

# An Effective Learning Environment for Teaching Problem Solving in Software Architecture

Kirti Garg

International Institute of Information Technology,  
Hyderabad, India  
+91-4023001967 Ext 301  
kirti@research.iiit.ac.in

Vasudeva Varma

International Institute of Information Technology,  
Hyderabad, India  
+91-4023001967 Ext 179  
vv@iiit.ac.in

## ABSTRACT

A software architect engages in solving Software Engineering (SE) problems throughout his career. Thus inculcating problem solving skills should be one of the learning objectives of SE academic and training programs. But structured problem solving is usually latent or missing in most of the current curriculum. In this paper, we describe an effective learning environment for SE education and training with problem solving as an integral part.

The learning environment is in accordance with Learning Sciences theory and practices. Our study strengthens our belief that such a problem based environment will help to create professionals well versed with theory and practice of software architecture and problem solving, and thus more productive and useful for the industry.

## Categories and Subject Descriptors

K.3.2 [Computer and Information Science Education]

**General Terms:** Design

## Keywords

Software architecture, Software Engineering education, learning environment, problem solving, systems thinking

## 1. INTRODUCTION

Software architects heavily use problem solving (PS) skills as problem solving is what they do most of the time. They face decision dilemmas, thus apply their knowledge and skills (Computer Science, SE and domain related) to analyze the given situation, evaluate various alternatives and develop a solution that satisfies various constraints (hence effective). Thus problem solving skills such as forming hypothesis, analysis, critical thinking and evaluation should be an integral part of the SE curricula along with other contents. These skills belong to the enduring engineering principles in any engineering course. Most current SE courses rarely teach (PS) skills, possibly because they are considered difficult to teach and assess. We studied desired properties of effective learning environments [1] and designed a learning environment that inculcates and nurtures SE as well as structured problem solving in software architecture

students. Students use systems thinking and engineering [2], and Polya's problem solving techniques [4] to work on software architecture problems given in form of case studies (contextualized problems). Learning happens in a cognitive apprenticeship [3] framework that uses the case studies for teaching and learning.

## 2. PROPOSED ENVIRONMENT

We designed a software architecture course that implements an effective learning environment and in accordance with Learning Science principles. We present main constituents of the learning environment and gluing of these components using the case study framework in a software architecture course. Various constituents of the proposed learning environment are:

- **Software Architecture as Context:** As per effective LS principle, learning is effective when it is situated i.e. is in a context [1]. Presence of a context provides inherent support for problem solving, such that the process of problem solving is clearly exhibited, practiced and its advantages are evident and comparable. Software architecture provides an excellent context for such a desired learning environment.
- **Problem Solving Model (Polya's):** Next required is appropriate problem solving knowledge, skills and dispositions. We find that G. Polya's famous work for mathematics "How to solve it?" [4] provides good theory and practical knowledge for structured problem solving. Polya also provides a dictionary (a catalogue) of problem solving heuristics. He suggests a four phase technique to solve any problem: 1) Understanding the problem (What is given, what is the unknown, what are the data, what are the conditions? 2) Devising a plan/solution (is there a similar problem? What steps will give a solution?) 3) Carrying out the plan (Execute the steps and check if each step is correct) 4) Examine the solution obtained (Looking back, what is the learning?).
- **Systems Thinking:** Another set of PS guidelines come from Systems thinking [2]. It is a proactive hybrid method that uses both analysis and synthesis. Systems' thinking requires iterative inquiry of inter-dependent variables to examine their inter-dependency. A holistic view of the system is required for its correct and complete comprehension. Iterations would establish validity of the assumptions and successively produce an integrated understanding [2].
- **Cognitive Apprenticeship (case study):** Use of case studies is a prominent method belonging to cognitive apprenticeship [3] approach. Here, students work on a

hypothetical or real problem or a challenge that is situated in a context. Solving this problem or 'case' involves analysis of problems, and devising a solution by application of concepts, tools, techniques and skills, evaluation of alternatives and decision making. Understanding the problem means understanding the associated people, organizations, past events and decisions. As a side-effect of solving a case, the domain and discipline knowledge (both conceptual and procedural) is reinforced and problem solving skills get practiced.

### 2.1 Putting it All Together

We now put these various components together to create a highly effective problem solving environment for teaching software architecture. The learning environment is being described through the software architecture course. The course consists of various learning activities, sub-activities, expected learning outcomes and assessment.

