SOFTWARE INSPECTION

Practical Code Inspection Techniques for Object-Oriented Systems: An Experimental Comparison
by Alastair Dunsmore, Marc Roper, and Murray Wood, pp. 21–29. Although inspection is an effective mechanism for detecting defects in procedural systems, object-oriented systems have different structural and execution models. This article describes the development and empirical investigation of three techniques for reading object-oriented code during inspection.

Prioritized Use Cases as a Vehicle for Software Inspections
by Thomas Thelin, Per Runeson, and Claes Wohlin, pp. 30–33. The authors combine principles from software inspection, use cases, and operational profile testing to create the usage-based reading technique, in which the user reads a design document guided by prioritized use cases. UBR takes the user’s viewpoint on software and the faults it might contain. Evaluation showed that the method is more effective and efficient in finding faults that are critical to the user than are checklist-based methods.

Making Software Timing Properties Easier to Inspect and Verify
by Jia Xu, pp. 34–41. A general method or tool that can verify the timing properties of any arbitrary piece of software will probably never exist. So, software with hard timing requirements should be designed using a systematic approach to make its timing properties easier to inspect and verify. Pre-runtime scheduling provides such an approach by placing restrictions on software structures to reduce complexity.

Tool Support for Fine-Grained Software Inspection
by Paul Anderson, Thomas Reps, Tim Teitelbaum, and Mark Zarin, pp. 42–50. An important part of the software inspection process is a detailed inspection of the source code, usually using a checklist. While many tools are available to manage the inspection process, few tools support detailed code analysis. Recent advances in static program analysis can help advance this aspect of software inspection. The CodeSurfer tool can analyze an entire software program statically, produce a dependence-graph representation, and reveal fine-grained details of the program’s semantics. The authors use examples from NASA’s Formal Inspection Guidebook to illustrate the tool’s usefulness.

FEATURES

Architecting Systems with UML 2.0
by Morgan Bjorkander and Cris Kobryn, pp. 57–61. Despite UML’s successes since its standardization in 1997, development tools have been slow to realize its full potential. In addition, the software industry has evolved considerably during the last six years, and the first version of UML (UML 1.x) is now dated. The proposed major revision of UML, version 2.0, will make the language more current and improve its expressive power and precision. The finalization of the major revision is expected to be followed by updates to modeling tools that implement the powerful new UML 2.0 features.

User Interface Modeling in UMLi
by Paulo Pinheiro da Silva and Norman W. Paton, pp. 62–69. The user interface is an essential part of most software systems, and often represents a significant proportion of the code developed during implementation. However, there has been little specific attention given to user interface issues in many object-modeling languages, including UML. User interface models can represent different features of an interface. You can support such functionality using both standard UML and the Unified Modeling Language for Interactive Applications (UMLi), a conservative extension of UML that offers explicit support for interface modeling.

Entity-Life Modeling: Modeling a Thread Architecture on the Problem Environment
by Bo I. Sandén, pp. 70–78. Object-oriented development identifies classes and relationships in the problem domain and carries them seamlessly into design and implementation. Entity-life modeling extends this modeling idea into the time domain. ELM identifies sequences of event occurrences in the problem space called event threads, which form the basis for threads in software. ELM supports the notion of a concurrency level, which indicates the number of optimal threads for a given problem.