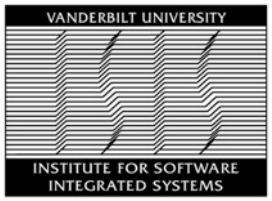


Workflow Automation for Cyber Physical System Development Processes

Charles Hartsell, Nagabhushan Mahadevan, Harmon Nine,
Abhishek Dubey, Ted Bapty, Gabor Karsai

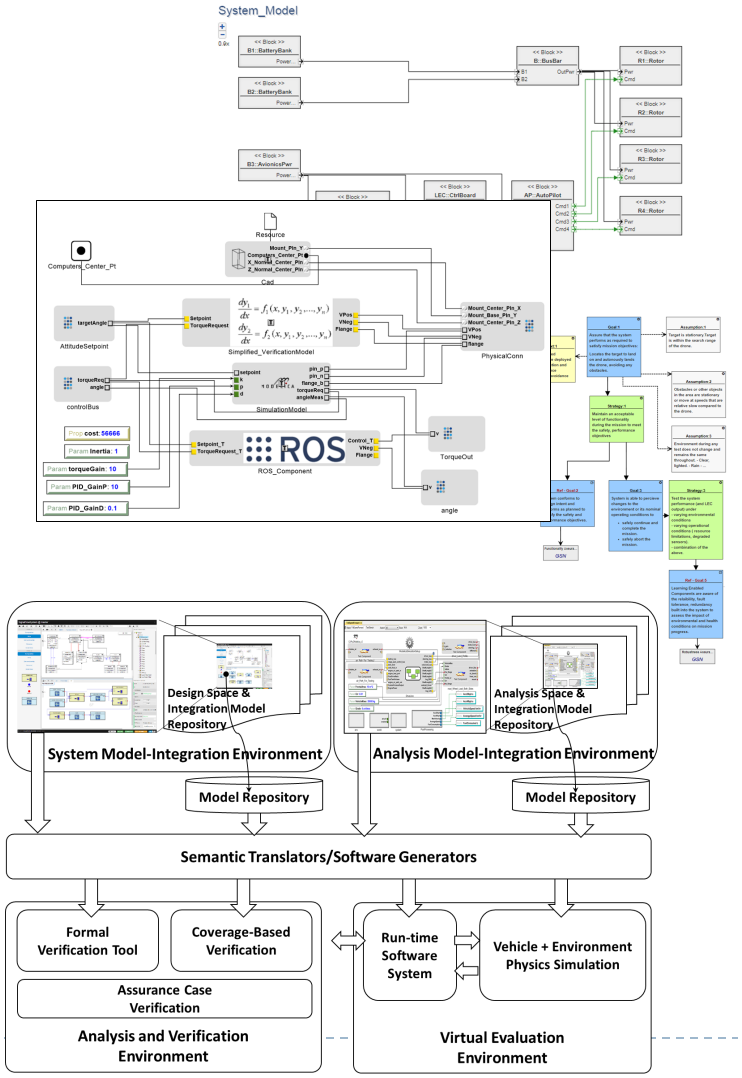
Supported by DARPA under FA8750-18-C-0089

Roadmap

- Typical CPS design workflow
- Heterogeneous models & tools for CPS
- ALC Toolchain workflow
- Workflow modeling language
- Examples
- Conclusions

Design Workflow for CPS

- Modeling
  - Requirements/specifications
  - Components
  - System architecture
  - Assurance
- Design
  - Specs → Components + Architecture
- Synthesis
  - Design → Implementation
- Verification
  - Design → ? → Specs
  - Implementation → ? → Design
- Assurance
  - Design + Implementation: safe?



Heterogeneous Languages & Tools

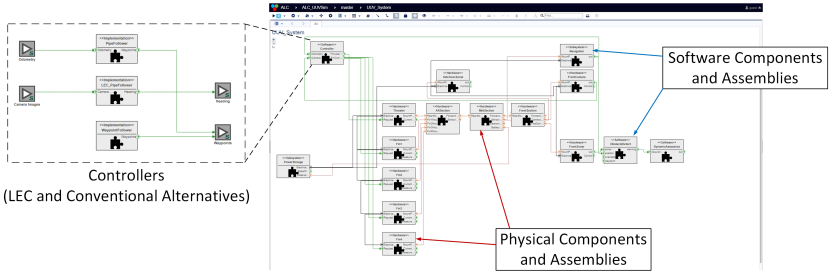
- Many different modeling languages, both general-purpose and domain-specific, with many supporting tools
- Heterogeneous interfaces between tools – automation is lacking
- Common architecture modeling languages and paradigms:
  - Architecture Analysis & Design Language (AADL) [1]
  - SysML [2]
  - Colored Petri Nets (CPN) [3]
  - Goal Structuring Notation (GSN) [4]
- Example supporting tools:
  - Functional Modeling Compiler (FMC) [5]
  - CPN Tools [6]
  - RESOLUTE [7]

Existing Workflow Description Languages

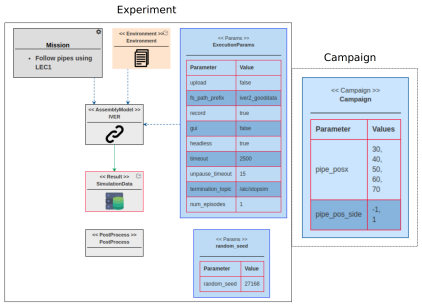
- Business Process Model and Notation (BPMN) [8]
  - Generic, natural language models commonly used for documenting business processes
  - Does not address interfacing problems – difficult to automate
- Project Worker [9]
  - Packages common, repeatable operations into *Engineering Automation Objects* (EAOs)
  - Highly generic, but conceptual ideas are not implemented and validated
- Formalism Transformation Graph (FTG) [10]
  - Describes how heterogeneous modeling languages are related to one another with *Model Transformations*

