An Ontological Teaching Of Software Engineering; Separation of Concerns  
Izzat Alsmadi  
Yarmouk University  

Keywords  
Software engineering, ontology, separation of concerns, project, product, process and people, CASE tools, software development methodology.  

Abstract  
As new field of study, software engineering teaching and subjects vary from one textbook to another. Despite the fact that most of the books cover similar subjects, however, students view of the subject is mixed. Some students have problems understanding the entire picture. Other students have problems connecting concepts with each other. In this research, an overall view of software engineering knowledge is presented. The knowledge is presented from four perspectives: Process, Project, People and Product. Those four are usually referred to as the 4Ps in literature. The goal is to make a distinction between the progresses in each area and explore the opportunities in finding windows for more research in any of those four views. In software project managements, managers need to separate their planning and evaluation among those four perspectives. Students should also differentiate between tools, concepts and standards used for each one of those views.  

1. Introduction  
In teaching software engineering courses to students, it is noticed that some students complain from the lots of models that they need to know without having to know the overall picture first. The same problem existed in another related field; formal methods. There are several formal method tools and sectors to learn without having an overall ontology that illustrates the connections between those tools or methods.  

An ontological study simplifies the structure of understanding domain knowledge. It includes abstraction, representations, and assumptions. An ontology or a conceptual model facilitates communication or knowledge sharing on common grounds. In any ontology, abstractions are selected to focus on necessary components and eliminate irrelevant details. By large, software engineering field of study is lacking a comprehensive ontology that covers the overall knowledge body.  

There are some other alternative views for the subject: software engineering development methodologies that usually get the main focus and concern. Other concerns can be divided into the dimensions; data; that focuses on entities, functional; which is concerned with functions or services, user and environmental views. This separation of views is important for many areas. For example, in software project planning and evaluation, managers need to plan and evaluate for each one of those dimensions separately. Similarly, in software design, designers draw different diagrams for the different views: class, activity, use case and sequence diagrams.  

Out of the four views listed above, the software process and project views are the two that have the major focus, documentation and models in literature. Nevertheless, they are tightly coupled that makes it hard to distinguish whether this is a process or project attribute. Part of this confusion is understood since the software project management can be seen as a software process or activity. However, we should differentiate between software processes that are product oriented such as requirements, design and coding or development, and the
processes that are people oriented such as the project management, personnel selections, and tasks distribution.

There are numerous software process models proposed to adopt while building a software. Those models can be classified according to the aspects and properties that distinguish each view. The software process is the set of activities and methods employed in the production of a software. The software project is how we manage and organize the process, the people and the product. This view makes the software project the main view that includes the three other views. The software product is the ultimate goal for the project, process and product. Software project success is measured through the product success. A successful project, people and process should result in a successful product. The people are the real and main resources in software projects. Without talented and dedicated people, we can’t have a successful project and product. This shows that the four views are highly coupled and depending on each other and that we have to guarantee each ones success to guarantee the overall success. Figure 1 shows an overall ontology for software engineering concerns.

In the following sections, the four dimensions are considered separately.

2. Literature Review

There are currently several known books that are usually used as textbooks in teaching software engineering [1,2, 4,5,6,7,8,9,10,11,12]. Those books vary in their presentation of the subject. There are some subjects such as formal methods or agile developments that get more focus on some books relative to others. Those two subjects usually divide teaching software engineering into two schools. A school that focuses more on traditional methods of teaching with presenting techniques that improve the correctness of the requirements and design in early stages of development. The other school following agile methodologies and their focus on the time and the flexibility factors relative to other factors such as correctness and quality.

The other part of the literature review are papers who discussed the issue of teaching software engineering and its different views [3,13,14,15,16,17,18]. Habra proposed two software engineering modules for an undergraduate education with focus on separation of concerns [3]. He defined the following software project dimensions to be considered by students: data, functional, user, reusing, and distribution. Hawker presented a model that combines: product, process and people elements [13]. He drew several UML diagrams presenting the different elements in the model along with their interaction with each other.

3. Software Processes; Activities and Models

As mentioned earlier, out of the 4P dimensions, the process is the one that has most of the existed literature or documentation. Software processes are the activities involved in producing and evolving the software. Examples of some of those major activities include;

![Figure 1. An ontology for software project concerns](image-url)
requirements gathering and specifications, software architectural and design, software implementation, testing and maintenance or evolution. We have to differentiate between software processes and software process models. Software process models are abstract representations for the models and their interaction. Similar to abstractions, models involve focus on particular concerns or perspectives. As a result, each model has certain scenarios that can be best used in. For example, the waterfall process model is used when we have fixed and stable requirements. On the contrary, agile methods are better when we have uncertainties in the project.

The difference between the different models is largely depending on how the processes interact with each other (that’s why they are process models). There are two possibilities on how processes can interact with each other.

A. **Straight forward processes.** In those models, each major software activity is completed first before moving to the next process. Once a process is completed, we can’t go back and modify it. The largely known and used model in this type is the Waterfall model. Waterfall is used when we have stable and fixed requirements as it can hardly deal with or accommodate changes.

B. **Iterative or evolutionary processes.** In those models, major activities are completed partially in cycles and evolve to reach the final product. The goal is to deal with the instability of requirements and the need to accept and accommodate changes. With the exception of the waterfall model, all other software process models, such as the incremental model, spiral model, prototyping, and agile models, are examples of the Iterative models. Some models iterate through all process activities, others gather all requirements, then iterate through the rest of the activities. Spiral models make explicit risk assessments in every cycle, agile models combine iterations with project and people techniques to include abilities to deal with evolution and accept changes.

