Embedded Systems Architecture

Session #7

Software Development Tools

- C Compilers
- Assemblers
- Library Builders
- Linkers
- Loaders
- Symbolic Debuggers
- Ancillary Tools

C Compilers

- C Preprocessor
- Parser
- Optimizer
- Code generator

C Preprocessor

- Performs preprocessor directives like:
  - #include *filename*
  - #define *name literal*
  - #undef *name*
  - #if *literal*
  - #ifdef *name*
  - #ifndef *name*
  - #else
  - #endif

Parser

- Lexical analysis
- Syntax check
- Feeds optimizer / code generator

Optimizer

- Maximizes execution speed or minimizes code size.
- Deletes *dead* or unreachable code.
- Type and level of optimization are typically user controllable.

Code Generator

- Converts parser/optimizer output to target specific assembly language code.

Assemblers

- Converts assembly language code to relocatable object code (I.e.: binary).
- Includes symbolic information.
- Relocatable object file formats:
  - a.out (original Unix)
  - COFF (Common Object File Format)
  - ELF (Executable and Linking Format)

Library Builders

- Creates library files from various relocatable object files.
- Libraries can contain both generic and application specific functions.
- Application specific functions include:
  - Hardware functions
  - RTOS functions

Linkers

- Combines all user relocatable object files.
- Resolves symbolic references.
- Searches libraries for as yet undefined symbolic references.
- Locates the different sections based on command file.
- Output can still contain symbolic references.

Loaders

- Takes linker output and loads it into target.
- Strips off symbolic information, if any.
- Takes care of any hardware specific programming requirements, if any.

Symbolic Debuggers

- Work in conjunction with in circuit emulators or simulators.
- Takes in object, symbolic, and source files.
- Provides the user with source level insight into the target.
- GNU Debugger (GDB) is an example.

Ancillary Tools

- Language Sensitive Editors
- Simulators & Interpreters
- Lint
- Make
- Revision Control
- Integrated Design Environments (IDE)
- Computer-Aided Software Engineering (CASE)

Language Sensitive Editors

- Render language keywords, variables, and constructs into various colors (fontification).
- Construct language specific templates (electrification).
- Emacs and vim are examples.
- Zeus

Simulators

- Simulates the target processor on host development system.
- Slower than real-time.
- Tracks execution time.

Interpreters

- Operate directly from the source code (or an intermediate form of it) instead of the compile/assemble/link method.
- Slower as a result.

Lint

- Catches questionable portions of C code that, while legal, may be in error.
- Used to help create portable code.
- Splint is open source version of lint.

Make

- Used to create (*make*) one file from one or more other files.
- The *makefile* contains the rules for how one file is made from another.
- If make finds that the destination file to be made is older than its source file, it is remade.
- Care must be taken in maintaining the makefile.

Revision Control

- Source code is checked in and checked out of revision control.
- Revision control of source code is important for large software projects.
- Different revision control systems exist:
  - Subversion, Rational ClearCase, Perforce
  - Concurrent Versions System (CVS)
  - SourceSafe (Micro$oft)

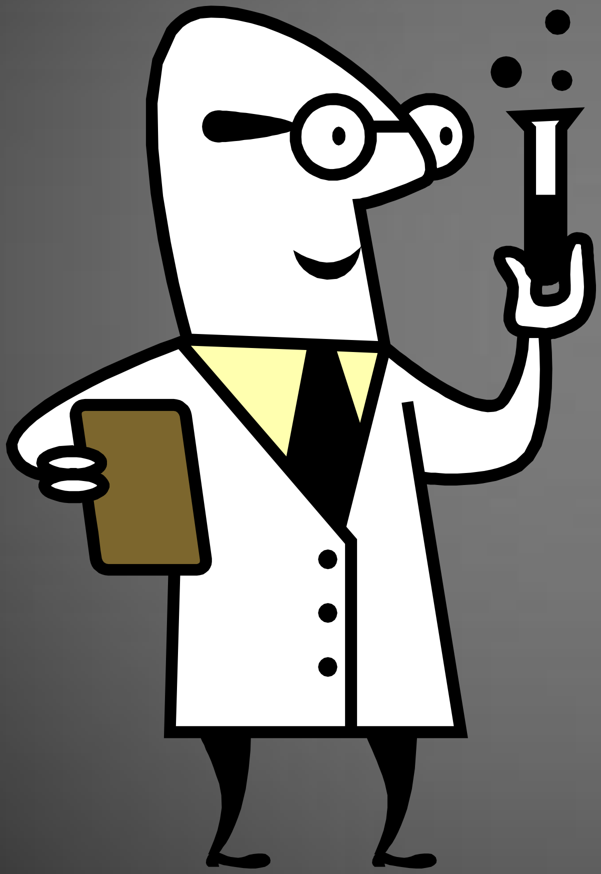
Integrated Design Environments (IDE)