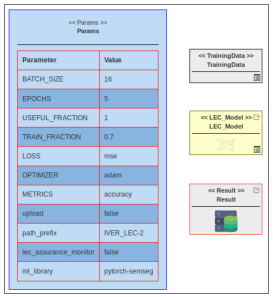
Assurance-based Learning-enabled CPS (ALC) Toolchain

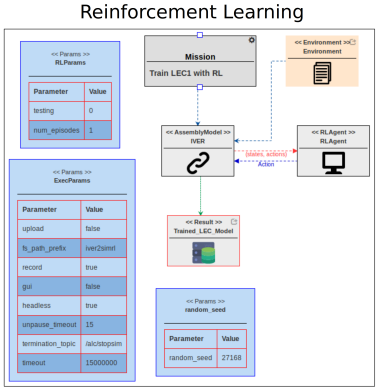
- Modeling
  - System Architecture / SysML



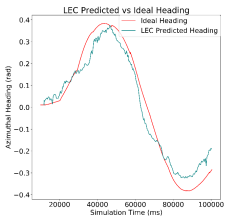
- Construction
  - Data collection
  - Training
  - Evaluation

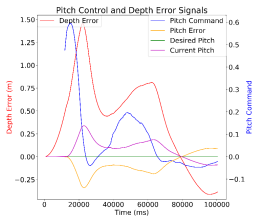


