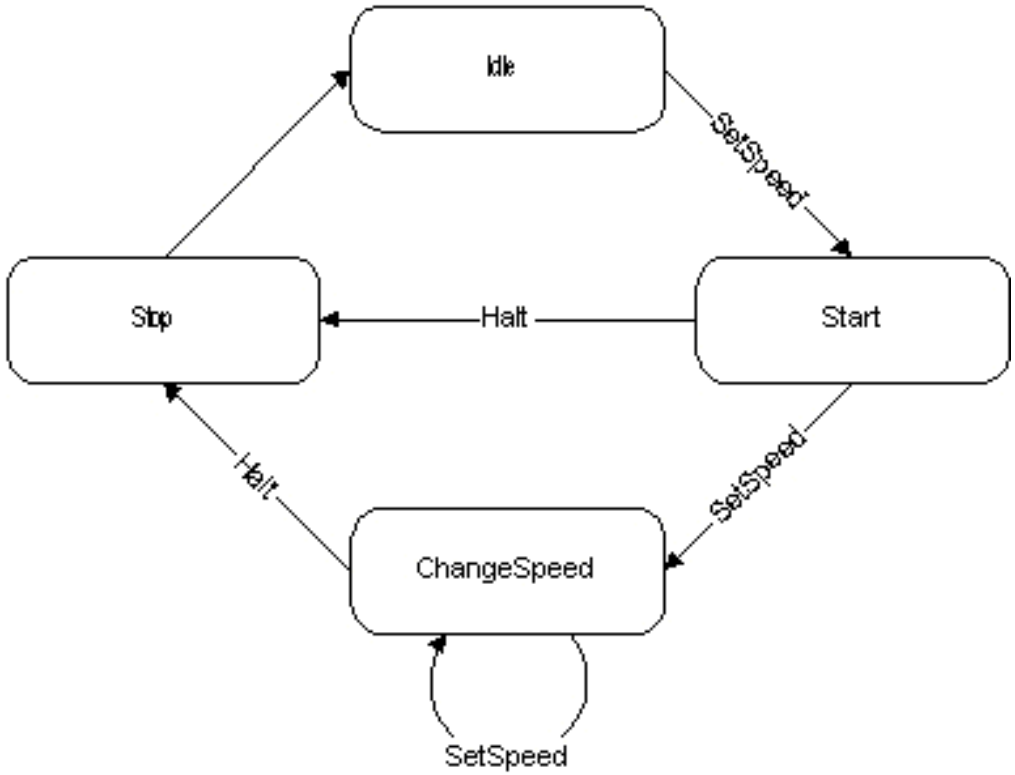
Introduction to OOP at PITZ Pt 3

Nonlinear Code Execution, State Machines & GUIs



HELMHOLTZ ASSOCIATION

DESY

Matlab mixin classes → Heterogeneous Arrays

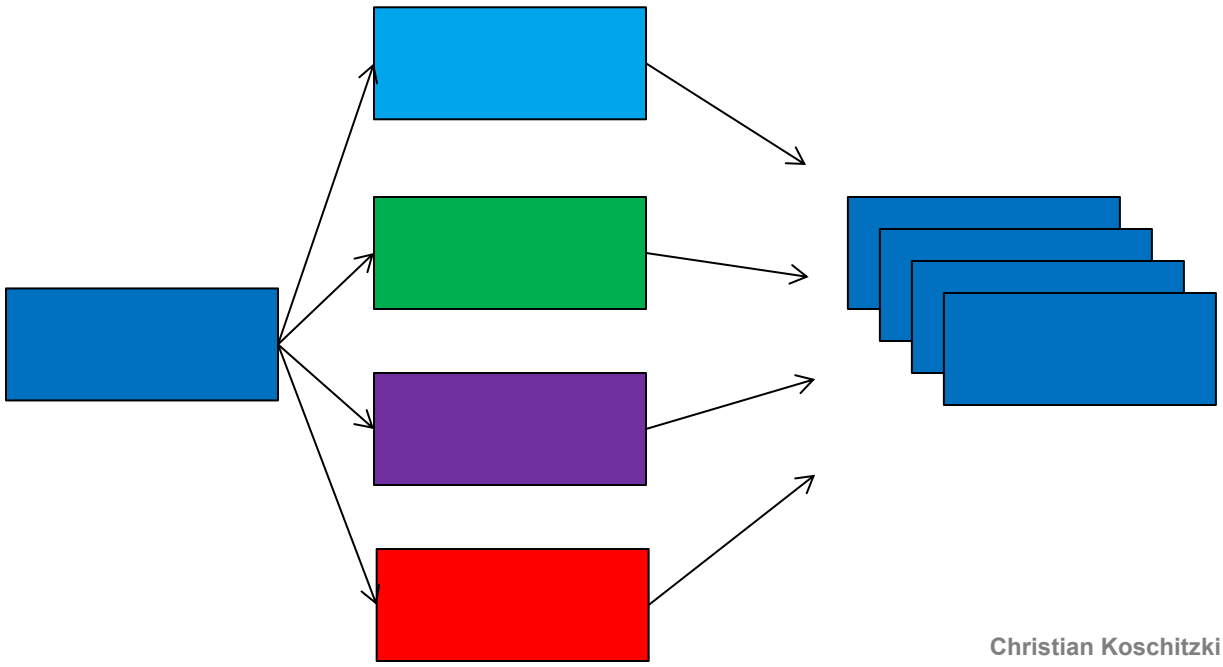
```
mot1=TL_DOOCS_Motor ('COM1',9600,…
         8,'even',1);
mot2=USBMotor('Port2');
motors=[mot1,mot2];
```

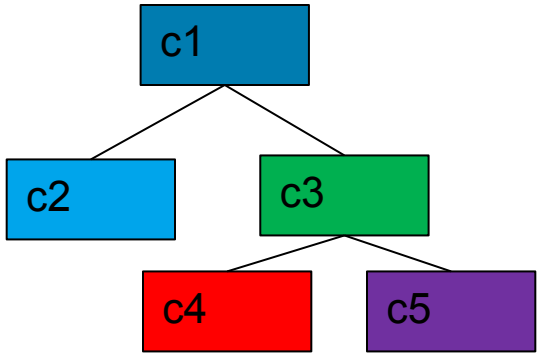
Matlab mixin classes are designed to customize object behavior and to be compatible for multiple inheritance

```
classdef Motor_class < handle & matlab.mixin.Heterogeneous
```

Prepare differently

Use as if the same





arr1=[c5,c5,c5]
arr1=[c5,c5,c4]
arr1=[c2,c5,c4]

