SOFTWARE ARCHITECTURE & ENGINEERING, INC.

TECHNOLOGY ROUNDTABLE

JULY 6, 1982

"The Software Cost Reduction (SCR) Project of the Naval Research Laboratory (NRL)"

NRL Principal Investigator: D. L. Parnas

Moderator: Grady Campbell

CONTEXT OF PROJECT

Sponsor: Naval Weapons Center, China Lake

Navy A-7 Operational Flight Program (OFP)
IBM System 4 PI Model TC-2 Computer
16 K 16-bit memory
22 hardware sensor/effector devices
Primary (real-time) functional modes:
- Navigation
- Weapon Delivery

PROJECT OBJECTIVES

Investigate methods for reducing costs of software maintenance

Evaluate the use of software engineering techniques

Evaluate development costs in designing for ease of change

Evaluate run-time costs of using software engineering techniques in real-time systems

Provide a model for future software development projects

APPROACH

Develop

- Requirements (existing OFP)
- Design (apply software engineering techniques)
- Implement

Document

- Products of each development phase
- Techniques used

Review

- Peer
- Expert/user

Evaluate

- Tests on A-7 simulator - correctness to requirements
- Tests on A-7 simulator - performance
- Sponsor ease-of-change

REQUIREMENTS PHASE

APPROACH

- Preparation
- Identification
- Formal specification

SUCCESSES

ISSUES

PREPARE FOR REQUIREMENTS PHASE

Identify questions that characterize all information needed.

Define specification document structure to achieve "separation of concerns."

Define documentation conventions/notations:

- templates for description completeness
- bracketing of symbolic names
- condition/event tables

IDENTIFY REQUIREMENTS

Determine available sources:

- interviews with application experts
- interviews with hardware experts
- interviews with users
- user documentation
- programmer documentation
- investigation of existing program behavior
- hardware specifications

Use sources to answer questions

Use answers to refine and expand question set

Use comments from document review to refine and expand question set

DOCUMENT REQUIREMENTS

Define:

- hardware constraints
- input/output data items
- external modes of operation
- functions (output generators)
- timing and accuracy constraints on functions
- undesired events and response
- useful subsets
- fundamental assumptions/potential changes

Identify questions to focus expertise of review audiences.

Record discrepancies found in reviews

Use review results to revise.

SUCCESSES

Focus on external behavior

Separation of concerns to focus expertise and for ease of document change.

Interaction via conceptual data items

Explicit source of requirements for design guidance

Reliance on requirements document for design to keep it valid as system definition.

Formal presentation for clarity and consistency

Preformulation of questions to prevent deferral of hard issues

Explicit fundamental assumption and potential change sets to aid completeness

Subset and potential change identification to allow design for each of change.

Review process to allow errors and misconceptions to be discovered early

ISSUES

Omission of system level description and use documentation.

Lack of explicit completeness in event/condition tables.

Difficulty in identifying system modes for user separation of concerns.

Failure to address actions at mode transition directly.

Ambiguity in event table semantics

Duality of events and conditions

Applicability to new systems

Applicability to larger systems

DESIGN PHASE

Information hiding and modules

Module structure design

Module interface design

Module internal design

Uses hierarchy and subsets

Successes

Issues

INFORMATION HIDING AND MODULES

Module: a programming work assignment; a unit of change; a set of closely related programs.

Goal of information hiding (for multiperson, multiversion software systems):

difficulty of change should be appropriate to the likelihood of change

Approach to information hiding:

- identify system details that can change; each such detail is known as a "secret"
- a module is defined to implement each secret; secrets which cannot change independently belong in the same module
- design the interface to each module so that it reveals only assumptions that are unlikely to change; the implementation of its secrets are hidden
- design the implementation of each module so that it uses no information about any other module's implementation

Module structure: the decomposition of the system into modules.

A-7 MODULE STRUCTURE

Hardware hiding

- Extended computer (EC)
- Device interface (DIM)

Behavior hiding

- Function driver (FD)
- Shared services (SS)

Software decision

- Application data type (ADT)
- Physical model (PM)
- Data banker (DB)
- System generation (SG)
- Software utility (SU)

MODULE INTERFACE DESIGN

Introduction - role of the module

Fundamental assumptions

Assumptions about undesired events

Access program descriptions

Access program effects

Additional events signalled

Local data types provided

Dictionary of terms used

System generation parameters provided

Information hidden

MODULE INTERNAL DESIGN

Module decomposition

Output produced

Function definition (event/condition tables)

- execution criteria
- value determination

Local dictionary

Mapping to requirements

Mapping to modules for interface terms referenced

Other implementation references

Timing and accuracy constraints

Secondary secrets

Design issues

USES HIERARCHY AND SUBSETS

"Uses": program A uses program B if the fulfillment of A's specification requires the presence and correct execution of B.

Purpose of uses hierarchy:

to identify possible program subsets

Purposes of subsetting:

- for incremental implementation and testing
- for reduced operational capability
- for alternative system versions

Characteristics of a subset:

- performs a subset of full system functions
- results from removing distinct sections of code and data structures with no other modification
- Uses resources comparable to a system designed to perform only that subset

SUCCESSES

Concise, explicit descriptions of module facilities

Module interaction references via symbolic interface terms.

Explicit assumptions to define constraints on the design of all facilities and any restrictions on facilities usage.

Abstract interfaces leading to independence of module implementations.

Localizing design change via information hiding.

Ease of document change due to separation of concerns.

Implementation notes to record designer's ideas.

Design issues to record the history and justifications for design choices.

ISSUES

Sensitivity to the requirements specification and nondeterminism of the module decomposition.

Reducing the behavior-hiding module's sensitivity to modes.

How to determine necessary facilities of each module.

Lack of attention to module internal design.

Appropriate traceability to the requirements specification.

IMPLEMENTATION PHASE

Coding

ITTI based pseudocode

Macro and procedure capabilities

In-line EC macrocode

Abstract data types

State Transition Event type class for process control

Virtual hardware coding

Software extensions (PM & SU)

SYSGEN constant parameters

SYSGEN effect procedures

SYSGEN formal parameter binding

IMPLEMENTATION PHASE

SYSTEM GENERATION (SYSGEN)

- Macroprocessor (EC to TC2)
- Conditional assembly
- Module version selection
- Static process creation and scheduling
- Execution of SYSGEN effect procedures
- Setting of SYSGEN constant parameters
- SYSGEN binding of formal parameters
- Development versus production system versions

ISSUES

Efficiency of object code produced

Manual translation from pseudocode to EC code

Methodology for deriving an efficient process structure

Difficulty of debugging without close source code/object code correspondence or source code level debugging facilities

Complexity of the SYSGEN process

Configuration control of implementation facilities and products