Formal Modeling of Air Traffic as System-of-Systems

German Aerospace Center (DLR)
Institute of Flight Guidance, Department of Air Transportation

Alexander Gillissen, Michael Schultz
Aviation System

• **Systems**
  - Airports, Groundhandler, Aircraft, ANSPs, Airlines, Governments, ...

• **Operations Planning**
  - Flight Schedule, Arrival-Departure Schedule, Turnaround, ATC, Trajectory Planning, ...

• **Architecture/Relations**
Aviation as a System-of-Systems

- Aviation is a System-of-Systems
- Aircraft:
  - Passengers, freight, “missions”
- Airports:
  - Landing/take off
  - Connection with passengers/freight
- ATC
  - Flight/airspace coordination
Aviation as a System-of-Systems

• Multiple systems are required to enable flight operations
• Systems operate independently, but depend on other systems to fulfill their own objectives
System-of-Systems Concept

• Describes a distributed systems architecture
• Comprised of multiple heterogeneous systems (airport, aircraft, ansp, etc.)
• Working towards a common goal (i.e. global aviation/flight operations)
• Systems still have their own operational objectives
• Systems are independent (authority over local operations)
• Systems are *interdependent*
SoS Properties

- System as I/O Black boxes
  - Implementation details do not matter for other systems
- Propagation effects throughout the network
Challenge

• Aviation systems are heterogeneous, interdependent, local decision makers
• Local optimization does not consider (up- &) downstream effects
• Thus remains local optimization
• Global optimization not really feasible
• How to create an integrated model of distributed decision makers?
• Collaboration between systems?
Idea

• Start with a formal model of the aviation system
• End with dynamic simulation environment

• Agents as operational decision makers
• Solvers “solve” operational decision problems of agents/systems
• Solvers are based on same formal model, thus can be exchanged, extended
Designing SoS

1) SoS concept
2) Modeling framework
3) Software architecture to implement (1) & (2)
4) Domain specific model based on (2)
5) Domain specific implementation based on (3)

(3) – (5) Simulation
Designing SoS: Goals I

• Not the same as software/hardware design
• But is implemented as a computer based simulation
• SoS is an abstract concept which accommodates multiple different systems
• We view SoS as a meta-architecture
• Implementation details of comprising systems are not part of the SoS design
  • But the relations between those systems
Designing SoS: Goals II

• Integration of heterogeneous system within simulation environment

• Separate SoS design/model from programming specific problems

• Strict modularisation and interface design
  • Exchange of solvers / algorithms
  • Exchange/extent/add features

• Accommodate different operational planning methods
Modeling Framework

$S_s = (\text{Attributes}_s, \text{Goals}_s, \text{Restrictions}_s, \text{Solvers}_s)$

• **Systems as Agents**
  • Systems are the abstract concept; Agents are instances of a system (Airport ↔ Specific Airport)
  • Described by attributes
  • Have their own goals and restrictions
  • Have operational planning problems
  • → Agents as **operational decision makers**
  • Have behavior (fly, move, decision making, etc.)
Modeling Framework

• **System Properties**
  - Systems have attributes with specific data types
  - All of them together are the state of the agent

• Hierarchy of agents
  - Horizontal: Inherit properties from higher level generalizations
  - Vertical: Integrate capabilities
Modeling Framework

- **Agent Behavior**
- Outputs of a system
- **Mission:**
  - Ordered set of operations
  - Example: Flight
- **Operation:**
  - Either used to order actions
  - Or to model operational decision making

- **Actions:**
  - Agents’ behavior
  - Change state of agent
  - Execution of operations
Modeling Framework

- Inheritance and Capabilities
- Compare with Traits, Mixins, Inheritance in OOP

System model:
- Attributes
- Actions, operations, missions
Modeling Framework

• **Solvers for Decision Making**
  - Formalization of operational decision problems
    → Fixed operational procedures (rule based)
    → Non-/linear programming
    → Heuristics
    → Diverse set of methods to solve specific problem
  - Inputs, Objectives, Constraints, Output
  - Deliver a specific output agents can then work on

\[ \text{Sol}_\text{sol} = (\text{Input}_\text{sol}, \text{Objective}_\text{sol}, \text{Constraints}_\text{sol}, \text{Output}_\text{sol}) \]
Modeling Framework

- Solvers for Decision Making
Modeling Framework

Workflow

1) Model all systems, their attributes, behavior
2) Model solvers for decision making
3) Compile SoS model to SoS simulation model
4) Implement SoS model specific behavior and solvers (domain model)
5) Execute simulation

Required: Implement general simulation architecture
Simulation Architecture

- **Scenario Manager**
  - **Scenario**
  - **DB**

- **Simulation Manager**
  - **Solver**
  - **Visualisation**

- **SoS Description Model**
  - **SoS Simulation Model**

- **Advance Scenario**
  - **AircraftAgent.tick()**
    - **AircraftA**
    - **AircraftB**
  - **AirportAgent.tick()**
    - **AirportA**
    - **AirportB**
  - **ATCAgent.tick()**
    - **ATCA**
    - **ATCB**
    - **...**
Simulation Architecture

