ISoLA 2022

UNIVERSITY *of York*

HUAWEI

Using Digital Twins in the Development of Complex Dependable Real-Time Embedded Systems

Xiaotian Dai, Shuai Zhao, Benjamin Lesage, Iain Bate
Department of Computer Science, University of York, UK

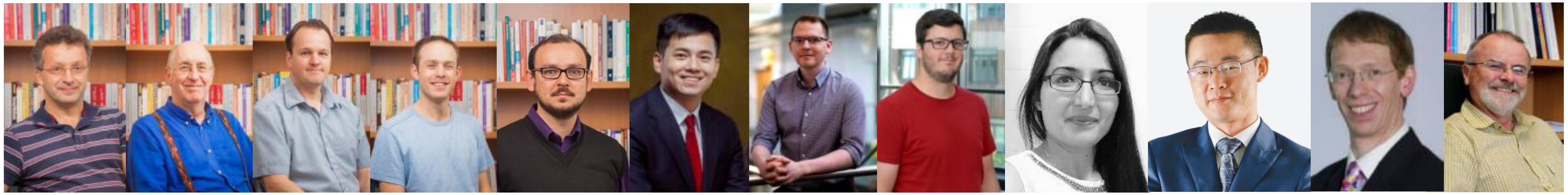
Dr. Xiaotian Dai

Real-Time Embedded Systems | Cyber-Physical Systems | Timing Analysis & Verification

- Research member of Real-Time and Distributed Systems Group (RTDS), University of York, United Kingdom
- 2020 - 2022, Postdoctoral Researcher in Real-Time Systems
- 2019 – 2020, Postdoctoral Researcher in High-Integrity Systems
- 2015 - 2018, PhD in Real-Time Systems, University of York

Real-Time and Distributed Systems Group

- Real-Time and Distributed Systems Group (RTDS)
- Established in 1990, it has undertaken research into all aspects of the design, implementation and analysis of real-time and distributed systems and is one of the world leading groups in those areas.
- Focus on systems where the distributed nature of the computation, the need for communication and coordination, and/or the timeliness of the system's actions are of critical importance to the overall functionality and to the end-users
- Application areas: embedded computing, Internet of Things (IoT), telecommunication, robotics, automotive, industrial process control, avionics, healthcare, and High-Performance Computing (HPC)

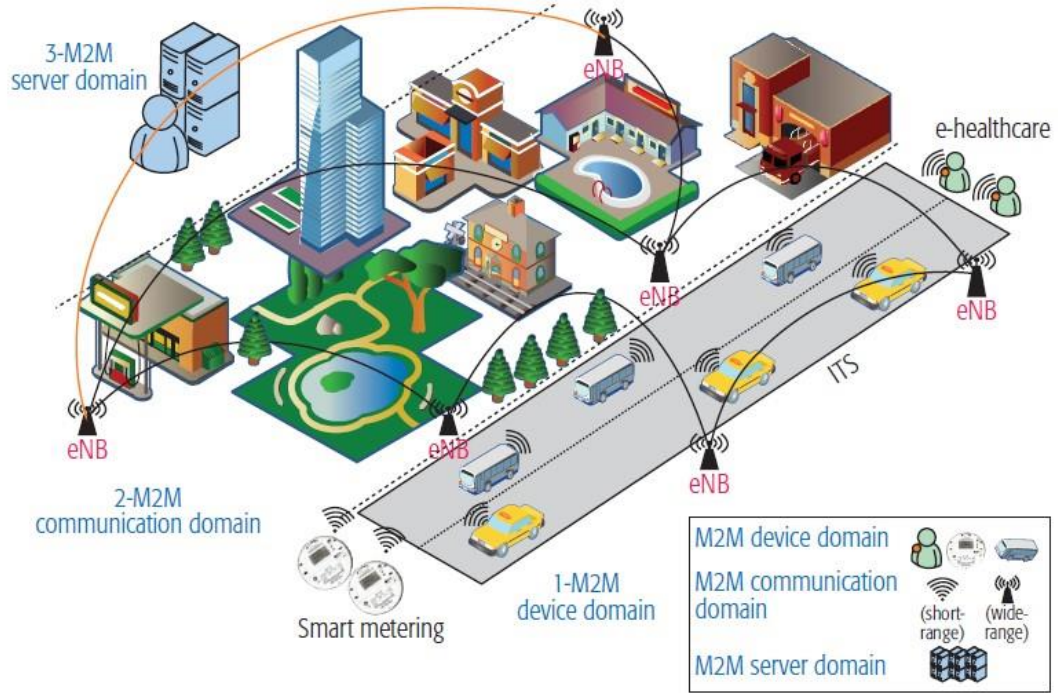


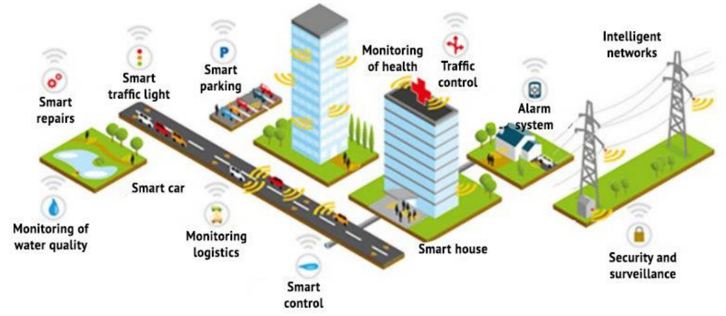
[1] L.S. Indrusiak, "RTDS Group Introduction", University of York