- IDEs provide the user with a single GUI (Generic User Interface) where files are edited, compiled, linked, loaded, and debugged.
- Eclipse is an open-source IDE.

Computer-Aided Software Engineering (CASE)

- Provides a higher level of abstraction to software engineering.
- Design entry methods include graphical as well as textual.

Lab Session #7

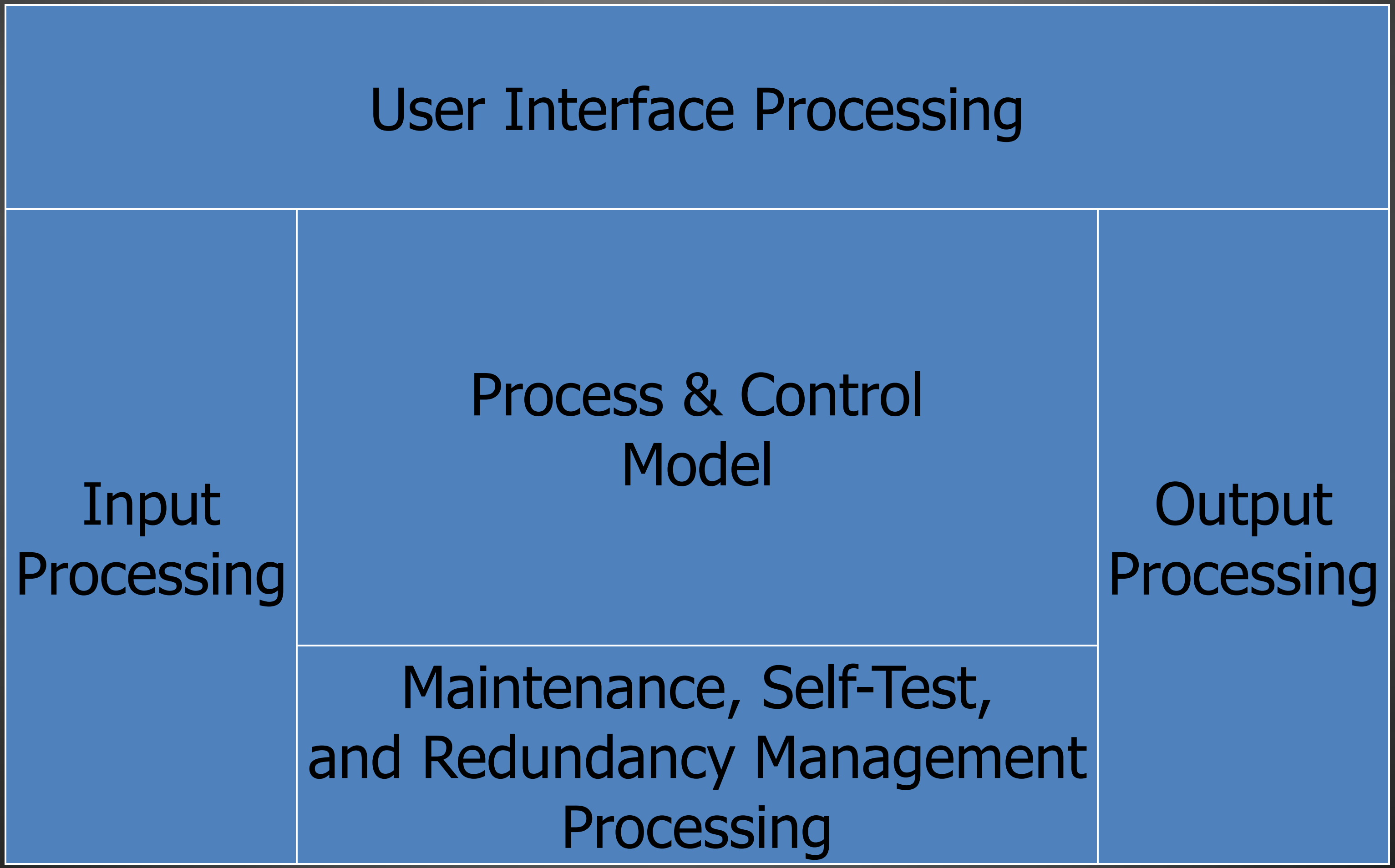


