Design Oracle for AI-Based CPS Design

1st Theodore Bapty*, 2st Sydney Whittington†, 3rd James Walker†,
4th Joseph Hite*, 5th Brian Swenson†, 6th Kathrine Owens*,
7th Fred Eisele*, 8th Jason Scott*, 9th Robert Owens*
*Institute for Software Integrated Systems Vanderbilt University, Nashville, TN, USA. theodore.a.bapty@vanderbilt.edu
†Southwest Research Institute, 6220 Culebra Rd, San Antonio, TX, USA. sydney.whittington@swri.org

***Abstract*—Automated design processes, especially using Machine Learning/AI techniques, require proposed systems to be evaluated across all relevant attributes, requirements, and concerns. Traditionally, teams create models in a set of engineering tools for design evaluation data. We describe a Design Oracle, where automated designers can submit a system as a graph of components and request a full range of evaluations, composing analysis workflows containing multiple engineering tools across multiple physical domains, fidelity, and scenarios. Manually created system models are not required. The system works across cloud architectures, supporting the use of parallel computation.**

The system leverages technology developed under the DARPA Adaptive Vehicle Make/META program, which created a uniform, multi-domain design and component representation, and tools for composition to engineering tool models across multiple tool types, domains, and fidelity levels [1]. The Design Oracle has been used within the DARPA Symbiotic Design for Cyber-Physical Systems, across two classes of target designs, Unmanned Air Vehicles and Unmanned Underwater Vehicles.

***Index Terms*—Cyber-Physical Systems, Design Automation, Model-Based Computing, Simulation, Analysis**

I. Introduction

There is significant interest in using machine learning (ML) techniques to assist humans in the design of cyber-physical systems (CPS). The DARPA Symbiotic Design for Cyber-Physical Systems addresses these goals, proposing a techniques for creating systems from arbitrary architectures of parameterizable components. Several efforts are developing ML techniques to synthesize and optimize these designs, both independently and with human input.

The metrics for quality of design depend upon evaluation across a range of functional and non-functional requirements. To keep the evaluation techniques from merging with the machine learning approach, ML systems need a "Design Oracle", where arbitrary designs can be submitted to the Oracle, and performance metrics, data, and evaluation scores are returned to the designer. These results pertain to the physical design, software controls, operational environment, and scenarios.

The functions of the Oracle are as follows:

- Design Specification Interface: accept design descriptions from design agents with a well defined schema. Descriptions reference components within the design corpus.
- Automation interface: accept requests for evaluation of designs across any number of analysis workflows.

This work supported by DARPA Symbiotic Design of Cyber-Physical Systems under contract FA8750-20-C-0541.

- Composition: Compose the requested design from the design graph and corpus components into the target analysis tool models used in an analysis workflow.
- Workflow Synthesis: Create the executable workflow, with inter-tool communication and sequencing.
- Execution: Execute the workflow on an available computational resource.
- Data Collection: Collect and package the data for consumption by the agent.

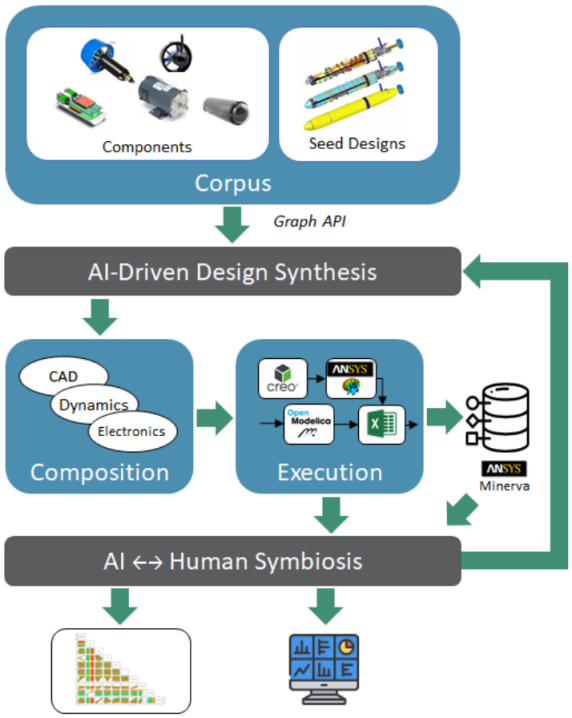

Fig. 1. High-Level Architecture.

