a simple and small unknown passage or text, newspaper, headings, road signs etc.
<i>Numeracy</i>	Read and write numbers up to 50 Do simple sums and subtraction without 'carry over'	Read and write numbers up to 1000	Read and write numbers up to 1000
		Do simple sums and subtraction with 'carry over'	Do sums and compute simple problems involving multiplication and division within 1000
		Do small sums in currency, weights and measures of decimal systems	Apply skills of reading, writing and numeracy in day-to-day life activities i.e., read and write letters, sign boards...

**Fig. 10** Examples of goals in adult literacy

propose a pattern to model goals based on these inputs. Figure §9 shows our proposed pattern for modeling goals based on Bloom’s taxonomy. Figure §10 shows some examples of goals for adult literacy based on IPCL methodology and Bloom’s taxonomy.

These goals can be refined for varied lessons as part of instructional design and can be customized for specific multiple Indian languages.

## 6.2 A pattern for modeling instructional processes

Instructional process is one of the critical aspects of instructional design as it facilitates the fulfillment of goals through a systematic process. However, most of the times it is not explicitly modeled by making it difficult for design of educational technologies. In this section, we will look at a commonly accepted way of teaching in the context of adult literacy in India based on IPCL methodology (*Handbook for Developing IPCL Material*, 2003) and present a structure for organizing that knowledge into a pattern. We discuss the instructional process in detail along with teaching philosophy as it forms the basis for a pattern that could be instantiated thousands of times for all Indian languages and dialects.

Figure §11 shows organizational structure of a *lesson* using the *pasi* pattern (Chimalakonda, 2017). The number and order of the plays, acts, scenes, instructions is not strictly fixed even though guidelines can be framed. For example, the first *play*, *act* and *scene* focus on providing motivation to the learner and the last *instruction* might be a summary of what has been learnt so far in a particular lesson. Figure §12 shows few examples of *acts* and some *scenes*. In this example, there are several *acts* each having its respective goals, and consisting of specific *scenes* and further *instructions*. For example *Act4* deals with the goal of teaching how to form new words from syllables with two

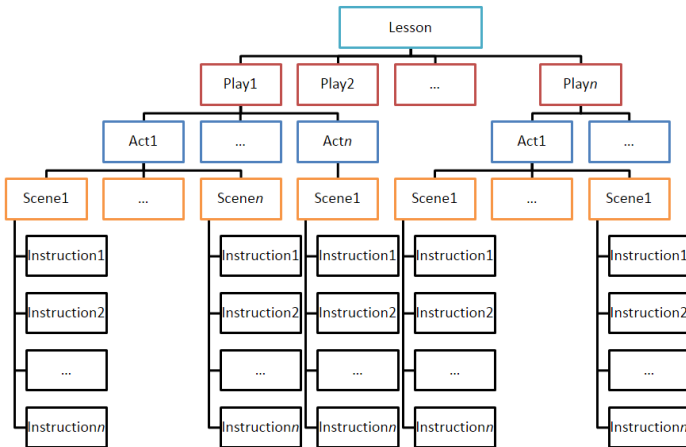
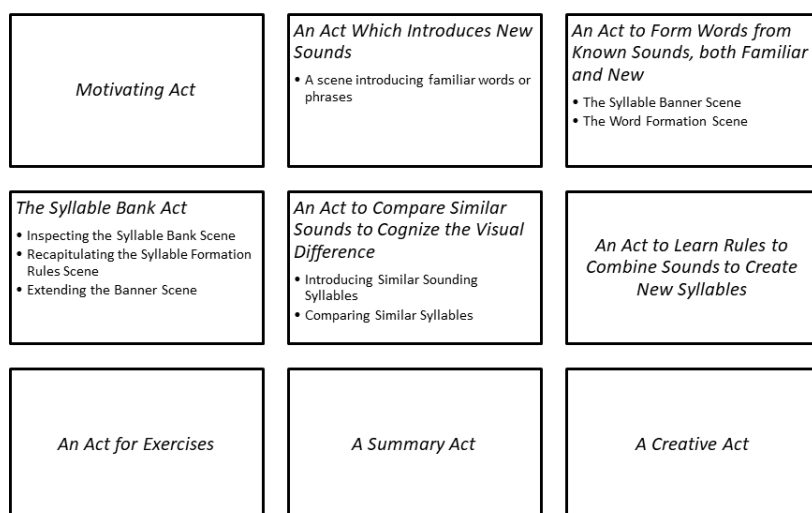