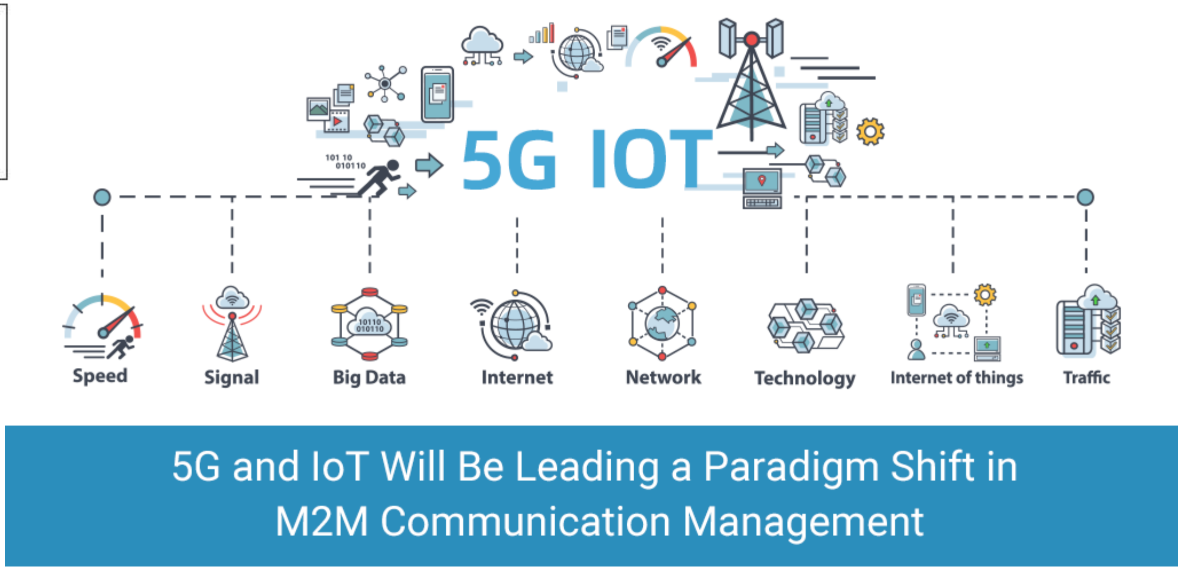
Outline of Presentation

- Background of Many-core Real-Time Embedded Systems
- DTiL-RTES framework to support resilience and adaptiveness
- DTiL-RTES for model refinement
- Eval. Explore model complexity
- Eval. Test of acceptance for model update
- Practical considerations
- Future work

The Fifth Generation (5G) Communication

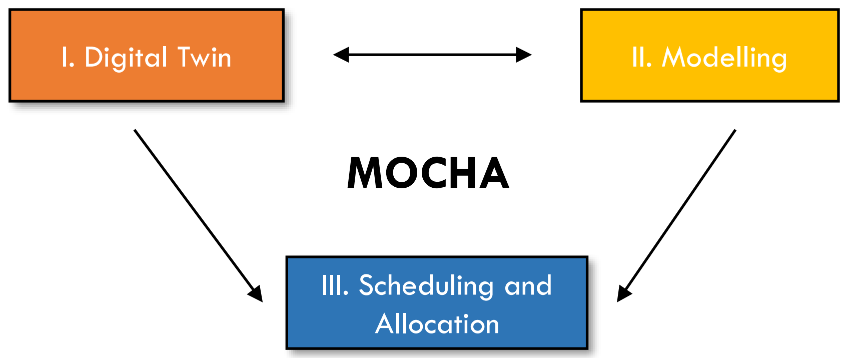


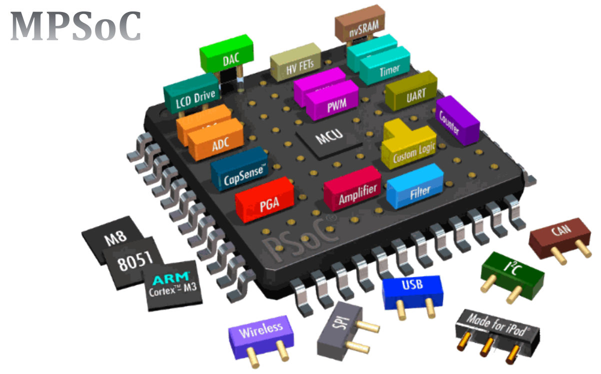



5G IOT
Speed
Signal
Big Data
Internet
Network
Technology
Internet of things
Traffic
5G and IoT Will Be Leading a Paradigm Shift in M2M Communication Management

MOCHA Research Project

- MOCHA: Modelling and Optimizing Complex Heterogenous Architectures. A three-year research project funded by Huawei.
- Working towards the next-generation of 5G/6G base stations to meet the increasing computing demands.
- Dealing with many-core systems with # of cores: 32, 64 or 128.
- How to develop efficient onboard task scheduling and allocation.



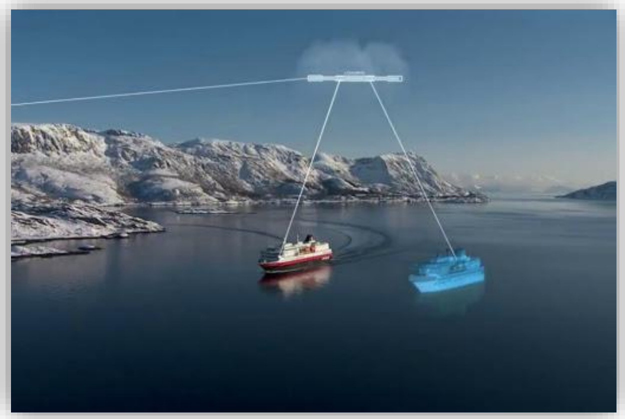


Challenges in many-core RTES for 5G

- As demands keep increasing, even for embedded devices or “edge” computing, are now using complex architecture with increasing cores.
- Execution environment is subjected to more uncertainties, which is hard to be predicted at design-time. However, some design decisions have to be made before software is available.
- Adopting multi-core is still challenge for most industrial real-time systems – timing and interference need to be explored with extensive test.
- Enhancing predictability, adaptiveness and performance (e.g. by better use of cache) is demanding.
- Assuring the system of meeting the timing requirements becomes essential. For example, the 5G URLLC (ultra-reliable low latency communications) has a deadline requirement as low as 1 *ms*.

What is a Digital Twin?

- "A digital twin is a *virtual representation* that serves as the *real-time* digital counterpart of a physical object or process."
- It is a duplication/Digital model that works *exactly* in the same way as the physical system.
- Attempts are being made in space and avionics, transportation, industrial automation, medical systems and autonomous driving.


Digital Twin for Maritime Transportation

Digital Twin for Process Control

Digital Twin for Avionic Engine Design and Control

What we think is a good Digital Twin?

A DT has to be useful. The DT’s role is different for the two phases:

- **DT @ Design-time**: evaluation and exploration of scheduling and allocation algorithms with more accurate models;
- **DT @ Run-time**: make (fast) decisions with approximated timing models, e.g., decisions of job admission.

The DT should meet the following principles:

- **Effectiveness** --- a satisfactory model without fully coverage profiling. The DT should not need the best accurate model but an effective one to produce a good result.
- **Efficiency** --- low computing and memory overhead.
- **Acceptability** --- following engineering practice, automating key processes and reduce human efforts from the engineers.

An Overview of DTiL-RTES

- We introduced *Digital Twin in-the-loop for scheduling and allocation of Real-Time Embedded Systems* (DTiL-RTES).
- To our best knowledge, we are <u>the first work</u> proposed that is using a Digital Twin for <u>scheduling</u> and <u>allocation</u> of RTES.