The challenges in supporting the Oracle for CPS designs include:

- The physical designs can be of arbitrary topology (within component-component compatibility). the direct use of hardwired simulators, where models are hand-crafted for specific system instances, are not feasible.
- Given the variety of types of CPS systems, and the physical domains of interest, the Oracle must be flexible to support a number of engineering tools

- Typically, multiple engineering tools and arbitrary workflows between these tools are required to compute metrics. These tools use different input model semantics, formats, and content. The Oracle must support mapping of the CPS design representation to each of these tools.
- Data postprocessing and extraction are needed to compute metrics and deliver information for training/feedback to the ML designers.
- A common, system-independent interface and evaluation capabilities will help to apply ML tools across a larger range of systems, e.g. Ground vehicles, UAVs, UUV's, etc.

For the purposes of the DARPA Symbiotic Design for Cyber-Physical Systems program, we have developed a Design Oracle infrastructure that can provide computation of the necessary metrics and raw data, be scaled across multiple compute servers, and provides high level metrics and low level data computed on the design. The system is based on Open Source software, and supports a variety of both open source and commercial engineering tools.

II. Related Work

Assured autonomy: Model-based design for CPS with learning-enabled components, [4] presents a tool workflow for training Learning Enabled Components within a set of scenarios. The system supports the architectural modeling of the system, including the software (ML and non-ML), and has the ability to define workflows for training, evaluation, and assurance. Integration with simulators allow the scenarios to be executed, generating data for training of LECs and assurance components. Significant capabilities include the automatic execution of complex simulation workflows and management of the input and output data. This offers a significant advantage in automation and traceability of training data with trained ML components. For applications such as CPS design, the system lacks the ability to compose different physical target system architectures and to incorporate complex physical simulation and analysis tools and workflows.

SimGAN, a Hybrid Simulator Identification for Domain Adaptation via Adversarial Reinforcement Learning [5], uses adversarial learning techniques to construct behaviors of a simulator to match training data. A suboptimal physics simulator is used as ground truth for robot behavior across several domains, including motion, part deformation, contact dynamics, and system power. The adversarial training system works to construct a hybrid system model surrogate for the target. This approach achieves a high degree of coupling and efficiency of operation, at the cost of generality.

III. Design Oracle for CPS

A. Component Representation

In summary, **Components** in the corpus consist of Connectors, Ports, Parameters, Properties, and executable artifacts. The AVM Component Model was created and documented under the DARPA META program as the ACM (AVM Component Model) Specification [6].

The definition of these are as follows:

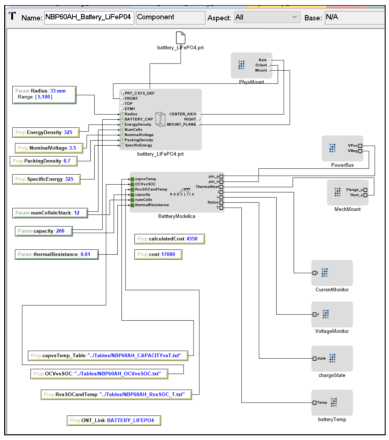

Fig. 2. Example Component Model (Battery).

- Connectors are composite connection end points which contain one or more domain-specific Ports. A connection between Connectors implies connection of all contained ports. Ports are domain-specific, representing interactions in geometry (alignment of Axis, Plane, or Coordinate Systems), power transfer (Modelica thermal, rotational, translational, electrical), electrical network (pins connecting SPICE and PCB/Schematics), and others.
- Property defines a non-changeable attribute of a component, which can define length, offset, mass, etc. of a component.
- Parameter defines an externally settable attribute of a component, otherwise equivalent to a property.
- Artifacts capture the individual domain models representing the component behavior or structure, with artifact ports mapped to component-level ports. Artifacts include CAD models, Modelica Classes, Spice models, PCB parts, etc.

