Supplementary Information For A Multi-Agentic Digital Twin Framework For Human-in-the-loop Workflows

Gayan Dihantha Kuruppu Kuruppu Appuhamilage[1],
Maqbool Hussain[1,*], Wajahat Ali Khan[1], Farid Meziane[1], Ben Anderson[2]

[1] School of Computing, College of Science and Engineering, University of Derby, Kedleston Road, Derby, DE22 1GB, United Kingdom
[2] AddQual Ltd, No. 27, Dunstall Park, Dunstall Park Road, Derby DE24 8HJ, United Kingdom

* Corresponding Author: Maqbool Hussain (m.hussain@derby.ac.uk)

Contents

Supplementary Table 1: Conceptual Data Model 2
Suppementary Table 2: Prompt Examples for Single-Agent Evaluations 2
Suppementary Table 3: Prompt Examples for Multi-Agent Evaluations 3
Suppementary Table 3: UR Cobot Features 4
Suppementary Table 4: Robotiq Gripper Features 5
Suppementary Table 5: ZEISS GOM Inspect Features 5
Supplementary Figure 1: Model Context Protocol 5
Supplementary Figure 2: Digital Twin 6
Supplementary Figure 3: Internet of Things 7
Supplementary Figure 4: Physical Twin 9

Supplementary Table 1: Conceptual Data Model

Data Type

Guideline

```
[
  {
    "id": "c3b800b8-8733-4bf6-8a87-36fc64b5a87c",
    "partitionKey": "guideline",
    "code": "o1",
    "text": "The UR cobot shall be operated within validated tool acceleration of 0.25,
             tool speed of 0.25, and payload limits."
  }
]
```

Plan

```
[
  {
    "id": "e954f22b-a50c-40ad-8a1e-27262aeefbe9",
    "partitionKey": "plan",
    "name": "p1",
    "description": "Aerospace item metrology scan plan for engine-item-b.",
    "stepModelList": [
      {
        "stepNumber": 1,
        "name": "Item Acquisition Step",
        "description": "This step involves acquiring the aerospace item for
                        metrology scanning.",
        "stateModelList": [
          { "twinId": "temperature_sensor_1", "property": "var_degree_celsius",
            "value": "20" }
        ],
        "onTrueTaskModelList": [
          { "twinId": "ur_cobot", "command": "control_show_popup",
            "payloadModelList": [{ "property": "popup_text",
                                   "value": "temperature is low" }] }
        ],
        "onFalseTaskModelList": [
          { "twinId": "ur_cobot", "command": "control_show_popup",
            "payloadModelList": [{ "property": "popup_text",
                                   "value": "temperature is too high" }] }
        ]
      }
    ]
  }
]
```

Suppementary Table 2: Prompt Examples for Single-Agent Evaluations

#	Prompt Structure				Target Task Type			
	Format		Rule					
	Structured Prompt	Unstructured Prompt	Conditional Prompt	Unconditional Prompt	Digital Twinning Tasks	Guideline Tasks	Planning Tasks	Not Existing Tasks

1	✓		✓		*"get only gomProgress property from the gom twin, if gomProgress is 0 then execute control_start_gom on the gom twin, if gomProgress is not 0 then execute control_start_gom on the gom twin."*	*"search for the g1 guideline. if it does not exist, create a new g1 guideline. this integrated system shall be treated as a single machine, and a system-level risk assessment shall be completed prior to operation."*	*"add a new on-task true condition for plan p1 step 1 with a ur cobot open popup control, with the payload popup text value: 'temperature is low, requires exactly 20 degrees.'"*	*"check for stock exchange mcp tools and let me know if any required stock exchange mcp tool is missing. ensure that no plan tools are used. if stock exchange mcp tools exists then, create a portfolio note that this trading system should be treated as a single strategy."*
2	✓			✓		*update the g1 guideline as follows: the universal robot cobot, robotiq gripper, sensors, and gom scanner which is zeiss inspect shall be used within manufacturer-specified limits."*	*"search for turbine-blade-x."*	
3		✓	✓		*"just check the act value from the robotiq gripper twin. if act is 0, run activate gripper on the robotiq gripper twin."*		*"in step 1 of pa, can you add a new state that sets temperature sensor 1 to 20 degrees C?"*	
4		✓		✓		*"let's update g6 with a few more details: calibration status needs to be clearly visible on the equipment, and if anything is out of tolerance, we need to segregate those results. also, we have to re-verify everything after any maintenance that could mess with the accuracy."*	*"can you make another plan called pb for scanning engine part A in the aerospace metrology setup?"*	*"update the portfolio note to say: users must always review order details before submitting."*