Fig. 11 Structure of instructional process

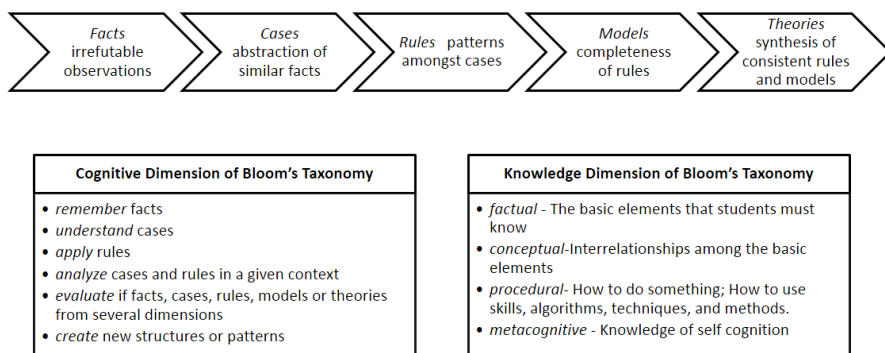


**Fig. 12** Example *acts* in adult literacy instructional design

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

While the *ProcessPattern* provides a goal-driven structure for modeling a pedagogy at a high-level, it does not include a strong philosophical attitude of instructional process and does not specify how these goals have to be achieved. It is here we analyzed the literature in instructional design and found that there are several perspectives of instructional design. There are also several ways of modeling instructional processes based on different instructional design theories or methodologies. Merrill has analyzed existing instructional design models and proposed that the following fundamental principles are critical to any instructional design (Merrill, 2012).

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



**Fig. 13** Pattern structure and mapping to cognitive and knowledge dimensions of Bloom's taxonomy

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

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

### 6.3 A pattern for modeling instructional material

Instructional material is at the center of instruction and we discovered a pattern for content as shown in Figure §13. This pattern is primarily derived from IPCL, scientific method but most importantly driven by our future need to facilitate reasoning in the subject. Can learners provide rationale and reasoning for their answers?

Figure §13 shows the progression of content from simple facts to be remembered to foundations in the subject. The mapping of this pattern to cognitive and knowledge dimensions of Bloom's taxonomy is also shown in the figure. We discussed the foundations of this pattern during its formative stages in (Chimalakonda & Nori, 2012). Figure §14 shows instances of the *ContentPattern* in *Hindi*, *Telugu* and *Gujarati* languages.

The *ProcessPattern* and *ContentPattern* are closely connected and can be considered as an instructional architecture pattern. In this pattern, teaching is structured, not simply as a sequence of lectures, but as a sequence of *models*



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

**Fig. 14** Example of *ContentPattern* instances for *Hindi*, *Telugu* and *Gujarati* languages

as needed by a *theory* to be taught. Here, we consider teaching as a *play* uncovering each *model* as the instruction unfolds, with learner participation! Each *model* in turn would be a sequence of topics. Uncovering a *model* is done in a succession of *acts*, with each *act* uncovering a topic pertinent to some *model*. Each *act* is presented as a succession of *scenes*, with each *scene* focusing on *cases* or *rules*. Finally, each *scene* is delivered as a succession of basic *instructions*, which uncover and play with *facts*.

## 7 Pattern-Oriented Software Architecture

Software architecture deals with most of the design decisions concerned with *structure*, *behavior*, *interaction*, *qualities*, and *implementation* of a software system (Taylor et al., 2009). In this paper, we are interested in identifying the variants that are possible in each of these aspects to support the desired requirements of scale and variety in educational technologies. In general, architectural patterns and reference architectures provide a baseline and a set of guidelines for creating specific software architectures tapping the variabilities of multiple related systems in a particular domain (Taylor et al., 2009). At a high-level, customization of software can be done at design level, requiring explicit modifications to architecture and design of software and also from a user perspective, requiring configurations (Mørch, 1997). Morche (Mørch, 1997) lists three common ways of tailoring software applications from literature:

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

In *integration*, users can have configuration options to integrate functionality that is outside the system. For example, a teacher might want to integrate learning management system like *Moodle* into the current system. This might require tweaking of components, extending interfaces and fixing interoperability issues. A GLUE-architecture was proposed to support integration of tools into virtual environments (Alario-Hoyos et al., 2013).

