

Runtime Deployment, Management and Monitoring of Web of Things Systems

Miguel Romero Karam
miguel.romero@tum.de

Embedded Systems and Internet of Things

Department of Electrical and Computer Engineering

Munich | December 14, 2020



1. Introduction

2. Approach

3. Evaluation

4. Conclusion

1. Introduction

1.1. Motivation

1.2. WoT Thing Description

1.3. WoT System Description

1.4. Problem Statement

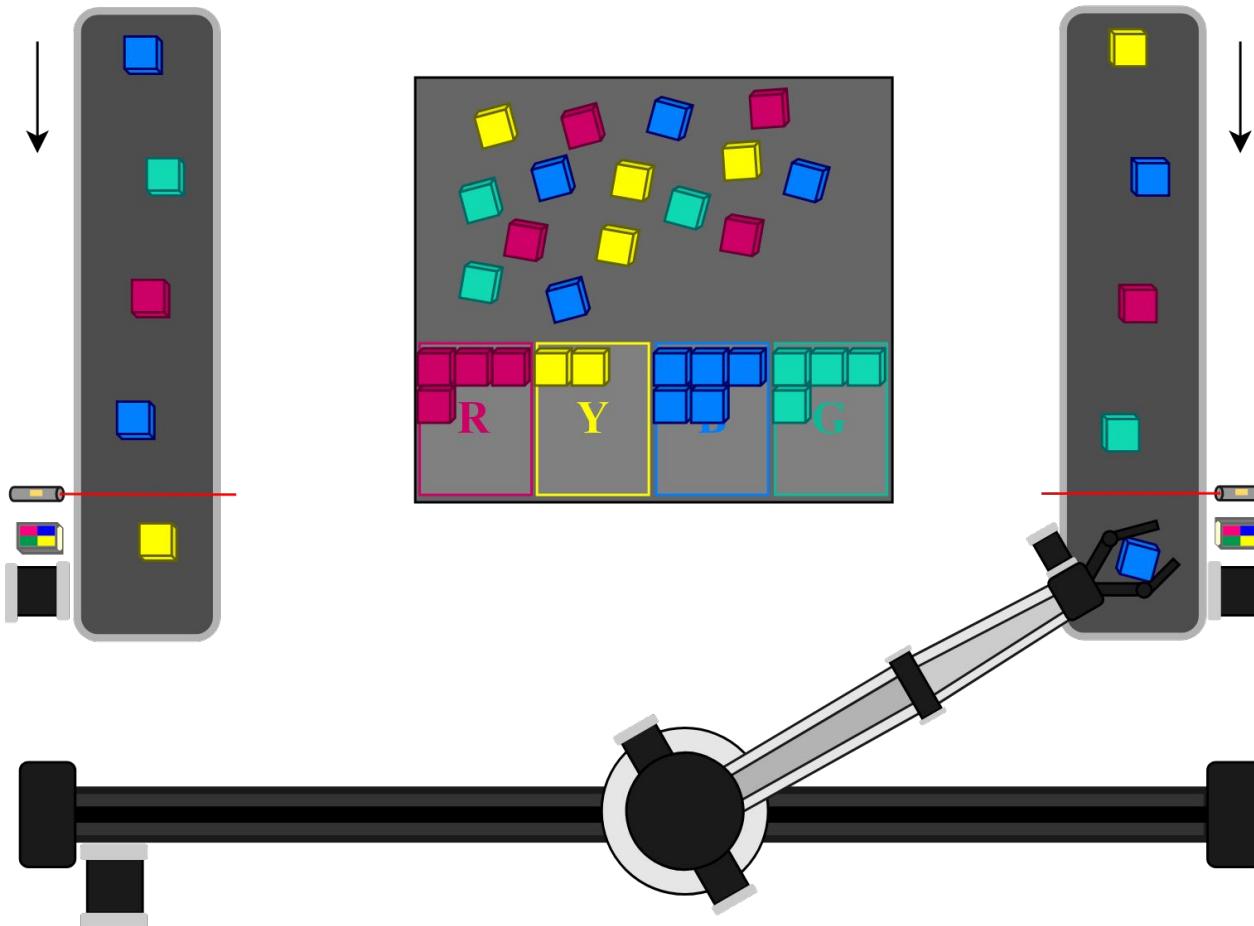
2. Approach

3. Evaluation

4. Conclusion

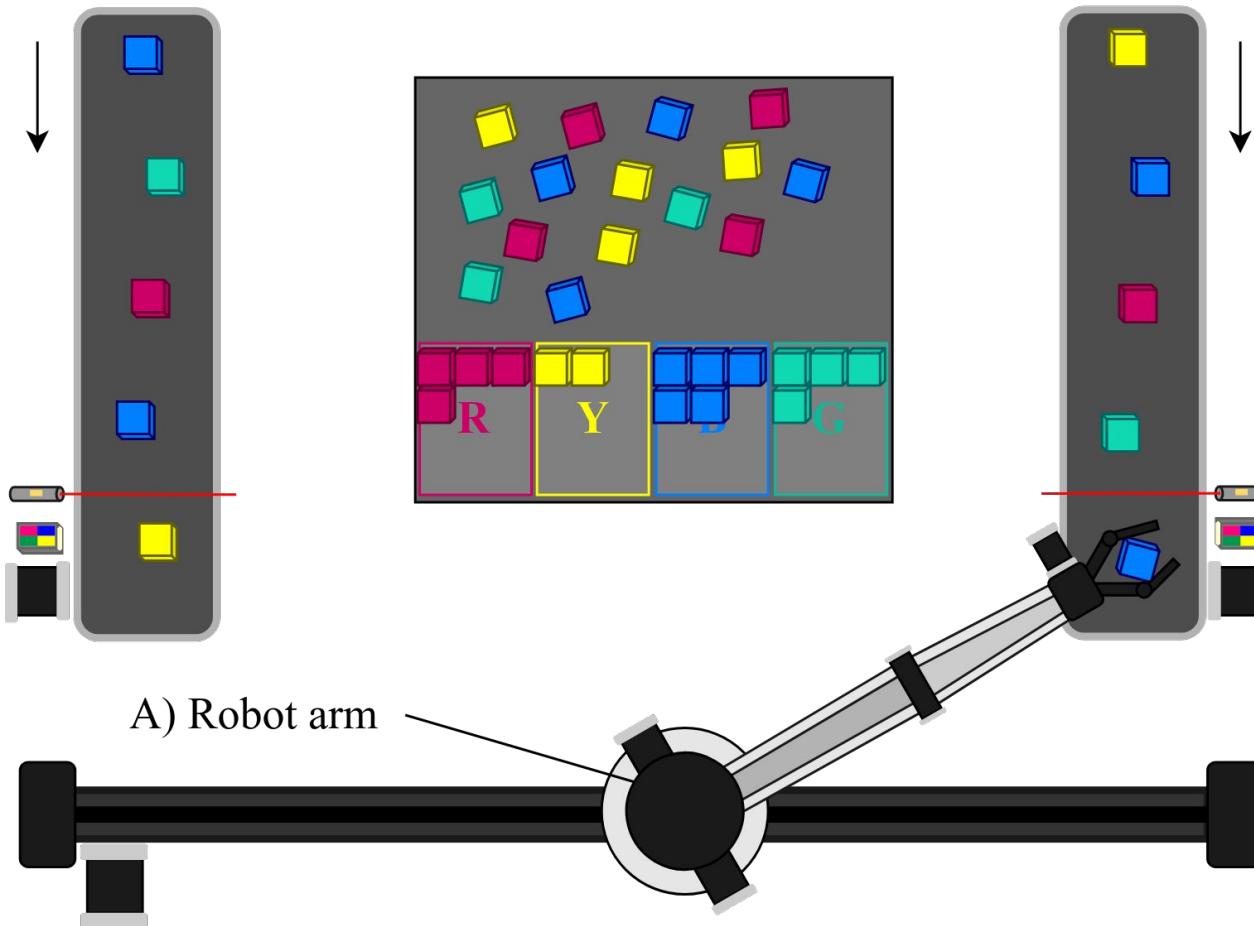
Motivation