Suppementary Table 3: Prompt Examples for Multi-Agent Evaluations

#	Prompt Structure				Target Task			
	Format		Rule					
	Structured Prompt	**Unstructured Prompt**	**Conditional Prompt**	**Unconditional Prompt**	**Digital Twinning Tasks**	**Guideline Tasks**	**Planning Tasks**	**Not Existing Tasks**
1	✓		✓		*"using digital twin agent, if state true then, execute on true task from step 1 from plan p1"*	*"using guideline agent, create a new guideline g1. digital twin id = infrared_sensor_1, property = var_detect_object, value = true"*	*"using plan agent, add a new state to plan p1 step 1 using digital twin id, state property and value in the text from guideline g1"*	

2	✓			✓		*"using guideline agent, create a new guideline g1. digital twin id = infrared_sensor_1, property = var_detect_object, value = true."*		
3		✓	✓		*"get step 1 from plan p3, check the states"*	*"add a new state to plan p1 step 1 using digital twin id, state property and value in the text from guideline g1"*	*"if state true then, execute on true task from step 1 from plan p1"*	
4		✓		✓		*"delete guideline g9"*	*"clear plan p3"*	

30 # Suppementary Table 3: UR Cobot Features

#	Feature Type	Feature / Function Name	Description
1	Telemetry	target_q	Target joint positions
2	Telemetry	target_qd	Target joint velocities
3	Telemetry	target_qdd	Target joint accelerations
4	Telemetry	target_current	Target joint currents
5	Telemetry	target_moment	Target joint torque moments
6	Telemetry	actual_current	Actual joint currents
7	Telemetry	actual_q	Actual joint positions
8	Telemetry	actual_qd	Actual joint velocities
9	Telemetry	joint_control_output	Joint control currents
10	Telemetry	actual_tcp_force	Generalised forces at the Tool Centre Point (TCP) compensated for payload-generated forces and torques
11	Telemetry	joint_temperatures	Temperature of each joint in degrees Celsius
12	Telemetry	joint_mode	Joint control modes; values are defined in the cobot's joint-mode constants table
13	Telemetry	actual_tool_accelerometer	Tool accelerometer values for the X, Y, and Z axes
14	Telemetry	speed_scaling	Speed-scaling value applied by the trajectory limiter
15	Telemetry	actual_momentum	Norm of the Cartesian linear momentum
16	Telemetry	actual_main_voltage	Main voltage of the safety control board
17	Telemetry	actual_robot_voltage	Robot voltage supplied by the safety control board, nominally 48 V
18	Telemetry	actual_robot_current	Robot current measured by the safety control board
19	Telemetry	actual_joint_voltage	Actual voltage of each joint
20	Telemetry	runtime_state	Program state, such as stopping, stopped, running, pausing, paused, resuming, or retracting
21	Telemetry	robot_mode	Robot mode, such as disconnected, confirm safety, booting, power off, power on, idle, backdrive, running, updating, powering off, or arm booting
22	Telemetry	safety_mode	Safety mode; deprecated and replaced by safety status
23	Telemetry	analog_io_types	Bit field representing analogue input 0, analogue input 1, analogue output 0, and analogue output 1 using bits 0-3
24	Telemetry	io_current	Input/output current
25	Telemetry	tool_mode	Tool operating mode, represented using tool-mode constants
26	Telemetry	tool_output_voltage	Tool output voltage
27	Telemetry	tool_output_current	Tool output current
28	Telemetry	standard_analog_out_0	Standard analogue output channel 0
29	Telemetry	standard_analog_out_1	Standard analogue output channel 1