- Architecture Model #2

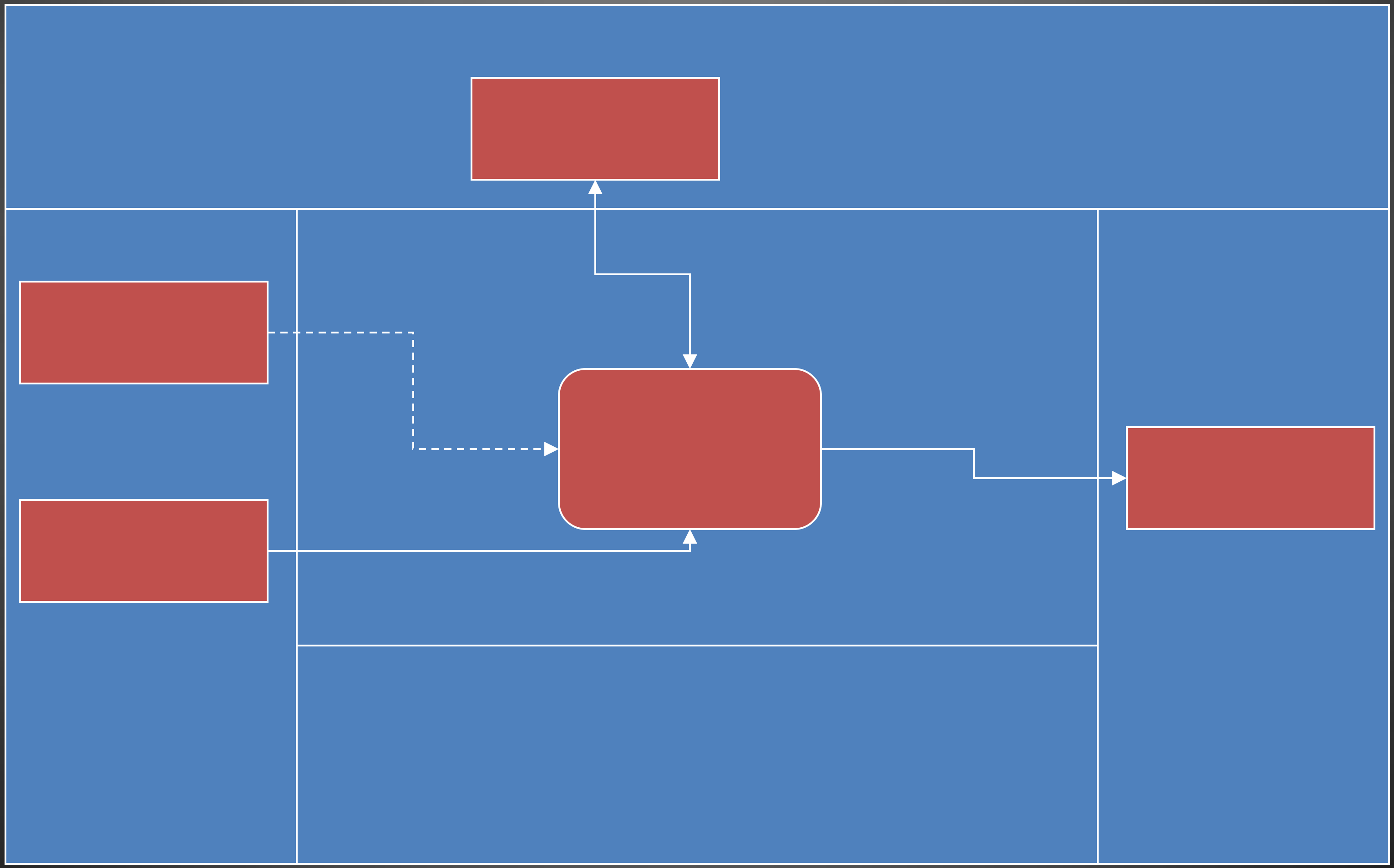
Architecture Diagrams

- The architecture context diagram (ACD) shows the physical boundaries of the system.
- The architecture flow diagram (AFD), shows the physical entities, called modules, in the system.
- The architecture interconnect diagram (AID) depicts the interconnection of modules in the system.