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

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

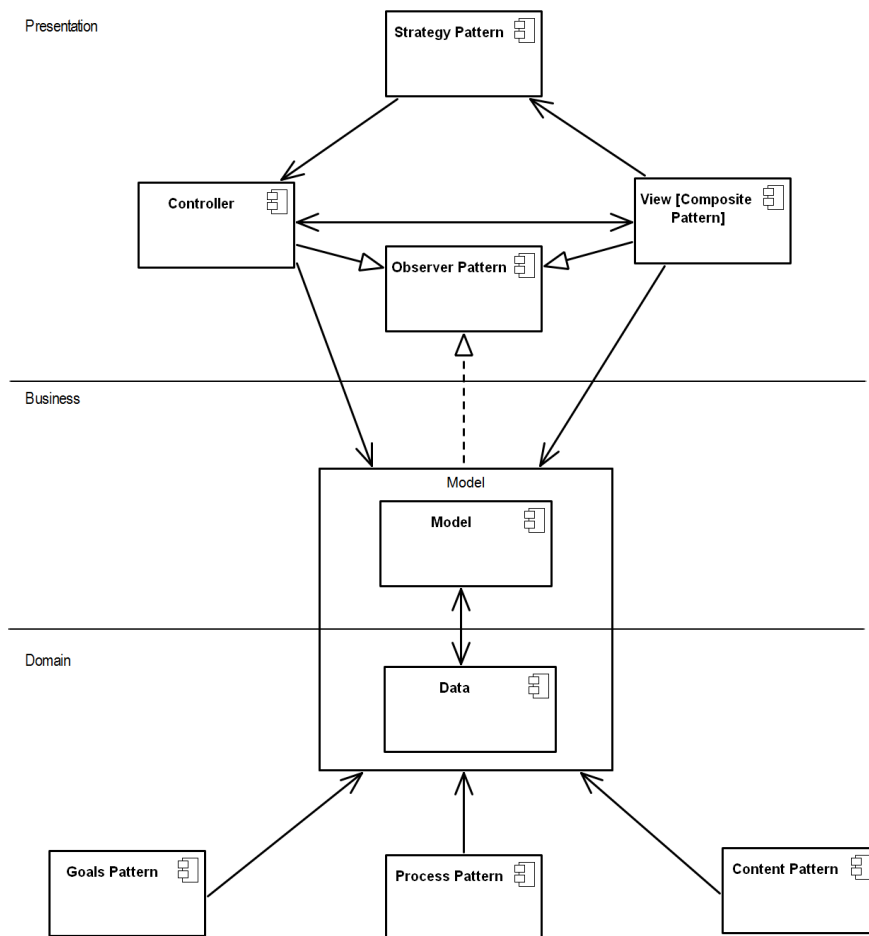
The core idea of POSA is to create software architecture using patterns such that these patterns can address variability and map to patterns in POID. Architecture patterns can further consist of design patterns with each of them addressing variability at different levels of granularity. In this section,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

We illustrate the idea of POSA through a commonly used pattern called Model-View-Controller (MVC). Kraner and Pope have described MVC for interactive user interface applications (Krasner et al., 1988). The key purpose of MVC pattern is to facilitate separation of the interactive application into three parts or components to efficiently address changing requirements. Models primarily represent the underlying application domain knowledge and act as core structure for Views and Controllers. Instructional design is the underlying domain in this paper and as such forms the basis for models. The representation of this model itself can vary based on how the domain is modeled. As emphasized by Kraner and Pope, the focus should be on modeling specific information about the application domain such that it can drive the other two parts. Views primarily focus on the user interface, graphical elements and what the users view as part of the system. A typical user interface consists of hierarchical views and the data for these views is fed from the model behind them. This is quite useful as a model can have many views associated with it facilitating variability from a user interface perspective. Consider a scenario where an *eLearning* System uses the same model but can be viewed using several user interface metaphors. The variabilities can range from simple color or themes to complex modifications emerging from instructional design. The order of these views itself can be a source of variability. Controller is the third

component of MVC pattern that acts as the interface between models and views.

It is not uncommon to integrate several architectural patterns while designing a software application to address varied requirements (Buschmann et al., 2007). For example, Figure §15 shows a simple overview of how MVC pattern can be integrated into a layered architectural pattern. Here the user interface, business logic and data are separated into three layers Presentation, Business and Domain. MVC pattern is spread across *presentation* and *business* layers. The MVC pattern itself is a composed pattern consisting of several design patterns and can be implemented in several ways leading to variations like Hierarchical MVC, PVC (Karagkasidis, 2008). Several design patterns are used to implement MVC and its variations. The model part of the MVC pattern is part of the domain strongly mapped to data parts of the domain i.e., pattern-oriented instructional design. A simple way to vary the application is to change these data parts in the POID without changing the interfaces exposed to POSA. Even MVC itself uses *Observer* pattern to notify views and controllers, providing a variation point. Composite pattern is commonly associated for constructing user interface elements in a hierarchical way and exposing only the top level view for the entire architecture leaving flexibility to change user interface elements. *Strategy* pattern is commonly used to alter between different controllers and use concrete strategies at different points of time to facilitate different behaviors for different triggers in the software application. One key difference in this layered MVC architecture as shown in Figure §15 is the use of components instead of classes emphasizing the need to model every class as a component with explicit interfaces to facilitate variability. The domain layer emphasizes the strong role of domain in this architecture than just traditional databases. The data is mapped to patterns in the domain as this can allow traceability of changes from domain to software. A 10-step process for MVC pattern (Buschmann et al., 1996) along with potential variabilities is shown below:

1. Separate human-computer interaction from core functionality  
*inputs, output behaviours, accessor functions*
2. Implement the change-propagation mechanism  
*publish-subscribe pattern, specific implementations*
3. Design and implement the views  
*appearance of views, display procedures, parameterized views, multiple draw methods*
4. Design and implement the controllers  
*specific behaviours for user actions, event handling*
5. Design and implement the view-controller relationship  
*initializations for which factory method pattern could be used, hierarchy of views and controllers*
6. Implement the set-up of MVC  
*initializations, events*