Object array type appears as of the highest common superclass

When “implicit” gets weird

```
ob1=MyClass();
ob2=MyClass();
arob=[ob1,ob2];

arob(:).method1();
```

```
>>arob.prop
ans =
    2

ans =
    1

>> [arob.prop]

ans =

    2    1
```

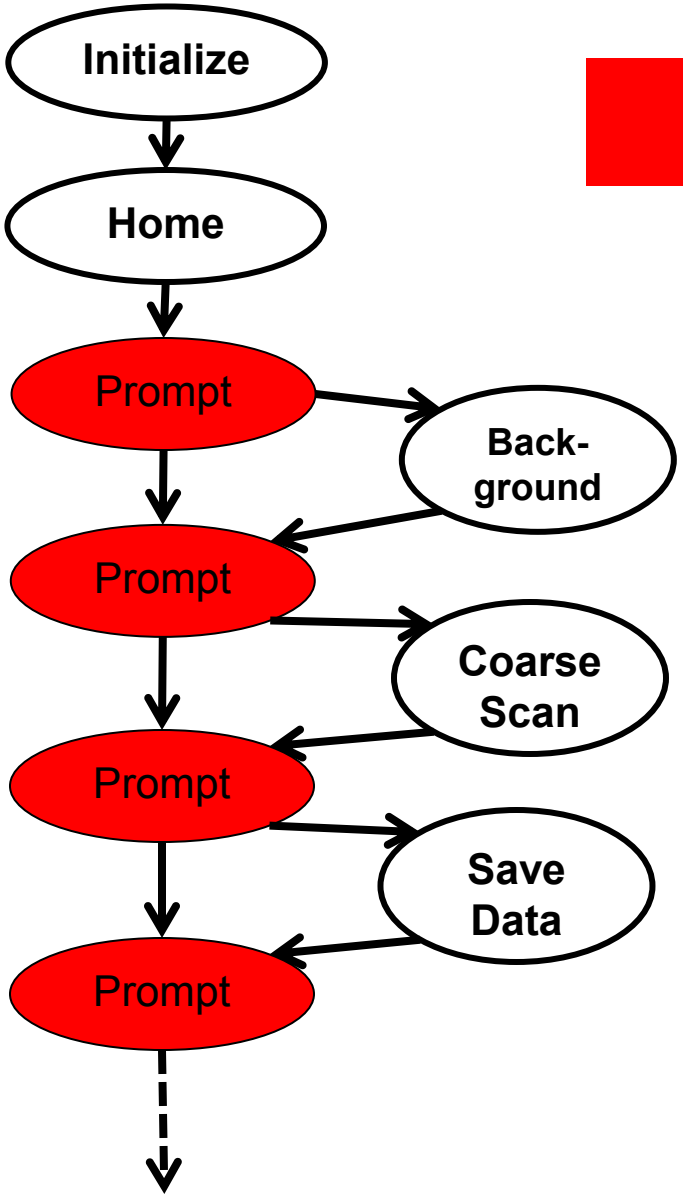
```
for i=1:length(arob)
    arob(i).method1()
end
```

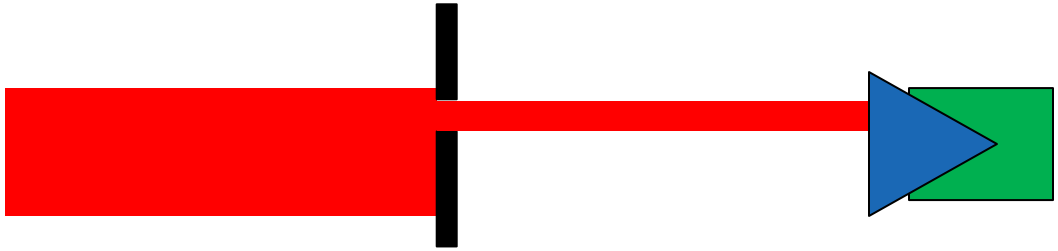
```
arrayfun(@(x) x.method1(),arob)
```

```
function method1(obj)   %obj=arob
somedat=obj.prop1;
end
```

In heterogenous arrays the common superclass method (array type) gets called, passing the object array.