30	Telemetry	standard_digital_out_0	Standard digital output channel 0
31	Telemetry	standard_digital_out_1	Standard digital output channel 1
32	Telemetry	standard_digital_out_2	Standard digital output channel 2
33	Telemetry	standard_digital_out_3	Standard digital output channel 3
34	Telemetry	standard_digital_out_4	Standard digital output channel 4
35	Telemetry	standard_digital_out_5	Standard digital output channel 5
36	Telemetry	standard_digital_out_6	Standard digital output channel 6
37	Telemetry	standard_digital_out_7	Standard digital output channel 7
38	Command	move_j_command	Executes a joint-space movement
39	Command	pause_command	Pauses the currently running robot program
40	Command	play_command	Starts or resumes the robot program
41	Command	close_safety_popup_command	Closes the active safety popup
42	Command	unlock_protective_stop_command	Unlocks the robot following a protective stop
43	Command	open_popup_command	Opens a popup on the robot interface
44	Command	close_popup_command	Closes an active popup
45	Command	power_on_command	Powers on the robot
46	Command	power_off_command	Powers off the robot
47	Command	start_free_drive_mode_command	Enables free-drive mode
48	Command	stop_free_drive_mode_command	Disables free-drive mode

31 Suppementary Table 4: Robotiq Gripper Features

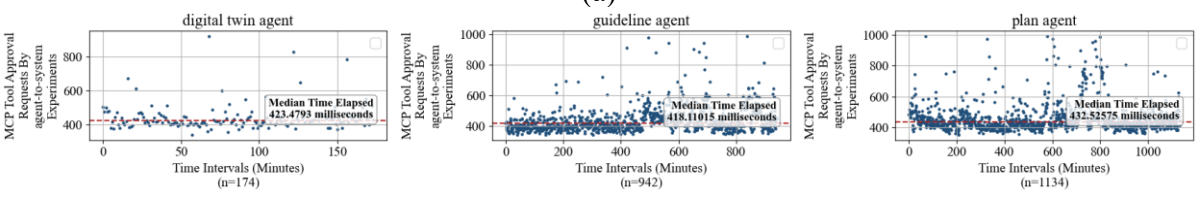
#	Feature Type	Feature / Function Name	Description
1	Telemetry	act	Gripper activation state
2	Telemetry	gto	Go-to action state
3	Telemetry	force	Configured gripper force
4	Telemetry	spe	Configured gripper speed
5	Telemetry	pos	Current gripper position
6	Telemetry	sta	Gripper activation or operating status
7	Telemetry	pre	Position-request echo
8	Telemetry	obj	Object-detection status
9	Telemetry	flt	Gripper fault status
10	Command	activate_gripper_command	Activates and initialises the gripper
11	Command	open_gripper_command	Opens the gripper
12	Command	close_gripper_command	Closes the gripper

32 Suppementary Table 5: GOM Features

#	Feature Type	Feature / Function Name	Description
1	Telemetry	is_gom_running	Indicates whether the GOM Inspect process is running
2	Telemetry	gom_progress	Current GOM inspection progress as a percentage
3	Command	start_gom_command	Starts the GOM inspection process
4	Command	stop_gom_command	Stops the GOM inspection process

33 Supplementary Figure 1: Model Context Protocol

(a)



(b)