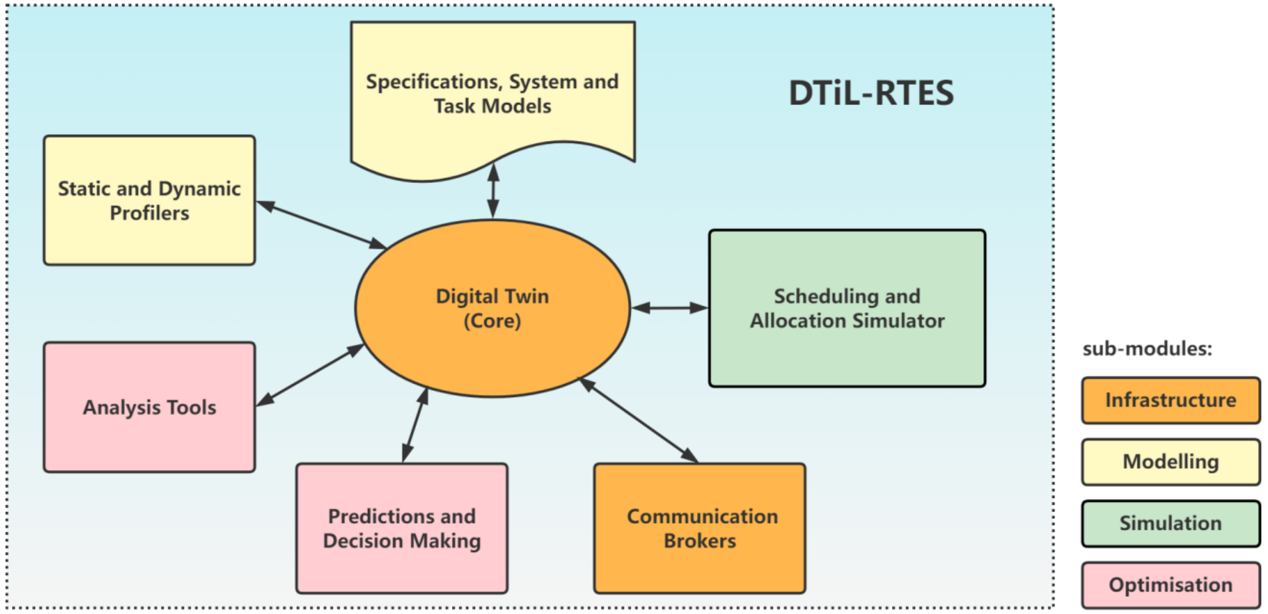
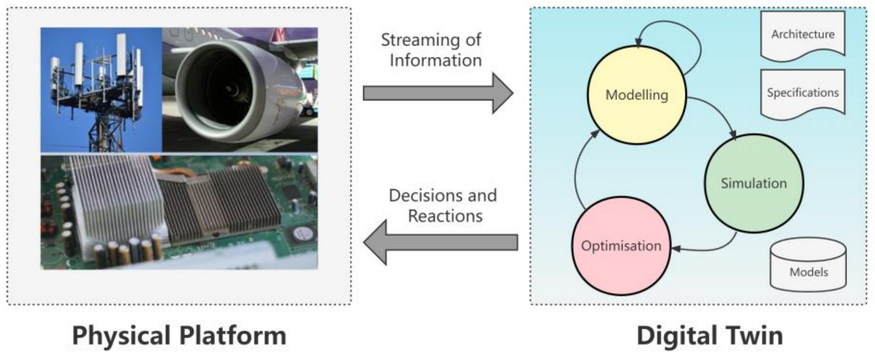
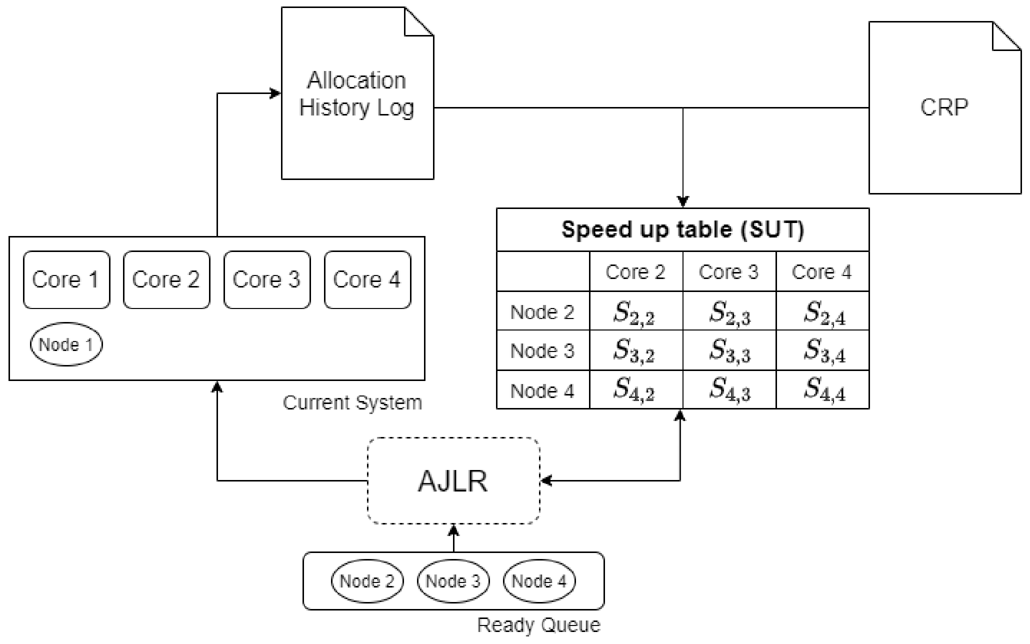

DTiL-RTES
Specifications, System and Task Models
Static and Dynamic Profilers
Digital Twin (Core)
Scheduling and Allocation Simulator
Analysis Tools
Predictions and Decision Making
Communication Brokers
sub-modules:
Infrastructure
Modelling
Simulation
Optimisation

Figure 1. The modules of DTiL-RTES

An Overview of DTiL-RTES

- Our framework does not require a specific scheduler, but it does require a timing model that has good abstraction level and with good observability.
- We have a built-in simulator that is consistent with the model and reflect the correct system architecture. It is also seen as a model of the system.
- **RQ. 1**: How to choose an effective timing model with adequate level of abstraction for the Digital Twin?





Speed up table (SUT)			
	Core 2	Core 3	Core 4
Node 2	$S_{2,2}$	$S_{2,3}$	$S_{2,4}$
Node 3	$S_{3,2}$	$S_{3,3}$	$S_{3,4}$
Node 4	$S_{4,2}$	$S_{4,3}$	$S_{4,4}$

Figure 2. Scheduling and Allocation Simulator

DTiL-RTES: Finding an Effective Timing Model

- *Cache Recency Profile* (CRP):

$$\boxed{\text{Execution time} = f\,(\text{Cache Recency, WCET, Allocation, …})}$$

- Data-driven approach that models the timing behavior from real executions (controversial to static program analysis)
- The model is then used to guide scheduling and allocation, for example, we have further proposed *Allocation and Jobs based on Learnt Recency* (AJLR)
- Under its hood, DTiL-RTES adapts CRP (as the predictive timing model) and AJLR (as the scheduler and allocator).

