Automated Software Design and Synthesis for Distributed Control of Aircraft Fuel Systems

Carlos C. Insaurralde
Institute of Sensors, Signals and Systems
Electrical, Electronic and Computer Engineering
Heriot-Watt University
Edinburgh, UK

Jose M. Giron-Sierra, Santiago Cifuentes, Juan F. Jimenez
Department of Computer Architecture and Automatic Control
Complutense University of Madrid
Madrid, Spain
Contents

• Introduction
• Background
• Design, Synthesis, and Verification
• Modular Simulator
• Simulation Results
• Conclusions
Introduction

• Increasing interest to develop methodologies to reduce costs and risks in development processes of control software systems.
• A suitable solution is a shortcut in the development V-diagram to verify and validate the software requirements in simulations at the design stage.
• The early verification and validation is even more effective if the software code synthesis is automated.
• This automation simplifies the code generation from models, and favors the software certification.
• Case study: Helicopter Distributed Fuel Control System (HDFCS).
• Development and testing of an automated design tool and a simulation process for SFCs.
• The methodological approach addresses the functional aspects of a HDFCS as to design and synthesis automation and simulation.
• The software (state-machine) synthesis automation is carried out by a computer tool that captures the behavior of the system components and system by flowchart diagrams.
Background

Centralized Control

• Central Computer
• Point-to-point connection
• Components without intelligence
• The control only depends on one component

Current Fuel Control System

• Central Computer
• Point-to-point connection
• Components without intelligence
• The control only depends on one component

Proposed Fuel Control System

• Elimination of the central computer
• Connection through a fieldbus
• Intelligent components
• Extend functionality; Auto-diagnosis
• Fault-tolerance by nature

Minimun wiring

Distributed Control

CAN Bus

Digital Components

Valves
Sensors
Pumps
FUEL COMPUTER
Monitoring and Control
Background

State Vector and Distributed Control System

States of the Components

<table>
<thead>
<tr>
<th>States of the Components</th>
<th>System Levels</th>
<th>System Modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>XXXX XXXX XXXX XXXX</td>
<td>XXXX XXXX XXXX XXXX</td>
<td>XXXX XXXX XXXX XXXX</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

State Vector: Bitmap

<<< Broadcasting State Vector >>>>

CAN Bus

RV1 FGT3 SP1 FGT1 FMC BP BV FGT4 HLS LLS FGT2 SP2 RV2

Subsystem 1

Tank 3

Tank 1

Subsystem 2

Tank 4

Tank 2

I/O
Background

Methodology based on Automata

- Global Automaton: *Fuel System*
- Specific Automaton: *System Operation Modes*
- Local Automata: *Intelligent Components*

<<< Broadcasting state vector >>>

CAN Bus
Background

Global Automaton

Specific Automaton

Local Automaton
Background

Experimental Platform

<table>
<thead>
<tr>
<th>Components</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BV</td>
<td>Balance Valve</td>
</tr>
<tr>
<td>BP</td>
<td>Balance Pump</td>
</tr>
<tr>
<td>SP11</td>
<td>Supply Pump 11</td>
</tr>
<tr>
<td>SP12</td>
<td>Supply Pump 12</td>
</tr>
<tr>
<td>SP21</td>
<td>Supply Pump 21</td>
</tr>
<tr>
<td>SP22</td>
<td>Supply Pump 22</td>
</tr>
<tr>
<td>RV</td>
<td>Refueling Valve</td>
</tr>
<tr>
<td>FGT1</td>
<td>Fuel Gauge Transmitter 1</td>
</tr>
<tr>
<td>FGT2</td>
<td>Fuel Gauge Transmitter 2</td>
</tr>
<tr>
<td>FGT3</td>
<td>Fuel Gauge Transmitter 3</td>
</tr>
<tr>
<td>FGT4</td>
<td>Fuel Gauge Transmitter 4</td>
</tr>
<tr>
<td>FGT5</td>
<td>Fuel Gauge Transmitter 5</td>
</tr>
<tr>
<td>FGT6</td>
<td>Fuel Gauge Transmitter 6</td>
</tr>
<tr>
<td>CV2</td>
<td>Control Valve 2</td>
</tr>
<tr>
<td>CV4</td>
<td>Control Valve 4</td>
</tr>
<tr>
<td>CV6</td>
<td>Control Valve 6</td>
</tr>
<tr>
<td>VV</td>
<td>Vent Valve</td>
</tr>
</tbody>
</table>
Design, Synthesis, and Verification

• The software development process proposed focuses on two automated stages, i.e. design and synthesis, and on partially-automated stage, i.e. simulation-based verification.

