**Critique\_1**

**By Anvesh Tummala (16173144)**

**Critique on “Essence and Accidents of Software Engineering”:**

About four essential difficulties

* Complexity - well explained by looking into complexity of communication, functions, structure and cause of management problems are well explained with examples.
* Conformity to complex external interfaces of a software system caused by not having standardizations in interfaces.
* Cause of changeability by pressure for need of extended functionality and to provide conformity to the new hardware changes is well justified.
* Communication effect due to invisibility of software is an interesting aspect.

About solutions for these difficulties

* High-level languages helps to solve the complexity of programming by abstract programming, where we only concentrates on operations, data-types and communications rather than detailed implementation.
* Time-Sharing can increase the productivity which intern helps in reaching the need for changeability of software. Even it helps in reduces the complexity of programming.
* Unified programming environments like UNIX, attacks these essential difficulties by providing integrated libraries, unified file formats, and piped and filters. This makes the conceptual structure more easily understandable. I.e. it helps in better visibility.
* Object oriented ideas like abstract data types and hierarchical type’s helps in removing higher order difficulties and allows designers more flexibility by not worrying on how to implement.

The attacks on conceptual essence by Buy versus build that reduces time of production, rapid prototyping helps in implementing main features of software first, thus reduces the time of production and “Incremental development by growing -not by building a software”, adding functions and provisions to an existing software to evolve it bigger seems more practical.

**Critique on “Foundations for the study of Software Architecture”:**

I like the way paper is organized. It starts by giving an idea of architecture from existing systems, introducing a simple and understandable model of elements, form, rationale, explaining that model with an example of three different overlapping views, benefits derived, and conclusion.

While giving intuition to software architecture from building architecture, I like the analogies of

* An architect who provides the floor plan for a builder, along with additional structural views to the software architect who needs a number of different views of software architecture for various uses and users. This simple analogy emphasizes on the need for different perspective of software architect for different users.
* Highlighting the need of design principles by stating “Different engineering principles are needed to move from the massiveness of the romanesque to lightness of the perpendicular.”
* “The materials have certain properties that are exploited in providing a particular style.” It makes us to be cautious on the selection of process or principles that might contradict with our design.

“Architecture is concerned with selection of elements, interactions and constraints to those elements whereas design is concerned with modularization and interfaces of design elements, their algorithms and procedures and data types.” gives a clear idea on the roles at architecture and design level.

The solution to architectural erosion and drift by the model of Elements, Forms, Rationale that has weighted properties (constraints on the elements) and relationships (constraints on the placement of elements) clearly defined in the Forms is a decent solution. As the process, data and connector view are interdependent, the architect should have all three views. In the example, selection of property-based scheme rather than type based scheme to capture better characteristics seems smart.

In justifying the statement “finding components for reuse is not the primary issue” the analogy with finding components in math library seems not well explained. It is not clear what author want to specify with this analogy, he might be trying to emphasize on cohesion inside a component. This is the best paper to get started for basic intuition on software architecture.