

# **CONTROL DESIGN FOR ROBOTIC SYSTEMS**

Eric Jackson

# Introduction

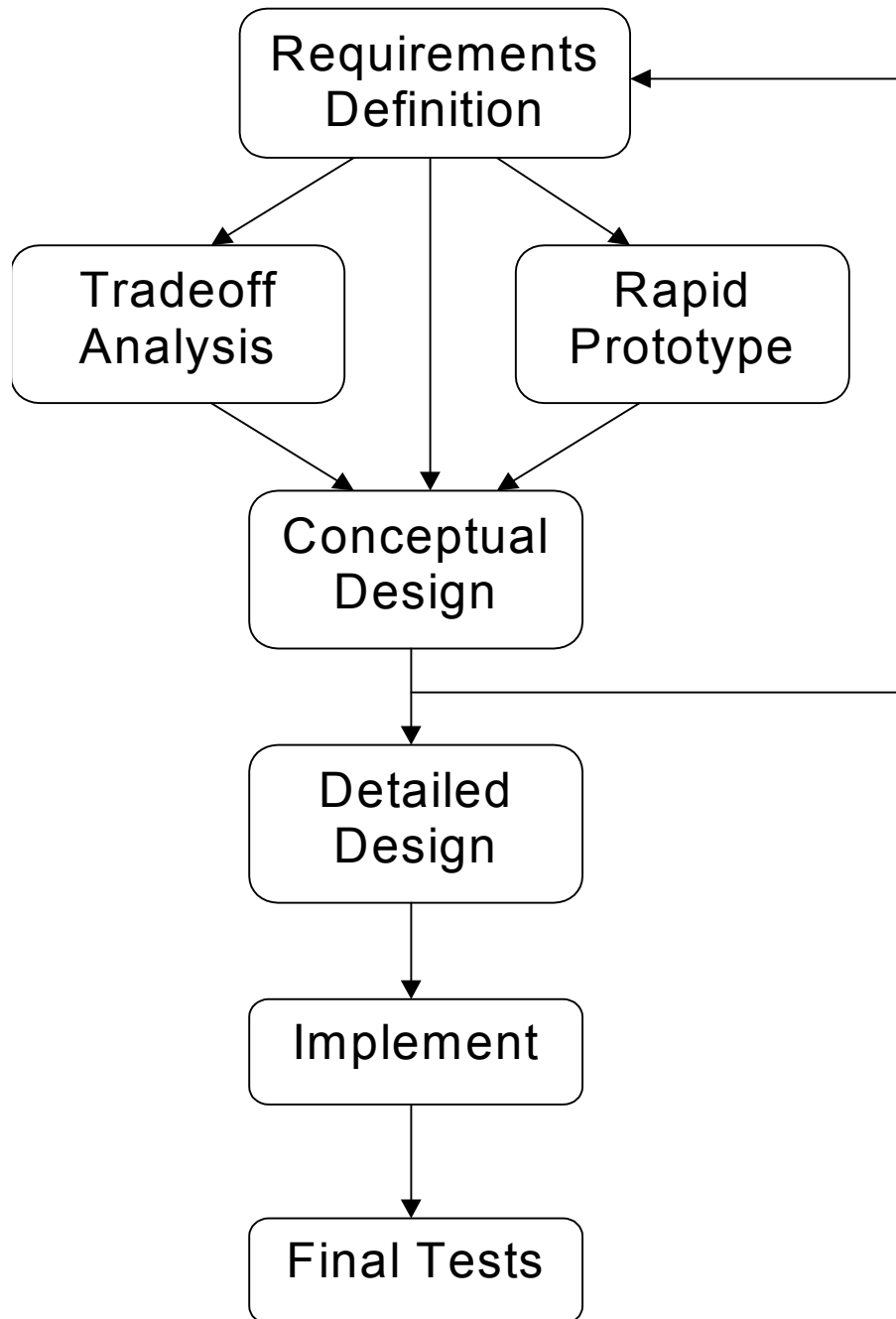
Robot projects generally require significant development.

Robot Control Systems are engineering-intensive hardware/software devices that form the brains and nerves of Robotic Systems.

Robot Control System development is a multi-disciplinary activity.

Successful projects should exploit any opportunities for parallel efforts, e.g. using multiple disciplines and simultaneous development and testing.

# PROCESS FOR DEVELOPMENT PROJECTS



# REQUIREMENTS DEFINITION

## General Considerations

Requirements are not firm - they are constrained by Budgets

Its better to define your own requirements and have the customer agree to them than to let the customer define them. Be proactive.