The example in figure 2 captures a battery component. Properties specify the geometry (Radius of the cylinder), technology properties (EnergyDensity, SpecificEnergy) and instance parameters (Capacity, numCellsInStack), and non functional parameters (Cost). The CAD artifact model specifies a CAD part file name/location, datum planes used for mounting and mapping to a connector. This artifact is used for composition of system geometry in PTC Creo.

A behavioral model artifact (Modelica) captures power interfaces (electrical, thermal, mechanical) and a set of monitoring ports for voltage, temperature, current, and charge state. This model can be used for a time-based assessment of component interactions and their contributions to system performance within a range of scenarios.

B. Design Representation

Designs are entered into the Oracle as a graph, consisting of V() ComponentInstances, SystemParameters and E() consisting of connections between componentInstance connectors and SystemParameters to ComponentInstance Parameters. The

specification is similar to SysML's Internal Block Diagram (IBD). The Design representation was created and documented under the DARPA META program as the ADM (AVM Design Metamodel) Specification.

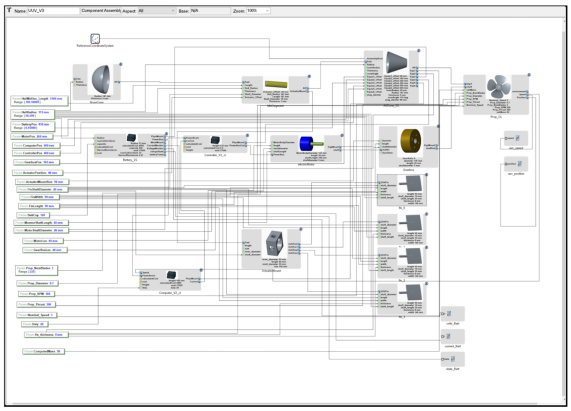

Fig. 3. Example System Design (UUV).

The figure 3 shows an example design for a UUV, consisting of nosecone, midsegment, tailcone, and prop. a simple internal powertrain contains a battery, motor controller, motor, and gearbox. Controls for the system use a set of aft fins, connected to an actuator, and a computer. A set of system parameters map to one or more component parameters to control system attributes such as radius, length, and fin sizes. Other parameters are used to specify desired performance characteristics that influence the design synthesis of the propeller.

C. System Composition For Execution

Converting from a Design and Component graph to individual engineering, or Composition of Tool Models, is encoded as an OpenMETA TestBench [1]. The Design Oracle leverages OpenMETA for composition of CAD, Modelica, Simulink, Gazebo, SPICE, and has extended facilities for SwRI's FDM, ANSYS Fluent CFD, and continues to add others as required by the DO users.

As an example, CAD composition involves traversing the Design Graph from a designated root component, inserting the part in the assembly constrained to the global coordinate system, then traversing connectors to each connected part. Each connected/child part is inserted and constrained to its parent via the ports/CAD datum objects. Following the recursive, depth-first traversal, parameters are applied to yield the final assembly. Operations in the CAD tool are then used to find mass, inertial matrix, and location of each of the components, and a result structure is produced for use a system metrics or to drive adjacent TestBenches.

D. Computational Workflows

Arbitrary computational workflows models are defined in the OpenMETA tool for the OpenMDAO framework [2]. The workflow models contain a Workflow Driver, a set of Testbenches, and a set of connections defining input/output relationships within the workflow. The Driver specifies a set of parameters that can be sampled over defined ranges with random distributions that connect to testbench/system input. Testbench metrics can drive other testbenches and/or be routed back to the driver for data collection. Raw data can also be routed to data archives for the design analysis.

E. Architecture

The Design Oracle supports a distributed system of design agents, interacting through standard interfaces, network-based interfaces. The infrastructure uses all open-source software, and is packaged as a set of Docker images for easy installation.

Design specifications are submitted to the graph database (JanusGraph [7]), using the Gremlin Query Language, using a schema based on ADM and ACM, with extensions for representation of Corpus ontologies. Queries can construct and edit design entities (ComponentInstances, SystemProperties, Connections), and extract design and component information. Additionally, python-based utilities are provided to construct designs from simple operations in a CSV file. The JanusGraph database is backed by Cassandra and ElasticSearch for persistence and text-based searching.