**Fig. 15** Layered Model View Controller Pattern

7. Dynamic view creation  
*components for managing views*
8. Pluggable controllers  
*different controllers*
9. Infrastructure for hierarchical views and controllers  
*composite pattern, chain of responsibility pattern*
10. Further decoupling from system dependencies  
*bridge pattern, higher levels of abstraction*

Today, there are several web frameworks such as *Django*, *Symfony*, *Rails* that are based on MVC pattern but most importantly implement their own variation, addressing specific requirements. Most of the variations are possible by tweaking one or more of the steps in the above process. For example, there

could be hundred controllers in a large scale application providing variability or the need to create varied interaction themes with users can be addressed by tweaking step 9. According to (Buschmann et al., 2007), MVC pattern references twelve design patterns such as *Facade*, *Observer* and so on.

POSA can also be designed by integrating several design patterns. Gamma et al. list 23 design patterns and their relationships in a single diagram (Gamma et al., 1994) and Zimmer provides a succinct classification of relationships between various design patterns into three layers: [design patterns specific to an application domain], [design patterns for typical software problems] and [basic design patterns and techniques] (Zimmer et al., 1995). To summarize, the core essence of this paper is to show how varying different aspects in patterns can facilitate variability needed for scale and variety in educational technologies.

In the next section, we discuss an implementation of our approach that uses patterns as the base for instructional designs and educational technologies.

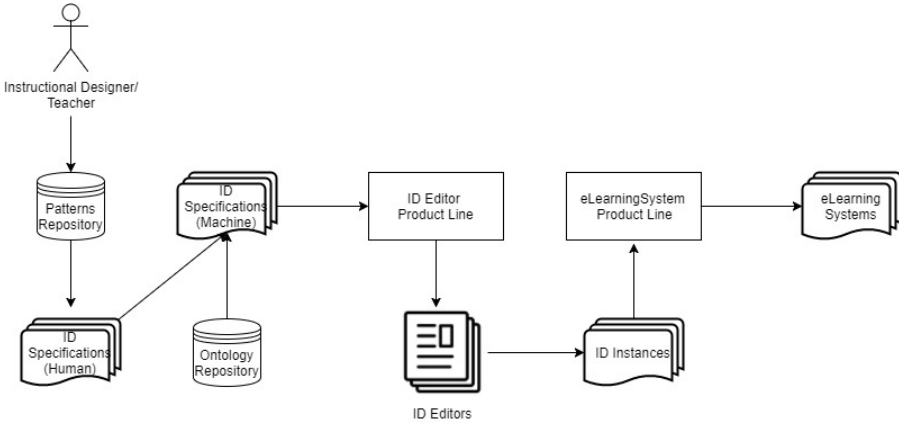
## 8 Implementation

Owing to the nature of our research work closely connected to the idea of solving societal challenges using computing, the implementation of our research work is made available at <http://rice.iiit.ac.in> and the work has been transferred to National Literacy Mission Authority, Government of India for further proliferation. The mobile version of the generated software is deployed on Google Play Store and is available at <https://play.google.com/store/apps/details?id=iiit.rice.al.telugu>. In addition, our work is also listed in the official websites of Department of Adult Education of Government of Telangana at <http://tslma.nic.in/> and State Resource Center, Government of Telangana at <http://srctelangana.com/>. In this section, we will provide an overview of our implementation.

Our initial implementation was a manual approach based on patterns in instructional design. We have designed an authoring tool called *EasyAuthor* that helps non-technical teachers to create educational content (Chimalakonda & Nori, 2013a) based on pedagogy patterns through our IDont framework (Chimalakonda & Nori, 2013b). This tool has wizards that help teachers to create different aspects of ID (context, goals, process, content, evaluation and environment) and eases the authoring process. Each of these wizards are based on patterns for corresponding aspects of the instructional design. This essentially connects different components of educational technologies to instructional design.

However, a critical need is to create a variety of authoring tools based on a variety of instructional designs. For example, there is a need for *EasyAuthor1* mapping to an *InstructionalDesignModel1*, *EasyAuthor2* for *InstructionalDesignModel2* and so on. To address this need of automating the development of *EasyAuthor(N)* tools for varied *InstructionalDesignModel(N)*, we have developed a software product line approach based on ontologies (Chimalakonda, 2017).

