Constrained Optimization Problem formulation for mechatronics Systems-of-systems

Herman Bruyninckx
University of Leuven, Belgium
Embedded Control Systems (H04P5)
http://people.mech.kuleuven.be/~bruyninc/ecs/

February 25, 2015

Objectives of this lecture
▶ modelling of constrained optimization problems
▶ for mechatronic systems of systems,
▶ composed together via the Composition Pattern,
▶ exploiting some “natural” hierarchies and interactions,
▶ and with impedance as the core model for motion interaction.

Note: the approach presented in this lecture is, by far, not complete, nor is it the only one to use when designing systems of systems! But since “mechanics” is everywhere, you will find few mechatronics applications that can do without the provided structure to guide your many design decisions...

Running examples
▶ mobile manipulators carrying same load
▶ drones carrying same load
▶ cars in traffic
▶ human in “shared control” wheelchair
▶ (… fill in your favourite…)
Constrained Optimization Problem

- **Objective functions**: \( \sum w_i f_i(x,y) \)
- **Optimization variables**: \( \min_x \)
- **Constraints**: \( g(x,y) \leq 0 \)
- **Tolerances**: \( \text{dist}(X,X_{\text{min}}) \leq \delta \)

Points of attention:
- Put contribution as *objective function* or as *constraint*?
- Optimization over: torque, speed, current, . . . ?
- Time horizon: beyond (first) eigenfrequency?
- Model Predictive Control: *to MPC or not to MPC*. . . ?

- There is always an “FSM” of COPs to be designed!
  - Coordinated COPs, often called “hybrid event systems”, “hybrid control”, “discrete event control”, . . .
  - During execution of behaviour of one FSM state, *parameters* in that state’s COP can also change continuously, for example: stiffening control towards end of “pick” motion; transition from “maximal torque control” to “accurate position control”; . . .
  - Three “time-based” types of constraints:
    - *Pre* conditions: to be satisfied at start of behaviour
    - *Per* conditions: to be satisfied during whole behaviour
    - *Post* conditions: to be satisfied at end of behaviour

  Lots and lots of opportunities for innovation!

Natural hierarchy #1: the “platform”

- COP “solved” at one(?) level: voltage, current, torque, speed
- “Other” levels provide *constraints*, and/or *terms* for the objective function, to the “solver” level
- Constraints: can be included in *solver* (= avoid them), and/or in *monitors* (= detect “violations” / “satisfactions”)
- Reaction to constraint events = knowledge in Coordinator
### Natural hierarchy #2: the “Coordinators”

<table>
<thead>
<tr>
<th>Coordination</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC</td>
<td>Strictly serialized behaviour</td>
</tr>
<tr>
<td>FSM</td>
<td>Single coordinator with “multi-threaded” functional Components</td>
</tr>
<tr>
<td>Orchestration</td>
<td>FSMs in hierarchical and/or peer-to-peer structure</td>
</tr>
<tr>
<td>Choreography</td>
<td>FSMs in friendly, shared-design structure</td>
</tr>
<tr>
<td>Competition</td>
<td>FSMs in adversary peer-to-peer structure</td>
</tr>
</tbody>
</table>

**Recall:** this hierarchy is behind this course’s definition of “levels of system complexity”.

### Natural hierarchy #3: the “objects”

<table>
<thead>
<tr>
<th>Plant</th>
<th>orchestrated line production ↔ “pallets”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line</td>
<td>orchestrated cell behaviour ↔ “packages”</td>
</tr>
<tr>
<td>Cell</td>
<td>coordinated device behaviour ↔ “items”</td>
</tr>
<tr>
<td>Device</td>
<td>process behaviour ↔ “motions”</td>
</tr>
</tbody>
</table>

- most often: interaction via Coordination on task execution
- exception: flow of material products