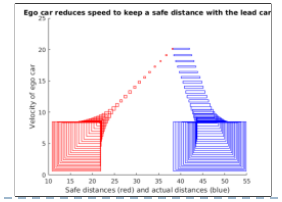


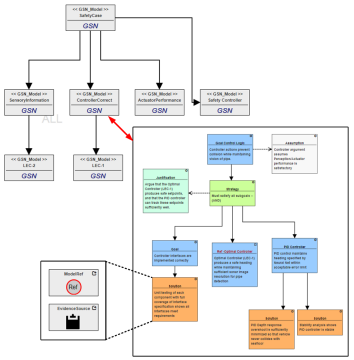


- Verification/Validation/Assurance - Evidence



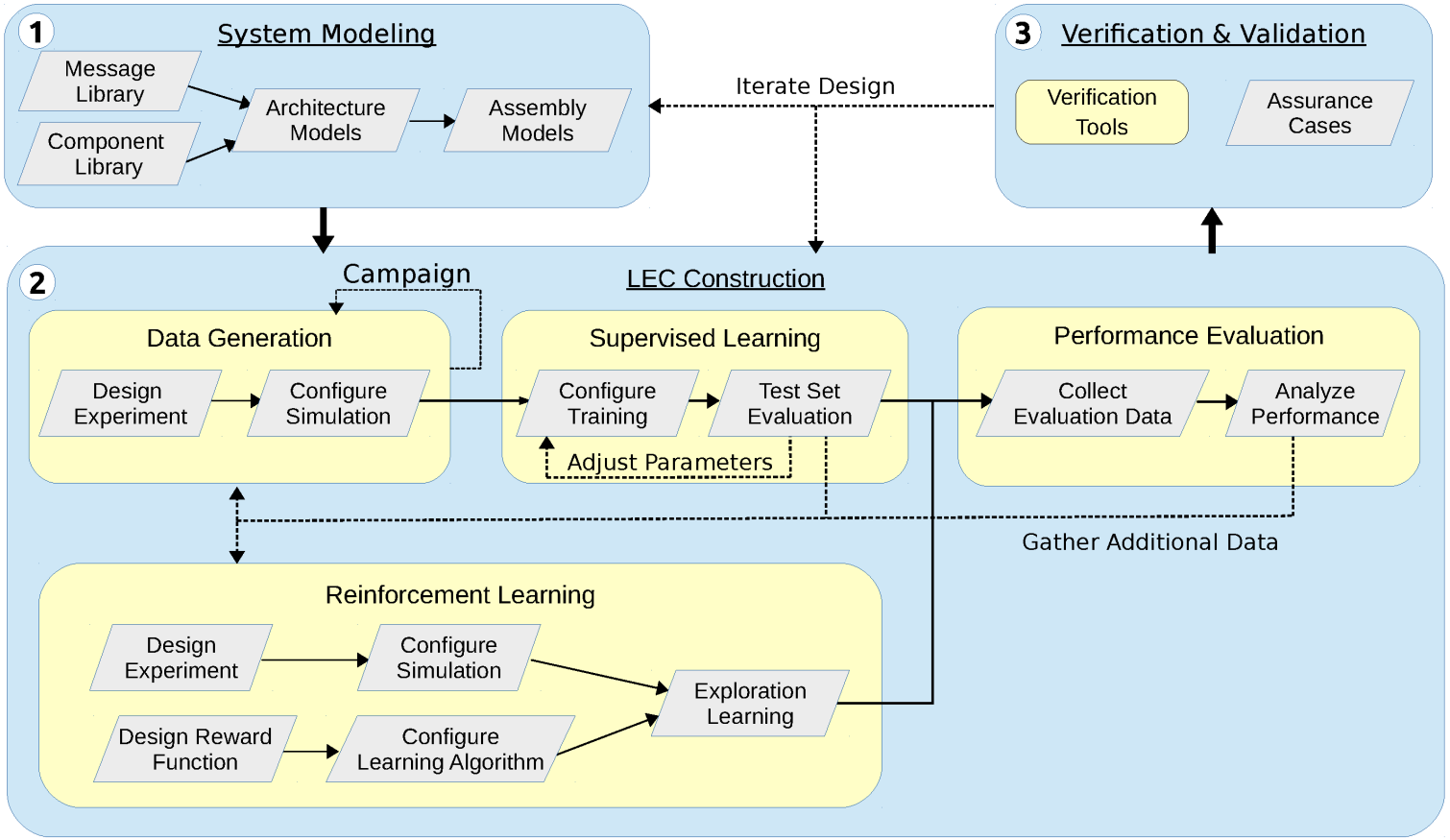






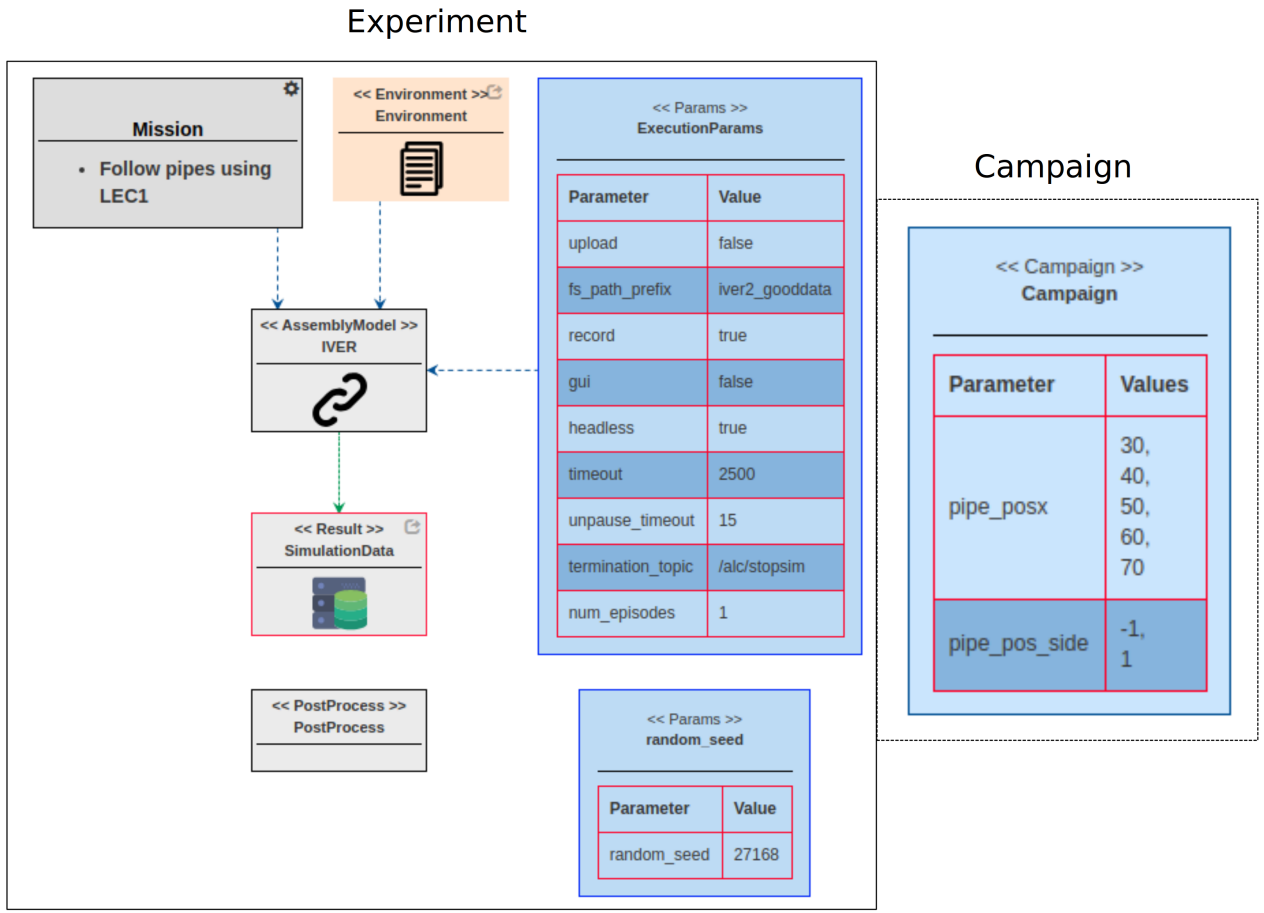
ALC Design Workflow

- Specialized for LEC development
- Consists of *Tasks* and *Activities*



ALC Model of Data Generation