Figure §16 shows an overview of the implementation of our approach. Here, an instructional designer first decides the instructional design model for a set



**Fig. 16** Overview of Implementation of Pattern-Oriented Approach

of courses that have to be taught. For example, the instructional designer may choose to use “project based learning” as a strategy to teach several courses. He creates a set of patterns for aspects such as *goals*, *process*, *content* either from scratch or by customizing existing patterns to suit the specific instructional design model. However, this model is mainly aimed at different stakeholders such as teachers, instructional designers, evaluators, policy makers and so on. These instructional design specifications have to be converted to specifications that are machine-processable such that automation is possible. To achieve this, we have used ontologies to concretely represent the patterns (Chimalakonda, 2017). These specifications are read by a tool called *ID Editor Product Line* that semi-automatically generates an “*ID Editor*” that essentially allows a teacher to create a specific instance of instructional design. This *ID Editor* is driven by the idea of connecting patterns in Pattern-Oriented Instructional Design and Pattern-Oriented Software Architecture. The *ID Instances* are then used as input to another product line for generating *eLearning Systems*. The details of how these product lines are implemented is beyond the scope of this paper and are detailed in the thesis (Chimalakonda, 2017). Further details on implementation and source code of implementation are available through <http://rice.iiit.ac.in>.

## 9 Conclusions & Future Work

We emphasized that lack of instructional design base is a critical challenge for design of educational technologies. So is the challenge of facilitating reuse and supporting a variety of instructional designs. To address these concerns, we presented a patterns-based approach that integrates patterns in instructional design and educational technologies. This approach has its roots in fundamental architecture principles in software engineering. Based on these principles, we presented an architecture that integrates *Pattern-Oriented Instructional Design* that is driven by instructional design methodologies and Pattern-Oriented Software Architecture that drives the design of educational

technologies. The essence of our approach is to systematically model different aspects of instructional design (*goals, process, content*) using *patterns* such as *GoalsPattern*, *ProcessPattern* (plays, acts, scenes, instructions) and *ContentPattern* (facts, cases, rules, models and theories). We demonstrated the application of our approach to model patterns in adult literacy case study in India. We then provided an implementation of our approach that generates instructional design authoring tools based on patterns. Finally, we see this paper as a major research direction that addresses challenges in design of educational technologies through solutions in software engineering.

This paper is an attempt to integrate research on patterns and patterns-based approaches in software engineering and instructional design to facilitate design of educational technologies that have a pedagogical basis and support systematic reuse. There many ways of improving our work. Firstly, there is no way to claim that the patterns we presented are the only possible patterns in instructional design nor they are comprehensive. In each category of patterns such as goals, process and so on, there is possibility of a variety of patterns in multiple contexts. In addition, there are still many manual steps in the pattern life cycle. While using patterns facilitates reuse, we see the following future research directions:

- Discovering new patterns is a critical future direction to support the breadth of instructional designs, so are the mechanisms to support evolution of existing patterns.
- Modeling notations for representing patterns and supporting techniques and tools for handling patterns throughout life cycle.
- Formal approaches for composing patterns either within the domain or software or between domain and software, validating assembly of patterns and their integration.
- Applying our approach to beyond adult literacy for schooling and other forms of education.

## Acknowledgements

We would like to thank TCS for providing us with initial inputs for this work, NLM for taking our work forward to create national impact, Government of Telangana for being one of the first adoptors of our technologies and all funding agencies for supporting several international research travels.

## References

- Adams Becker, S., Cummins, M., Davis, A., Freeman, A., Hall Giesinger, C., & Ananthanarayanan, V. (2017). NMC horizon report: 2017 higher education edition. *Austin, Texas: The New Media Consortium*.
- Alario-Hoyos, C., Bote-Lorenzo, M. L., Gómez-Sánchez, E., Asensio-Pérez, J. I., Vega-Gorgojo, G., & Ruiz-Calleja, A. (2013). GLUE!: An architecture for the integration of external tools in Virtual Learning Environments. *Computers & Education*, 60(1), 122–137.



- Alexander, C., Ishikawa, S., & Silverstein, M. (1977). A Pattern Language: Towns, Buildings, Construction (Center for Environmental Structure Series).
- Alper, A., & Gülbahar, Y. (2009). Trends and issues in educational technologies: A review of recent research in TOJET. *TOJET: The Turkish Online Journal of Educational Technology*, 8(2).
- Anderson, L. W., Krathwohl, D. R., Airasian, P. W., Cruikshank, K. A., Mayer, R. E., Pintrich, P. R., ... Wittrock, M. C. (2001). A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives, abridged edition. *White Plains, NY: Longman*.
- Apel, S., Batory, D., Kästner, C., & Saake, G. (2013). *Feature-oriented software product lines: concepts and implementation*. Springer Science & Business Media.
- Association, C. R., et al. (2003). Grand research challenges in information systems. In *A Conference Series on Grand Research Challenges in Computer Science and Engineering*.
- Avgeriou, P., Papasalouros, A., Retalis, S., & Skordalakis, M. (2003). Towards a pattern language for learning management systems. *Educational Technology & Society*, 6(2), 11–24.
- Bayne, S. (2015). What's the matter with technology-enhanced learning? *Learning, Media and Technology*, 40(1), 5–20.
- Bednar, A. K., Cunningham, D., Duffy, T. M., & Perry, J. D. (1992). Theory into practice: How do we link. *Constructivism and the technology of instruction: A conversation*, 17–34.
- Berczuk, S. (1994). Finding solutions through pattern languages. *Computer*, 27(12), 75–76.
- Berger, C., & Kam, R. (1996). Definitions of instructional design. Retrieved January, 30, 2006.
- Bergin, J., Eckstein, J., Volter, M., Sipos, M., Wallingford, E., Marquardt, K., ... Manns, M. L. (2012). *Pedagogical patterns: advice for educators*. Joseph Bergin Software Tools.
- Bingimlas, K. A. (2009). Barriers to the successful integration of ICT in teaching and learning environments: A review of the literature. *Eurasia Journal of Mathematics, Science & Technology Education*, 5(3).
- Borchers, J. O. (2001). A pattern approach to interaction design. *Ai & Society*, 15(4), 359–376.
- Botturi, L., Derntl, M., Boot, E., & Figl, K. (2006). A classification framework for educational modeling languages in instructional design. In *6th IEEE International Conference on Advanced Learning Technologies (ICALT 2006)*, Kerkrade (The Netherlands) (pp. 1216–1220).
- Botturi, L., Stubbs, S. T., & Global, I. (2008). *Handbook of visual languages for instructional design: Theories and practices*. Information Science Reference Hershey.
- Bower, M., Howe, C., McCredie, N., Robinson, A., & Grover, D. (2014). Augmented Reality in education—cases, places and potentials. *Educational Media International*, 51(1), 1–15.