Building the Offline Timing Model

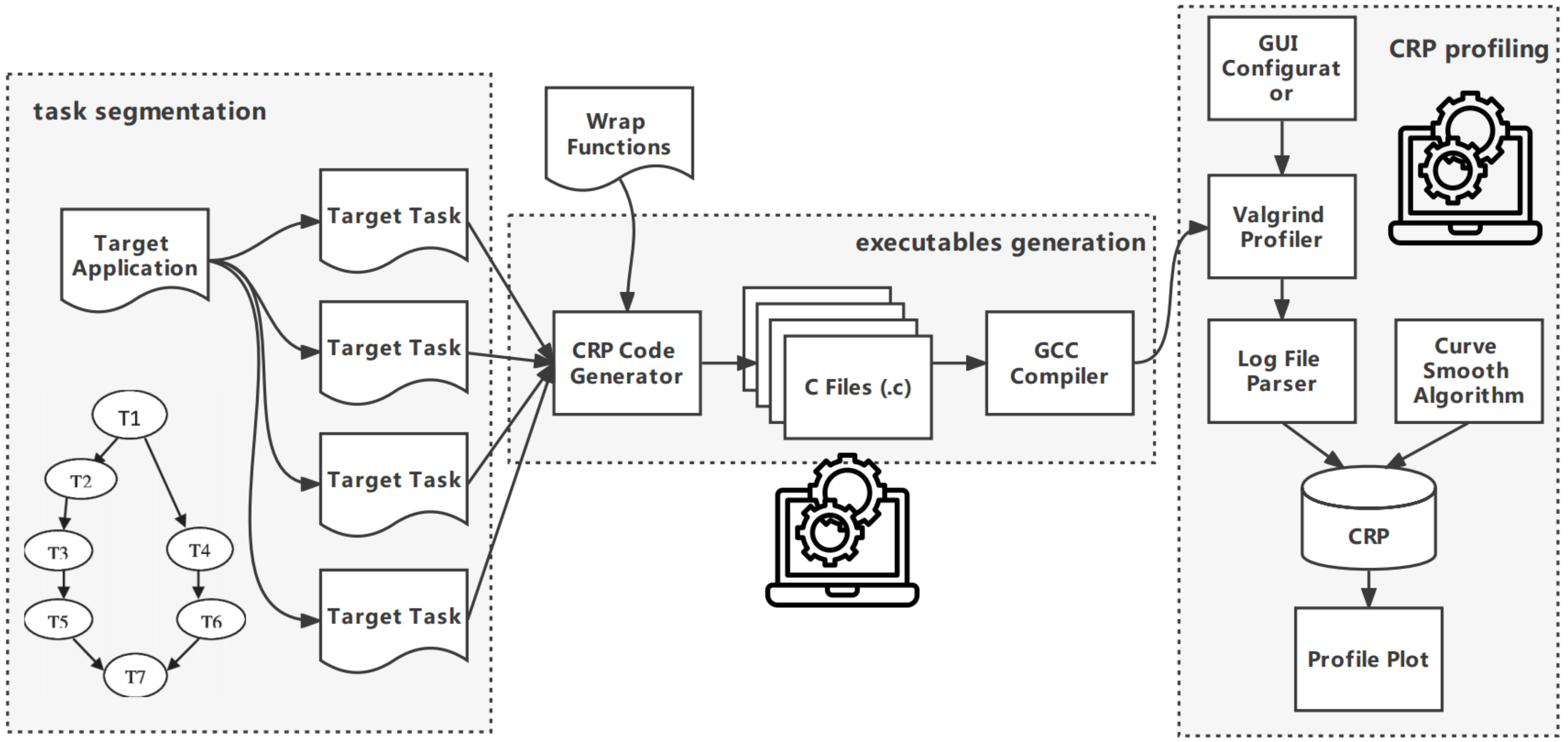


Figure 3. Task profiling of CRPs with an offline automated process

Issues of the Offline Model

- Offline-built models are inevitably *not sufficiently accurate*, at least *not accurate across all scenarios*, or otherwise being *overly pessimistic* (e.g. worst-case execution time; WCET).
- Representative counter-scenarios:
  - Input data in the test cases is not representative;
  - Task was profiled in isolation within a controlled environment;
  - The OS can add extra interferences;
  - The workload can be changed over time;
  - The hardware behaves differently;
  - and many more…!

(More) Research Questions

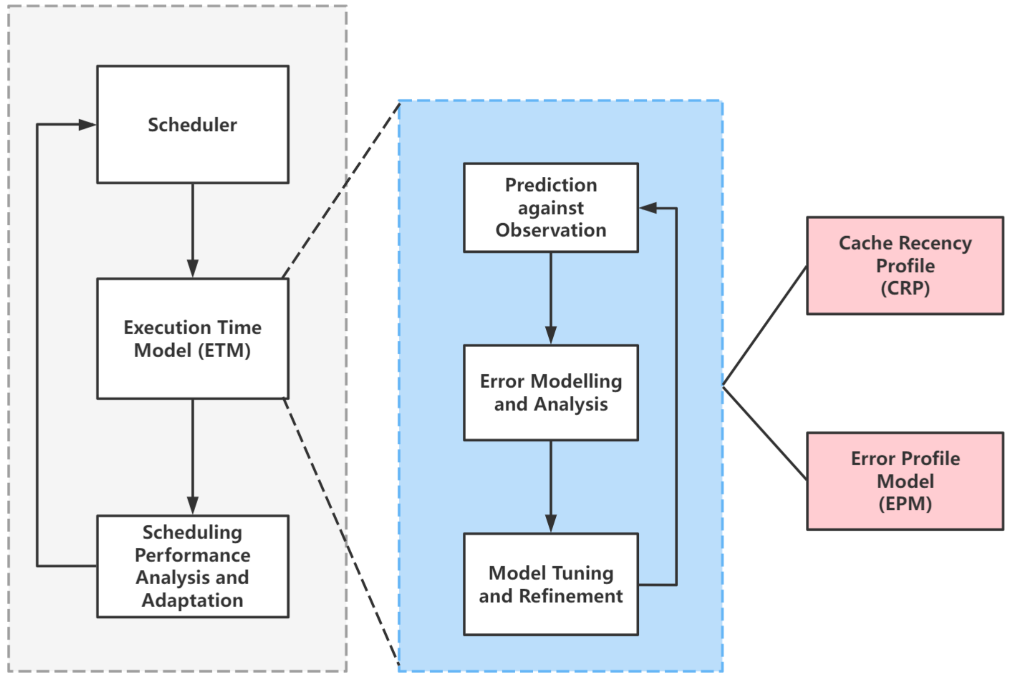
- **RQ. 2**: When the data observed from the real system and the model are disagreed, how to evaluate and how to align them again?
- **RQ. 3**: How to understand the sources and impact of errors, and *how* to refine the model *when* the system can be improved (w.r.t. model inconsistency).