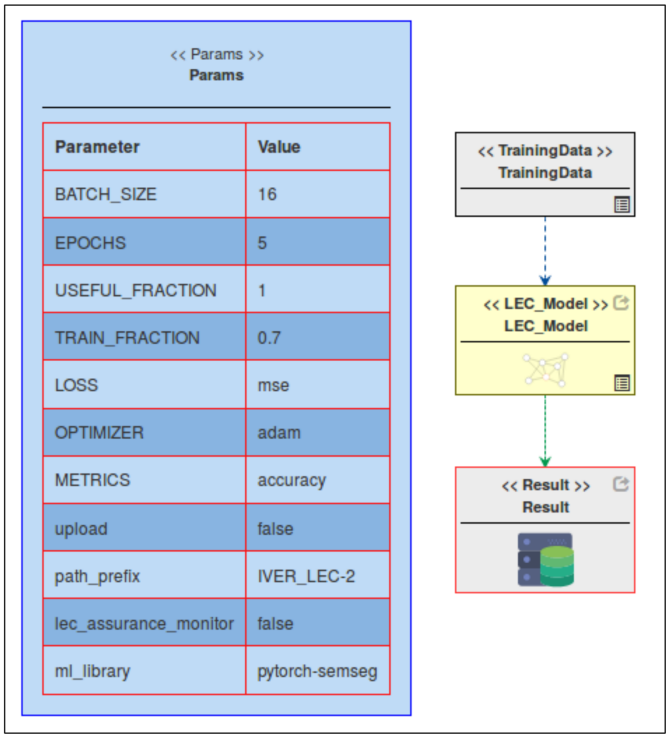
- *Tasks*: design experiment & configure simulation
- *Activity:* execute experiment to collect data



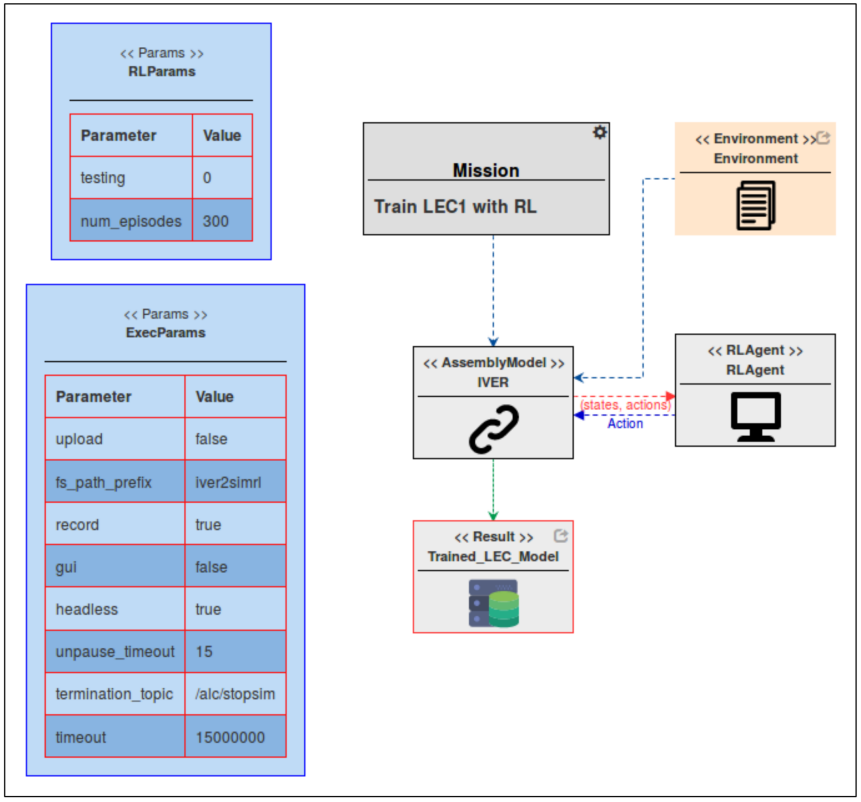
ALC Model of LEC Training

- *Tasks*: configure training & evaluate against test set
- *Activity:* perform training with selected ML framework

