ABSTRACT

Although there is a large body of research in dependability, architectural level reasoning about dependability is only just emerging as an important theme in software development. This is due to the fact that dependability concerns are often left until too late in the process of development. In addition, the complexity of emerging applications and the trend of building trustworthy systems from existing untrustworthy components are urging dependability concerns to be considered at the architectural level. This tutorial will present the current challenges and promising solutions for structuring dependable systems at the architectural level. In addition of providing basic concepts related to dependability and software architectures, the rest of the tutorial is presented in the context of the dependability technologies. Throughout the tutorial, case studies will be used to exemplify the key concepts.

Categories and Subject Descriptors
D.2.11 [Software Engineering]: Software Architectures;
D.4.5 [Operating Systems]: Reliability

General Terms
Design, reliability, verification

Keywords
Dependability, Software architecture, Fault-tolerance, Trustworthiness

1. INTRODUCTION

The aim of this tutorial is to provide an overview of the principles associated with both software architectures and dependability, and provide an insight on how the structuring of software systems at the architectural level is fundamental for the development of dependable systems. Taken as a basis the different dependability technologies, the intention is to show how dependability should be considered at the architectural level, and the impact this should have when developing dependable systems. Existing architectural approaches do not provide the necessary means for reasoning about dependability, hence the need to know what are the general principles associated with software architectures, what is being developed in terms of dependability technologies, and what are the challenges lying ahead. The main objectives of this tutorial are following:

- to establish the major principles associated with software architectures and dependability that are relevant when reasoning about faults at the architectural level;
- to introduce and discuss existing approaches for architecting dependable systems;
- to identify the main challenges that lie ahead when considering the structuring of dependable systems at the architectural level.

Reasoning about dependability at the architectural level has lately grown in importance because of the complexity of emerging applications, and the trend of building them through the integration of pre-existing software components. This component-based trend requires trustworthy systems to emerge from the integration of untrustworthy components, whose actual implementations may even not be known in advance. These new applications demand for dependability concerns to be considered at the architectural level, rather than late in the development process. From the perspective of software engineering, which strives to build software systems that are free of faults, the architectural consideration of dependability compels the acceptance of residual and unanticipated faults, rather than relying only in their avoidance. Thus the need for novel notations, methods and techniques that provides the necessary support for reasoning about faults at the architectural level [3].

2. ARCHITECTING FOR DEPENDABILITY

The structure of a system is what enables it to generate its intended behaviour, from the behaviour of its components. The architecture of a software system is an abstraction of the actual structure of that system. The identification of the system structure early during its development allows abstracting away from details of the system, thus assisting the understanding of broader system concerns [2]. One of the benefits of a well-structured system is to avoid overly complex relationships between its components, which in turn should lead to a more dependable system. Dependability can be defined as the ability of a system to deliver service that can justifiably be trusted [1].

System dependability is measured through its attributes, such as reliability, availability, confidentiality and integrity, and there are several technologies for attaining these attributes, which can be grouped into four major categories
3. TUTORIAL OUTLINE

The main topics of the tutorial are the following.

General Introduction The purpose of this section is to provide a context for the emerging field of architecting dependable systems.

Basic Concepts This is an introductory section to software architectures and dependability.

- **Software Architectures** - what are the key issues in software architectures that are relevant when structuring dependable systems.
- **Dependability** - what are the dependability principles and concepts.
- **Software Architectures in the Design of Dependable Systems** - what are the key properties of an architecture from the dependability perspective.

Architecting for Dependability The impact of reasoning about dependability at the architectural level will be analysed in the context of dependability technologies.

- **Rigorous Design** - presents how rigorous and formal notations, in addition to design principles can prevent the introduction of faults.
- **Verification and Validation** - presents how faults can be removed from an architectural representation.
- **Fault Tolerance** - presents how faults can be tolerated at the architectural level. This ability would allow a system to deliver its service despite the presence of faults.
- **System Evaluation** - presents how to measure the dependability attributes at the architectural level, considering that not all faults can be removed.

Case Studies The intent is to use two case studies throughout the tutorial to exemplify the different concepts.

Future Trends The key challenges in the area will be discussed. New approaches have to be devised for handling residual and unanticipated faults during run time, since not all faults can be removed during the development process.

4. ACKNOWLEDGMENTS

The author wishes to thank Paulo Asterio de C. Guerra for collaborating in the preparation of an early version of this tutorial, which was presented at LADC 2005.

5. REFERENCES