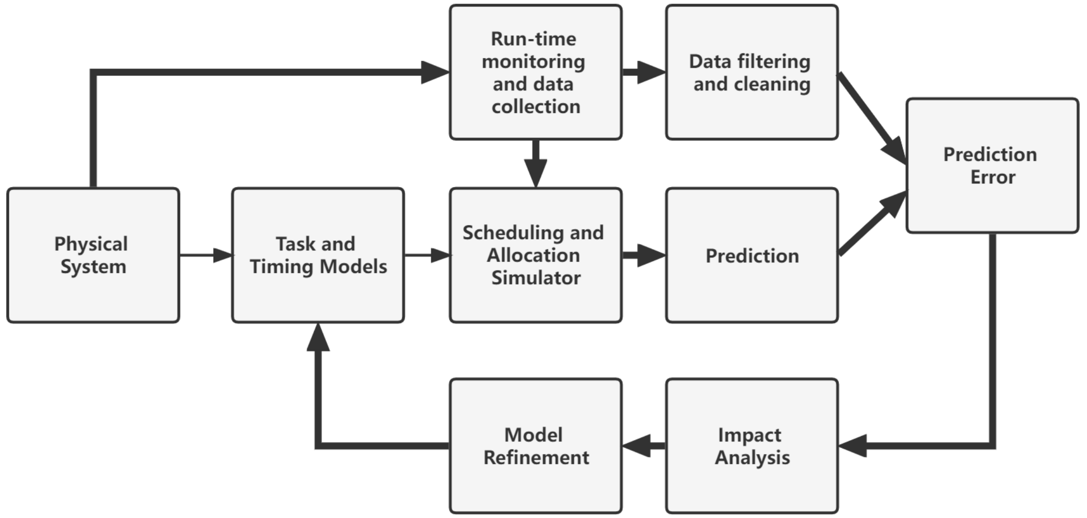
▷ **Model Mismatch**

▷ **Model Refinement**

How DTiL-RTES closes the gap?

- DTiL-RTES uses *online measurements* to build a new CRP model, *predict the performance* with a scheduling and allocation simulator, and make a decision to (or not to) *update the timing model* through *impact analysis*.


Scheduler
Execution Time Model (ETM)
Scheduling Performance Analysis and Adaptation
Prediction against Observation
Error Modelling and Analysis
Model Tuning and Refinement
Cache Recency Profile (CRP)
Error Profile Model (EPM)


Physical System
Task and Timing Models
Run-time monitoring and data collection
Data filtering and cleaning
Scheduling and Allocation Simulator
Prediction
Prediction Error
Impact Analysis
Model Refinement

DTiL-RTES: Model Refinement

Model Refinement Process:

1. The system initially has an offline profiled CRP model, i.e., without considering inter-task and OS interference.
2. Once the system is up and running, operational data in the production environment can be collected.
3. Data is filtered and classified into the right task profile.
4. A new model is then fitted with the (noisy) measurements with light-weight machine learning approach.

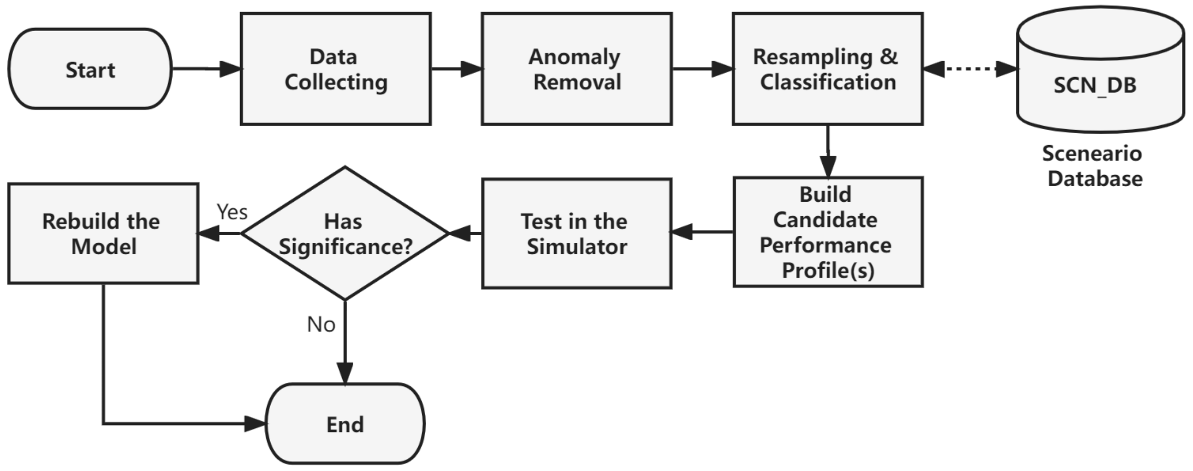


Figure 4. Model Refinement Process

The data is fitted through **PieceWise-Linear Regression (PWLR)** with the **Least Square Error (LSE)**

DTiL-RTES: Model Refinement

Model Refinement Process:

5. The comparison of the real data and the output from the model will produce some prediction error, which will trigger a second-stage evaluation process.
6. The impact of the model is produced by a prediction-based evaluation method, i.e., from a simulator of AJLR and CRP.
7. Finally, the results are compared with statistical test to make a decision to change the model.

