Planning and Prototyping:
Welcome to the world of Project Management

Gabriel Hugh Elkaim
Announcements

1. Beer Challenge
   Wed @ 6 pm in BE-138

   Persona - 2.5 kg/1.5 kg

   Robots

   NO CODING IN 138
"IT'S OUR NEW ASSEMBLY LINE. WHEN THE PERSON AT THE END OF THE LINE HAS AN IDEA, HE PUTS IT ON THE CONVEYOR BELT, AND AS IT PASSES EACH OF US, WE MULL IT OVER AND TRY TO ADD TO IT."
The Idea Selection Process

EXHIBIT 4  Concept selection is an iterative process leading to a single concept upon which subsequent development activities will be focused.
### Rating the Options

<table>
<thead>
<tr>
<th>Selection Criteria</th>
<th>A: Master Cylinder</th>
<th>B: Rubber Brake</th>
<th>C: Ratchet</th>
<th>D: (reference) Plunge Stop</th>
<th>E: Swash Ring</th>
<th>F: Lever Set</th>
<th>G: Dial Screw</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ease of handling</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ease of use</td>
<td>0</td>
<td>-</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Readability of settings</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strength of attachment</td>
<td>0</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Durability</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ease of manufacture</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Portability</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sum +’s</td>
<td>2</td>
<td>1</td>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Sum 0’s</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>Sum −’s</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Net Score</td>
<td>2</td>
<td>−1</td>
<td>−2</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rank</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Combine</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**EXHIBIT 5** The concept screening matrix. For the syringe example, the team rated the concepts against the reference concept using a simple code (+ for "better than," 0 for "same as," − for "worse than") in order to identify some concepts for further consideration. Note that the three concepts ranked "3" all received the same score.
More Comprehensive Rating

<table>
<thead>
<tr>
<th>Selection Criteria</th>
<th>Weight</th>
<th>Weighted Score</th>
<th>Weighted Score</th>
<th>Weighted Score</th>
<th>Weighted Score</th>
<th>Weighted Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ease of handling</td>
<td>5%</td>
<td>3</td>
<td>0.15</td>
<td>3</td>
<td>0.15</td>
<td>4</td>
</tr>
<tr>
<td>Ease of use</td>
<td>15</td>
<td>3</td>
<td>0.45</td>
<td>4</td>
<td>0.6</td>
<td>4</td>
</tr>
<tr>
<td>Readability of settings</td>
<td>10</td>
<td>3</td>
<td>0.3</td>
<td>3</td>
<td>0.3</td>
<td>5</td>
</tr>
<tr>
<td>Dose metering accuracy</td>
<td>25</td>
<td>3</td>
<td>0.75</td>
<td>3</td>
<td>0.75</td>
<td>2</td>
</tr>
<tr>
<td>Durability</td>
<td>15</td>
<td>3</td>
<td>0.45</td>
<td>5</td>
<td>0.75</td>
<td>4</td>
</tr>
<tr>
<td>Ease of manufacture</td>
<td>20</td>
<td>3</td>
<td>0.6</td>
<td>3</td>
<td>0.6</td>
<td>2</td>
</tr>
<tr>
<td>Portability</td>
<td>10</td>
<td>3</td>
<td>0.3</td>
<td>3</td>
<td>0.3</td>
<td>3</td>
</tr>
<tr>
<td><strong>Total Score</strong></td>
<td>3.00</td>
<td></td>
<td></td>
<td>3.45</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Rank</strong></td>
<td>4</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Continue?</strong></td>
<td>No</td>
<td></td>
<td></td>
<td>Develop</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**A (reference):** Master Cylinder

**DF:** Lever Stop

**E:** Swash Ring

**G+:** Dial Screw+
# Satellite Morphologic Design Space

<table>
<thead>
<tr>
<th>Sub-Functions</th>
<th>Sub-Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Structural</strong></td>
<td></td>
</tr>
<tr>
<td>Configuration</td>
<td>Sphere, Box</td>
</tr>
<tr>
<td>Material</td>
<td>Aluminum, Steel, Kevlar</td>
</tr>
<tr>
<td>Design</td>
<td>Sheet, Sheets, Tapes, Titanium, Composites</td>
</tr>
<tr>
<td><strong>Navigation</strong></td>
<td></td>
</tr>
<tr>
<td>Estimation</td>
<td>TRAS, GPS</td>
</tr>
<tr>
<td>Control</td>
<td>GPS, Cables, New</td>
</tr>
<tr>
<td><strong>Attitude</strong></td>
<td></td>
</tr>
<tr>
<td>Passive Control</td>
<td>GPS, Sat, Sensor</td>
</tr>
<tr>
<td>Active Control</td>
<td>Magnet, Magnetometer, Torque, Cables</td>
</tr>
<tr>
<td><strong>Mission Comm. B. Control</strong></td>
<td></td>
</tr>
<tr>
<td>Antenna</td>
<td>Power, Antenna, High Power, Low Power</td>
</tr>
<tr>
<td>Operation Control</td>
<td></td>
</tr>
<tr>
<td>Configuration</td>
<td></td>
</tr>
<tr>
<td><strong>Thermal</strong></td>
<td></td>
</tr>
<tr>
<td>Central</td>
<td>Activa, Heaters, Coolers</td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td></td>
</tr>
<tr>
<td>Generation</td>
<td>Solar, Nuclear, Dynamic</td>
</tr>
<tr>
<td>Storage</td>
<td>Battery, Power, PEM, PSA, DMF, PSL, System</td>
</tr>
<tr>
<td><strong>Tracking, Telemetry &amp; Command</strong></td>
<td></td>
</tr>
<tr>
<td>Modulation</td>
<td>AM, AER, APR, P33, PSL, System</td>
</tr>
<tr>
<td>Track Station</td>
<td>RAD Code, Coherent Drive</td>
</tr>
<tr>
<td>Processing</td>
<td>Reduction, Compression, Essential 1, Essential 2, Command, Communication, Encryption, Authentication, Validation</td>
</tr>
<tr>
<td>Data</td>
<td>Binary, 1-0, Analog 0-5V, Serial, Parallel</td>
</tr>
</tbody>
</table>
~ x \pi^2