Supervised Learning

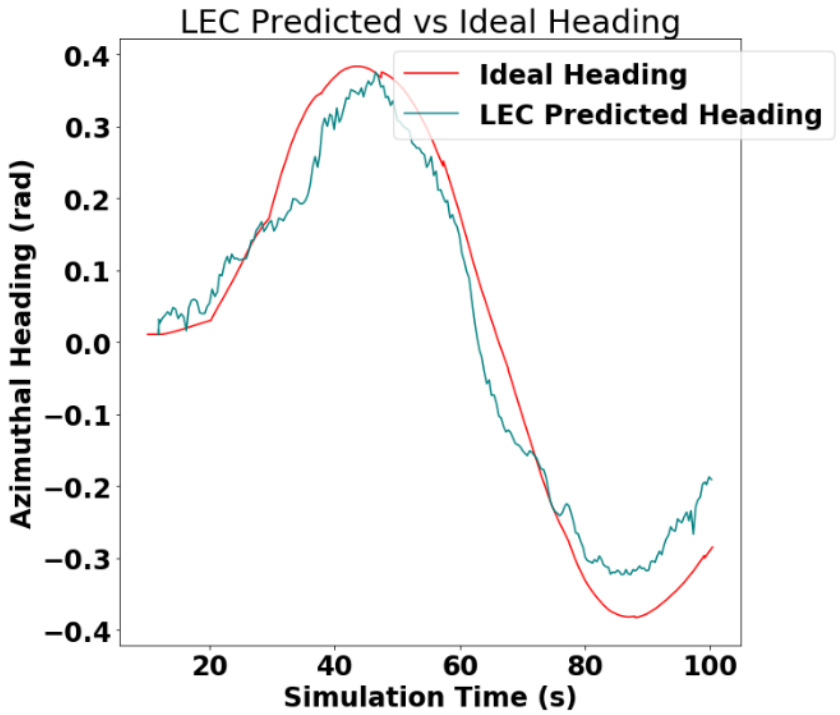


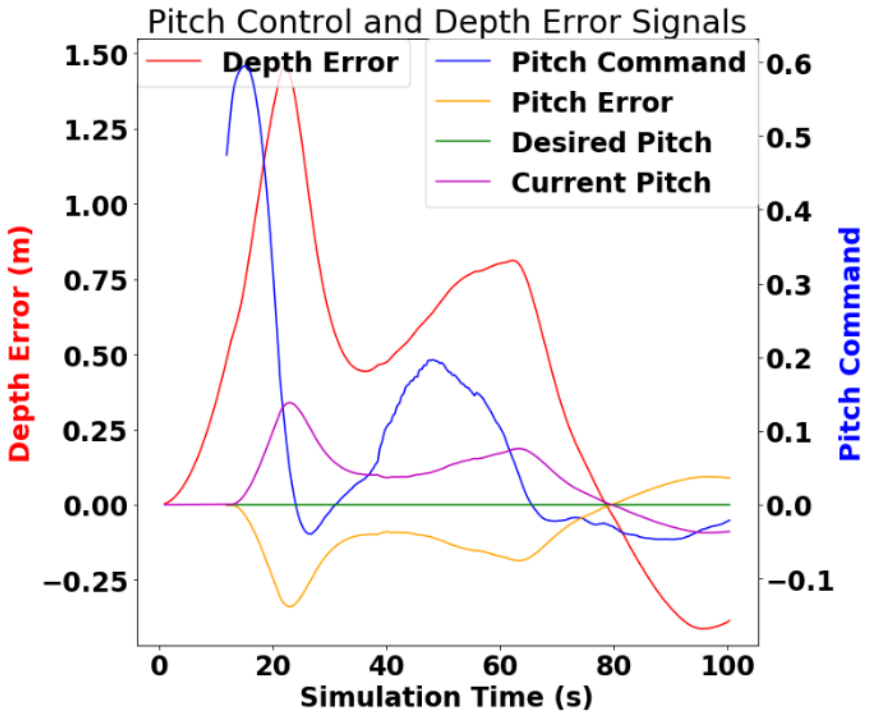
Reinforcement Learning



ALC Model of Performance Evaluation

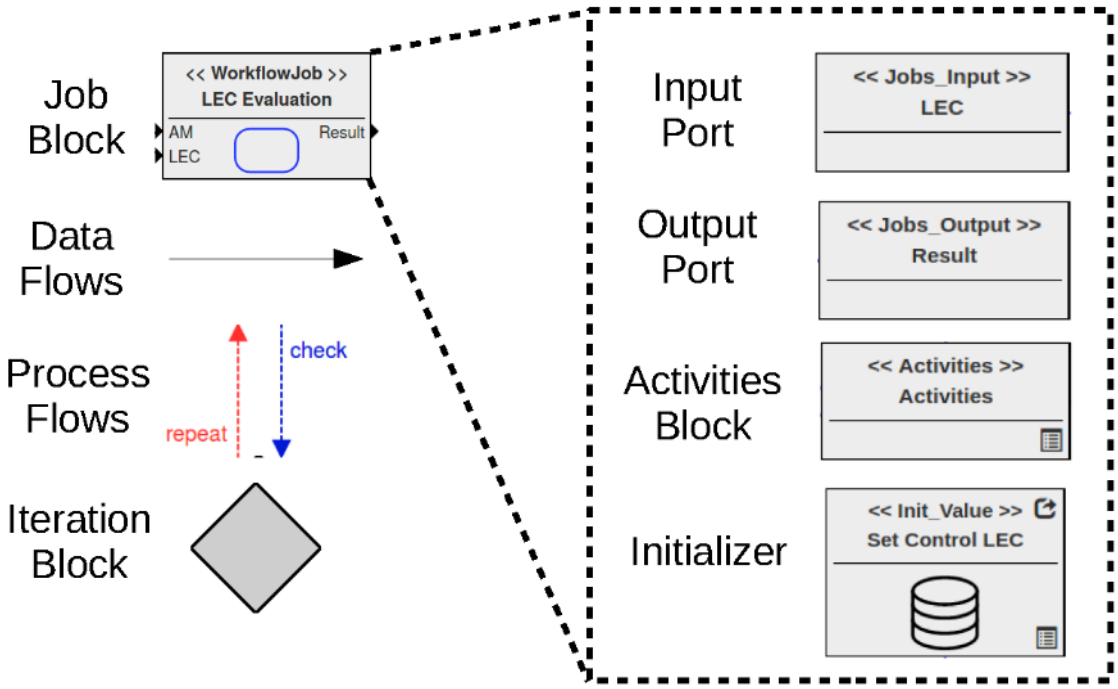
- *Tasks*: design test scenarios & analyze results
- *Activity:* execute experiment & analysis
- Same experiment model as data collection