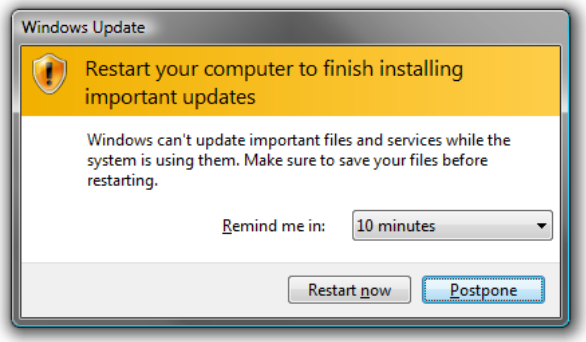
Linear Execution and Prompting



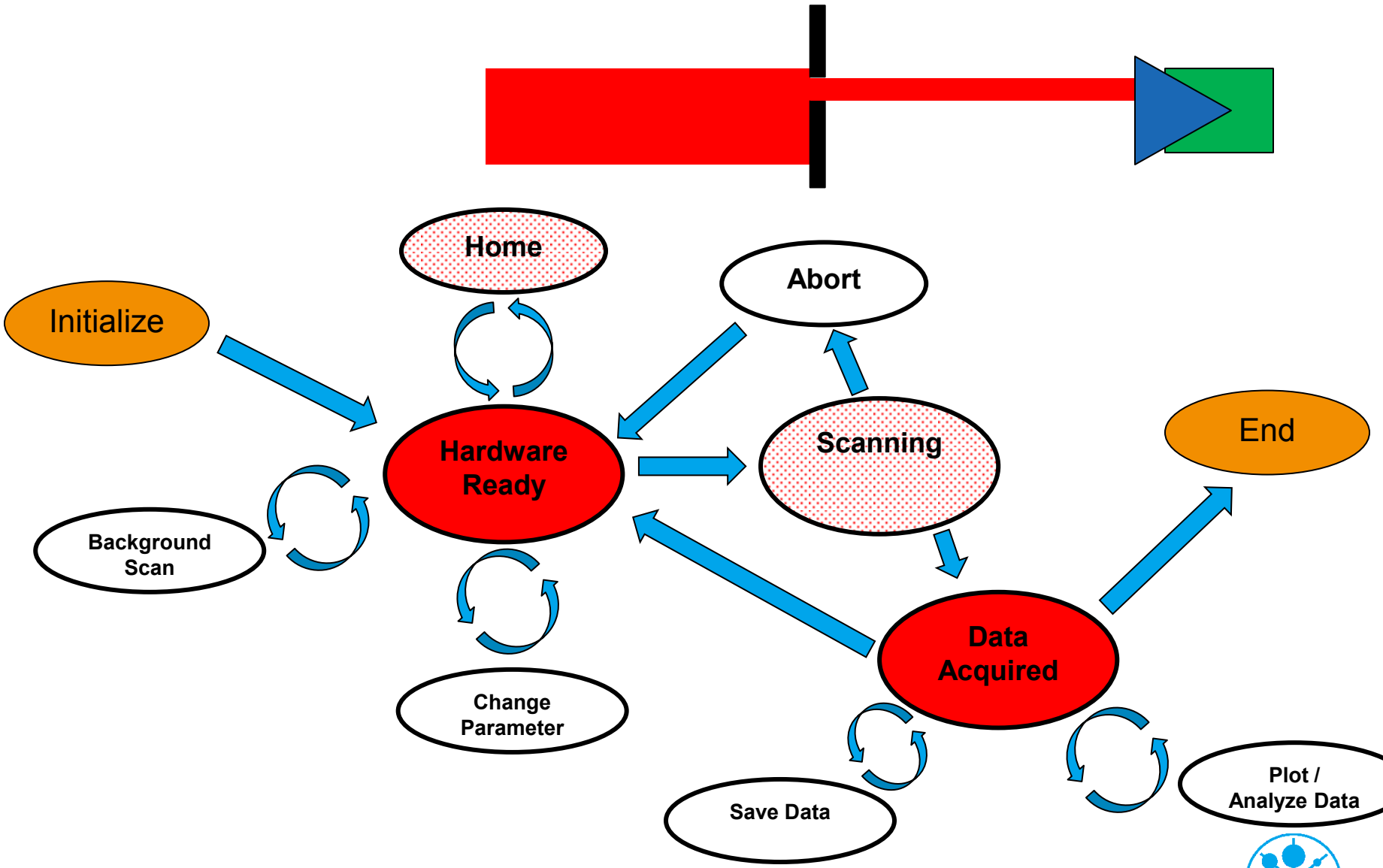


Good Prompting:

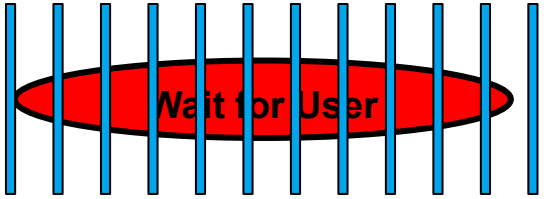
- happens rarely
- Urgent
- No other way to proceed