Time Management

DILBERT® by Scott Adams

It is physically impossible for me to finish both of my projects on time. Which one is more important?

Hmm... if I absolutely had to choose between them, I'd say... do them both on time.

Wow. When you do that with your arms, it creates the illusion that you're thinking.

What you need is a third project.

---

Figure 7-1. Keeping Engineers Busy With Multiple Projects (Reprinted with permission from United Media.)

Manage Up: { Under Promise, Over Deliver }
The Basic Gantt Chart

Figure 7–2. Gantt Chart for Automobile Bumper Design Project

Time →

Manage the critical path.

Tasks

Resource allocation
### More Complete Gantt Chart

<table>
<thead>
<tr>
<th>Task Name</th>
<th>hrs</th>
<th>Days</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary Design</td>
<td>30</td>
<td>1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31</td>
</tr>
<tr>
<td>Build Prototype</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Test Prototype</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>Final Design</td>
<td>40</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 7-3. Enhanced Gantt Chart for Automobile Bumper Project**

- Software Integration
- Finalize
- Testing

Failure is not an option - It is the result.

---

Gabriel Hugh Elkaim
CMPE 118/218 – Intro. to Mechatronics
# Gantt Chart as Tracking Tool

<table>
<thead>
<tr>
<th>Task Name</th>
<th>hrs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary Design</td>
<td>30</td>
</tr>
<tr>
<td>Build Prototype</td>
<td>20</td>
</tr>
<tr>
<td>Test Prototype</td>
<td>25</td>
</tr>
<tr>
<td>Final Design</td>
<td>40</td>
</tr>
</tbody>
</table>

**Figure 7-4.** Keeping Track of Project Progress on a Gantt Chart

Mythbusted: man-month

Keep a project notebook

Gabriel Hugh Elkaim

CMPE 118/218 – Intro. to Mechatronics
Common Pitfalls

“I ran out of time!”
“I need the computer too!”

“It had to do that too?”
“It would be nice if…”
“There must be a better way”

“Why didn’t you do it this way?”

“Why don’t these work together?”

“But how do I make that?
“I’m not sure what’s wrong”

“It worked once…”

- task awareness / manage critical path
- coordinate schedules

- focus on requirements...
- and only the requirements
- generate alternatives
- trade-off analysis
- manage subsystem interfaces

- consider fabrication aspects
- consider testing aspects

- test and integrate EARLY and OFTEN
REMOTE SWITCH

Questions?
Modular Programming:
Where the rubber meets the Code

Gabriel Hugh Elkaim
The three uses of static

1. Allocates a variable of static type & retains
   the value between function calls = \$ permanence

2. Scope restriction - cannot be accessed from
   outside the module
   static int a;

3. Pseudo-Global Variable / Module level variable
   static int c;

Public/Private
Encapsulation (1.2)

```
int foo

FunctionA()

FunctionB()

FunctionC()

char bar
```

- Functions
- Data

Implemented my interface
Helper function

Data is specific
Encapsulation (2.2)

.h - interface definition
   "contract"

.c - implement the functionality

int foo
char bar

FunctionA()
FunctionB()
FunctionC()

 insults/rc_init/RC_end
RC_addPin/RC_removePin
RC_setPin/RC_getHum

Module name.h
name.c
What goes into a header file?

**Header Guard**
- Block comment - explaining module (or package)
- Prototypes for *public* functions
- Definitions for *public* constants
- Documentation comments
- Public data structures (e.g., arrays, structs, enums, unions, bit fields, etc.)
What **shouldn't** be in a header file?