# **REQUIREMENTS DEFINITION**

## **Requirements Process**

System Requirements are generated by the Robotic Work Task:

- Operating Environment - temperature, operating medium, speed, altitude, depth, disturbances
- Work Task - bandwidth, degrees of freedom, geometry, manipulation, tools
- Supervisory Interaction - operator availability, communications latency and bandwidth
- Processing Constraints

# **REQUIREMENTS DEFINITION**

## **System Requirements Flow Down to Subsystems, e.g.**

- Structure
- Ballast
- Propulsion
- Manipulators
- Power Distribution
- Navigation
- Communications
- Control
- User Interface

# CONCEPTUAL DESIGN

A conceptual design is required in order to assign requirements to subsystems

It is generally more cost-effective to choose and prototype an approach early on than try to find the optimum solution up front - until you try something, you often don't know enough to find an optimum

Tradeoffs are often required in order to generate a Conceptual Design, e.g.

- store onboard energy vs. provide power through a tether,
- level of autonomous vs. teleoperated control

## CONCEPTUAL DESIGN

At this point, Analyses and Rapid Prototypes may be required, e.g.

- analyze defined tradeoffs
- test prototype algorithms
- sensor tests
- actuator power and dynamic response tests

Recommendation - prototype the integrated system, as it may exhibit unimagined behaviours.

# CONCEPTUAL DESIGN

Two Methods for Generating a Control System Requirements Specification:

- Hierarchical Task Decomposition - time domain hierarchy derived from the overall work task requirements
- Device-Oriented Analysis - derived from the other subsystems

Don't wait for requirements from customers or other subsystems - delays in control system requirements can delay a project. Be proactive.

# CONCEPTUAL DESIGN

## Time-Domain Hierarchical Task Decomposition

- Define hierarchy of “Task Verbs” for operational task definition, e.g. Survey, Retrieve, Deploy
- Define top-down Task Verb decomposition rules and algorithms, e.g.
  - Deploy -> Position, Approach, Grapple,  
Extract, Position, Ungrapple
- Define bottom-up servo control hierarchy e.g.
  - ◆ control surfaces, attitude, line-following, or
  - ◆ arm joint, end effector, trajectory

# CONCEPTUAL DESIGN

## Time-Domain Hierarchical Task Decomposition

- Define knowledge and sensory information requirements to decompose each task verb and perform each control servo
- Define exception handling
- Define User Interface capabilities (end user and developer) at each level

This procedure will define all non-I/O processing functions

# CONCEPTUAL DESIGN

## Device-Oriented Analysis

Define control system requirements for each of the subsystems, e.g.

- Propulsion control,
- Variable Ballast control,
- Manipulator control,
- Power System control and monitoring
- Navigation algorithms,
- Communications software

Correlate the results of the two methods

# PROTOTYPE TESTING

## Types of Tests

- Acceptance Tests - easy to write up but inflexible
- Characterization Tests

## Test Reports

- Minimum is Scientific Method
- More reasonable is to show a network of constraints, objectives, designs, tests, and decisions

Now is the best time to negotiate changing the requirements, statement of work, schedules, budget.

# DESIGN

## Overall Application

- Maintain the time domain hierarchy developed in the conceptual design
- Group processing functions by bandwidth and time domain behaviour
- Distribute processing between computers based on bandwidth and time domain considerations
- Respect the electromechanical system - time delays due to sampling, processing, and communications add directly to phase lag, leading to instability

# DESIGN

## Software Design

- Let controls engineers design the application, and let software engineers manage the software
- For the application design, use software components with triggers and data flow
- For software component design, use object-oriented methods
- New graphical application design tools are well-suited for this division of responsibility

# IMPLEMENTATION

A parallel implementation and testing approach should be used:

- Integrate and test the I/O software with the target hardware first
- Integrate and test additional functions from the lowest control level upward

Integrate early. The results are often non-intuitive and sometimes amazing.

## **FINAL TESTING**

Final acceptance tests are usually required. Generate a Test Plan and have it approved by the customer well in advance.

The Test Plan should show how the final tests will verify that the system meets the requirements. Keep the planned tests to a minimum. They are expensive.

Test Procedures are not normally part of the Test Plan. They are expensive to generate, but they may be worthwhile in cases where test methods might be disputed.

# CONCLUSIONS

Be proactive - requirements, conceptual design.

Integrate early.

Document - requirements, prototyping results, final test plan.

Generate and maintain a time domain hierarchy.

Divide the problem by disciplines.

Exploit parallel development opportunities.

Time delays are bad.

Respect the physics of the hardware.