Hugh hoang -1833106

There are multiple applications that comprise the specification of simple programs,data types and processes as well as of complex hardware and software systems. Formal and semi-formal approaches have their advantages and disadvantages: the informal diagrammatic methods are easier to understand and to apply but they can be ambiguous. Due to the different nature of the employed diagrams and descriptions it is often difficult to get a comprehensive view of all functional and dynamic properties. The formal approaches are more difficult to learn and require mathematical training. But they provide mathematical rigor for analysis and prototyping of designs. Verification is possible only with formal techniques.The basic idea of the approach presented here is to provide a common semantic domain, given by transformation systems as basic elements, that allows the interpretation of all the different languages. By referring to the same semantic element the correspondence of the syntactic elements can then be deduced.The goal is to establish a method for the systematic development of formal models for high assurance systems. The aim is to reduce errors in requirement definitions of software based safety critical systems to get safe specifications. This is achieved by combining formal models with safety analysis techniques from engineering. Formal specifications are used to describe the system model and to formulate and verify safety properties. Design errors and safety faws are detected by safety analysis techniques. The benefit of combining both techniques is three-fold.A formal semantics of fault tree analysis (FTA) was developed. A common feature of present-day algebraic specification languages is the provision of operations for building large specifications in a structured fashion from smaller and simpler ones.The main idea is that architectural specifications describe branching points in system development by indicating units (modules) to be independently developed and showing how these units, 4 once developed, are to be put together to produce the overall result. The GENGED approach developed at the Technical University of Berlin allows already the generic description of visual modeling languages based on formal graph transformation and graphical constraint solving techniques and tools.Plan is a language being developed at UPenn for programming active networks. Plan programs are sets of active packets that travel through a network, executing code on specified nodes. The active nature of Plan programs makes it important to have a formal specification of the semantics. The Object Constraint Language OCL offers a formal notation for constraining the modeling elements occurring in UML diagrams. The integration of several different modeling techniques into a single formal method has turned out to be advantageous in the formal design of software systems. An important issue in the area of software architecture is the specification of reconfiguration and mobility of systems. This talk presents an approach for the specification of software architecture styles using hyperedge replacement systems and for their dynamic reconfiguration using constraint solving. The semantic framework we have chosen for presenting our results is tile logic, which has the advantage of allowing a uniform treatment of goals and observations and of applying abstract categorical tools for proving the results. This approach is based on concepts of partial algebra. Partially ordered runs are constructed via process terms that use operators for sequential and for concurrent composition. Concurrent composition is defined only partially, depending on the respective occurrence rule. This work provides a unique overall view of information system 17 analysis and points out systematically the weakness of an MIS, and the relation to organization performance. Defning Visual Languages means dealing with two different aspects, namely the description of the logical stucture of the language and its graphical layout.UML (Unifed Modeling Language) [10, 11] is a de-facto standard visual modeling language used to specify, visualize, construct and document the structure and the behavior of software systems.