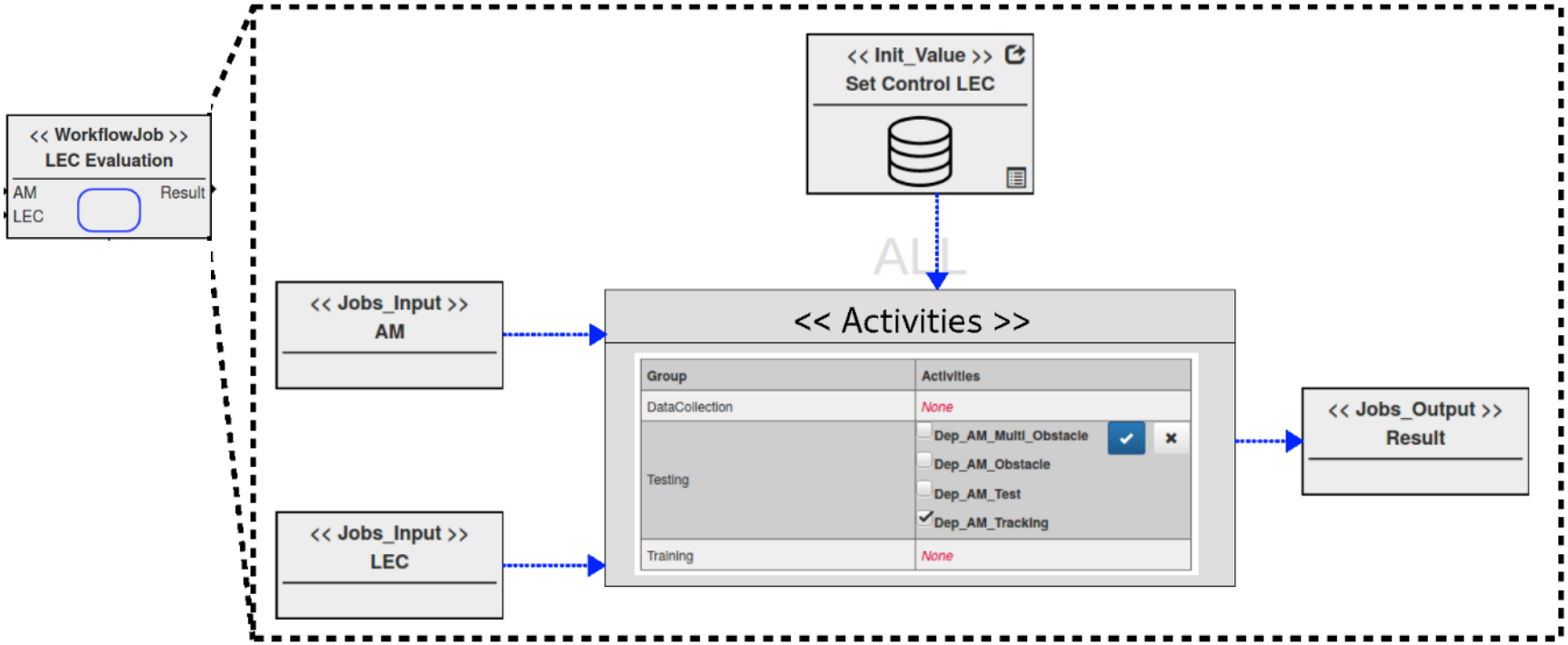
ALC Workflow Modeling Language

- *Jobs* have:
  - *Inputs* and *Outputs*
  - *Activities*
  - *Initializers*
- *Data flow* from *Outputs* to *Inputs*
- *Process flow* to control *iteration*
- *Workflow Iteration* logic defined in *Iteration Blocks*



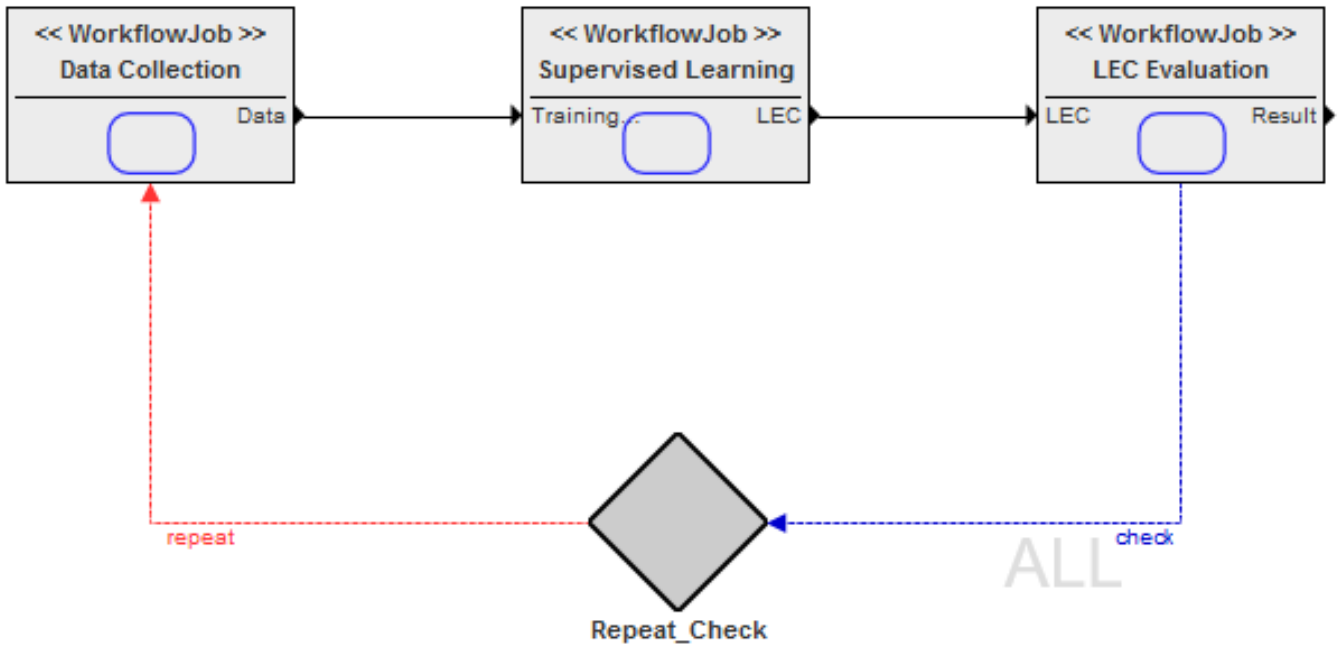
ALC Workflow Job Model

- Example *LEC Evaluation* job
- Two inputs, one output, and an initializer
- One evaluation activity model selected