There are some software engineering books who consider some other software process models such as formal specification or Commercial Off-The shelf Software (COTS) as process models. The usage of formal specification or off-shelf software can be in any of the previously mentioned models and need not to be a separate model. For example, formal methods are used to verify requirements formally before starting the construction process. This is a specific extra activity that is added to the requirement stage.

### 4. People, the software project resources.

A software process model is an explicit description of the process through which software artifacts (or products) are implemented. Most of those process models focus on the products, time or goals as the factors that control the tasks for each stage of the model. In real environment the employees are the main resource and in each stage each employee, should have a specific task. Many of those current software process models do not consider some scenarios where a company may start a new project and each team or team member, whether from the business analysis’s team, from the development, testers or from the document writers, is expected to do some work at any development stage.

Humans play an important role in the success of software projects. Communication is very important to ensure all team collaborations and contributions. Some methods, such as agile development models are customer oriented where the customer suggestions and concerns are always considered.

In most of the traditional software development models, the focus is in achieving the goals or the requirements that the application is expected to fulfill. In agile methodologies, the time is more sensitive giving the fact that requirements and/or many other factors may change with a relatively short time.

Although those models target most of business scenarios in developing projects, in some cases we may have all company employees required to work simultaneously. Following
for example a traditional approach in that case, requires developers to wait or take a vacation if they don’t have other projects to work on till business analysis’s team finish collecting the requirements. Testers are also expected to wait longer time waiting for developers to design or implement the product or part of it. In a small business environment, this is expected to happen specially when there are no earlier versions of the application that those testers or document writers can work on through early stages.

Software project managers define tasks by roles, or individuals. It is better, however, to define project tasks by roles only. If the individual got busy in any other task, any other individual in the same class or role can be assigned the task. New software process methodologies such as Scrum, tries to mix software engineering roles to make all individuals capable of working in the different roles in different times. In reality, few people can perform all types of tasks professionally.

5. Tools, the people helpers

Tools play a major role in software engineering processes and development. In several cost estimation models such as COCOMO, the amount of assistance tools gave to developers is an important information needed to estimate the development time. They can noticeably improve the overall productivity of the team members.

Tools (also called Computer Aided Software Engineering, CASE) can be classified in several ways. They can be classified according to the software process stage they are working on (e.g. requirement tools, design tools, coding tools, etc). They can also be classified according to the number of stages they are working into tools, workbenches and environments that start from an individual stage to tools that support all software engineering activities.

6. The software product, the goal of the software project.

The software product is the ultimate deliverable or output that the team will produce. Any project’s ultimate goal is to provide a product with the right functionalities and qualities. However, some projects may not have a deliverable product; instead, they will have objectives to fulfill. No matter how much successful the team was, or what tools, or techniques they used, if the product fails, all will be considered so. This means that logically successful project, process and people should produce a successful product. However, this is not always the case.

In the product teaching section, COTS can be introduced as a subject, quality assurance and software metrics are two other main subjects to be covered. This section will also focus on classifying products according to their business domain. Products that share same domain are expected to have several common characteristics.

7. The software project; the umbrella that covers all views

The software project is the umbrella that holds all the earlier concerns together. Project management takes care not only of the people, or resources of a project, but it also deals with the process and the product management. Project management major activities include: preparing the feasibility study, tasks’ planning, allocating and scheduling, cost estimation, evaluation and measurements, risk assessment, and management, etc.

8. Software Engineering Education

One of the early papers that discussed software engineering education is that of Mills in the early 80s [20]. He discussed some of those early challenges and requirements that faced education in the new field. Later on, in a paper he published in 1988 [21] he acknowledged
that the software engineering education may vary depending on the degree and university requirements, industry needs and expectations, and many other factors that may eventually route the software engineering education to its ultimate destiny. He indicated that (even in the late 80s), the problem is not with the lack of technologies, but with the difficulties or problems of education management.

Software Engineering Body Of Knowledge (SWEBOK) is a general manual or guide from IEEE Software Engineering experts started in the late 90s on the general knowledge and information about this field [22]. In 2004, SWEBOK define the following knowledge areas in the software engineering field: Software requirements, Software design, Software construction, Software testing, Software maintenance, Software configuration management, Software engineering management, Software engineering process, Software engineering tools and methods, and Software quality. This work was an important document that contributed to the knowledge and education in this field, however, known researchers such as Cem Kaner and Grady Booch believed that such document needs a thorough reevaluation process.

The compromise between the theoretical and the practical teaching or education was always an issue of difference between curriculum or course contents. Another related issue is the ability to bridge the gap between the academia and the industry. Many progresses in this field are out of synch if we compare it between the academia and the industry. Universities are then in a challenge whether to produce good work force members or good researchers.

9. Conclusion and future work

The goal of this separation of concerns is to organize the software engineering project into smaller manageable parts that can be easy to understand. It should reduce complexity and improve clarity. This concept is at the core of software engineering. The 4Ps concerns have some overlapping and distinct features. Concepts such as; ontology, abstraction, modeling and views or separation of concerns always include some sort of abstraction or focus. The goal is to draw a better image or understanding for the problem. The goal of the separation of the concerns in software engineering projects is to improve the understandability and consider only relevant properties for each perspective.

In another goal, we hope that the separation of concerns will help software engineering students better understand the large number of modeling and terminology concepts that may overlap and hence seem ambiguous.

This paper suggests a methodology to teach software engineering on the basis of the different perspectives or view. Such views are expected to develop and overall conceptual understanding that seems to be missing for many students who learn introductory software engineering courses. We described those different views in brief to proof the concept. As this is a suggestion for a book or a course, it should include more details and elaborations. We will introduce all software engineering terms, and concepts in terms of this view.

9. References


Author:
Izzat Alsmadi
Department of Computer Science and Information Technology, Yarmouk University
ialsmadi@yu.edu.jo