SoS Description Model

SoS Simulation Model

AircraftAgent.tick()

ATCA

ATCB

AircraftAgent.tick()

Mission: Flight

Operation X

Turnaround

AirportA

AirportB

AirportAgent.tick()

Scenario

Scenario Manager

Simulation Manager

Visualisation

Solver

DB

Simulation Architecture

Action A

Action B

Action C

Refuel

Deboard

Unload
Domain Model

• Aircraft
  • Has schedule of flights which it conducts (1-8)

• Airport
  • Arrival-Departure management (3,7)
  • Ground movement planning (2,8)
  • Turnaround (1)
Domain Model

- En-route Control
- Trajectory planning (5)
Domain Implementation

• Implement agents’ behavior
  • Flight performance model (take off, climb, decent, ...)
  • Vehicle driving abilities
• Implement solvers to generate solutions to previously modeled decision problems
• Let agents’ work these solutions
Properties: Modularity

• Exchange parts of the model with other implementations
• Adhere to defined interface of systems
  • Or expand as needed
• Example:
  • Use a different method to solve a specific operational problem
    • Solution quality vs. computation time
Properties: Interdependencies

• Decisions and actions of systems (agents) within a SoS affect other systems
• The presented modeling framework helps to identify where such IDs occur:
  • Input-Output relationships of solvers
  • All inputs and outputs are modeled explicitly
  • Out of the formal system model

\[ Sol_{s,sol} = (\text{Inp}_{s,sol}, \text{Obj}_{s,sol}, \text{Con}_{s,sol}, \text{Out}_{s,sol}) \]
Properties: Collaboration

• **Local Planning**
  Systems’ local optimum

• **Collaborative Planning**
  Optimum for collection of systems

• **Global Planning**
  Optimal solution for the overall SoS
Properties: Collaboration

• Again the SoS perspective
• In light of interdependencies and the previously discussed properties, ...
• ... aviation system wide optimization can only be achieved if operations planning is done between multiple agents
• Consider:
  • Planning of optimal arrival-departure-schedule
  • Ground Holding Problem
Properties: Collaboration

• This cannot be solved by a single airport alone
• Solution: Introduce a new solver, which considers multiple airports
• Integrate this new solver with the corresponding agents (airports, en-route control/ATC)

$$\text{Sol}_{\text{ERCAP,ADPTRAJ}} = \left( \text{Inp}_{\text{ERCAP,ADPTRAJ}}, \text{Obj}_{\text{ERCAP,ADPTRAJ}}, \text{Con}_{\text{ERCAP,ADPTRAJ}}, \text{Out}_{\text{ERCAP,ADPTRAJ}} \right)$$

$$\text{Inp}_{\text{ERCAP,ADPTRAJTraj}} = \text{Inp}_{\text{AP,ADP}} \cup \text{Inp}_{\text{ERC,trajectory}}$$

$$\text{Out}_{\text{ERCAP,ADPTRAJ}} = \{ \text{Out}_{\text{AP,ADP}}, \text{Out}_{\text{ERC,trajectory}} \}$$

- The presented SoS model and simulation architecture is also a mission & operations planning system
- Define missions a system can fulfill
  - and operations
- Generate set of operations to enable the mission to be conducted
- Generate an operational plan
  - When, what, how to execute the mission
Performance Characteristics

- Solving an operational decision problem might need a lot of computation time – if feasible at all
- A solver with better run time characteristics might be needed at the cost of solution quality
- Presented architecture lets you do just that:
  - Choose an appropriate solver for specific needs within a scenario
Final Remarks

- Very formal within the paper
- Good for compact writing within paper
- But not really usable inside computer program

\[ S = \{ AC, AP, ERC \} \]
\[ S_{AC/AP/ERC} = \left( A_{AC/AP/ERC}, Goal_{AC/AP/ERC}, \right) \]
\[ Restr_{AC/AP/ERC}, Sol_{AC/AP/ERC} \]

\[ A_{AC} = \left\{ A_{AC,Position}, A_{AC,Origin}, A_{AC,Dest}, \right. \]
\[ Out_{ERC,Traj}, A_{AC,PlanApproachTime}, \]
\[ A_{AC,PlanDepTime}, Out_{AP,ADP,SchedApproachTime}, \]
\[ Out_{AP,ADP,SchedDepTime} \}

\[ Sol_{ERC,Traj} = \left( Inp_{ERC,Traj}, Obj_{ERC,Traj}, \right) \]
\[ Con_{ERC,Traj}, Out_{ERC,Traj} \]
Final Remarks

- Actual implementation deviates from this
- But holds true to the presented modeling procedure

→ Directly modeled in Python
Formal Modeling of Air Traffic as System-of-Systems

Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)
German Aerospace Center

Institut für Flugführung

Alexander Gillissen
Department of Air Transportation
Telefon 0531 295-2917
alexander.gillissen@dlr.de