Nonlinear Execution

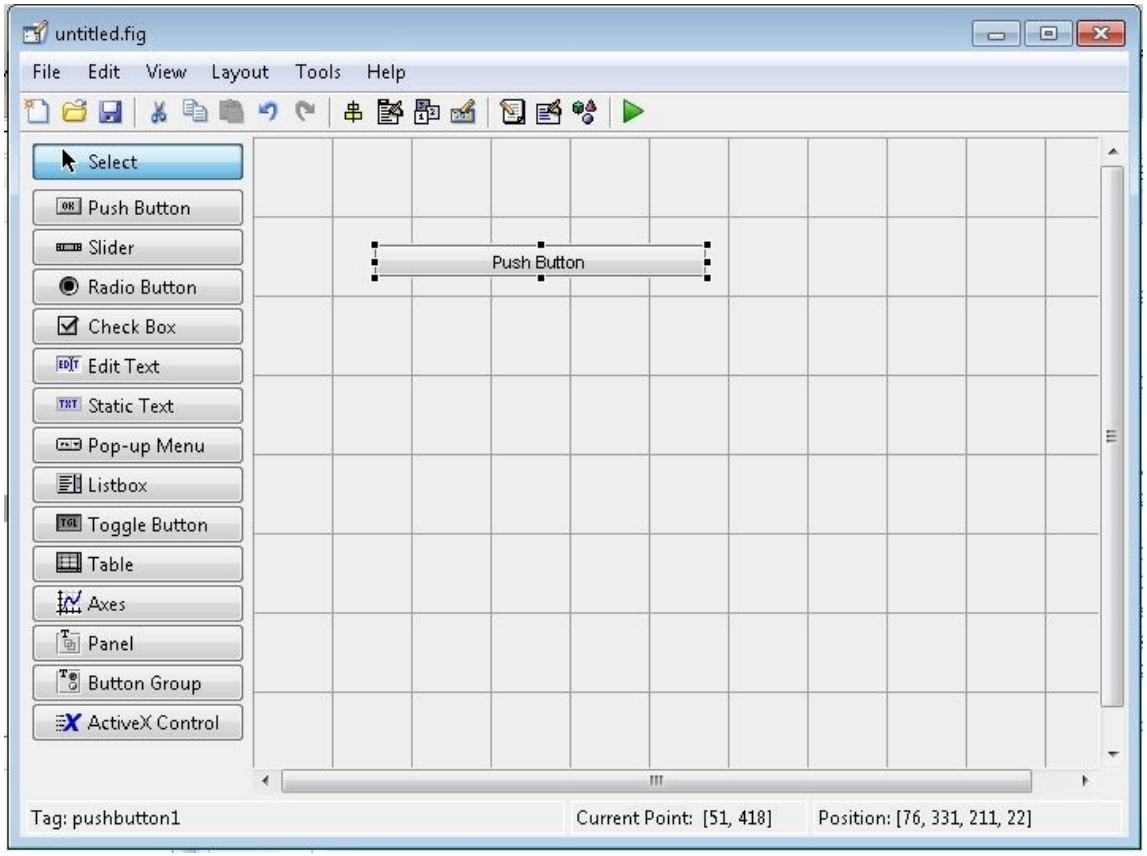


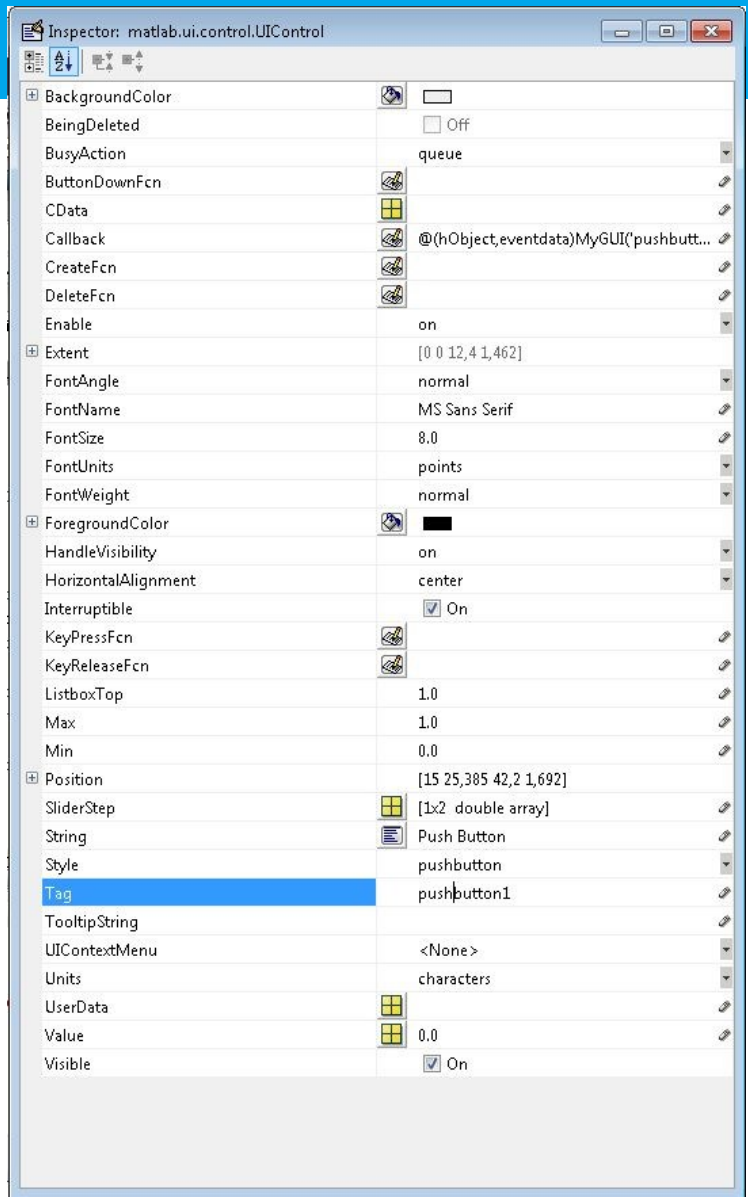
The GUI State Machine in Matlab



- Do not block execution
- Trigger state changes
- Respond to button events

Graphical User Interface Development Environment

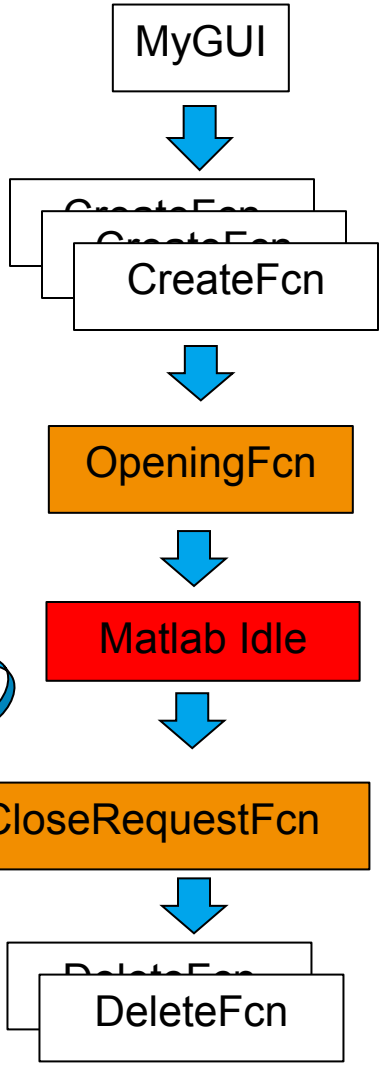




GUIDE backpanel

```
function varargout = MyGUI(varargin)

function MyGUI_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);

function varargout = MyGUI_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function pushbutton1_Callback(hObject, eventdata, handles)
DoTheThing(hObject, handles);

function pushbutton1_CreateFcn(hObject, eventdata, handles)
handle.pushbutton1.Color='red';

function MyGUI_CloseRequestFcn hObject, eventdata, handles)
delete(hObject);
```