Not going in .h files:
- No functions
- No executable code
- No module variables
- No static function prototypes (private)
Where do you `#include` the header file?

```c
#ifdef _NOTOMEHSE_
define _NOTOMEHSE_ 0
#endif
```

```c
... code ...
```

```c
#endif
```
Programming Style Issues

Layout within a module

KAR

ALT-SHIFT-F

One "True" Indentation Style

if (NewTime() && ((GetTime() % 1050) == 0))
    return 1;
else
    return 0;

void putKey($SERVICE_PARAM)
{
    putchar(GET_SHARED_BYTE());
    putchar('.');
}

Use of White Space

Use of Comments

Module Level Comment
#includes
library/module headers
Public header for this module
Private Constant Definitions
Private Macro Definitions
Private Type Definitions
Private Variables
Private Function Prototypes
Code

TEST NUMBERS

while (1)
{
    MES_HandleEvents();
}

temp[0] + 123;  // mind out

Naming Conventions

Gabriel Hugh Elkaim

CMPE 118/218 – Intro. to Mechatronics
Module Design by Interface Specification

- **View**
  - The module provides **Services** to the rest of the code

- **Design Activities**
  - **Specify the services**
    - Describe Functionality
  - **Name the Services**
  - **Design the implementation**
Module: **Communications to UI on PC**

To avoid hanging up the master during the transmission or reception of messages, this module should implement buffered, interrupt driven transmit and receive. The communications routines for this module will need to be interrupt driven because the UI may send its message at any time.

**Char InitializeUICommunications(void);**
Do whatever hardware and software initialization necessary to prepare for communications with the UI on SC1.

**Void TellUINewUserReady(void);**
Should send the message to the UI that a new iButton has been inserted and read.

**Unsigned char IsNameReady(void);**
Should check to see if a new name is ready from the UI. Return TRUE if a new name is ready, FALSE otherwise.

**Unsigned char GetNewName( unsigned char NameSpace[]);**
Should copy the name gotten from the UI into the array NameSpace. The copy operation should copy no more than 16 characters, including the terminating NULL. Should return TRUE if there was a new name ready, FALSE otherwise.
Design the interfaces to modules

• Design interface for:
  – Driving the platform
  – Gathering Sensor data

• Produce
  – Public Interface specification
  – What are the details that are being hidden?
uint8_t DriveInit(void) - DriveFullStop(void) DriveStraight(int8_t speed) DriveTurn(enum type, int8_t speed) DriveEnd(void)
Hierarchical State Machines

- Harel 1980's Statecharts

Allows us to zoom in or out depending on behavior
Work out State Diagrams to Implement Finding Tape
Finding Tape

Diagram:
- Enter
- Acquiring
- Beacon Acquired
- Diving to Tape
- Stop
- Hit Tape

Paths:
- Enter → Acquiring
- Acquiring → Beacon Acquired
- Beacon Acquired → Diving to Tape
- Diving to Tape → Stop
- Stop → Hit Tape
Implementing Hierarchical State Machines

- What do you need?
State Machine Function Template

If current state is state one
  Execute During function for state one
  If an event is active
    If event is event one
      Execute action function for state one: event one
      Decide what the next state will be
    Endif
    If event is event two
      Execute action function for state one: event two
      Decide what the next state will be
    Endif
    Repeat the block above as required for each of the possible events affecting this state.
  If next state is different from current state
    Execute exit function for state one
    Execute entry function for new state
    Modify state variable to reflect the new state
  Endif
Endif
Return from state machine function
Module
d:\me218b\Lectures\Lecture 29\SMTemplate.c

Description
This is a template file for implementing state machines.

Notes

History
When     Who     What/Why
---------- --- ----
02/18/99 10:19 jec  built template from MasterMachine.c
02/14/99 10:34 jec  Began Coding

/****************************************************************************
#include header files for this state machine as well as any machines at the
next lower level in the hierarchy that are sub-machines to this machine
*/

/****************************************************************************
#define constants for the states and event for this machine

void RunStateMachine(unsigned char CurrentEvent )
{
    unsigned char NextState = CurrentState;

    switch ( CurrentState )
    {
        case STATE_ONE :  // If current state is state one
            DuringState0ne();  //Execute During function for state one
            if ( CurrentEvent != NO_EVENT )  //If an event is active
            {
                switch ( CurrentEvent )
                {
                    case EVENT_ONE :  //If event is event one
                        // Execute action function for state one : event one
                        NextState = STATE_TWO;  //Decide what the next state will be
                        break;
                }
                // If next state is different from current state
                if ( NextState != CurrentState )
                {
                    // Execute exit function for current state
                    // Execute entry function for new state
                    CurrentState = NextState;  //Modify state variable
                }
            }
    }
    return;
}
Function
   StartStateMachine

Parameters
   None

Returns
   None

Description
   Does any required initialization for this state machine

Notes

Author
   J. Edward Carryer, 2/18/99, 10:38AM

*******************************************************************************/

void StartStateMachine ( void )
{
    CurrentState = ENTRY_STATE;
    // call the (optional) entry function for the ENTRY_STATE
    // any other initialization necessary to re-start the state machine
}
Questions?