Software engineering in the large

by JOHN C. CHIANG

Hayes Microcomputer Products, Inc.
Norcross, Georgia

Software engineering practices typically develop, first, from coding activities and then move to more global applications of tools and engineering practices that affect the entire development cycle. There is a growing recognition that successful software engineering depends upon factors beyond practices that apply to a single project. These factors encompass multiple projects, longer time frames, and affect many groups in the organization. Borrowing from DeRemer and Kron’s “Programming in the Large,”1,2 we will refer to these broader factors as “Software Engineering in the Large”. The five examples below illustrate topics from this broader view software engineering.

Software Development Environments Extend Over Time

A series of related projects may be developed by an organization over an extended period of time. Projects in the series might be elements in a family of products. The stability of the development environment across time and projects becomes an important factor in development productivity. UNIX, with its tool box approach, provides an opportunity to stabilize the development environment.3 Gandalf4 is another example of a set of similar development environments that offer stability over projects.

The Culture of Communication

Desire alone does not fulfill the long recognized need for communication in software development. There should be a central system to archive documents, to make knowledge available to more personnel, and to serve as a future reference source. Individual project books or libraries tied to single projects do not support the larger family of projects over extended time periods.

Early Prototypes

One solution to rapidly changing project requirements and technology is to develop early prototypes that solidify requirements and build up technical know-how before implementation begins. The key to successful prototypes is to resist the temptation to patch the original prototype into a finished product.5

Life Cycle Model in Software

The traditional life cycle model1 partitions the software life cycle into rigid steps, such as specification, design, and implementation. This model, also referred to as “The Waterfall Model,” does not properly reflect the dynamic nature of software development. The newly formed “Spiral Model,”6 recognizing the dynamic nature of software development, emphasizes risk analysis and discipline in planning. It is believed that this model will be widely accepted, and a host of supporting tools and practices will emerge from it.

Physical Environment and Organization

The concept of a “software factory,”7,8 addresses office arrangements, computing environments, project team structure, and supporting organizations. The result of applying this approach can be finely tuned software development machine.

REFERENCES