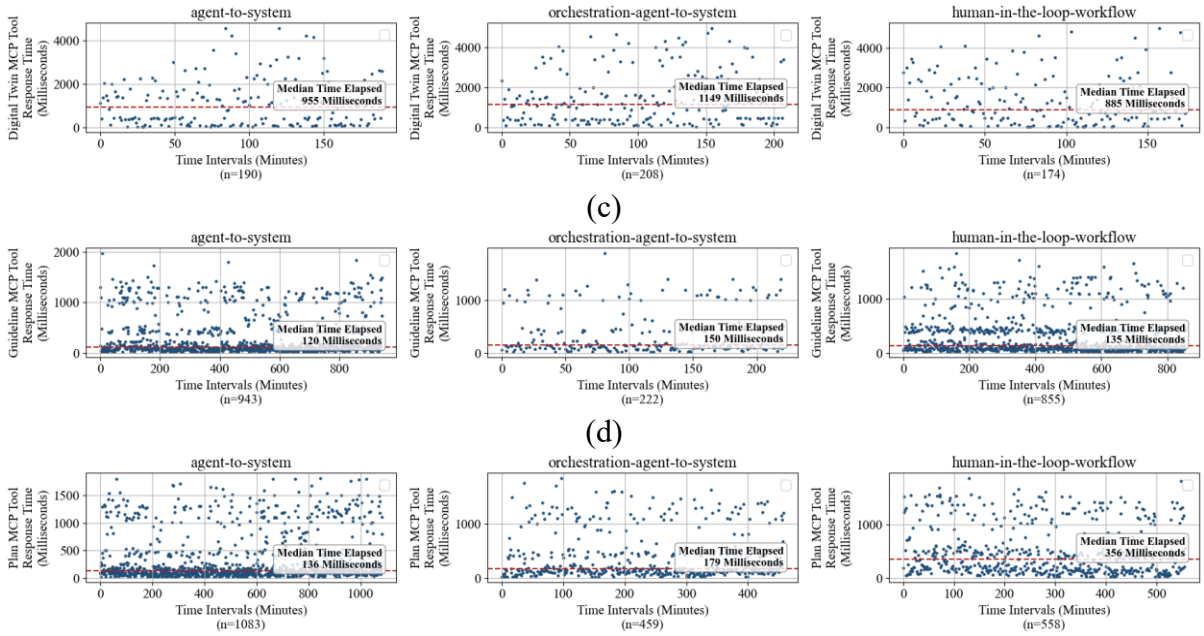


Figure 1. Model Context Protocol (MCP) Aspects.

a. ***MCP Tool Approval Requests By Agent-To-System Experiments*:** Time elapsed for an agent to approve model context protocol request in agent-to-system experiments. The metric is used to measure time it takes to perform authorisation for a particular request. Based on results, it takes on median time elapsed 418 to 432 milliseconds to perform this task. The maximum amount of time framework took to authorise is 10,552.27 milliseconds in a human-in-the-loop workflow prompt. This metric is only available for agent-to-system experiments as the current framework does not currently have capability to capture this information in connected agents.

b. ***Digital Twin MCP Tool Response Time*:** Time elapsed to perform digital twin operations in each request. This metric is implemented to measure exact time consumption required to process HTTP requests. During this experiment, 864 digital twin operations are performed including 479 get-state operations and remaining 379 set-state operations. On median, each operation took between 885 and 1,149 milliseconds to process. The maximum time recorded is 32,191 milliseconds for a move-joint command in a UR Cobot digital twin as its physical twin takes time to move its arm between given positions.

c. ***Guideline MCP Tool Response Time*:** Time elapsed to perform guideline operations in each request. Like the digital twin MCP tool this metric is implemented to process HTTP requests related to guideline API endpoints. During this experiment, this framework processed 2,542 guideline operations. On median, these operations took between 120 and 150 milliseconds as less time is required due to communication between an Azure service. The maximum time recorded is 43,862 milliseconds as at starting container apps consumed additional time to cold start its application.

d. ***Plan MCP Tool Response Time*:** Time elapsed to perform plan operations in each request. As previously mentioned, this metric is implemented to process HTTP requests related to plan API requests. During this experiment, this framework processed 2,457 plan operations with an median time elapsed between 136 and 356 milliseconds. Unlike guideline tool, the plan tool requires additional querying and this affected median time elapsed. The maximum time recorded is 56,053 milliseconds at the beginning of a cold start in container apps.