- Boyle, T., & Cook, J. (2001). Towards a pedagogically sound basis for learning object portability and re-use. In *Meeting at the Crossroads. Proceedings of the 18<sup>th</sup> Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education. The University of Melbourne* (Vol. 101, p. 109).
- Brown, A. H., & Green, T. D. (2015). *The essentials of instructional design: Connecting fundamental principles with process and practice*. Routledge.
- Burgos, D. (2015). A Critical Review of Ims Learning Design. In *The Art & Science of Learning Design* (pp. 137–153). Springer.
- Buschmann, F., Henney, K., & Schimdt, D. (2007). *Pattern-oriented Software Architecture: On Patterns and Pattern Language* (Vol. 5). John Wiley & Sons.
- Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). {Pattern-oriented Software Architecture Volume 1}.
- Caeiro, M., Llamas, M., & Anido, L. (2014). PoEML: Modeling learning units through perspectives. *Computer Standards & Interfaces*, 36(2), 380–396.
- Caeiro-Rodríguez, M., Llamas-Nistal, M., & Anido-Rifón, L. (2006). A separation of concerns approach to educational modeling languages. In *Frontiers in Education Conference, 36<sup>th</sup> Annual* (pp. 9–14).
- Carroll, J. B. (1963). A model of school learning. *Teachers college record*.
- Chimalakonda, S. (2017). *A Software Engineering Approach for Design of Educational Technologies* (PhD thesis). International Institute of Information Technology-Hyderabad.
- Chimalakonda, S., & Nori, K. V. (2012). Towards a Model Driven eLearning Framework to Improve Quality of Teaching. In *Technology for Education (T4E), 2012 IEEE Fourth International Conference on* (pp. 138–143).
- Chimalakonda, S., & Nori, K. V. (2013a). EasyAuthor: supporting low computer proficiency teachers in the design of educational content for adult illiterates. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems* (pp. 649–654).
- Chimalakonda, S., & Nori, K. V. (2013b). IDont: An Ontology Based Educational Modeling Framework for Instructional Design. In *Advanced Learning Technologies (ICALT), 2013 IEEE 13<sup>th</sup> International Conference on* (pp. 253–255).
- Consortium, I. G. L., et al. (2003). *IMS learning design specification*.
- Cristea, A., & Garzotto, F. (2004). Designing patterns for adaptive or adaptable educational hypermedia: a taxonomy. In *World Conference on Educational Multimedia, Hypermedia and Telecommunications* (Vol. 2004, pp. 808–813).
- Cuban, L., & Jandrić, P. (2015). The dubious promise of educational technologies: Historical patterns and future challenges. *E-Learning and Digital Media*, 12(3-4), 425–439.
- Czarnecki, K., & Eisenecker, U. W. (2000). Generative programming. *Edited by G. Goos, J. Hartmanis, and J. van Leeuwen*, 15.
- Dalziel, J. (2003). *Implementing learning design: The learning activity man-*