• The software development process for the HDFCS involves the following documents
  • R1. Specification of functional requirements of the HDFCS in natural language.
  • R2. Software specification by means of representation by means of state flow diagrams.
  • R3. Design report that includes the state-machine interpretations according to the function requirements specified.
  • R4. C-code implementing state-machine for Matlab/Simulink based on function blocks.
  • R5. C-code to be integrated with the target hardware.
Software Development Process

- The software development process for the HDFCS involves the following activities
  - A2. Matching of the design report with the functional requirements specifications.
  - A3. Automatic code generation from the above graphical representation.
  - A4. Backward procedure on the logic equations of the automata.
  - A5. Software code verification by means of extensive simulation based on Simulink function blocks.
  - A7. System components and system verification.
Software Description (Design)

• The design stage (A1), software description, has as input the specification of functional requirements (R1), and the state flow diagrams as output (R2).

• In addition, a similarity-based matching (A2) is performed on the design report (R3), and a feedback for the specification of functional requirements (R1).

• The functional behaviors of the SFCs can be represented by a state-machine based on the specification of the functional requirements (R1).

• This machine is automatically generated in the design processes in order to produce state flow diagrams (R2).
Automatic Code Generation (Synthesis)

- The synthesis stage (A3), the automatic code generation, has as input the state flow diagrams (R2), and the design report (R3), the Simulink function block c-code (R4), and the integration-ready c-code (R5) as outputs.
- The development tool for automatic code generation turns the logic represented by the state-machine into c-code source files for Simulink function block (R4), and also the final c-code versions of SFC software to be integrated with the target hardware (R5).
- In addition, a document to report the design specifications is provided by the above tool (R3).
System Test (Verification)

- The verification stage (A4) in simulation has as input the Simulink function block c-code (R4), and the simulation results (R6), and as output.
- The final software code compilation (A6) has the integration-ready c-code (R5) as input, and the software-hardware integration (R7) as output.
- The verification stage (A7) in rig, has the system test has the simulation results (R6), and the software-hardware integration (R7) as input, and the verification report (R8) as output.
- Both verifications are test-driven processes.
System Test

- **A2**: The functional requirements specifications are pre-verified by carrying out a similarity-based matching process on the software description (state flow diagrams). This activity is based on a language specification and format definition.

- **A4**: The software implementation are pre-verified by test cases obtained from careful examination of the flow-chart diagrams, and the logic equations which govern the transitions between state nodes. It is done by applying a modified backward procedure to the above equations so the set of test cases can be obtained.
Modular Simulator

- The simulator objective is to run system simulation tests and verify these tests against component requirements.
- The simulator has a modular nature that is achieved by selectable simulation levels.
- The system simulator implements all the defined functions.
- The logic simulation is established to be able to validate the functional requirements specification (software implementation, including state-machine).
- The operation of the systems logic and the validation of the above specification are included in a verification report to define the state-machine functionality and logic simulation.
- The simulator integrates two main simulation models: (1) the modules of the HDFCS components (including SFCs), and (2) the modules of the fuel flow using respective tanks, and fuel pipes.
- The former (1) provides a detailed simulation of the internal processes of the HDFCS components.
- The latter (2) is described in the next Subsection.
Simulator Architecture

- The coupling of the both model raises the challenge of integrating two different domains: discrete and continuous simulation.

![Simulator Architecture Diagram](image_url)
Fluid Model

- The modeling of HDFCS components is made by following the specifications given in requirements reports.
- The approach considered to model the fluid flow is based on the key fluid variables: pressures and flow rate.
- The fuel is considered as incompressible so the flow quantity is considered as invariant, i.e. the total flow remains constant from pressure sources to pressure sinks.
- The following equation to model the flow across the different elements (valves, pipes, pumps) which connect pressure sources and sinks (tanks, fuel supply hoses and engines, system2) can be applied

\[ P_{out} - P_{in} = \frac{1}{2} \rho R(v) v^2 \]  

- where \( P_{in} \) and \( P_{out} \) represent the pressure in the ends of the element, \( u \) is the fluid velocity, \( \rho \) the fluid density, and \( R \) is a generic resistance coefficient.