Analyses are requested to and orchestrated by a Jenkins [8] as Pipelines. Pipelines are used to sequence a set of operations on a design. Typical analysis pipelines execute a series consisting of:

- Prepare and initialize the execution environment, clearing out any prior data from previous executions on the worker node.
- Fetch the design from the JanusGraph database, executing a Gremin query on the requested design, resulting in a GraphML or set of JSON files.
- Convert the design GraphML into an OpenMETA design representation.
- Import the design and insert the model into the requested workflow and contained TestBenches.
- Use OpenMETA to compose the executable models and construct the OpenMDAO workflow files and directory structures.
- Execute the workflow under OpenMDAO, across a set of requested parameters.
- Package the results and upload the data to both the Jenkins server and the MinIO (S3) server.

An arbitrary number of Jenkins Agents can be provisioned to allow parallel execution of workflows. Jenkins capabilities are used to perform the load balancing and dispatch to waiting execution nodes.

F. Use Cases and Results

1) UUV Use Case: The DO has been used to support the AI designer and Human Symbiosis tools in SymCPS in two system design challenges. The initial deployment included a corpus for design of Unmanned Underwater Vehicles (UUV), and was used for Design Oracle architectural development and user evaluation. The engineering analysis tools used were: PTC Creo for CAD Assembly, Modelica for system power

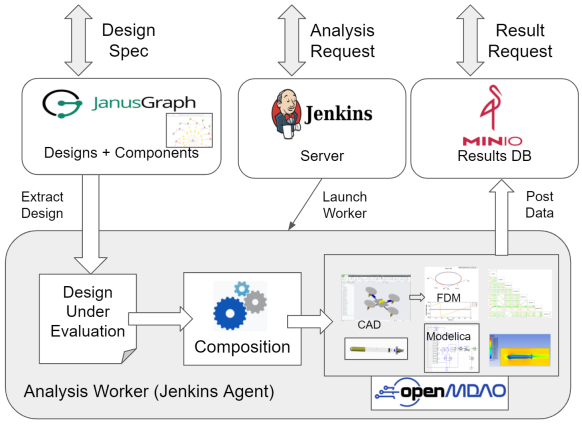

Fig. 4. Design Oracle Architecture.

analysis and 1D motion, ANSYS Fluent for drag calculation, OpenProp for propeller synthesis and performance, and Gazebo for 6-DOF dynamic simulation.

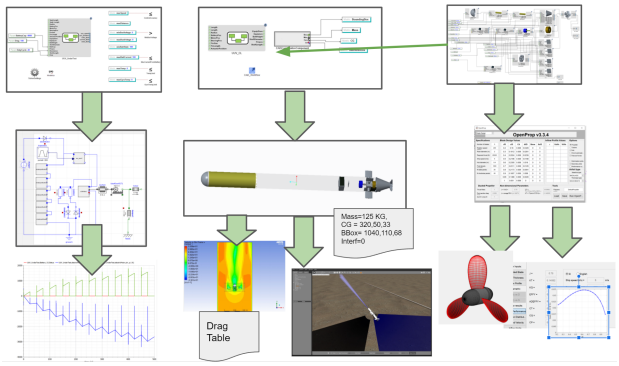

Fig. 5. Computational Workflow for UUV Analysis.

CAD was used for system sizing, mass calculation, and ability to fit components without interference. A simple drag surrogate, using CFD simulations for calculation of drag on an nominal vehicle, was scaled by frontal area for a specific test case, and computed a drag vs. speed table. OpenProp was used to synthesize a propeller design for the specific drag calculated by the drag surrogate, at the vehicle's target speed. OpenProp also provided tables of thrust vs. RPM/Speed and torque demand vs. RPM/Speed. The drag, thrust, and torque tables were used in a dynamic simulation executed in Modelica. The simulation used a PID controller to match a speed trajectory, simulating system speed (1D) considering mass, drag, and thrust, dynamic propeller thrust and torque, feeding back to the motor, speed controller, and battery across electrical and rotational power domains, computing all voltage, current, torque, velocity, and thermal properties of each component. Simple metrics were calculated based on end battery state, maximum thermal state, and ability to meet speed setpoints across time.