34 # Supplementary Figure 2: Digital Twin

(a)

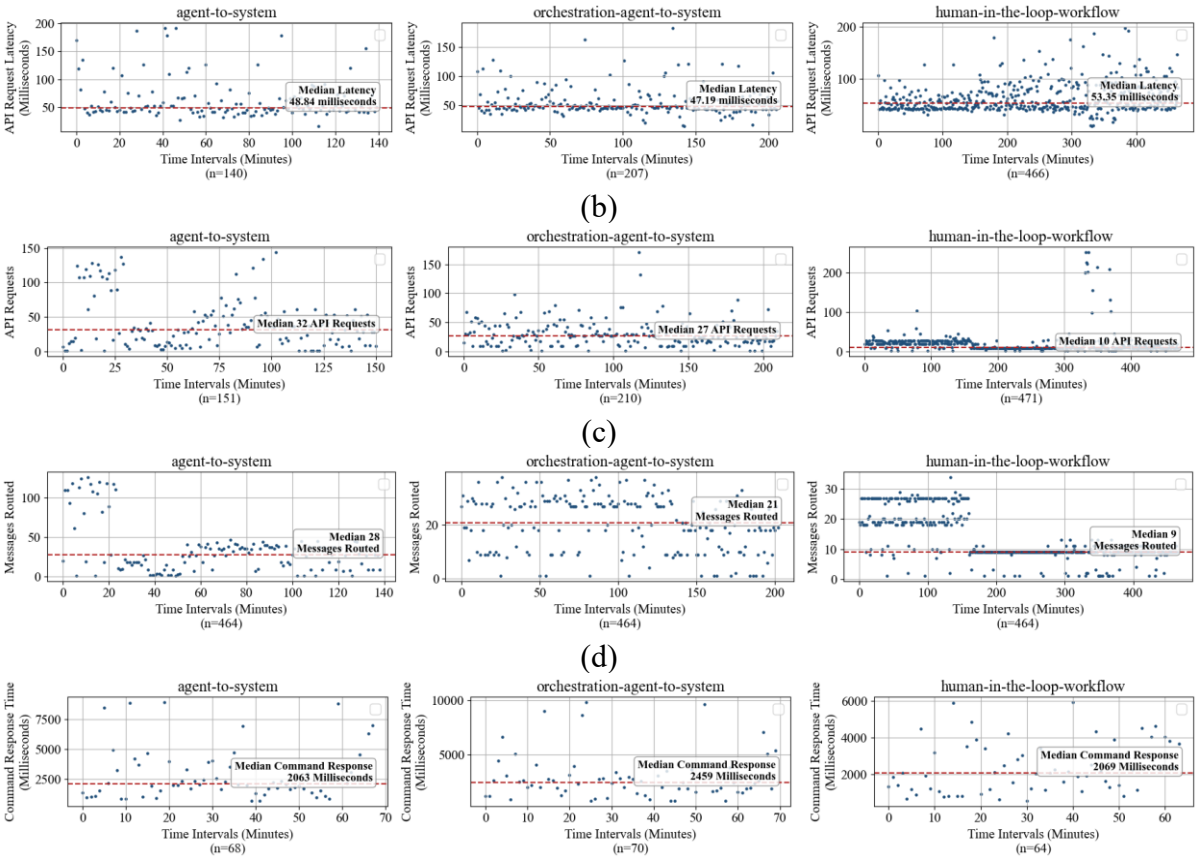


Figure 2. Digital Twins Aspects.

a. ***API Request Latency*****:** Response time in milliseconds for an API request from when request is received by Azure Digital Twins until a success or failure result is returned for digital twin read, write, delete, and query operations which is recorded by cloud. Based on results, it took 47.19 to 53.35 milliseconds on median to read or write a digital twin in cloud. The maximum time elapsed for an API request is 198.22 milliseconds.

b. ***API Requests*****:** Total number of API requests made for read, write, delete, and query operations on digital twins as recorded by cloud. The results suggested that framework performed 10 to 32 API requests on median to read or write digital twins. Moreover, the framework performed 832 total API requests during experimentation. The maximum number of API requests made in a minute is 253 during a human-in-the-loop experiment.

c. ***Messages Routed*****:** Total number of messages routed. This metric can be used to measure total number of digital twin update events performed through telemetry events and command events recorded by cloud. The results suggest that median number of messages routed which indicates number of times digital twins are updated ranged from 9 to 28 messages per minute. During agent-to-system experiments 126 messages are routed in one minute.

d. ***Command Response Time*****:** Time elapsed to process command features through Internet of Things and update them in a digital twin as recorded by framework. Based on findings, framework took a median of 2,063 to 2,459 milliseconds to perform a command feature. However, it also took 31,197 milliseconds to perform some command features such UR Cobot's move-arm command.