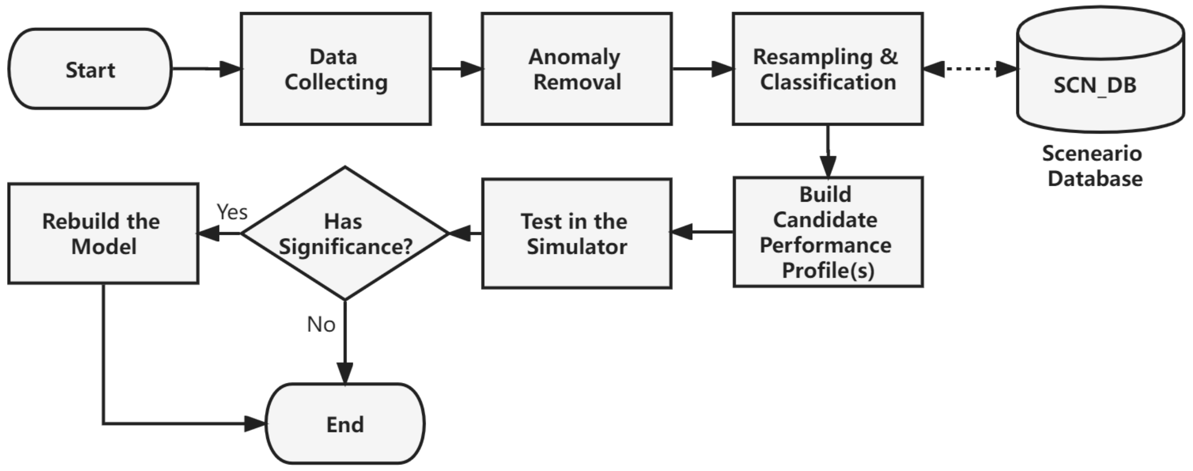
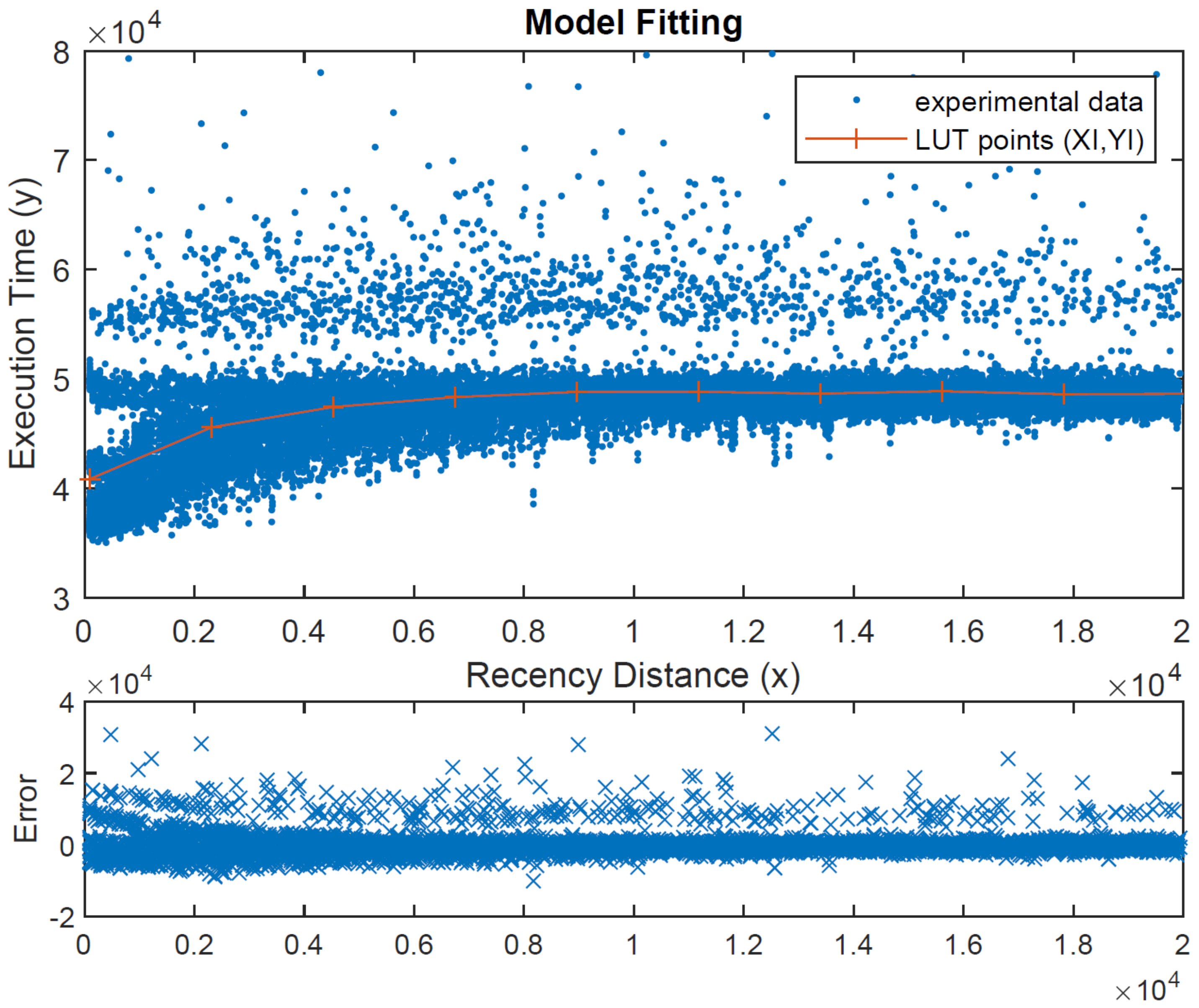


Figure 4. Model Refinement Process

The significance is tested through a non-parametric **Kolmogorov–Smirnov (K-S)** statistical test.