Equation 2 is used to carry out the instantaneous flow rate along (flow line)

\[ P_{out} - P_{in} = \frac{1}{2} \rho R(\Phi/A) \Phi^2/A^2 \]  

- where \( A \) represents the section of the element.
Fluid Model

\[ \Phi = \text{sign}(\Delta P) \sqrt{\left| P_{\text{out}} - P_{\text{in}} \right| \rho R/2A^2R(\Phi/A)} \]  (3)

- where \( \text{sign}(\Delta P) \) represent the sign (+/-) of the pressure difference between the ends of the element.
- A positive flow rate has been implicitly associated with a fluid flowing from the pressure source \( P_{\text{in}} \) to the pressure sink \( P_{\text{out}} \), and a negative flow rate with a fluid flowing in the opposite sense.
- Schematic view of a flow line, connecting a tank, which acts as a pressure source, with an engine (a pressure sink).

- Equation 3 is an implicit equation, as far as the resistance factor \( R \) depends on the flow rate \( \Phi \).
Fluid Model

- Equation 2 provides a model of the fuel system in a modular manner.
- Several elements can be concatenated between a pressure source and a pressure sink.
- The resulting system can be described by the next two equations, which can be combined in a single equation to obtain again the flow rate along the complete flow line.

\[
P_{\text{out}} - P_{\text{link}} = \frac{1}{2} \rho R_2 \left( \Phi^2 / A^2 \right) \tag{4}
\]

\[
P_{\text{link}} - P_{\text{in}} = \frac{1}{2} \rho R_1 \left( \Phi^2 / A^2 \right) \tag{5}
\]

\[
\Phi = \text{sign}(\Delta P) \sqrt{\left( |P_{\text{out}} - P_{\text{in}}| \right) \left( \rho / 2 A^2 R_2 + \rho / 2 A^2 R_1 \right)} \tag{6}
\]

- The generalization for a set of $n$ elements in the flow line is trivial:

\[
\Phi = \text{sign}(\Delta P) \sqrt{\left( |P_{\text{out}} - P_{\text{in}}| \right) \left( \rho / 2 A^2 R_n + \ldots + \rho / 2 A^2 R_1 \right)} \tag{7}
\]
Fluid Model

- The whole HDFCS system can be described in terms of flow lines (HDFCS flow-line model).
Simulation Results

Fuel Transfer for Balance (Engine Supply)

- The simulation of the balance system demonstrates the automation performance of such operation.
- The simulation shows an automatic balance after takeoff, and is to compensate the initial fuel imbalance between both systems.
Simulation Results

Fuel Transfer for Balance (Engine Supply)

• The imbalance is detected by the state-machine after the starting up of the (model) system. After starting both engine and setting the WoW signal to flight, the balance pump starts to send fuel from system 1 to system 2.

• It is interesting to notice how the effect of the balance is a decrease in tank 5 and tank 6 levels and the corresponding increase of tank 1 and tank 2.

• The reason is that tank 3 and tank 4 are completely full, and the jet pumps activity tends to hold the tank full. Thus, although the transfer of fuel during the balance process takes place between tanks 3 and 4, tank 3 is automatically replenished by the jet pumps of system 1.

• In subsystem 2, the fuel arriving to tank 4 is immediately sent back to tanks 1 and 2 through the respectively tank overflow connections.
Simulation Results

Fuel Transfer for Balance

• The evolution of the fuel flow along the balance line during the first 30 second of the test. The WoW signal is set to flight around second 4.
• Right after, the flow in the balance line begins to rise.
• The figure on the right shows the evolution of the pressure drop between both sides of the balance pump.
• As it can be seen, after roughly 520 s the pump stops because the imbalance between the two systems is less than 25 kg.
Simulation Results

Refueling of Tank Systems

• The simulation of the refueling of tank systems demonstrates the automation performance of such operation. It sets the preselected value for refueling to 300 kg. In this way the total amount of fuel refueled fits into tanks 3 and 4.

• The figures show evolution of subsystem 1 and subsystem 2 respectively. There is an unexpected event that takes place at time ~24s. Although the preselected fuel mass should fit into tanks 3 and 4, part of the fuel enters into the other tanks (1, 2, 5 and 6).
Conclusions

• Automated design and synthesis processes to develop a HDFCS have been presented.

• Tests carried out on simulations demonstrate that the results (main focus on the software code automatically generated, i.e. state-machine) obtained from the above processes are according to what has been specified.

• The state-machine generated works as expected in the simulator. However, there are some cases (near the limits of the tank capacities) in which the state-machine (software implementation, i.e. code generated automatically) makes some states of valves oscillate.
Future Work

• Future research work will include the integration of different modeling languages by means of combining models with diverse nature.

• The above further investigation makes it possible to pre-verify the HDFCS at early stages of development (analysis and design) by considering the cross-domain impact of changes from different viewpoints.