35 # Supplementary Figure 3: Internet of Things

(a)

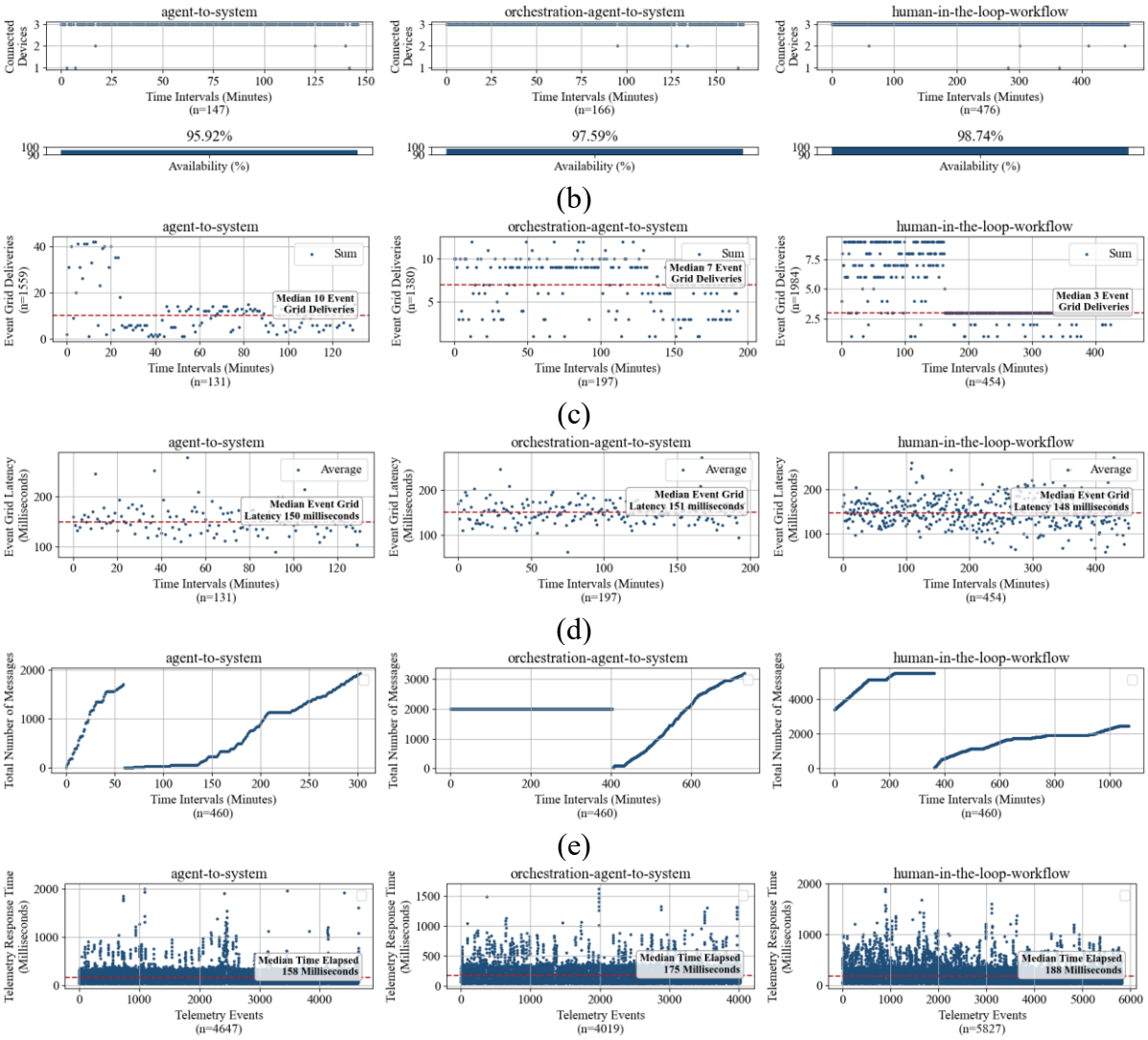


Figure 3. Internet of Things Aspects.

