The Simplex approach

Georg Köster
Outline

- Motivation: how and where other methods fail
- Simplex architecture
- Dependable upgrade
- Dynamical systems theory and robust control
- Examples
Ariane 5 tells us: software fault tolerance needed

Possible answers?

☐ Fault avoidance
  ▪ Assumes attainability of perfection
  ▪ Relies on specification / no fall-back if perfection is missed
  ▪ Expensive
  ▪ But: Ariane raises doubts

☐ N-Version programming
  ▪ Relies on redundancy and multi-perspective approach
  ▪ Ariane might have survived
  ▪ Expensive
  ▪ But: Empirical analysis raises doubts

☐ No place for fancy new technology like e. g. neural networks
Software fault tolerance revisited

□ Extremely expensive even for modestly complex systems
  ▪ Fault avoidance software development processes (e. g. FAA’s DO 178B)
  ▪ N-Version programming needs more than twice the effort

□ ‘Commercial of the shelf’-components (COTS) are usually not sufficiently reliable
  ▪ Provide superior performance
  ▪ Significantly lower cost
Never touch a running system! (?)

Cosmonauts out for a walk – space station shuts down for software update

☐ Downtime costs may outweigh possible gains

☐ Engineering problem: dependable upgrade
Outline

- Motivation: how and where other methods fail
- Simplex architecture
- Dependable upgrade
- Dynamical systems theory and robust control
- Examples
Rethink the fallback idea

- Assume well-formed diversity
  - The fallback component is more reliable by design
  - Analytically redundant with main component: both fulfill some model

- Surprisingly simple it is often applicable in the real world
  - Reliable components providing basic functionality are often readily available. E. g. predecessors or controllers engineered by robust control theory

- In the case of the simplex approach dynamical systems theory provides reliable controllers
Simplex architecture

Unreliable controller → U_new → Reliable controller → U_rel → Decision logic → U_new/ U_rel

System under control

Simplex controller
Requirements and addressed fault types

☑ Required: a system model against which the controller outputs may be checked

☑ Required: a truly reliable controller – provided by dynamical systems theory and robust control (presented later)

☑ These provided forward error recovery is possible and every type of fault in the unreliable controller is tolerated. Even semantic faults (Byzantine)
Components of the simplex architecture

- Unreliable controller
  - Might fail: produce incorrect output, miss deadlines (crash)
  - May be developed with COTS-components and/or not verifiable components like e. g. neural networks

- Reliable controller
  - Acquired through a verifiable development process or verified in practice

- Decision logic
  - Checks unreliable controller’s output ($U_{new}$, the system input) against current system state and switches control to the reliable controller when $U_{new}$ will lead to an inadmissible system state
Decision logic

- Checks unreliable controller’s output ($U_{\text{new}}$) against current system state and switches control to the reliable controller:

- Have:
  - Current system state at time $t$: $s(t)$
  - Output of the unreliable controller ($U$) at time $t$: $U_{\text{new}}(t)$

- Need to project the system state into the future taking into account system input $U_{\text{new}}(t)$. Switch if $s(t + 2)$ inadmissible

<table>
<thead>
<tr>
<th>Input to system</th>
<th>$s(t)$</th>
<th>$s(t + 1)$</th>
<th>$s(t + 2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_{\text{new}}(t - 1)$</td>
<td>$U_{\text{new}}(t)$</td>
<td>$U_{\text{new}}(t + 1)$</td>
<td></td>
</tr>
</tbody>
</table>

- $t$: $U_{\text{new}}(t)$ calculated
- $t + 1$: $U_{\text{new}}(t)$ applied
- $t + 2$: $U_{\text{new}}(t)$
Outline

- Motivation: how and where other methods fail
- Simplex architecture
- Dependable upgrade
- Dynamical systems theory and robust control
- Examples
Requirements for dependable upgrade

☐ New version of unreliable controller must be integrated into the running system

☐ Forward recovery: faults in the new version should be tolerated by anticipatory switching to the preceding version (called ‘baseline controller’)

☐ Before switching the system should be brought into a stable state – prevent switching to an unreliable controller when the system is in a state that might be unknown (baseline is not a reliable controller)
Simplex dependable upgrade architecture

- **Unreliable controller**
- **Baseline controller**
- **Reliable controller**

Decision logic

- $U_{\text{new}}$
- $U_{\text{bl}}$
- $U_{\text{rel}}$

Simplex controller

System under control

Actuators

Sensors
Course of events for an upgrade and a failure recovery

1. An upgrade is started: the new controller becomes the unreliable controller and the old one becomes the baseline controller.

2. The reliable controller takes over control and brings the system into a stable state.

3. The unreliable controller takes over – upgrade complete.

4. Upon the discovery of system state deviation the reliable controller takes over and brings the system into a stable state.