### Natural hierarchy #4: the “resources”

<table>
<thead>
<tr>
<th>Domain framework</th>
<th>Movelt, HDF5, Simulink,... to hide domain complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW framework</td>
<td>Containers, OSGi, MQTT,... middleware components to hide OS variability</td>
</tr>
<tr>
<td>Operating System</td>
<td>process, virtual memory, IPC,... for behaviour containers for deployment on hardware resources</td>
</tr>
<tr>
<td>HW framework</td>
<td>CPU, bus, storage HW resource components</td>
</tr>
</tbody>
</table>

- optimization: deployment of the resources
- interaction via Life Cycle State Machine Coordinators
Natural hierarchy #5: the “deployment”

- creating
- configuring
- running
- pausing
- deleting
- ready
- deploying
- active

Life Cycle State Machine (LCSM)
- this Coordinates “life” (“process”, “agent”, “activity”, . . .)
- the activity of a “higher” resource can only be “deployed” when the “lower” resource has become “active”
- resources at same level use LCSM to start up shared tasks

Natural hierarchy #6: the “information”

- Understanding
  - data with prediction
  - declarative
  - intention
  - why? trade-offs
- Knowledge
  - data with meaning
  - procedural
  - context
  - how? optimization
- Information
  - data with interpretation
  - episodical
  - who? what?
  - where? when?
- Data
  - facts
  - measurements
  - signals
  - symbols

- higher up, innovations have more value
- first step: in the human designer
- next step: in the software
- ultimate step: in the device

Composition #1: the “task”

- Mission
  - gives context to Tasks
  - selects trade-offs & tolerances in the production
- Task
  - coordinates & configures
  - capabilities by adding artificial constraints & artificial optimums
- Capability
  - uses resources to realise a process
  - via control
- Resource
  - physical capabilities
  - physical constraints
  - physical optimums

- learning objective of this course: skill design!
Composition #2: the “application”

- Robot platform
  - What are the robots’ motion capabilities?

- World Model
  - What can exist in the world, and where is it?

- Task
  - What does the application want the robot to achieve?

- Object Affordances
  - Which manipulation and perception activities are possible on objects?

- Common Knowledge
  - What should every application system know?

▶ this is the “real stuff”! (But it is not a hierarchy!)
▶ beyond the learning objectives of this course!
▶ it is the only thing that users want to pay for...

Composition Pattern #1: the “hierarchy”

“impedance” models:
▶ “energy transformations” towards mechanical domain
▶ “I^2T” current monitor during torque control; “dead zone” inequality constraints through hysteresis in pneumatic cylinder; mechanical spring-damper effects in hydraulic tubes

Composition Pattern #2: the “interaction”

Mechanical impedance couples relative motions:
\[ F = m(\dot{X}_1 - \dot{X}_2) + c(X_1 - X_2) + k(X_1 - X_2) \]

- “master” imposes motion; “peer” exerts force
- forces can be added, motions can’t!
- nonlinear mass-spring-dampers: finite horizon of action
- control not only motion/force, but also inertia, stiffness and damping!
"peer-to-peer" interaction can also be applied in "hierarchical" interaction; for example:
- actuator controllers are distributed over the platform;
- interaction requires "observation", realised in separate "estimator" functional Components;
- interactions can be sensed but also communicated:
  - by a subsystem’s own monitoring:
    1. 1D Self-Progress parameter: how close is behaviour to finish?
    2. 1D Self-QoS parameter: how well is behaviour performed?
    3. 1D Self-Uncertainty parameter: how confident is sub-system about its own performance?
  - by a subsystem’s functional feedforward:
    - what is it going to do in the immediate future?
    - what is the final intention of the current motion?
    - what Coordination event is about to be fired?
  - by a subsystem’s Coordination feedforward:
    - what Coordination event is about to be fired?
    - what Choreography model is being used now?
    - can we back track to previous State, please?

Conclusions
- this lecture presents "best practice" structures that can help in keeping the complexity of the system of system design process manageable.
- the software in the practical sessions uses these structures (but a lot of improvements can still be made!)
- in your own practical & design projects, you can only focus on a small subset of it.
- most innovation does not go far beyond the mentioned "small scopes".
- no full implementation of all presented structures is known to the lecturer...