ALC Composed Workflow Diagram

- Composed workflow for LEC Construction
- *Iteration* block update models in the *Data Collection* job based on the results from the *LEC Evaluation* job



Workflow Executor

- Two parts:
  - Model interpreter – Parses workflow diagram to build a JSON object description of the workflow
  - Job executor – Uses Gradle[11] to build a task dependency Directed Acyclic Graph (DAG) and execute each task
- Provides API for that allows *workflow models* to interact with *activity models*
- API used when defining *Iteration* and *Initialization* scripts

Monitoring Workflow Progress

- Status monitor indicates *Pending*, *Finished*, or *Error* for each activity in the workflow
- If an *Error* occurs, workflow executor automatically skips all activities which finished successfully before the error

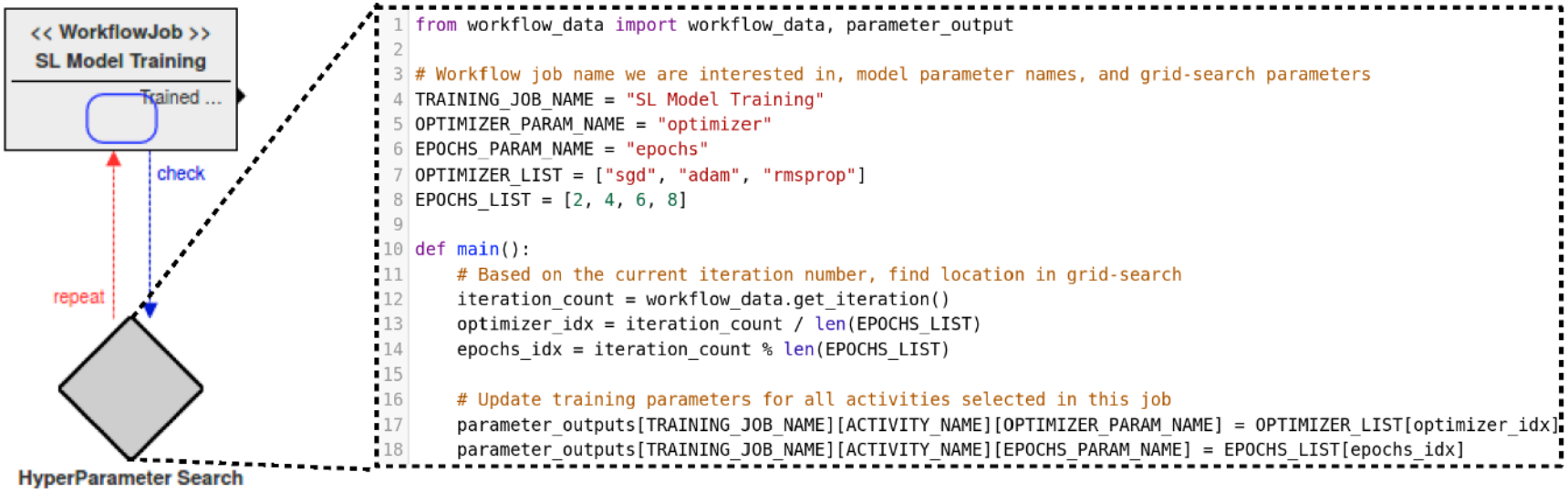
Execution	Job	Activity	Status
RL_WF			Pending
	RL Exploration		Finished
		RLTrainingLEC1	Finished
	LEC Evaluation		Pending
		Dep_AM_Tracking	Pending

Example 1:
Reinforcement Learning & Evaluation

- Two step workflow:
  1. Output an LEC trained with Reinforcement Learning activity model
  2. Input trained LEC to Evaluation activity model and output evaluation results
- No iteration, and final output is unused

Example 2: Hyperparameter Search

- Iteratively train an LEC with a Supervised Learning activity model
- Use iteration logic to find optimal hyperparameters via simple grid-search