a. ***Connected Devices*:** Number of devices connected to Internet of Things during a given time period. This is used to generate device availability rates during experimentation. In this experiment, this work used three IoT devices namely UR Cobot, Robotiq Gripper, and GOM Inspect 3D Scanner with availability rates ranging from 95.92% to 98.74%. During experiment our framework had total disconnections ranging from 0.006% to 0.02% over 789 minutes of experimental time.

b. ***Event Grid Deliveries*:** Number of IoT events such as telemetry events and command response events delivered to Event Grid during a given time period. During experimentation, this work recorded 4,923 events from Internet of Things to digital twins. On median, this is between 3 and 10 events per minute. During agent-to-machine experiments this work evaluated this without time-delay guardrails as this framework can support 42 updates per minute.

c. ***Event Grid Latency*:** Latency in milliseconds from event generation to publication. During experiments, this work recorded 782 minutes of latency data with median latency ranging from 148 to 151 milliseconds. The minimum time recorded is 58.5 milliseconds while maximum time recorded is 278 milliseconds.

d. ***Total Number of Messages*:** Total number of messages consumed per day. This metric includes both telemetry events command events, and other default events generated by Internet of Things for UR Cobot, Robotiq Gripper, and GOM Inspect 3D Scanner devices. In orchestration-agent-to-system and human-in-the-workflow experiments, overlaps occurred because metrics are not reset at end of the day and evaluations on other aspects such as planning and managing guidelines are also conducted on same day. The incremental growth in message consumption by each device is 3,641 messages for agent-to-system experiments, 3,211 messages for orchestration-agent-to-system experiments, and 4,570 messages for human-in-the-loop workflows.

e. ***Telemetry Response Times*:** Time elapsed to process a telemetry event and send it to the respective digital twins. This framework processed 14,493 of these update events. The framework took a mean of 158 to 188 milliseconds to update digital twins with response times ranging from 35 to 1,999 milliseconds.