Industrial Automation System



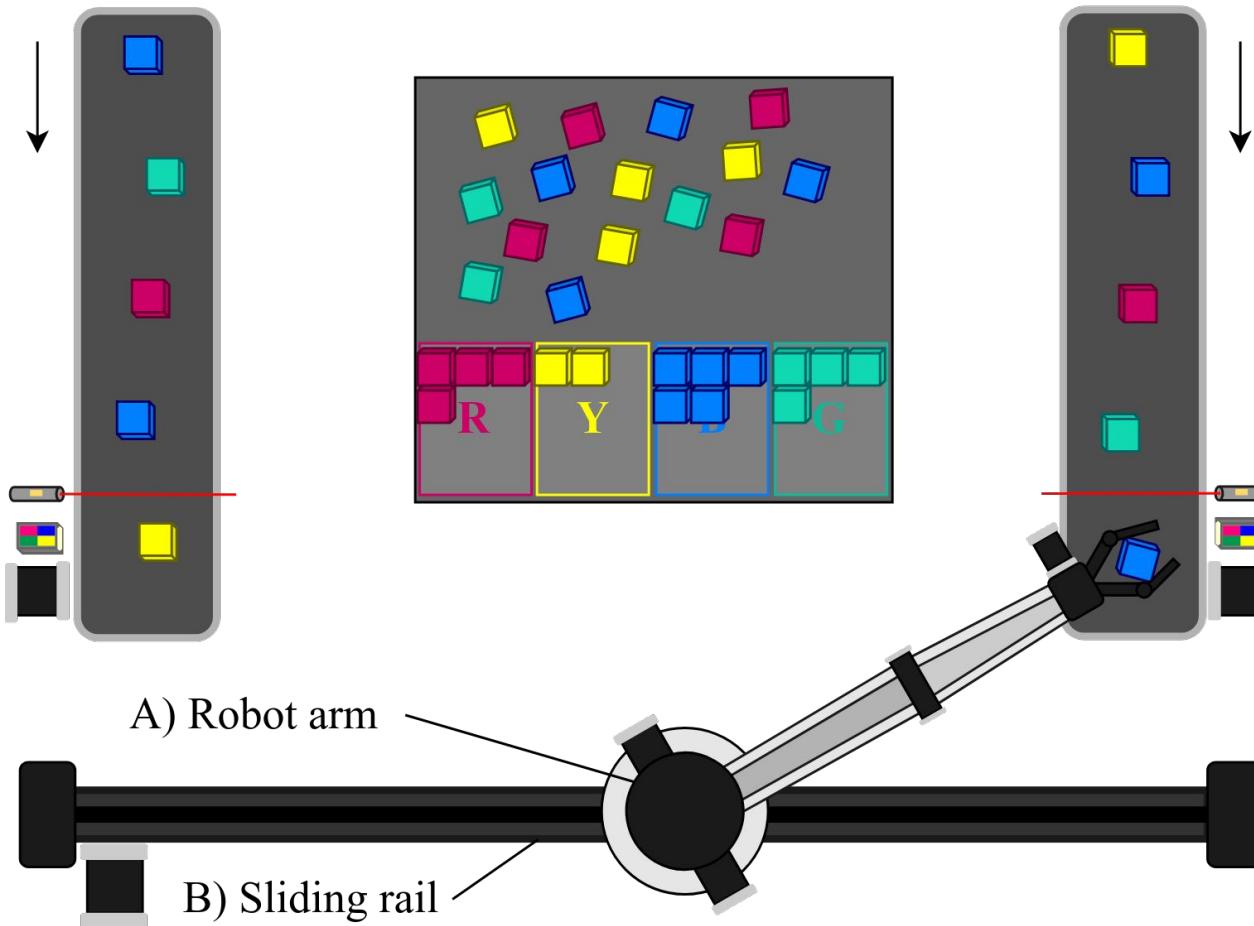
Motivation

Industrial Automation System



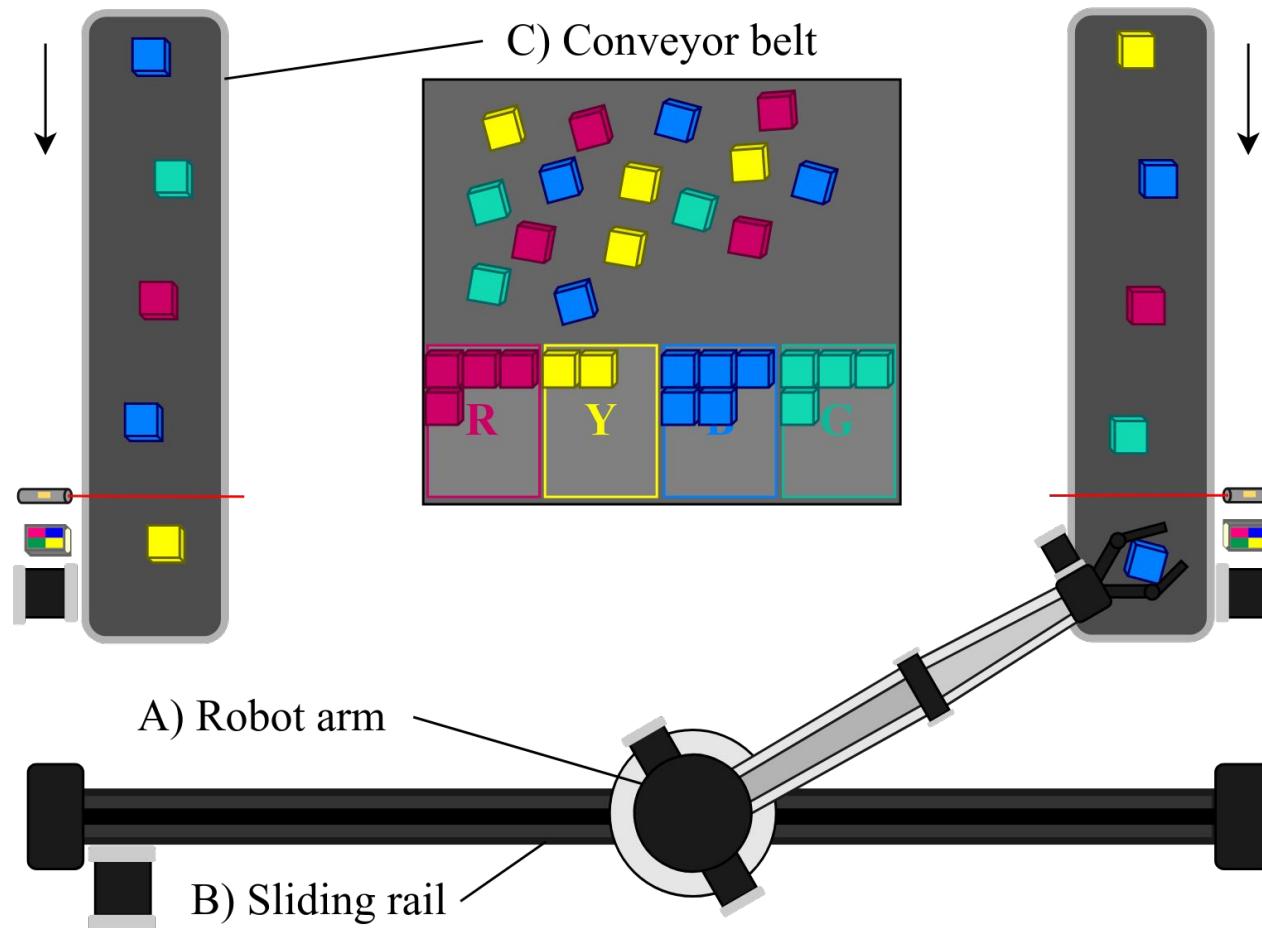
Motivation

Industrial Automation System