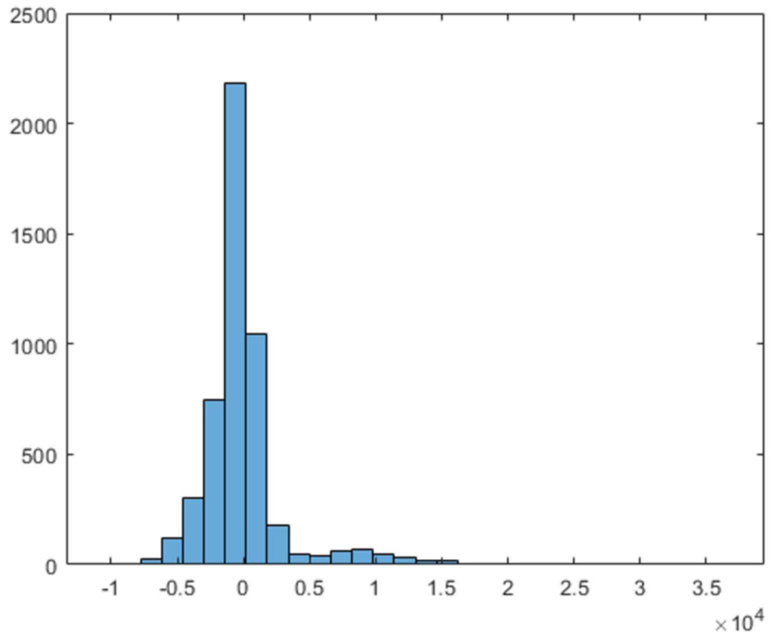
Fitting a Model with Noisy Measurements



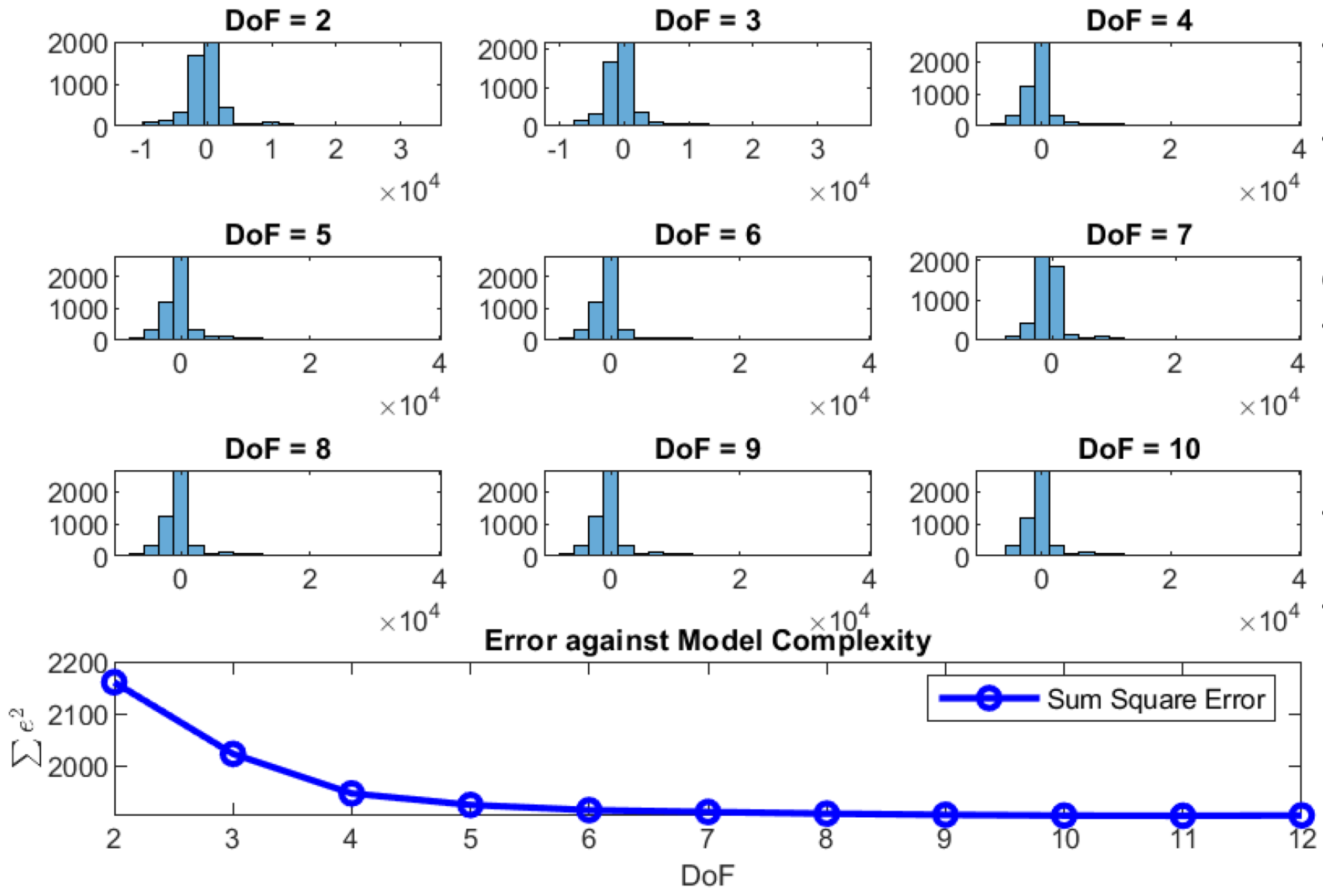
PSWL: Piece-wise Linear Regression

$$y(x)=\begin{cases}\eta_1+\beta_1\left(x-b_1\right), & b_1<x\leqslant b_2\\ \eta_2+\beta_2\left(x-b_2\right), & b_2<x\leqslant b_3\\ \cdots & \\ \eta_n+\beta_n\left(x-b_{n-1}\right), & b_{n-1}<x\leqslant b_n\end{cases}$$

$$RMSE=\sqrt{\sum\left(\hat{y}_i-y_i\right)^2/n}.$$



Evaluation of Model Complexity against Error

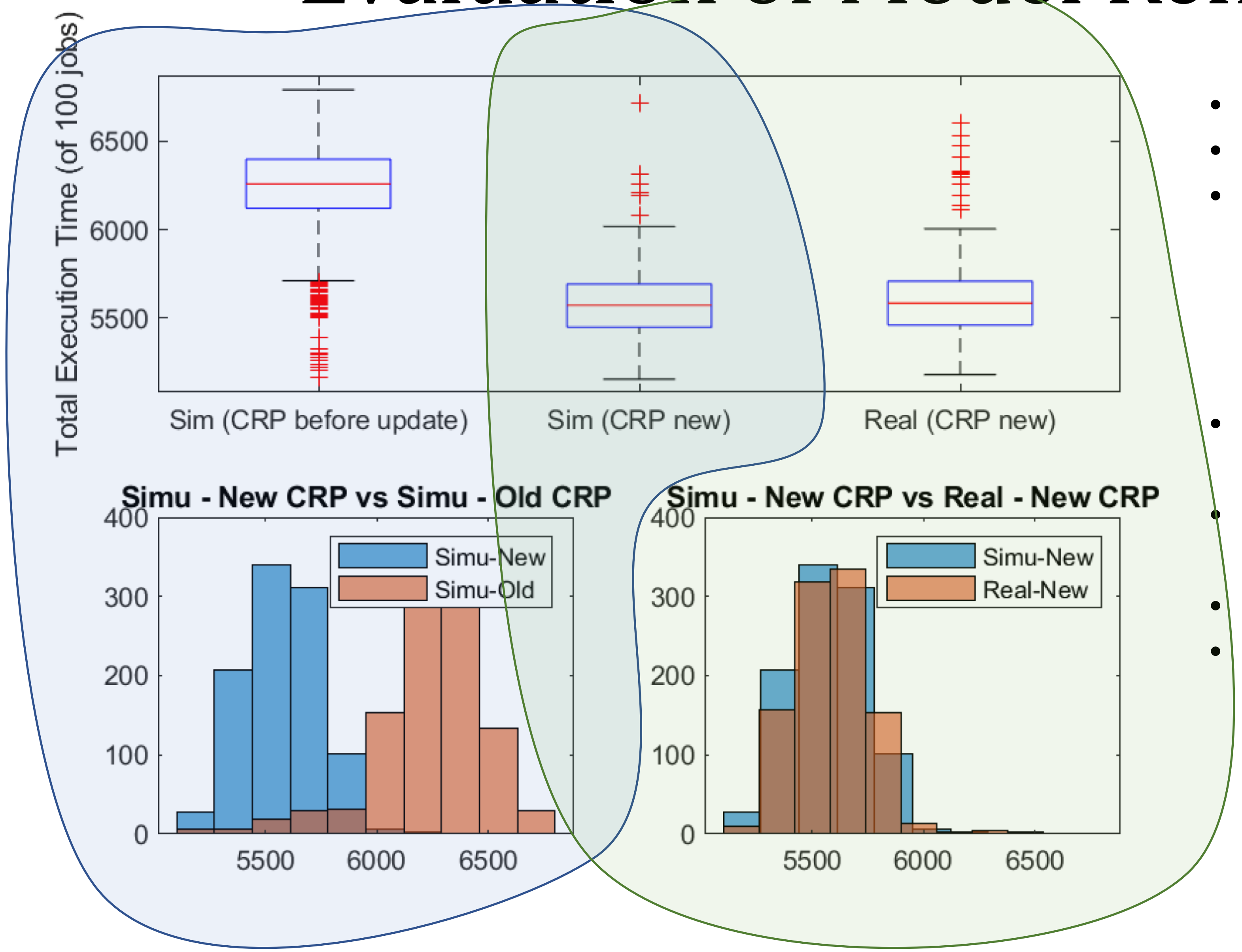


- Error is defined as the real data against model predictions.
- 80% data for training and 20% for testing.