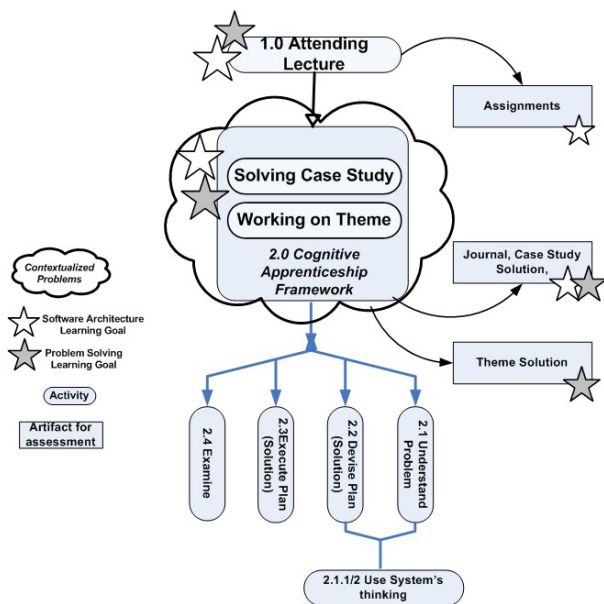


Figure1. Learning Environment for Software Architecture

Software architecture provides the context of learning. Software architecture knowledge (conceptual & procedural) and problem solving theory is imparted to students through lectures. Students then move to various activities in the cognitive apprenticeship model, namely solving case studies and working on theme. Challenges from a case are the problems that need to be solved. The case provides the context (organizational, project specific and domain). Solution to the challenges requires thorough understanding and application of software architecture knowledge and skills, thus re-enforcing the learning from lectures i.e. conceptual & procedural knowledge and skills.

We use generalized problem solving principles and heuristics given by Polya to solve the problems in a structured manner. The 4 step cycle (described earlier) forms our general process of solving the problems. Systems' thinking is used to illuminate the problem and the solution space. As software architecture also deals with systems that consist of inter-connected and inter-dependent components, systems thinking can provide problem solving guidelines for systematic and holistic understanding and

design of software systems. A correct and complete comprehension of the problem space by understanding the existing conditions/factors, and their relations helps to design an optimal solution. Further this understanding of the complexity and essential nature of the system forms the basis of evaluation. Once the problem is understood well, further steps of Polya's cycle can be applied, i.e. execution and evaluation. These steps are also related to the requisite meta-cognitive activities and hence are important.

Product (case study solution) and a student's understanding and application of the PS process are used for evaluation, thus making the assessment aligned with the learning outcomes and instruction, leading towards an effective learning environment. Solutions are discussed with peers. This and the first hand experience of applying the learnings to real problems makes the instruction learner centric and knowledge centric.

Use of this systematic approach to solve complex, abstract and large problems of the real world definitely increases the understanding of domain, of types of problems and problem solving heuristics and process in general, thus promoting deeper learning. Thus an excellent problem solving environment (Figure 1) can be created for Software architecture education. Table 2 gives a mapping of Learning Science principles and guidelines [1] and our proposed learning environment.

Table 2: Summary of the Proposed Learning Environment

Feature of Course	Mapping to Guidelines for Effective Learning Environment [1]
Use of Case Studies	Contextualization, promote deep approaches for learning and meta-cognitive learning, active learning, alignment of learning goals, instruction & assessment
Use of Polya's approach,	Provide guidelines for systematic problem solving, promote meta-cognitive learning
Use of systems thinking	Provide guidelines for systematic problem solving, promote meta-cognitive learning
Use of Theme	Promoting learner and knowledge centered learning, opportunity to practice problem solving

### 3. REFERENCES

- [1] Bransford, J. D., et al(Eds.), How People Learn: Mind, Brain, Experience and School, Expanded Edition. Washington, DC: National Academy Press, 2000
- [2] Checkland P., System Thinking, Systems Practice, John Wiley and Sons, 2000
- [3] Collins A. et al, Cognitive Apprenticeship: Teaching the Craft of Reading, Writing, and Mathematics, in Cognition and Instruction: Issues and Agendas. (L.B. Resnick, editor), Lawrence Erlbaum, 1989.
- [4] Polya, G, How To Solve It. Princeton, NJ: Princeton UP, 1973.