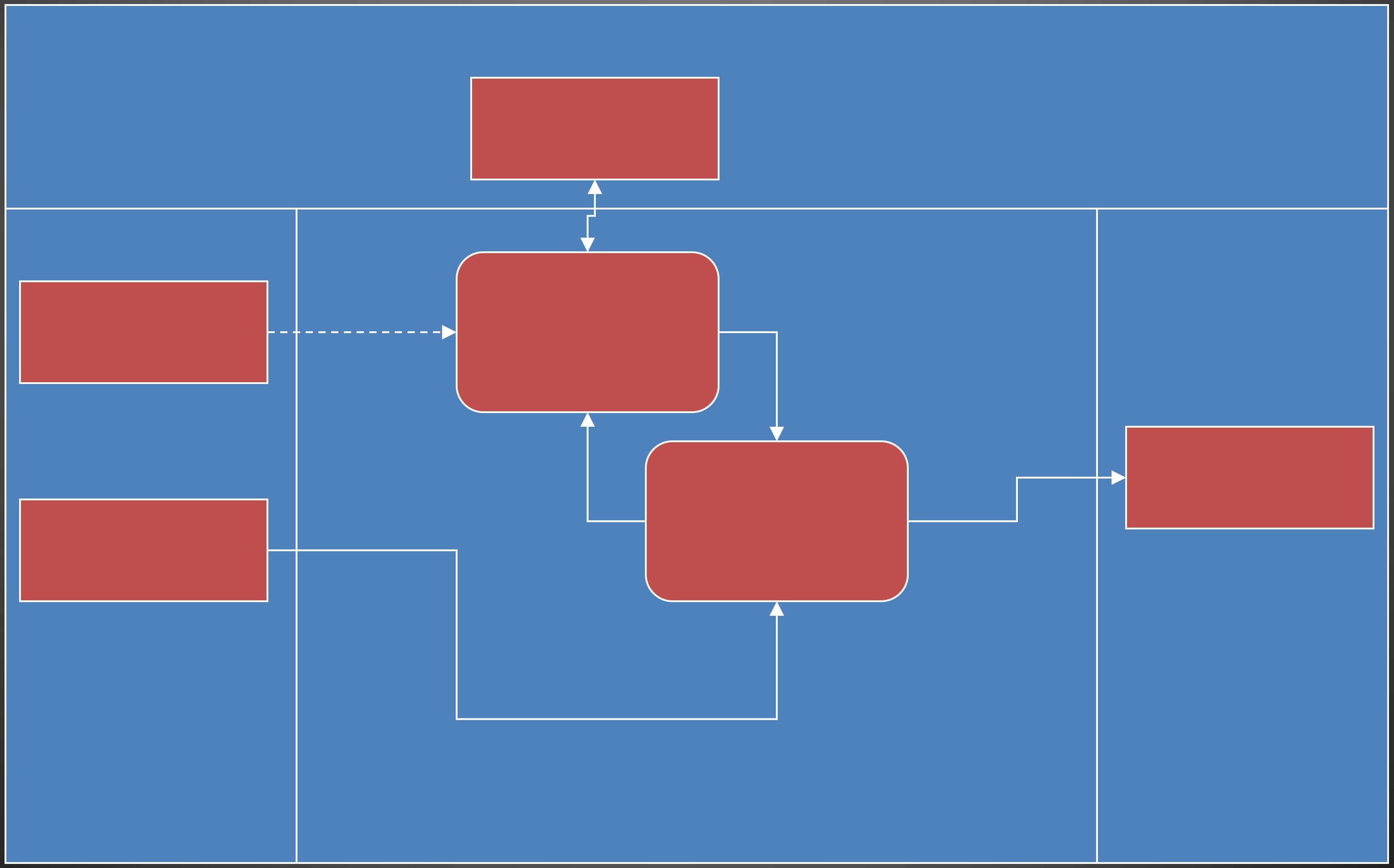
Architecture Template


User Interface Processing
Input Processing
Process & Control Model
Maintenance, Self-Test, and Redundancy Management Processing
Output Processing

Architecture Context Diagram



Architecture Flow Diagram



Architecture Interconnect Diagram