Callbacks



handle =GUIclass Property

OpeningFcn = GUI Constructor

CloseRequestFcn = GUI Destructor

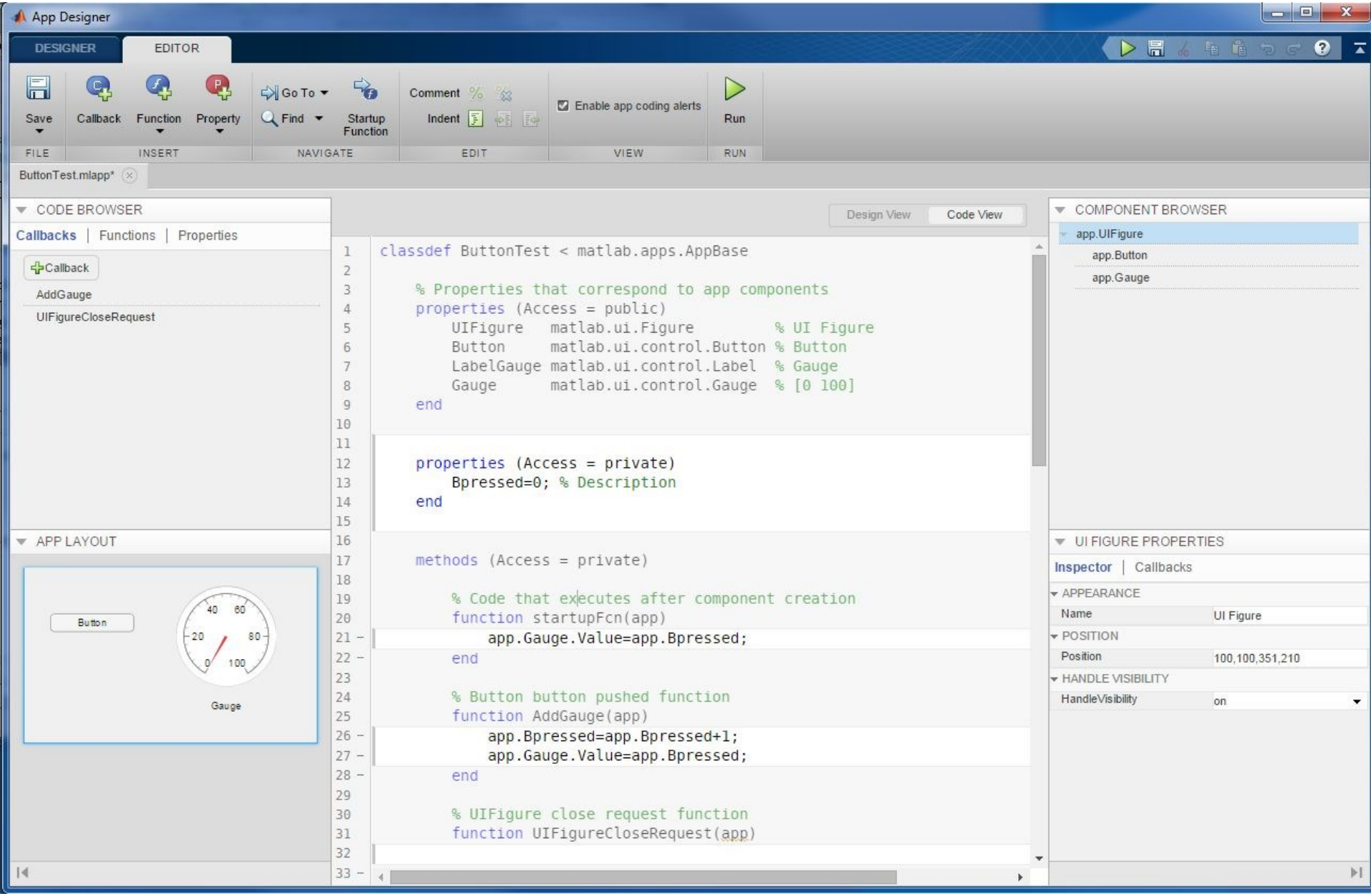
DESY

Matlab AppDesigner

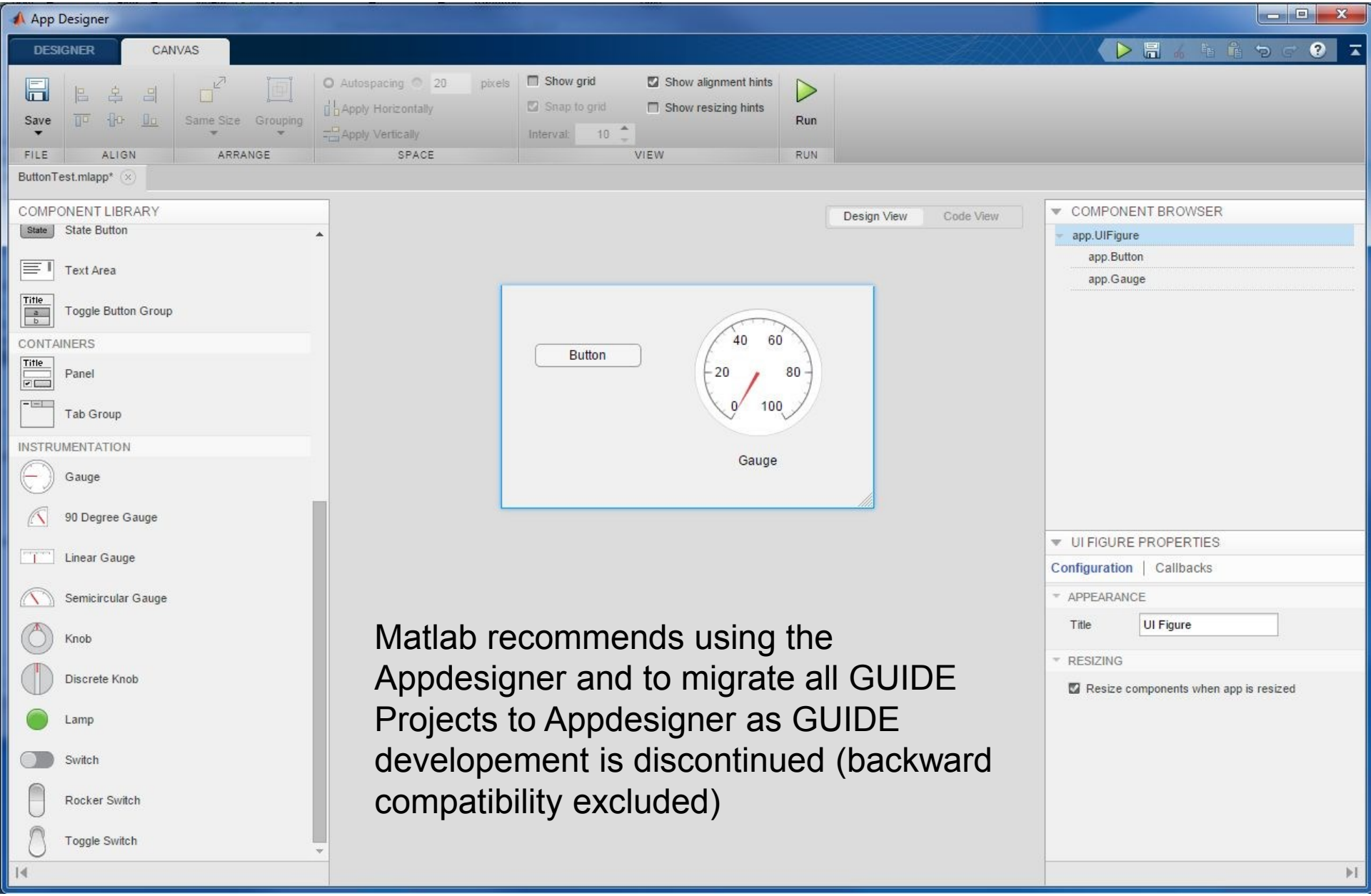
```
classdef CKMyApp < matlab.apps.AppBase
% Properties that correspond to app components
properties (Access = public)
UIFigure matlab.ui.Figure
Button matlab.ui.control.Button
end

methods (Access = public)
% Construct app
function app = CKMyApp()
% Create and configure components
createComponents(app)
% Register the app with App Designer
registerApp(app, app.UIFigure)
if nargout == 0
clear app
end
end
% Ebefore app deletion
function delete(app)
% Delete UIFigure
delete(app.UIFigure)
end
end
end
```

```
% App initialization and construction
methods (Access = private)
% Create UIFigure and components
function createComponents(app)
% Create UIFigure
app.UIFigure = uifigure;
app.UIFigure.Position = [100 100 420 328];
app.UIFigure.Name = 'UI Figure';
setAutoResize(app, app.UIFigure, true)
% Create Button
app.Button = uibutton(app.UIFigure, 'push');
app.Button.ButtonPushedFcn = …
createCallbackFcn(app, @AddGauge);
app.Button.Position = [123 299 100 22];end
end
```