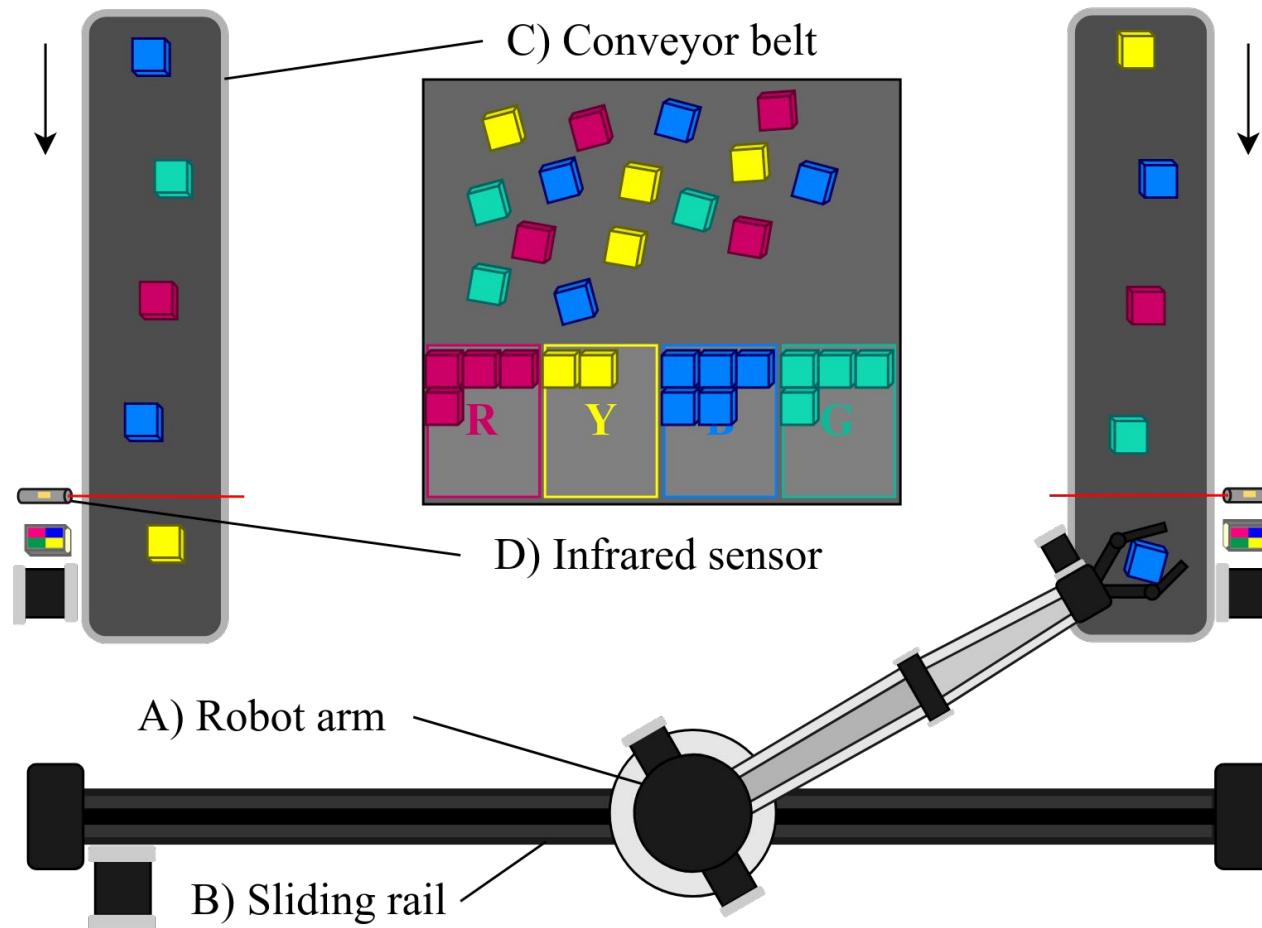
Motivation

Industrial Automation System



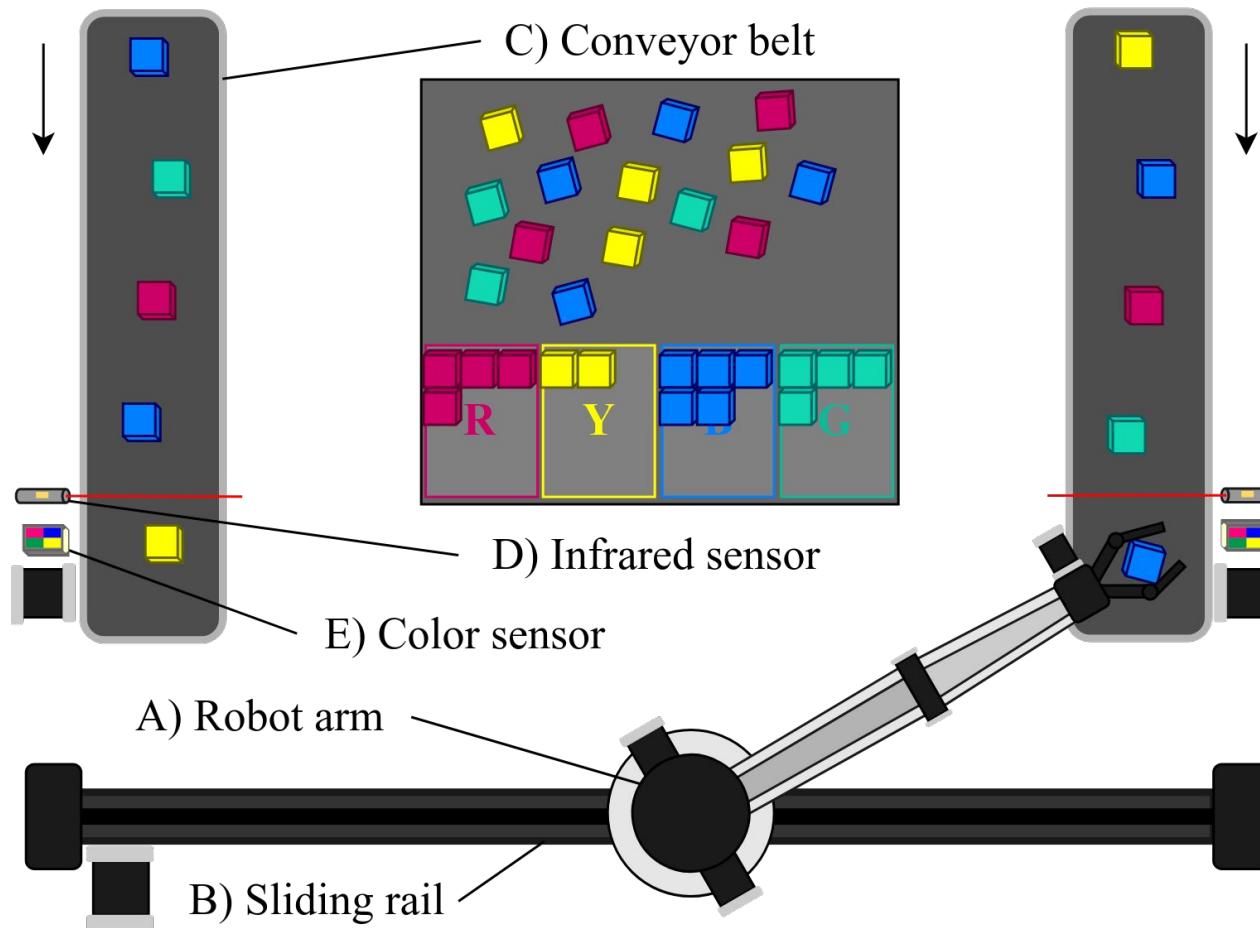
Motivation

Industrial Automation System