5. The baseline controller takes over – recovery complete.
Outline

☐ Motivation: how and where other methods fail

☐ Simplex architecture

☐ Dependable upgrade

☐ Dynamical systems theory and robust control

☐ Examples
Dynamical systems theory

- Provides base for the reliable controller and the decision logic

- A dynamical system is a mathematical object that describes the evolution in time of physical, biological or other actually existing systems. (Ein dynamisches System ist ein mathematisches Objekt zur Beschreibung der Zeitentwicklung physikalischer, biologischer und anderer real existierender Systeme. Desktop Bronstein)

- Rooted in studies of Newton and Poincaré – chaos theory extending it
A schematic system

Environment

- Dynamic
  - Input processing
  - State
  - Feedthrough
  - Output processing

System

- (Composite) controller
- Dynamics
- Dynamic
- State

System
Dynamical systems

- Simple linear dynamical system:
  - $x(t)$ system state at time/iteration $t$
  - $u(t)$ input at $t$

\[
x(t), u(t) : \mathbb{R} \rightarrow \mathbb{R}, \quad x(t + 1) = d(t)x(t)
\]
\[
d(t) = 2 + 1u(t)
\]

- State space $S$

\[
S = \mathbb{R}
\]
Dynamical systems

Of course there are usually more variables in state and input. This can be handled by the introduction of matrices and vectors:

\[ x(t + 1) = Ax(t) + Bu(t) \]

\[
\begin{pmatrix}
  x_{1,t+1} \\
  x_{2,t+1}
\end{pmatrix} =
\begin{pmatrix}
  a_{1,1} & a_{1,2} \\
  a_{2,1} & a_{2,2}
\end{pmatrix}
\begin{pmatrix}
  x_{1,t} \\
  x_{2,t}
\end{pmatrix} +
\begin{pmatrix}
  b_{1,1} & b_{1,2} \\
  b_{2,1} & b_{2,2}
\end{pmatrix}
\begin{pmatrix}
  u_{1,t} \\
  u_{2,t}
\end{pmatrix}
\]

State space \( S \):

\[ S = \mathbb{R} \times \mathbb{R} \]
Robust system control

- Idea is to find a controller $K$ that stabilizes the system
  - Target behavior is called setpoint
  - Even in the presence of disturbances (limited by max. amplitude)

\[ x(t + 1) = Ax(t) + BKx(t) \]

- Practically this means that in a specific region of the state space, the system state converges to the setpoint

$U$ is the recovery region
Designing the recovery region

- State constraints for the controlled system known
  - Divide state space into erroneous and error-free states
- Recovery region is described by an elliptic function (Lyapunov-function)
  - Use LMI (linear matrix inequality) optimization to maximize region
Outline

- Motivation: how and where other methods fail
- Simplex architecture
- Dependable upgrade
- Dynamical systems theory and robust control
- Examples
Examples

- Inverted pendulum with two HPC (high performance controller) and a reliable controller: dependable upgrade
  - Unreliable controller: provides fast motion to reach a new position
  - Baseline controller: modestly fast – should illustrate old version
  - Example of the system’s orbit with a fault in unreliable HPC:

![Graph showing system's orbit with a fault in unreliable HPC.]
Examples

Boeing 777

- Composite controller
- Composite digital controller
- Direct control (human)

Boeing 777

- High performance 777 controller
- 747 controller (basic)

- Decision logic

- U_new
- U_bl

- U_new/ U_bl
Examples

- Chip plant – wafer manufacturing
  - Simplex used for an experimental process-control software
- NSSN (New attack submarine program)
  - COTS-based fault tolerant submarine control system
- F-16 auto-pilot upgrade (Dependable system upgrade/Dynamic Control System Upgrade Using Simplex Architecture)
References

- Simplex Architecture Website, Software Engineering Institute, Carnegie Mellon University, [http://www.sei.cmu.edu/simplex/simplex_architecture.html](http://www.sei.cmu.edu/simplex/simplex_architecture.html)


- Lions, J. L. et al., Ariane 501 Inquiry Board report, [http://www.esa.int/esaCP/Pr_33_1996_p_EN.html](http://www.esa.int/esaCP/Pr_33_1996_p_EN.html)
Questions
Pendulum addendum

- System: state has +/- distance from setpoint, cart velocity, pendulum angle, pendulum angular velocity

- \( u(t) \) is the control input (motor voltage) and calculated by feedback control: \( K \) is constant gain vector

\[
u(t + 1) = Kx(t)
\]

- Constraints are:

\[
\{ | x_1 | \leq 0.2 \text{m}, \ | x_2 | \leq 1.0 \text{m/s}, \ | x_3 | \leq 30^\circ, \ Kx | \leq 4.95 \text{Volt} \}\]