

# Document Driven Disciplined Development of Software

David Lorge Parnas, P.Eng, Ph.D, Dr.h.c., Dr.h.c., FRSC, FACM, FCAE  
*SFI Fellow, Professor of Software Engineering*  
*Director of the Software Quality Research Laboratory (SQRL)*  
*University of Limerick, Ireland*

## *Extended Abstract*

It is no accident that the branches of Engineering are called “disciplines”. Every properly educated Engineer has learned that the design of quality products requires discipline and a willingness to follow standard procedures. Engineers understand that they must produce a specified set of documents and perform a variety of analyses whose results must be included in the documents. Engineers who do these things are less likely to produce a defective product. In many jurisdictions, engineers who fail to follow the standard discipline may be considered to have been negligent. Software development should not be different but most developers have not been taught the appropriate discipline and neither their employers nor the customers know what to demand.

Something has gone wrong in the field that we call “Software Engineering”. Where a Mechanical Engineering journal might publish a paper on how to make sure that an impeller will not break if the fluid is viscous, or how to control a milling machine more precisely, Software Engineering journals publish papers on how to get people to cooperate better. In other words, we are focusing on aspects of project management and forgetting the problems of design and analysis that are at the heart of other areas of Engineering. Other papers describe theories that are so far from the issues that are important in practice that practitioners cannot see their relevance. In mathematics, the relationship between the “real world” and the axioms is not important; in Engineering it should be viewed as essential.

Nowhere is this clearer than in the area of requirements. An Engineer is responsible for making sure that the product is “fit for us”. When designing a control system or a circuit, an Engineer will analyze the environment together with the product, using physics and mathematics to see how the combination of environment and product will work together. In contrast, software engineers seem to be satisfied with going to potential users and asking, “What would you like?” Functional requirements for physical products can be checked for completeness and consistency; in software we accept the fact that our requirements will have neither property.

Where the “architecture” of a physical product would be documented by complete specifications for the main components, software engineers are satisfied with pretty graphs that show only a fraction of the interactions between components.

Where an Electrical Engineer would analyze a mathematical “design” for a circuit before producing physical components, most Software Engineers will have a vague design in their head as they write the code. The code is the first precise description of the component that can be reviewed by others.

At the heart of the problem is our failure to agree on a set of documents that contain analyzable (and sometimes simulatable) descriptions and specifications. At best we begin with highly simplified “models” that have a vague resemblance to what we have in mind. Most of these models have properties that could never be implemented and ignore critical facts that are essential for trustworthy products.

This talk will describe a set of documents that contain the information required for disciplined development. We define the documents by their content, not their format. A developer who completes these documents properly will have performed the analyses necessary to assure that the product will be of high quality. This talk does not describe a “process”. Discipline does not require that documents be completed in any particular order - only that they are all eventually completed properly. It is hoped that this talk will stimulate a discussion that will lead to an improved set of document standards.