36 # Supplementary Figure 4: Physical Twin

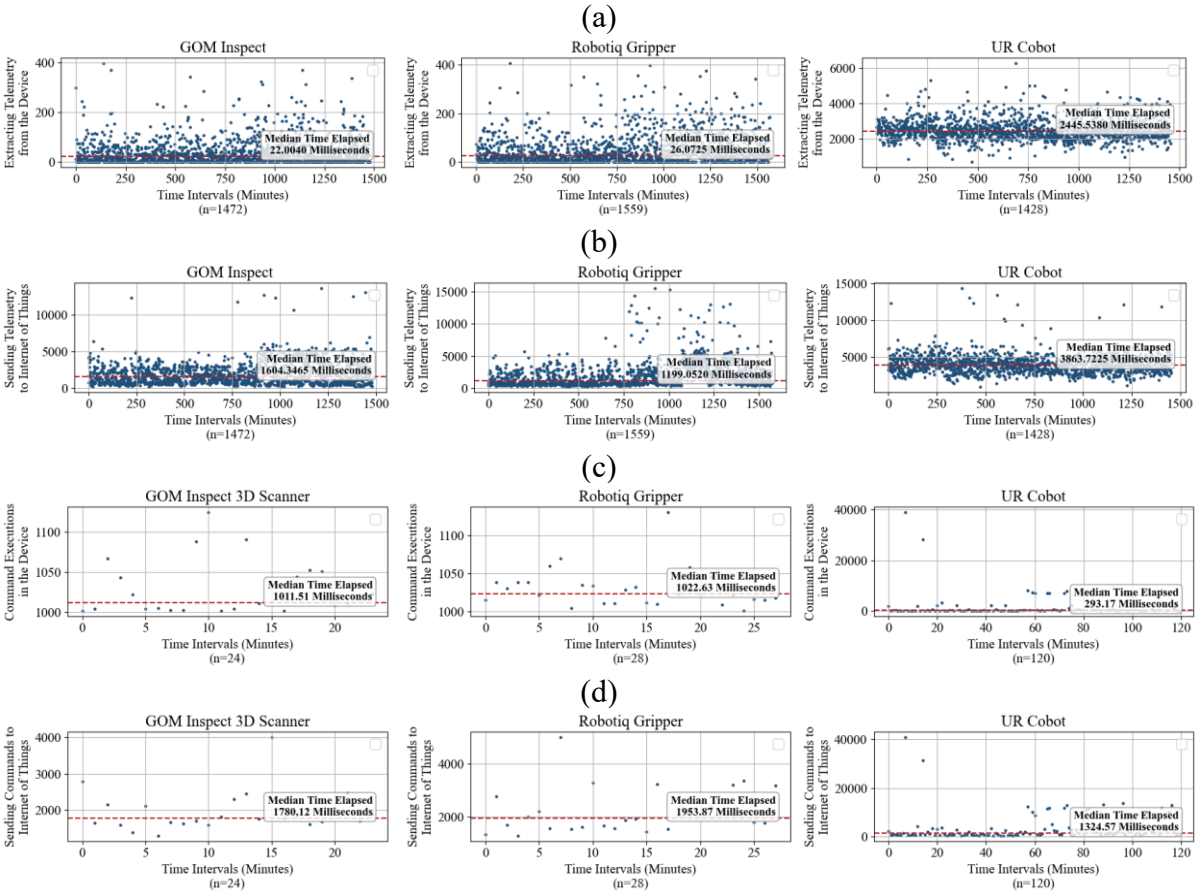


Figure 4. Physical Twin Aspects.

a. ***Extracting Telemetry from Device*:** Time elapsed to extract a telemetry event from a device. The framework extracted 4,505 telemetry events from GOM Inspect 3D Scanner 1,479 telemetry events, Robotiq Gripper 1,570 telemetry events, and UR Cobot 1,456 telemetry events. The median time for GOM Inspect 3D Scanner is 22 milliseconds with a maximum of 398.97 milliseconds to extract a telemetry event. For Robotiq Gripper, median is 26.07 milliseconds with a maximum of 406.86 milliseconds. However, for UR Cobot it

recorded a median of 2,445.53 milliseconds, a minimum of 681.54 milliseconds, and a maximum of 6,299.78 milliseconds.

b. ***Sending Telemetry to Internet of Things*****:** Total time elapsed from extracting telemetry from a device to sending that telemetry event to Internet of Things and receiving a success or failure response. On median, GOM Inspect 3D Scanner took 1,604.34 milliseconds with a minimum of 203.52 milliseconds and a maximum of 13,669.62 milliseconds. For Robotiq Gripper, median is 1,199.05 milliseconds with a minimum of 176.28 milliseconds and a maximum of 15,575.38 milliseconds. For UR Cobot, median is 3,863.72 milliseconds with a minimum of 914.03 milliseconds and a maximum of 14,394.25 milliseconds.

c. ***Command Executions in Device*****:** Amount of time it took to execute a command on a device. The framework executed 172 command events, 24 commands by GOM Inspect 3D Scanner, 28 commands by Robotiq Gripper, and 120 commands by UR Cobot. As a median, GOM took 1,011.51 milliseconds with a minimum of 1,001.27 milliseconds and a maximum of 1,125.28 milliseconds. For UR Cobot, median is 293.17 milliseconds with a minimum of 5.19 milliseconds and a maximum of 39,130.75 milliseconds.

d. ***Sending Commands to Internet of Things*****:** Total time elapsed from executing a command to sending response payload to Internet of Things and receiving a success or failure response. On median, GOM took 1,780.12 milliseconds with a minimum of 1,290.81 milliseconds and a maximum of 4,008.14 milliseconds. For Robotiq Gripper, median is 1,953.87 milliseconds with a minimum of 1,271.74 milliseconds and a maximum of 5,018.34 milliseconds. The UR Cobot had a median of 1,324.57 milliseconds with a minimum of 218.08 milliseconds and a maximum of 40,934.61 milliseconds.

37