CycleFree™ Software

The CycleFree Methodology

A Simple Approach to Building Reliable, Robust, Real-Time Systems

Dick Mays
CTO, Simtrol
other stuff

&

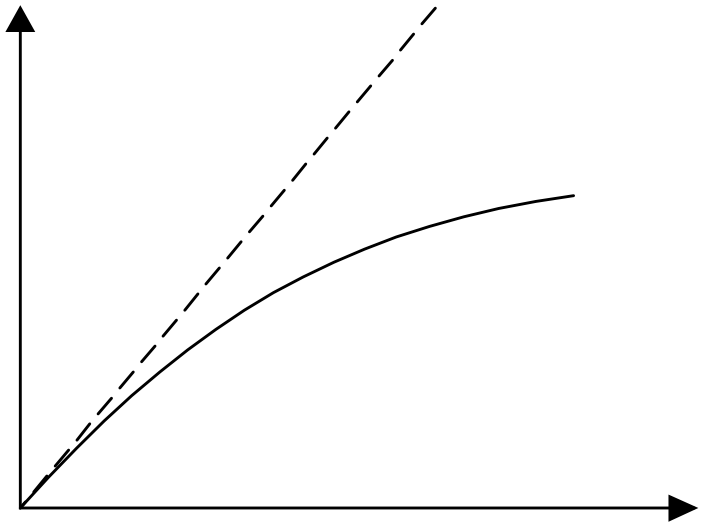
Richard LeBlanc
Professor, Georgia Tech
Seattle University

Dick's Background

- 1980 NCR – Winn Dixie Cash Registers
- Graduate School at Georgia Tech
  - Taught Operating Systems
- Smartcom III, multi-threaded
- Smartcom for Windows – state driven
- Graduate School again
- VSI/Simtrol

Result of Complexity

- Testing real-time control systems is extremely difficult.
- Programmer productivity declines over time.

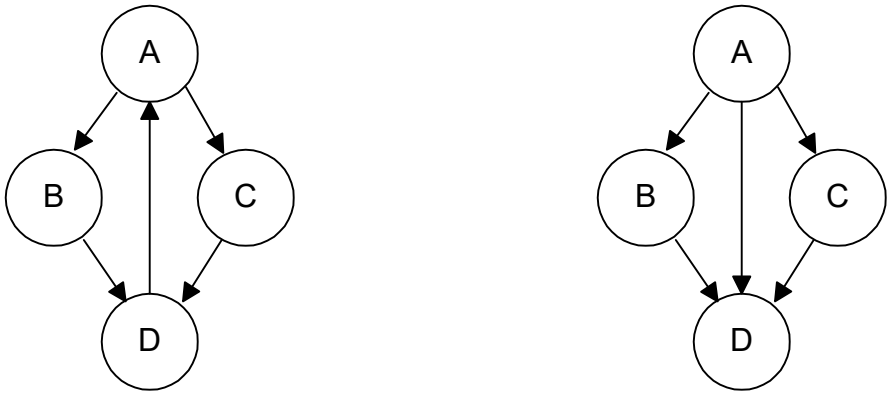


CycleFree Results

- Omega Development
  - Domain: Videoconferencing Device Control
  - Time Frame: 6 elapsed years, 21 man years
  - Productivity: 343,000 line/code, 65 lines/man-day
- ONGOER Development
  - Domain: General A/V Real Time Device Control
  - Fully Concurrent CycleFree Kernel
  - Time Frame: Two years, 12 man years

Invocation Hierarchy

- Layering concept taken seriously (enforced by the kernel)



- Drastically reduces program complexity

Transient Error

- Single Threaded Model
  - Non Reentrant Code
  - Unbounded Recursion
- Multi-threaded Model
  - Unprotected Critical Sections
  - Cyclic Deadlock
  - Priority Inversion

Goal

- **Keep**
  - ✓ Prioritized Preemption
  - ✓ Wait-For Statement
  - ✓ True Concurrency