AppDesigner Codeview



The AppDesigner Design View



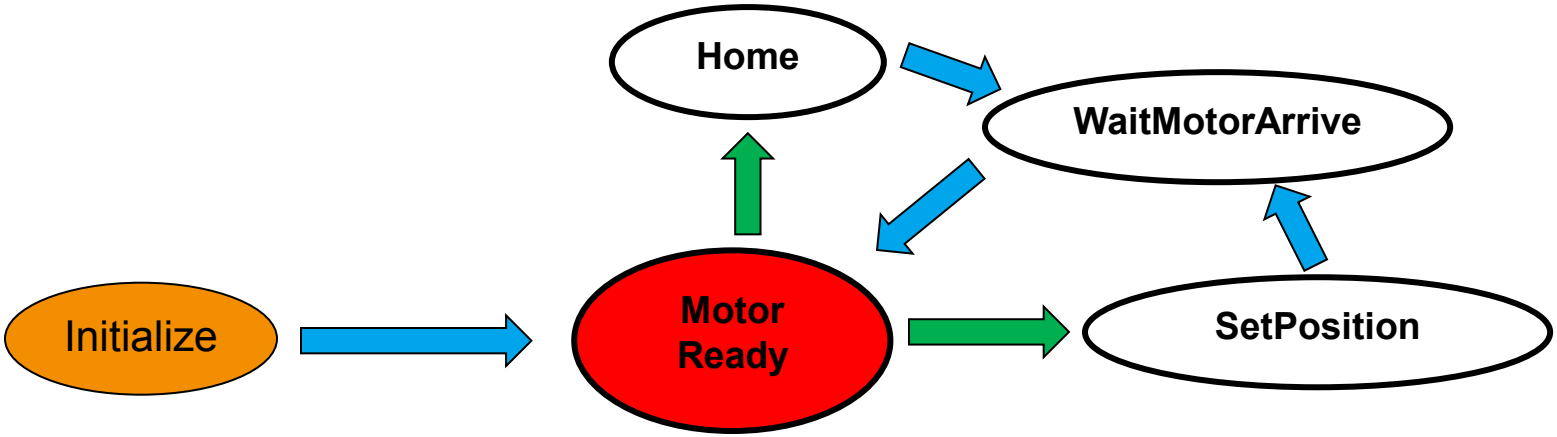
Matlab recommends using the Appdesigner and to migrate all GUIDE Projects to Appdesigner as GUIDE developement is discontinued (backward compatibility excluded)

GUIs Manage User Interfaces

- Breaking backward compatibility to introduce OOP is worth it
- GUIs are objects and thus SOLID should apply (?)
- GUI backpanels have a Single Responsibility: Manage User Interfaces!!
  - They should **not** implement solutions to the problem they visualize

```
function HomeButton_Callback(hObject, eventdata, handles)
[~] = xcomm(addr.IRSactA,1,'ACCESS','W','connection','sync');
```

  - They should redirect user input to method invoking and visualize output.
  - Implement a subset of possible usages of (possibly one) handled object.



  - Expect Every possible Button to be pressed.

Periodic State

```
function WaitMotorArrive(obj)
            while obj.moving
                obj.PollStatus;
                pause(0.2);
            end
end
```

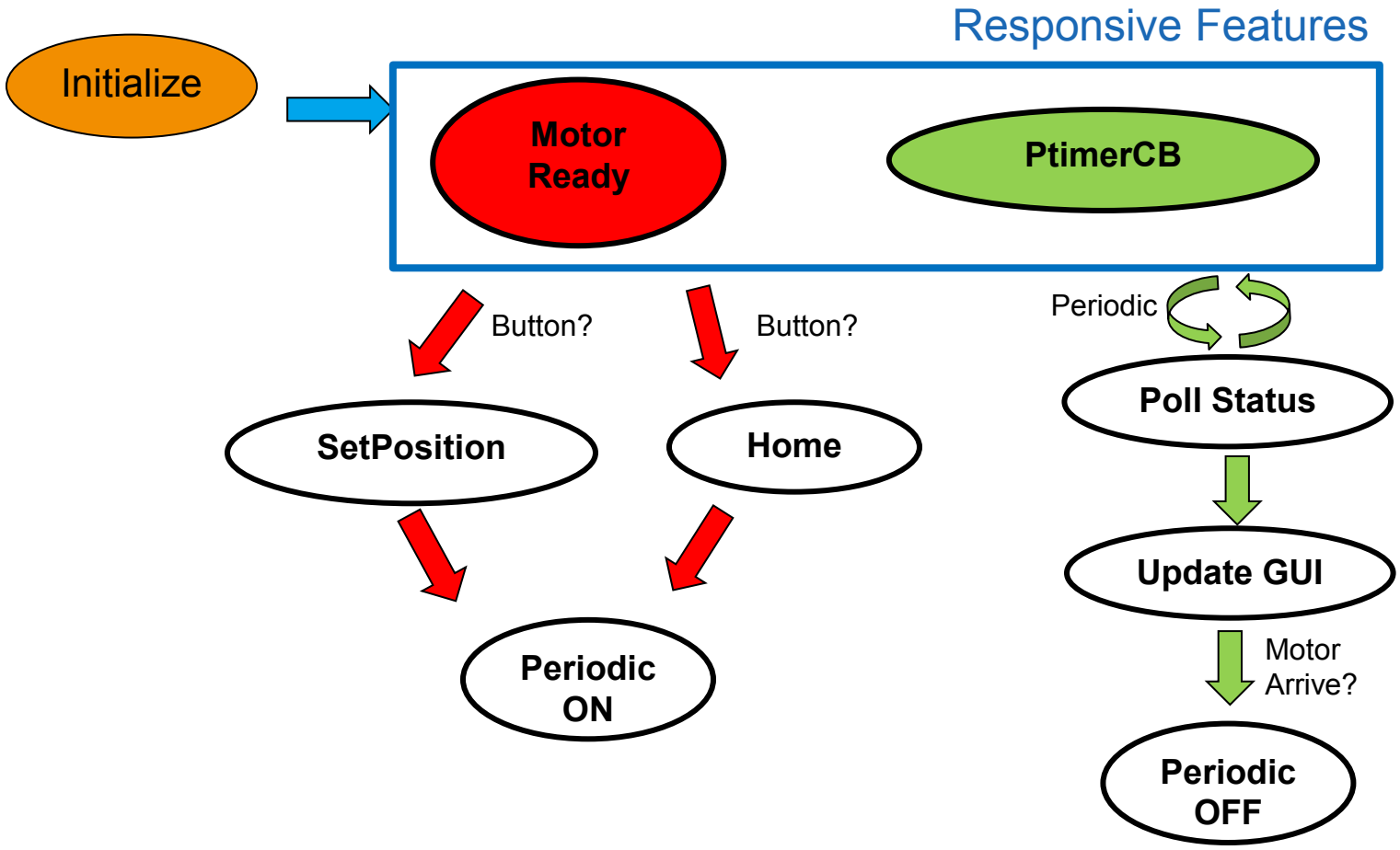
Supposed to mean:
→ Execute PollStatus every 200 ms