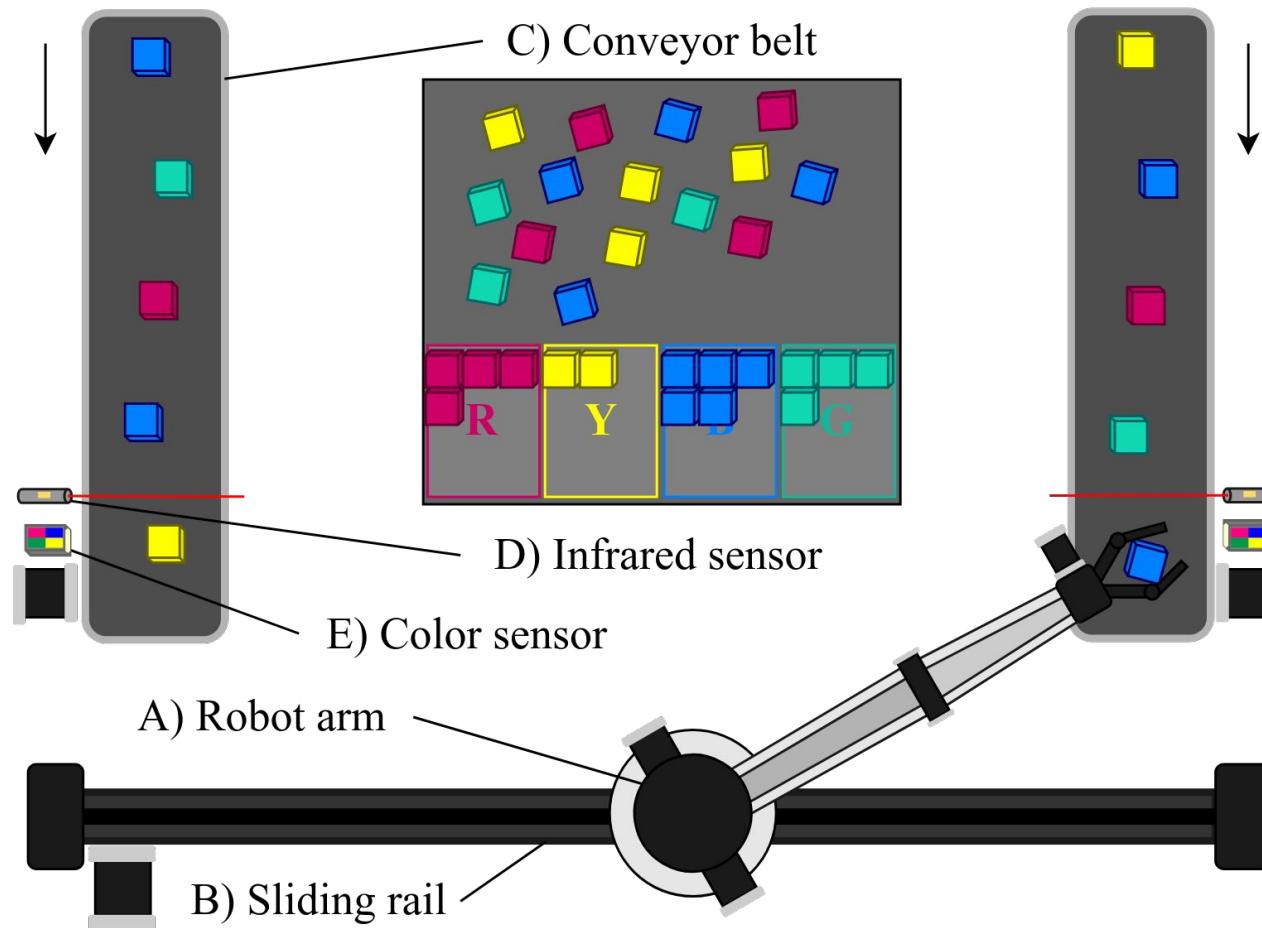
Motivation

Industrial Automation System



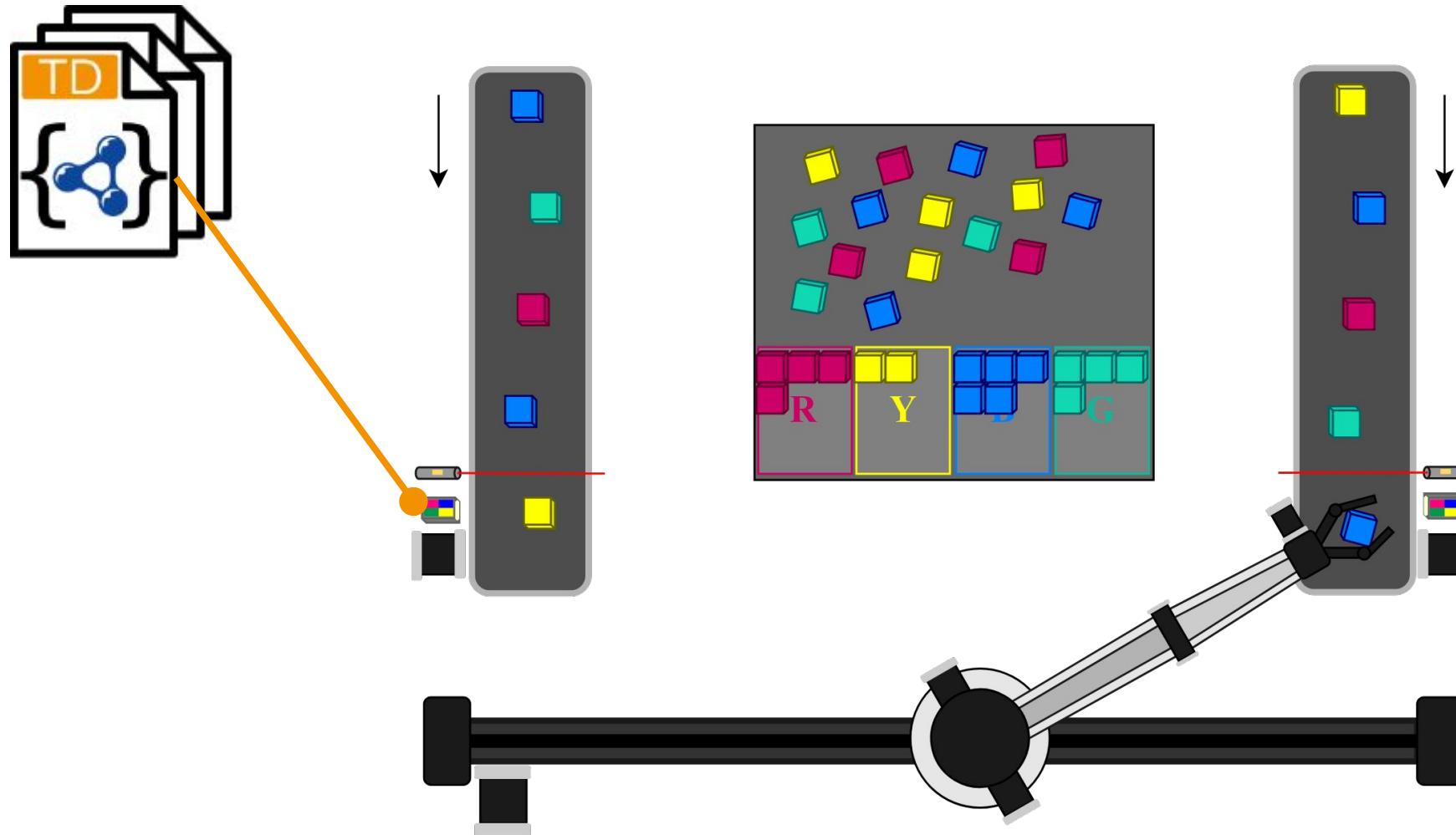
Motivation

Industrial Automation System



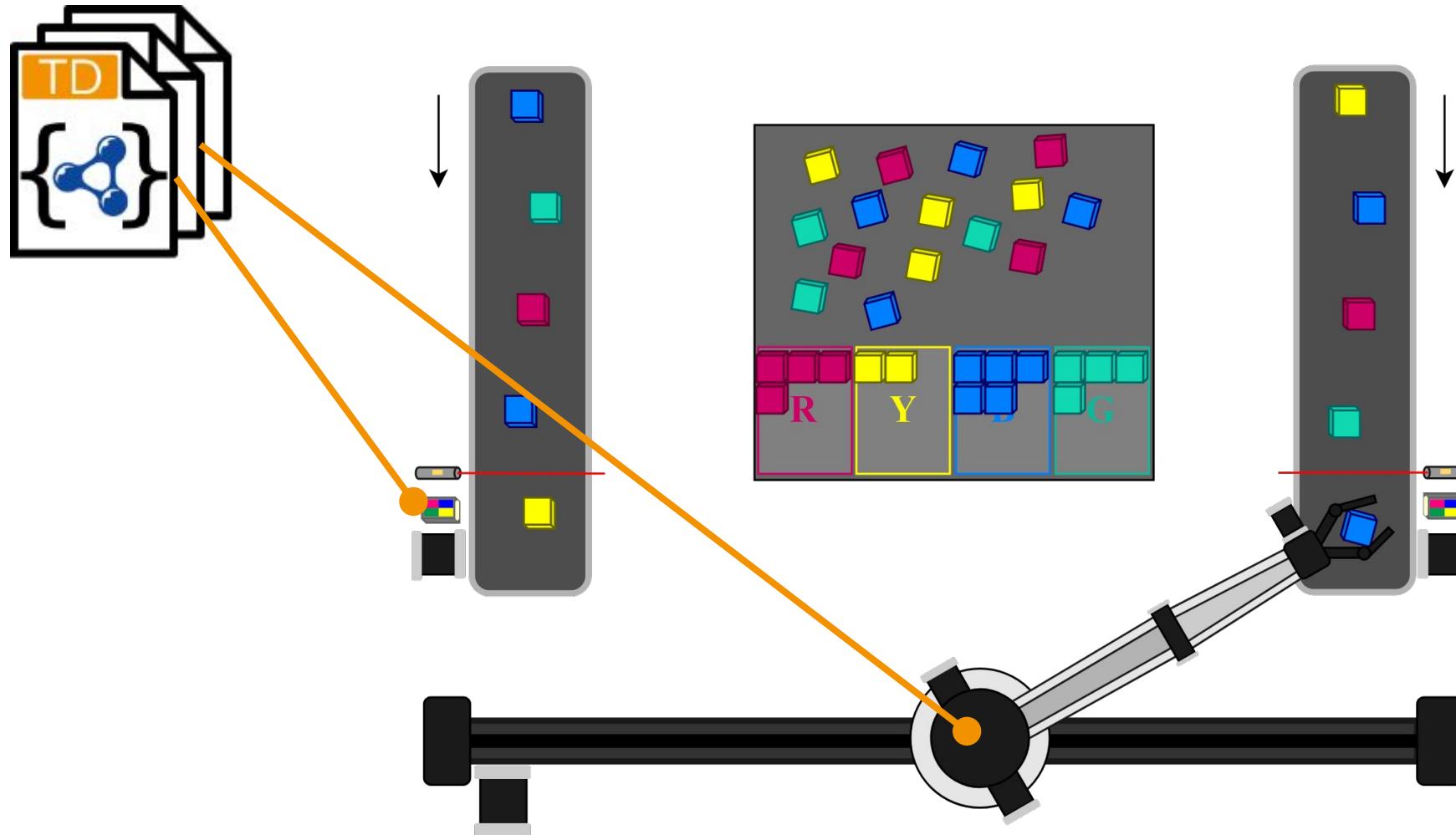