2) UAV Use Case and Design Hackathon: The primary use case for the system has been with the UAV design challenge for the SymCPS program. The phase 1 UAV corpus contains 744 unique components, with structural members (tubes, hubs, flanges, mounting plates) and a large variety of batteries, motors, and propellers. Component connections are primarily physical (e.g. Hub-to-Tube, Tube-to-Flange, Flange-to-Motor) and power connections (Battery-to-ElectronicSpeedController(ESC), ESC-to-Motor) and a combined connection (Motor-to-Propeller).

The seed designs are shown in figure 7. These include variants of a QuadCopter with all components coplanar and one with the ability to adjust the center of propulsion vs the center of mass. A HexCopter explores a increasing the number of propulsion units, while another uses a combination of horizontal and vertical mount propellers and wings. These seed designs served as a starting point for the AI algorithms to explore the design space.

The Web interfaces to the tools are shown in 6. These are provided for manual parameterization and execution of a workflow an visualization of execution status. The MinIO interface is also shown, with data from previous executions. Note that REST interfaces are the preferred method for execution, typically integrated into the ML training process.

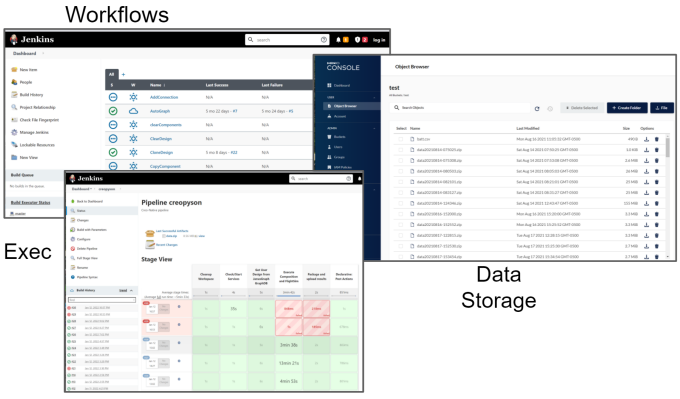

Fig. 6. Web Interfaces to the Design Oracle.

Designs were run through several trial trajectories which scored performance across several performance modes. The first of the trial trajectories was a simple climb and hover test, which evaluated vertical takeoff and landing (VTOL) capabilities and flight endurance, where the specified design was asked to climb to an altitude of 150 meters and hover. This was graded with first a threshold of a 200 second hover time, under which the design got 0 points, and an additional scaling score of 1 point per second of hover time up to a total of 400 points. This was viewed to be the most important feature of the design challenge, as the VTOL ability necessitated several design choices, and if a specific design could not earn points on this challenge, then that design was not considered a valid UAV.

The second and third trial trajectories worked with similar hard and soft point scoring lines. The second test bench was a horizontal distance flight, with 200 points coming after 2 kilometers of level flight, plus an additional point per 10

meters up to 4 kilometers of flight. Points were also subtracted for lateral error at a rate of 10 times the lateral error in meters, which indicated a lack of controllability. The third test bench evaluated level flight in a one kilometer diameter circle, focusing on accuracy. Designs were awarded 300 points for completing the circle, regardless of speed (assuming battery supplies lasted long enough to complete the trial), and penalized 50 times their lateral error in meters.

The final trial acted as the stretch goal requirement, where there was not a trial-based upper limit on design performance. The course taken was a racing oval of approximately 3.4km in length, with 200 points awarded for completing the course, and an additional point earned for every second less than 350 seconds of total flight time. Departing from the course by 10 meters reduced the score to 0, but inaccuracy was not further penalized, due to the higher speeds and curves. Ultimately, the top score possible was limited by our calculated trim states, which capped out at 50 m/s, which would yield a possible point total of 482, based on the total distance of the oval.

Given the corpus and tools, the teams were able to explore hundreds of thousands of design instances. From each instance, the Design Oracle returned the primary physical metrics and detailed reports from the flight simulator containing the available trim states for various target speeds, forces and accelerations at these speeds, power consumption and other voltage/current maxima at operating points for evaluation of motor/propeller/battery interaction. Additionally, scores for performance of dynamic flight on the set of trial trajectories, along with the position log during the flight and an overall score.