Observations:

- Higher DoF means higher model complexity but more memory/computation overhead.
- Higher complexity does not always assure higher accuracy.
- An appropriate level can be found by setting an accepted error threshold. This can be set as statical or adaptive.

Evaluation of Model Refinement



- Fix model DoF = 3
- Initial CRP was first profiled offline
- Improved CRP profiled online with measurement on a testbed environment with tasks from a benchmark (TacleBench)

- Sim: result evaluated through the simulator
- Real: result obtained through real measurement
- - Old: with model before update
- - New: with model after update

The model is aligned after the refinement process. Inconsistency was reduced from 20% to less than 5%.

Deployment of Digital Twin (1/2)

- Onboard DT: The Digital Twin is running in a dedicated partition of the system. For example, allocate 2 cores to the Digital Twin and the rest 30 cores to run the applications.
- Offboard (Distributed) DT: The Digital Twin is deployed on a dedicated remote machine (i.e. DT server). Data is passed through the client(s) to the DT server.
- In both cases the communication layer needs to be re-implemented (ported), which is platform- and application-dependent.

Deployment of Digital Twin (2/2)

	Pros	Cons
Onboard DT	Lower data transmission overhead. Data can be passed with low overhead, e.g. through shared memory.	Need to reserve/share some capacity. For example, a dedicated partition.
Distributed DT	Offload the computation on the host machine.	Subjected to communication reliability and security.
		Takes more time to response due to networking delay, etc.

DTiL-RTES: Fitting into the Safety Case

- We have also shown how the DTiL-RTES can be fitted into a safety case to support the integrity of embedded systems.
- Digital Twin can provide posterior statistical correction.
- The purposes of the Digital Twin are:
    - Use empirical estimates to process to target reliability.
    - Showing the conclusions from offline simulation are still validated online.
    - Support detection of significant errors and the strategy for logging and correction.

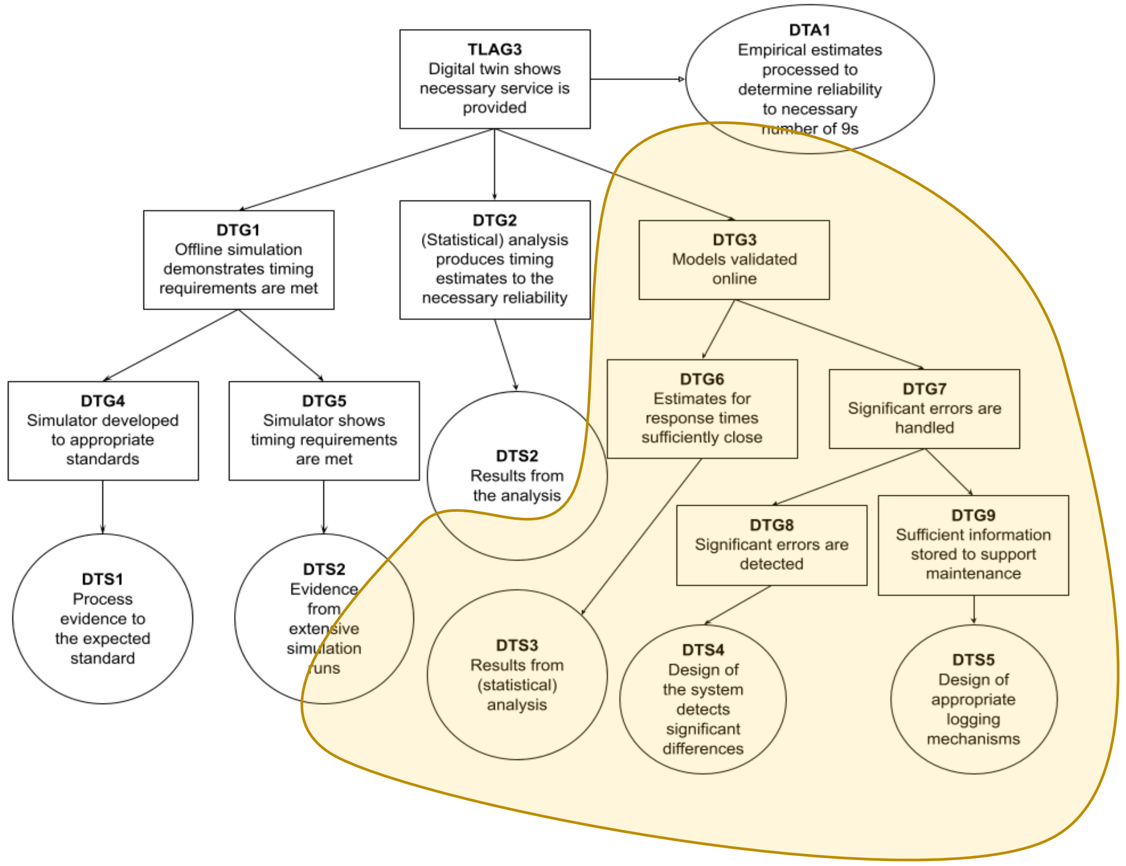

TLAG3 Digital twin shows necessary service is provided
DTA1 Empirical estimates processed to determine reliability to necessary number of 9s
DTG1 Offline simulation demonstrates timing requirements are met
DTG2 (Statistical) analysis produces timing estimates to the necessary reliability
DTG3 Models validated online
DTG4 Simulator developed to appropriate standards
DTG5 Simulator shows timing requirements are met
DTS2 Results from the analysis
DTG6 Estimates for response times sufficiently close
DTG7 Significant errors are handled
DTG8 Significant errors are detected
DTG9 Sufficient information stored to support maintenance
DTS1 Process evidence to the expected standard
DTS2 Evidence from extensive simulation runs
DTS3 Results from (statistical) analysis
DTS4 Design of the system detects significant differences
DTS5 Design of appropriate logging mechanisms

Figure 5. Digital Twin Safety Argument

Summary

- In this work, we proposed DTiL-RTES, the first Digital Twin framework for scheduling Real-Time Embedded Systems.
- We demonstrated how DTiL-RTES has the potential to improve the timing model, and consequently improve the adaptiveness, resilience and robustness.
- We then discussed practical problems including choosing the right degree of freedom against the residual error.
- Finally, we demonstrated an example of a study of model refinement on a testbed environment with real programs.

Future Work

- **Short-term plan**
  - Make a workable prototype digital twin on an industrial system that can adapt to errors in the decision-making process and improve the system by adaptively changing the parameters of the underlying models.
- **Long-term plan**
  - To make on-line decisions relevant to scheduling (e.g., admission of new tasks to schedule or allow more events to be processed).
  - To formulate system-of-systems and load balancing across base stations.
  - To adapt this method beyond scheduling, for example, to answer "what if" questions for *design space exploration* (DSE).
  - To adopt *formal verifications* into the process of decision making.

Thank You!