Motivation

Industrial Automation System



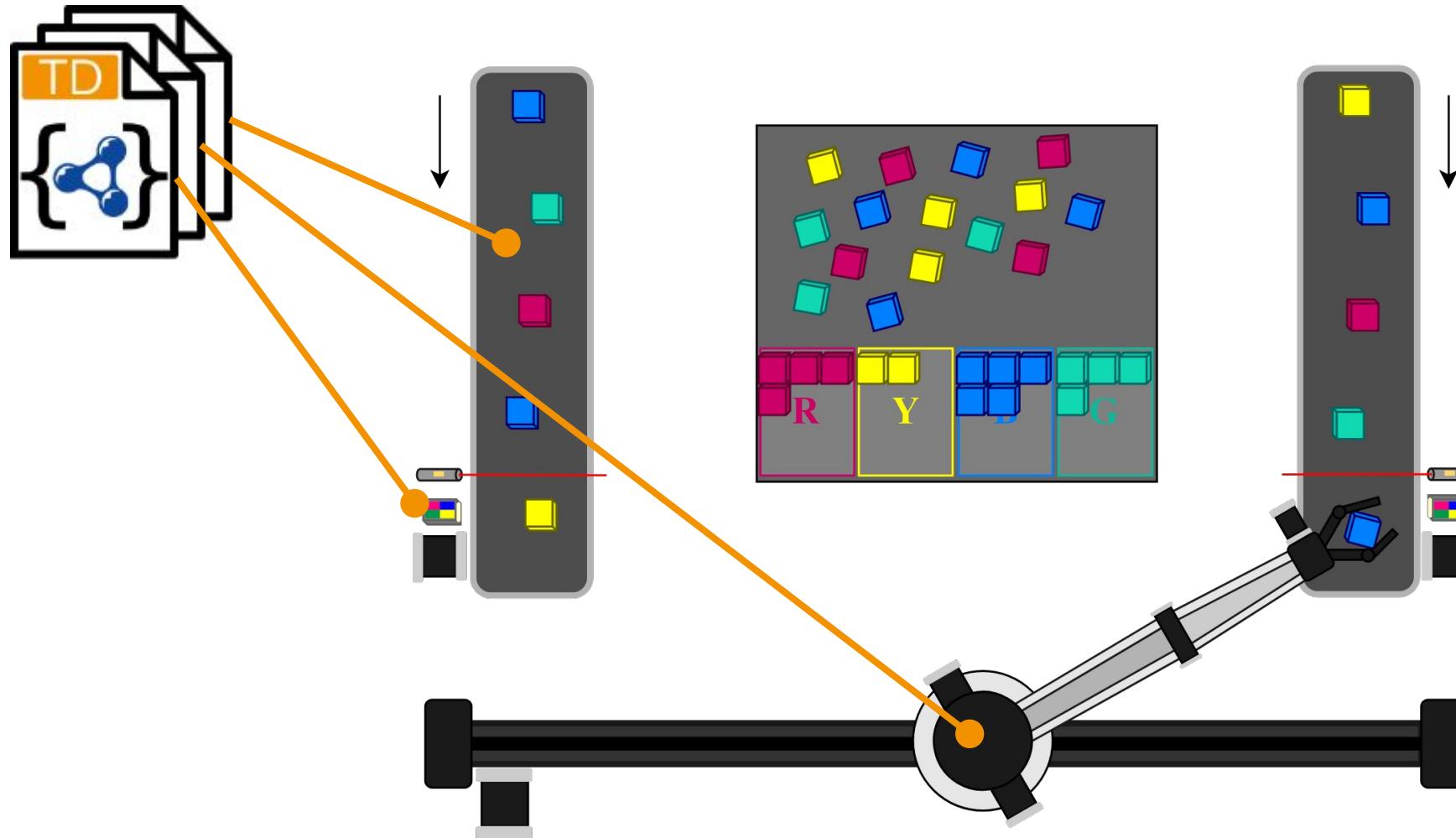
Motivation

Industrial Automation System



Motivation

Industrial Automation System