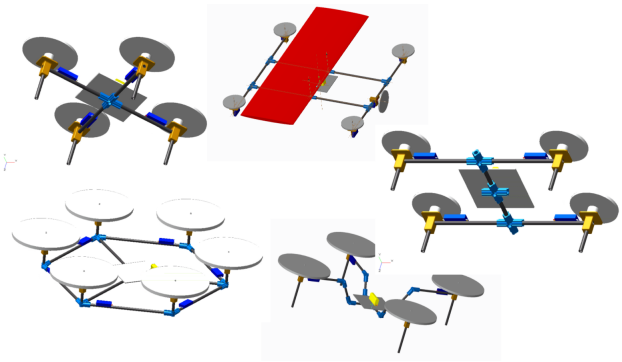

Fig. 7. Corpus Seed Designs

IV. Conclusion

The Design Oracle has been developed and refined under the SymCPS program. It leverages the OpenMETA Framework, developed under the DARPA Adaptive Vehicle Make program, and subsequently extended and maintained by MetaMorph Inc. The architecture is based on additional open source software for the distributed architecture and execution coordination, including JanusGraph, Cassandra, and ElasticSearch for design representation using the Gremiln Query Language. Execution is coordinated via Jenkins, and data storage using the S3-compatible MinIO system. The system is distributed using Docker images, and has been installed and used at all of SymCPS performers.

Two challenge problem corpora/seed designs have been supported by the system, and the tools were used for the UAV design challenge. Multiple workflows support composition of CAD models, computation of mass and moments of inertia, and location/orientation of propellers and wings, and a full 6 DOF flight simulator that supports arbitrary vehicle composition. Other workflows calculate lower fidelity system estimation of hover time and power usage.

Execution time and computational overhead was a limitation on the users of the system. Typical execution time was ˜120 seconds for a single point design, with ˜30 seconds for additional points. Given the desire for large data sets, we have improved the execution time to ˜40 seconds per architecture, plus 3-to-8 seconds per design point. We expect this time to increase as architectures get more complex and analysis fidelity increases.

V. Acknowledgements

This project was supported by DARPA under the Symbiotic Design for Cyber-Physical Systems with contract FA8750-20-C-0541. The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

We also thank the feedback of multiple participants of the TA1 and TA2 teams from SRI, Siemens, Peraton, Vanderbilt, MIT, Berkeley, CRA, and STR, for feedback on the tools that helped (and continue) to guide the capabilities of the system.

Finally, we thank MetaMorph Inc., for their support of the OpenMETA framework, which provides much of the analysis composition capabilities used in this effort.

References

[1] Sztipanovits, J., Bapty, T., Neema, S., Howard, L., Jackson, E. (2014). OpenMETA: A model-and component-based design tool chain for cyber-physical systems. In From programs to systems. The systems perspective in computing (pp. 235-248). Springer, Berlin, Heidelberg.

[2] Gray, J.S., Hwang, J.T., Martins, J.R.R.A. et al. OpenMDAO: an open-source framework for multidisciplinary design, analysis, and optimization. Structural and Multidisciplinary Optimization, 59, 1075–1104 (2019). https://doi.org/10.1007/s00158-019-02211-z

[3] J. Sztipanovits, T. Bapty, X. Koutsoukos, Z. Lattmann, S. Neema and E. Jackson (Sept. 2018) "Model and Tool Integration Platforms for Cyber–Physical System Design," in Proceedings of the IEEE, vol. 106, no. 9, pp. 1501-1526.

[4] Hartsell, C., Mahadevan, N., Ramakrishna, S., Dubey, A., Bapty, T., Johnson, T., Karsai, G. (2019, April). Model-based design for cps with learning-enabled components. In Proceedings of the Workshop on Design Automation for CPS and IoT (pp. 1-9).

[5] Jiang, Y., Zhang, T., Ho, D., Bai, Y., Liu, C. K., Levine, S., Tan, J. (2021). SimGAN: Hybrid Simulator Identification for Domain Adaptation via Adversarial Reinforcement Learning. arXiv preprint arXiv:2101.06005.

[6] Nagel, A, et al. AVM Component Model Specification AVM_Component_Spec_241a.pdf. https://cps-vo.org/taxonomy/term/5521

[7] Sharp, Austin et al., 2019. JanusGraph, Available at: https://janusgraph.org/.

[8] https://www.jenkins.io/doc/