Eliminate

Unbounded Recursion

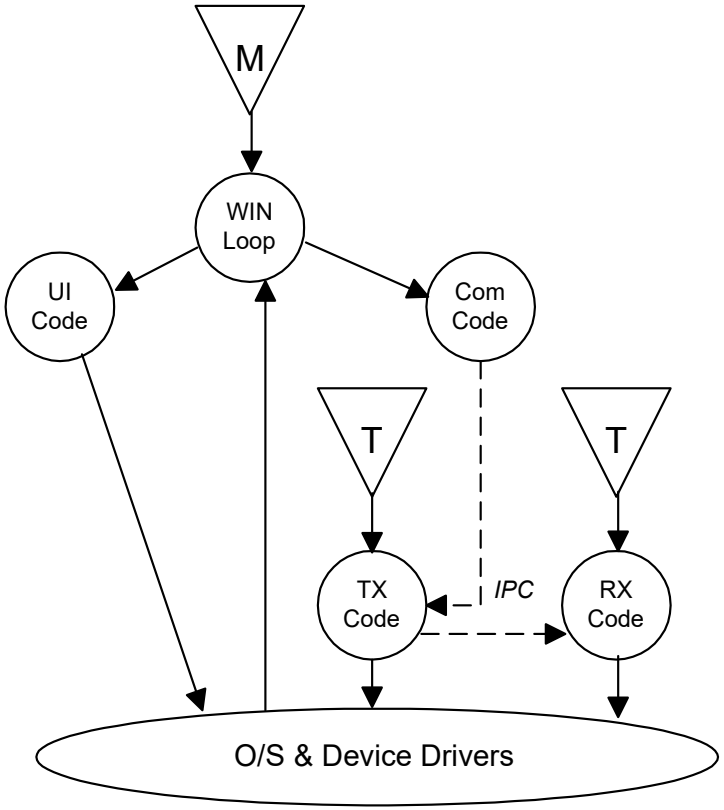
Non-Reentrant Code

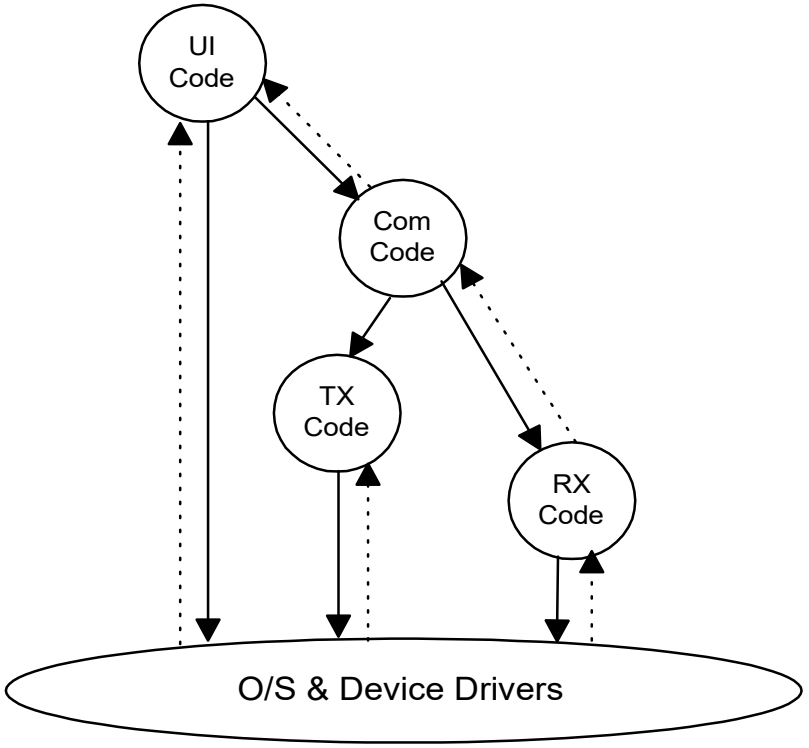
Critical Sections

Cyclic Deadlock

Priority Inversion

Structural Comparison





Definitions

- Context – Like an Object with a Level
- Event – Like a Semaphore with a Routine
- W++ is C++ extended with Preprocessor
- CycleFree Kernel
  - Enforces Invocation Hierarchy
  - Determines Event Capture
  - Scheduled Execution of Event Routines

Great programming ideas...

…have been about introducing *constraints* into the art:

- High Level Languages – *elimination of machine specific coding practices*
- Structured programming – *elimination of unconstrained control structures*
- Type systems – *elimination of arbitrary interpretation of data*
- Data abstraction – *elimination of unconstrained data access*
- Memory Management – *elimination of direct address (pointer) manipulation*

Each of these constraints eliminates a large class of errors.

The CycleFree Method

Significantly constrains the static and dynamic structures of programs, thus eliminating the most common sources of transient errors.