But also states:
→ Explicitly do nothing in rest of time

```
 methods
 function obj=TL_Doocs_Motor(doocs_adr)
          obj.ptimer = timer('Name',[obj.name, '/ptimer'],     ...
                             'Period',1,                        ...
                             'StartDelay',1,                    ...
                             'TasksToExecute',inf,              ...
                             'ExecutionMode','fixedSpacing', ...
                             'TimerFcn',{@(~,~) obj.ptimerCB,obj});
          obj.PollStatus;
 end
 end
 methods
 function ptimerCB(obj)
              obj.PollStatus;
              if obj.moving
                  obj.StopPollTimer;
              end
              obj.UpdateGUI;
 end
 end
```

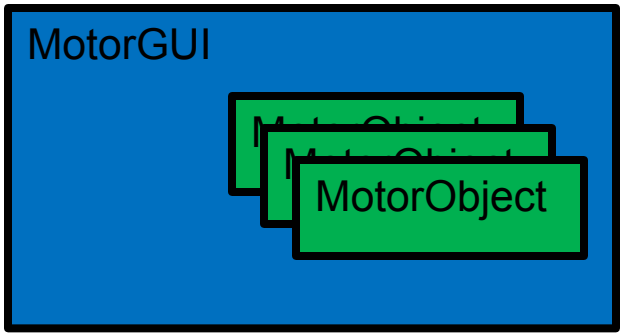
Ptimer objects can periodically call a function handle.
→ Great to implement polling

Periodic State



Watch out that period delay >> execution time of periodic callback

Object events



Ptimer gave the MotorObject the ability to execute/change “uncontrolled” by the GUI program. Changes in the GUI have to be initiated by the object.

initiated but not implemented

Dependency Inversion Principle

- High-level modules should not depend on low-level modules. Both should depend on abstractions.
- Abstractions should not depend on details. Details should depend on abstractions.

Objects should be able to inform anonymous (to the object) recipients about their activity

https://de.mathworks.com/help/matlab/matlab_oop/events-and-listeners-concepts.html

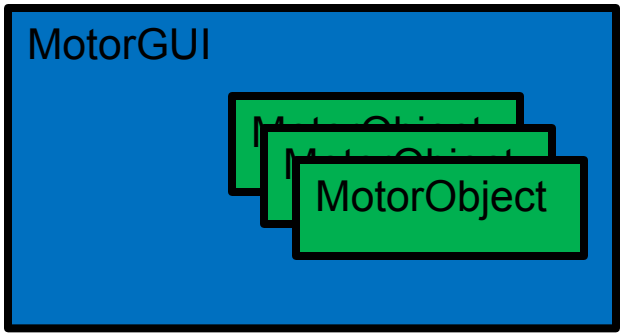
Object event notification

```
classdef TL_Doocs_Motor < handle

properties
end

events
     StatusPolled
end

methods
       function PollStatus(obj)
            [s] = xcomm(obj.STATUSadr,'connection','sync');
           obj.mstat=s.data;
           %Trigger notification for updated Motor
           notify(obj,'StatusPolled')
        end
end
```

Notify makes an object broadcast, without knowledge of recipients.

Interested recipients have to tune in to the broadcast

Listening to Object Events



If owned by script or GUIDE app

```
lh = addlistener(eventSourceObj,'EventName',@functionName);
```

If owned by another object

```
lh = addlistener(eventSourceObj,'EventName',@obj.methodName);
```

```
handles.motlistener=addlistener(handles.cmot,'StatusPolled'  @(src,evnt) MotorStatusCB(src,evnt,handles));
```

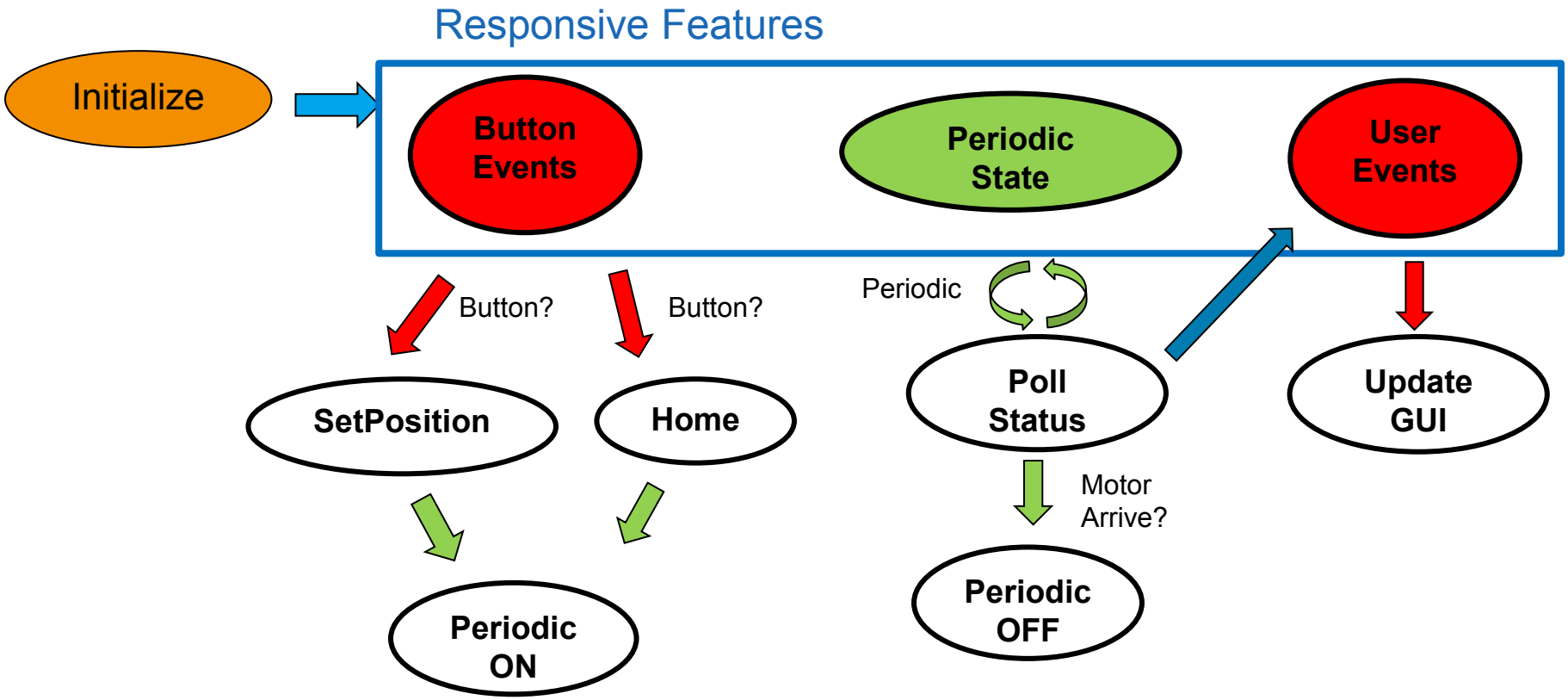
```
function MotorStatusCB(src,eventdat,handles)
            handles.CurPos.String=[num2str(src.rbkpos,'% 100.3f')];
            if src.moving
            handles.MotorStatus.String='Moving';
            handles.MotorStatus.BackgroundColor='red';
            else
            handles.MotorStatus.String='Settled';
            handles.MotorStatus.BackgroundColor=[0.94,0.94,0.94];
            end
            drawnow;
end
```

handles status isn't updated.