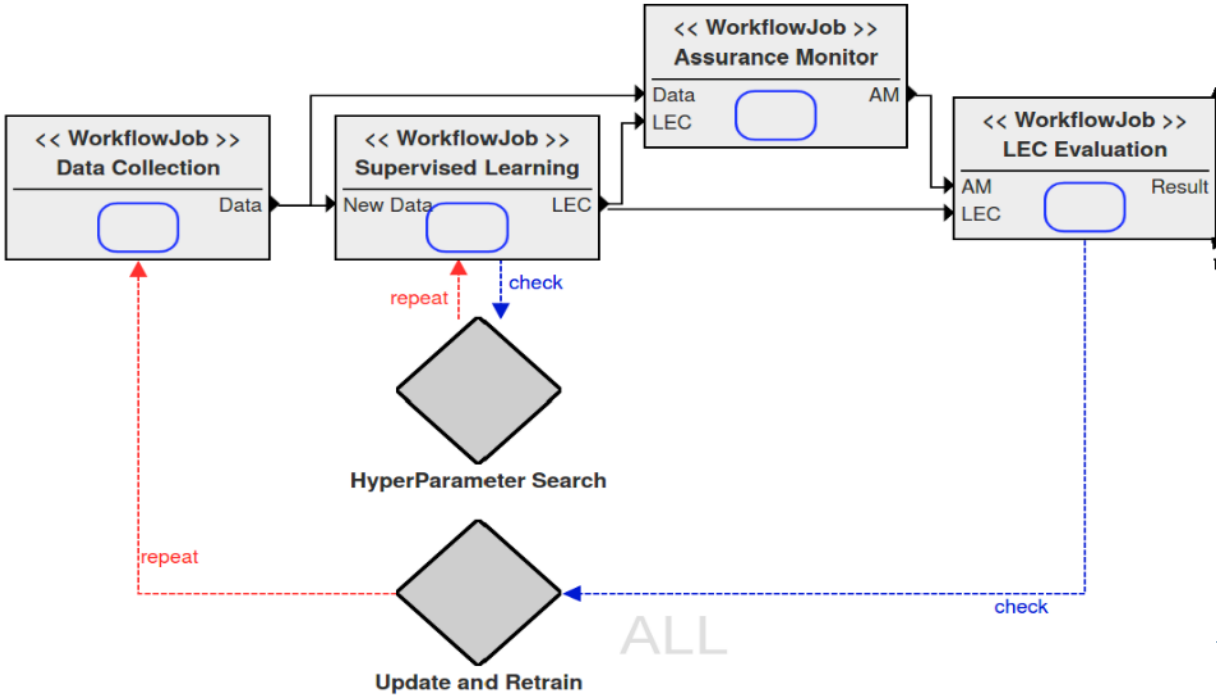

```python
from workflow_data import workflow_data, parameter_output

# Workflow job name we are interested in, model parameter names, and grid-search parameters
TRAINING_JOB_NAME = "SL Model Training"
OPTIMIZER_PARAM_NAME = "optimizer"
EPOCHS_PARAM_NAME = "epochs"
OPTIMIZER_LIST = ["sgd", "adam", "rmsprop"]
EPOCHS_LIST = [2, 4, 6, 8]

def main():
    # Based on the current iteration number, find location in grid-search
    iteration_count = workflow_data.get_iteration()
    optimizer_idx = iteration_count / len(EPOCHS_LIST)
    epochs_idx = iteration_count % len(EPOCHS_LIST)

    # Update training parameters for all activities selected in this job
    parameter_outputs[TRAINING_JOB_NAME][ACTIVITY_NAME][OPTIMIZER_PARAM_NAME] = OPTIMIZER_LIST[optimizer_idx]
    parameter_outputs[TRAINING_JOB_NAME][ACTIVITY_NAME][EPOCHS_PARAM_NAME] = EPOCHS_LIST[epochs_idx]
```

Example 2: Results

Learning Optimizer	Number of Epochs			
	2	4	6	8
SGD	1.057	0.996	0.951	0.904
Adam	0.057	0.032	0.022	0.011
RMSProp	0.147	2.485	0.010	0.010

Example 3: Complex Development Workflow

- Combines iterative LEC Construction process with Hyperparameter optimization
- LEC with minimal loss is output from hyperparameter search to Assurance Monitor training and Evaluation



Recent & Future Improvements

- Recent improvements
  - Parallel execution is supported
  - Reduced scripting requirements – additional model elements
  - Hierarchical workflows
  - Gradle replaced with optimized Python-based implementation
- Future improvements
  - Full-support for conditional branches in the workflow
  - Post-processing & user notification of workflow results
  - Library of workflow templates (eg. Hyperparameter search)
  - Improved debugging
  - Integrate additional *activities* such as verification & assurance

Summary

- CPS involves complex, iterative design workflows
- Many domain-specific models and tools with heterogeneous interfaces
- Existing automation between tools is minimal
- Extensible workflow language & executor allows for automation between tools
- Tools publicly available on CPS-VO at:
  - https://cps-vo.org/group/ALC

References

[1] P. H. Feiler, B. Lewis, S. Vestal, and E. Colbert, An Overview of the SAE Architecture Analysis & Design Language (AADL) Standard: A Basis for Model-Based Architecture-Driven Embedded Systems Engineering. Boston, MA: Springer US, 2005, pp. 3–15.

[2] OMG. OMG Systems Modeling Language (OMG SysML), Version 1.5, 2017

[3] K. Jensen and L. M. Kristensen, Coloured Petri Nets - Modelling and Validation of Concurrent Systems. Springer, 2009.

[4] Tim Kelly and Rob Weaver. The goal structuring notation–a safety argument notation. In Proceedings of the dependable systems and networks 2004 workshop on assurance cases, page 6. Citeseer, 2004.

[5] A. Canedo and J. H. Richter, "Architectural design space exploration of cyber-physical systems using the functional modeling compiler," Procedia CIRP, vol. 21, pp. 46–51, 2014.

[6] A. V. Ratzer, L. Wells, H. M. Lassen, M. Laursen, J. F. Qvortrup, M. S. Stissing, M. Westergaard, S. Christensen, and K. Jensen, CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 450–462.

[7] Gacek, A., Backes, J., Cofer, D., Slind, K., & Whalen, M. (2014). Resolute: an assurance case language for architecture models. *ACM SIG Ada Letters*, *34*(3), 19-28.

[8] OMG, "Business Process Model and Notation (BPMN), Version 2.0," https://www.omg.org/spec/BPMN/2.0/PDF, Object Management Group, 2011.

[9] R. Maier, S. Unverdorben, and M. Gepp, "Efficient implementation of task automation to support multidisciplinary engineering of cps," in 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE). IEEE, 2018, pp. 1388–1393.

[10] S. Mustafiz, J. Denil, L. L´ucio, and H. Vangheluwe, "The ftg+pm framework for multi-paradigm modelling: An automotive case study," in Proceedings of the 6th International Workshop on Multi-Paradigm Modeling, 2012, pp. 13–18.

[11] https://gradle.org/