- agement system (LAMS). December.
- Derntl, M. (2005). *Patterns for person centered e-learning*. Citeseer.
- Derntl, M., & Motschnig-Pitrik, R. (2004). Patterns for blended, person-centered learning: Strategy, concepts, experiences, and evaluation. In *Proceedings of the 2004 ACM symposium on Applied computing* (pp. 916–923).
- Dick, W., Carey, L., Carey, J. O., et al. (2001). *The systematic design of instruction* (Vol. 5). Longman New York.
- Dijkstra, E. W. (1982). On the role of scientific thought. In *Selected writings on computing: a personal perspective* (pp. 60–66). Springer.
- Duffy, T. M., & Jonassen, D. H. (2013). *Constructivism and the technology of instruction: A conversation*. Routledge.
- Evans, E. (2004). *Domain-driven design: tackling complexity in the heart of software*. Addison-Wesley Professional.
- Fischer, F., Wild, F., Sutherland, R., & Zirn, L. (2014). *Grand Challenges in Technology Enhanced Learning: Outcomes of the 3rd Alpine Rendez-Vous*. Springer.
- Gagne, R. M., & Briggs, L. J. (1974). *Principles of instructional design*. Holt, Rinehart & Winston.
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design patterns: elements of reusable object-oriented software*. Pearson Education.
- Garzotto, F., Retalis, S., & Chapter, V. (2009). A critical perspective on design patterns for e-learning. *Learning Objects: Is-sues, Applications and Technologies*, 346–372.
- Ghezzi, C., Jazayeri, M., & Mandrioli, D. (2002). *Fundamentals of software engineering*. Prentice Hall PTR.
- Gibbons, A. S. (2013). *An architectural approach to instructional design*. Routledge.
- Goddard, T., Griffiths, D., & Mi, W. (2015). Why has Ims Learning Design not Led to the Advances which were Hoped for? In *The Art & Science of Learning Design* (pp. 121–136). Springer.
- Goodyear, P., Avgeriou, P., Baggetun, R., Bartoluzzi, S., Retalis, S., Ronteltap, F., & Rusman, E. (2004). Towards a pattern language for networked learning. In *proceedings of networked learning* (pp. 449–455).
- Govindasamy, T. (2001). Successful implementation of e-learning: Pedagogical considerations. *The Internet and Higher Education*, 4(3), 287–299.
- Greenfield, J., Short, K., Cook, S., Kent, S., & Crupi, J. (2004). *Software factories: assembling applications with patterns, models, frameworks, and tools*. Wiley Pub.
- Gustafson, K. L., & Branch, R. M. (1997). *Survey of instructional development models*. ERIC.
- Handbook for Developing IPCL Material*. (2003). Directorate of Adult Education, India.
- Hernández-Leo, D., Chacón, J., Prieto, L. P., Asensio-Pérez, J. I., & Derntl, M. (2013). Towards an integrated learning design environment. In *Scaling up Learning for Sustained Impact* (pp. 448–453). Springer.

- Hernández-Leo, D., Moreno, P., Chacón, J., & Blat, J. (2014). LdShake support for team-based learning design. *Computers in Human Behavior*, 37, 402–412.
- Hwang, G.-J., & Tsai, C.-C. (2011). Research trends in mobile and ubiquitous learning: A review of publications in selected journals from 2001 to 2010. *British Journal of Educational Technology*, 42(4).
- Iba, T., & Miyake, T. (2010). Learning patterns: A pattern language for creative learning ii. In *Proceedings of the 1<sup>st</sup> Asian Conference on Pattern Languages of Programs* (p. 4).
- Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., & Peterson, A. S. (1990). *Feature-oriented domain analysis (FODA) feasibility study* (Tech. Rep.). DTIC Document.
- Karagkasidis, A. (2008). Developing GUI Applications: Architectural Patterns Revisited. In *EuroPLOP*.
- Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.-M., & Irwin, J. (1997). Aspect-oriented programming. In *European conference on object-oriented programming* (pp. 220–242).
- Koper, R. (2005). *An introduction to learning design*. Springer.
- Krasner, G. E., Pope, S. T., et al. (1988). A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming*, 1(3), 26–49.
- Kruchten, P. B. (1995). The 4+ 1 view model of architecture. *Software, IEEE*, 12(6), 42–50.
- Kruse, K. (2002). Introduction to instructional design and the ADDIE model. Retrieved January, 26, 2005.
- Laurillard, D. (2012). *Teaching as a design science: Building pedagogical patterns for learning and technology*. Routledge.
- Laurillard, D. (2013a). *Rethinking university teaching: A conversational framework for the effective use of learning technologies*. Routledge.
- Laurillard, D. (2013b). *Teaching as a design science: Building pedagogical patterns for learning and technology*. Routledge.
- Laurillard, D., Charlton, P., Craft, B., Dimakopoulos, D., Ljubojevic, D., Magoulas, G., ... Whittlestone, K. (2013a). A constructionist learning environment for teachers to model learning designs. *Journal of Computer Assisted Learning*, 29(1), 15–30.
- Laurillard, D., Charlton, P., Craft, B., Dimakopoulos, D., Ljubojevic, D., Magoulas, G., ... Whittlestone, K. (2013b). A constructionist learning environment for teachers to model learning designs. *Journal of Computer Assisted Learning*, 29(1), 15–30.
- Lowyck, J. (2014). Bridging learning theories and technology-enhanced environments: A critical appraisal of its history. In *Handbook of research on educational communications and technology* (pp. 3–20). Springer.
- Mager, R. F. (1962). Preparing instructional objectives.
- Martínez-Ortiz, I., Moreno-Ger, P., Sierra, J. L., & Fernandez-Manjon, B. (2007). Educational modeling languages. In *Computers and Education* (pp. 27–40). Springer.

