Using Ptolemy II as a Framework for Virtual Entity Integration & Orchestration in Digital Twins

MODDIT 2021 WORKSHOP AT MODELS 2021, OCTOBER 12\textsuperscript{TH}, ONLINE OCTOBER 2021

David A. Manrique Negrin
Loek Cleophas
Mark van den Brand
Agenda

- Problem definition
- Background
- Solution design – Ptolemy II implementation
- Results
- Conclusion & future work
Our vision on Digital Twin architecture

Using Ptolemy II for integration & orchestration – D.A. Manrique Negrin
Problem definition

• Different **tools** have **different semantics and syntax**.

• **Manual integration** becomes **unfeasible**, automation is needed.

• A **framework** to **automate** the **integration** is required. Commercial options exist, but they only support their own tools for modeling.
Problem definition

Our aim: Find a framework to integrate heterogenous models & extend it.

❖ 2 aspects for integration:
  • Communication: interface & encapsulation
  • Orchestration: sequence of execution (control flow) and how data is exchanged (data flow)

• Ptolemy II is a promising framework for integration.
Background – Ptolemy II

- **Objective**: support analysis & experimentation with CPSes
- **Actor oriented**
- **Directors** are orchestrators of actors
- **Separated control flow and data flow**

Why Ptolemy II:
1. **Open-source**
2. Accessible and sufficient **documentation & support**
3. Artefacts are **modular & re-usable**
Background – AES-Lab [1]

• **Scaled-down truck** for driving and docking in a DT.
• Used for **testing of autonomous driving**: path generation, avoidance of obstacles & driving control.

AES-Lab Digital Twin

Models:
1. **Unity Game Engine** – visualization model & truck dynamics
2. **Simulink** – path selection & control
3. **IBM Rhapsody** – requirements, structure & logical behavior
Final behavior results

Use case for validation:

1. Truck at initial position.
2. Signal a docking station as destination for truck.
3. Truck generates a path.
4. Truck follows path, avoiding obstacles & with similar maneuvers (speed & steering) as in [1].

Original implementation vs our objective

Implementation in [1]:
• Starting the execution requires manual setup & startup.
• Complex Integration between models.

Our implementation aims:
• Decrease complexity of integration, set-up & startup.
• Reproduce original behavior.

First steps for implementing of DT in Ptolemy II

Simulink models were split:

- To test director execution & model integration.
- In a real application the separation would aid maintainability.

Integration of models:

- Choose final destination
- Start truck movement

Path planner

Control

GUI

Using Ptolemy II for integration & orchestration – D.A. Manrique Negrin
Ptolemy II Implementation - Actor

REQUIREMENTS:

- R1: Execution of all the models
- R2: Simulink simulation state visible
- R5: Reduction of complexity in integration & setup
- R3: Model’s communication on each time-step
- R4: Enable configuration of Simulink solver (time-step)

Our Solution:

- Selected Simulink actor for Simulink models & Exec actor for Unity models.
- New Simulink actor:
  - Reduction of complexity to call Simulink models.
  - Enable visibility on simulation states.
- S-function generation for Simulink models & communication over TCP-IP using Python actor.
- Simulink actor enables configuration of Simulink time-step.
All models required **concurrent exchange & execution** of data

Our implementation → **PN director** for orchestration
Communication → **TCP/IP protocol** with python actor
Results - Analysis

Results: successful execution

- Decreased complexity of integration:
  - Simulink actors execute models with **only 2** parameters
  - Unity models executed with **1 parameter**

- Reproduced original behavior:
  - Using **TCP/IP communication**
  - Orchestration of all models by **PN director**

- Setup and startup **complexity reduced**:
  - **One click** to start execution

---

Conclusions

• **Ptolemy II facilitates models’ integration.**
• **Actors reduce complexity** of such integration & aid **scalability.**
• **Director** can **orchestrate** control flow to reproduce DT behavior.
• **Simplified startup.**

Future work

• **Implement data flow** control for orchestrator - Implement the data exchange within the **Simulink actor.**
• **Simplify further** the integration for Simulink models - **Automate** the generation of **S-function** for Simulink models.
• **Experiment on integration technologies** for other type of models, e.g., FMI.
Thank you.