gets passed as it was by the time addlistener() was called

https://de.mathworks.com/help/matlab/matlab_oop/listener-callback-functions.html#brc8nnr

The 3 Reaction channels



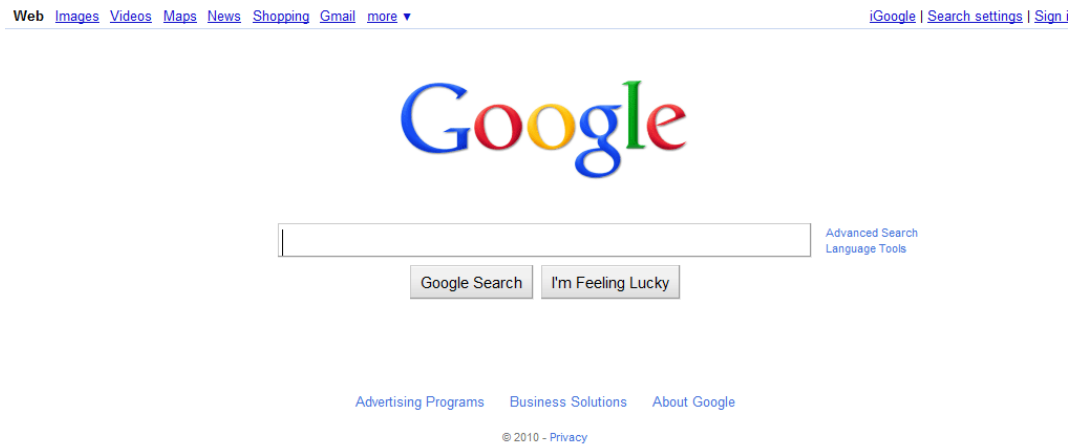
GUI call

Object call

Broadcast

GUI front panel design

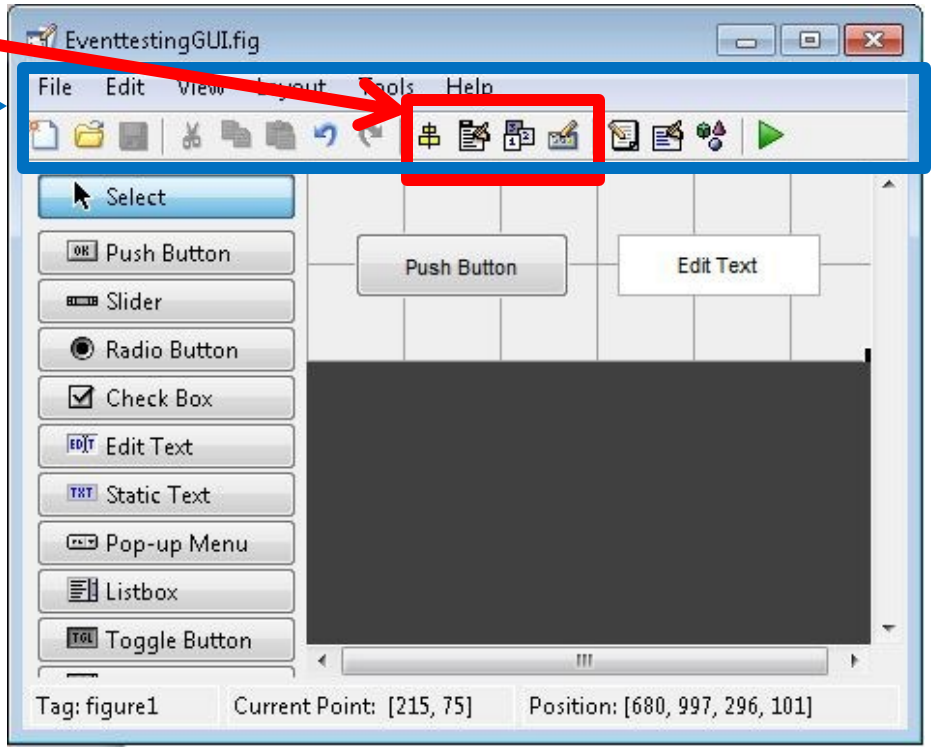
Those Companies successes might have to do with their GUI design.?



- Main controls centered, big and obvious.
- Secondary controls small, intuitively grouped and few
- Use subpanels (Application Manager)
- OR SubGUIs (GUIDE)
- labels should (exist) be short and descriptive

GUI frontpanel design 2

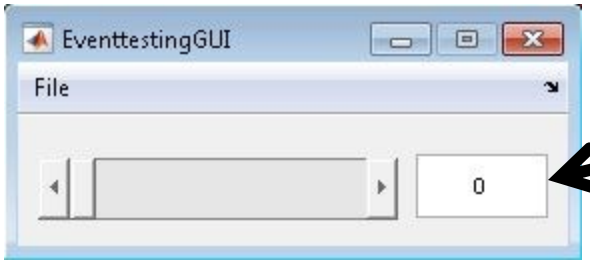
Use these
to create menus and toolbars like this



Menus and Toolbars can help unclutter your GUI

People are used to a File Menu with Save and Load Options.

Keep your buttons intuitive



This is the precision input for the same control as the slider

A recipe for the design of intuitive GUIs

- Draw your GUI on **paper**
- Use one paper for every subGUI
- Bring in a colleague who understands the context of your GUI
- Show him the Main GUI
- Ask him what reaction he expects from the buttons
- Don’t explain your drawing !!!!!!
- If a button opens a subGUI, show the paper with the subGUI
- Ask if that is the subpanel/ subGUI he expects to see
- Repeat above steps
- Don’t explain!!! **Listen!**

If you program an example GUI instead of using paper, people will not give certain criticism, because it would be to much to work to change it (they think).

Find ME

The presented GUI for Telescope design can be found at

https://github.com/Kritzek/VarTelDesigner

Questions to
christian.koschitzki@desy.de