- Merrill, M. D. (2012). *First principles of instruction*. John Wiley & Sons.
- Meszaros, G., & Doble, J. (1998). A pattern language for pattern writing. *Pattern languages of program design*, 3, 529–574.
- Midgley, C. (2014). *Goals, goal structures, and patterns of adaptive learning*. Routledge.
- Millwood, R. (2014). *The Design of Learner-Centred, Technology-Enhanced Education* (Unpublished doctoral dissertation). University of Bolton.
- Mizoguchi, R., Hayashi, Y., & Bourdeau, J. (2007). Inside theory-aware and standards-compliant authoring system. In *SW-EL'07* (pp. 18–pages).
- Mor, Y. (2014). *Practical Design Patterns for Teaching and Learning with Technology*. Springer.
- Mor, Y., Warburton, S., & Winters, N. (2012). Participatory pattern workshops: a methodology for open learning design inquiry. *Research in Learning Technology*, 20.
- Mørch, A. (1997). Three levels of end-user tailoring: Customization, integration, and extension. *Computers and design in context*, 51–76.
- Neumann, S., Klebl, M., Griffiths, D., Hernández-Leo, D., De la Fuente-Valentin, L., Hummel, H., ... others (2009). Report of the results of an IMS learning design expert workshop.
- Paquette, G. (2014). Technology-Based Instructional Design: Evolution and Major Trends. In *Handbook of Research on Educational Communications and Technology* (pp. 661–671). Springer.
- Parnas, D. L. (1972). On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12), 1053–1058.
- The Pedagogical Patterns Project*. (2014, May). <http://www.pedagogicalpatterns.org/>.
- Reich, J. (2015). Rebooting MOOC research. *Science*, 347(6217), 34–35.
- Reid, P. (2014). Categories for barriers to adoption of instructional technologies. *Education and Information Technologies*, 19(2), 383–407.
- Reigeluth, C. M. (2013a). *Instructional-design theories and models: A new paradigm of instructional theory* (Vol. 2). Routledge.
- Reigeluth, C. M. (2013b). *Instructional design theories and models: An overview of their current status*. Routledge.
- Reigeluth, C. M., & Carr-Chelman, A. A. (2009). Instructional-Design Theories and Models: Building a Common Knowledge Base. Volume III. *Education Review//Reseñas Educativas*.
- Rodríguez-Artacho, M., & Maillo, M. F. V. (2004). Modeling educational content: the cognitive approach of the PALO language. *Educational Technology & Society*, 7(3), 124–137.
- Sampson, D. G., & Zervas, P. (2014). A hierarchical framework for open access to education and learning. *International Journal of Web Based Communities*, 10(1), 25–51.
- Sharp, H., Manns, M. L., & Eckstein, J. (2000). The pedagogical patterns project (poster session). In *Addendum to the 2000 proceedings of the conference on Object-oriented programming, systems, languages, and applications (Addendum)* (pp. 139–140).

- Sheu, F.-R., & Chen, N.-S. (2014). Taking a signal: A review of gesture-based computing research in education. *Computers & Education*, 78, 268–277.
- Spector, J. M. (2013). Emerging educational technologies and research directions. *Journal of Educational Technology & Society*, 16(2).
- Spector, J. M., Merrill, M. D., Elen, J., & Bishop, M. (2014). *Handbook of research on educational communications and technology*. Springer.
- Taylor, R. N., Medvidovic, N., & Dashofy, E. M. (2009). *Software architecture: foundations, theory, and practice*. Wiley Publishing.
- Tobias, S., Fletcher, J. D., & Wind, A. P. (2014). Game-based learning. In *Handbook of Research on Educational Communications and Technology* (pp. 485–503). Springer.
- Toyama, K. (2011). There are no technology shortcuts to good education. *Educational Technology Debate*, 8.
- Venkatesh, V., Croteau, A.-M., & Rabah, J. (2014). Perceptions of effectiveness of instructional uses of technology in higher education in an era of Web 2.0. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on* (pp. 110–119).
- Villasclaras-Fernández, E., Hernández-Leo, D., Asensio-Pérez, J. I., & Dimitriadis, Y. (2013). Web Collage: An implementation of support for assessment design in CSCL macro-scripts. *Computers & Education*, 67, 79–97.
- Woolf, B. P., Lane, H. C., Chaudhri, V. K., & Kolodner, J. L. (2013). AI Grand Challenges for Education. *AI Magazine*, 34(4), 9.
- Wu, W.-H., Wu, Y.-C. J., Chen, C.-Y., Kao, H.-Y., Lin, C.-H., & Huang, S.-H. (2012). Review of trends from mobile learning studies: A meta-analysis. *Computers & Education*, 59(2), 817–827.
- Zimmer, W., et al. (1995). Relationships between design patterns. *Pattern languages of program design*, 57.