1. Introduction

1.1. Motivation

1.2. WoT Thing Description

1.3. WoT System Description

1.4. Problem Statement

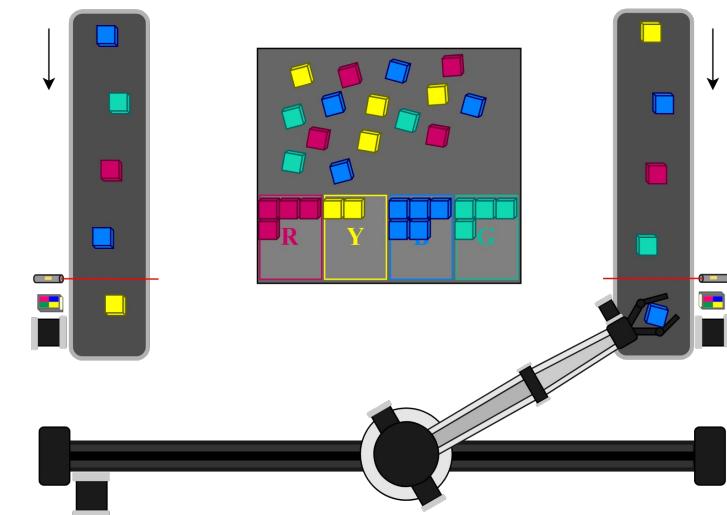
2. Approach

3. Evaluation

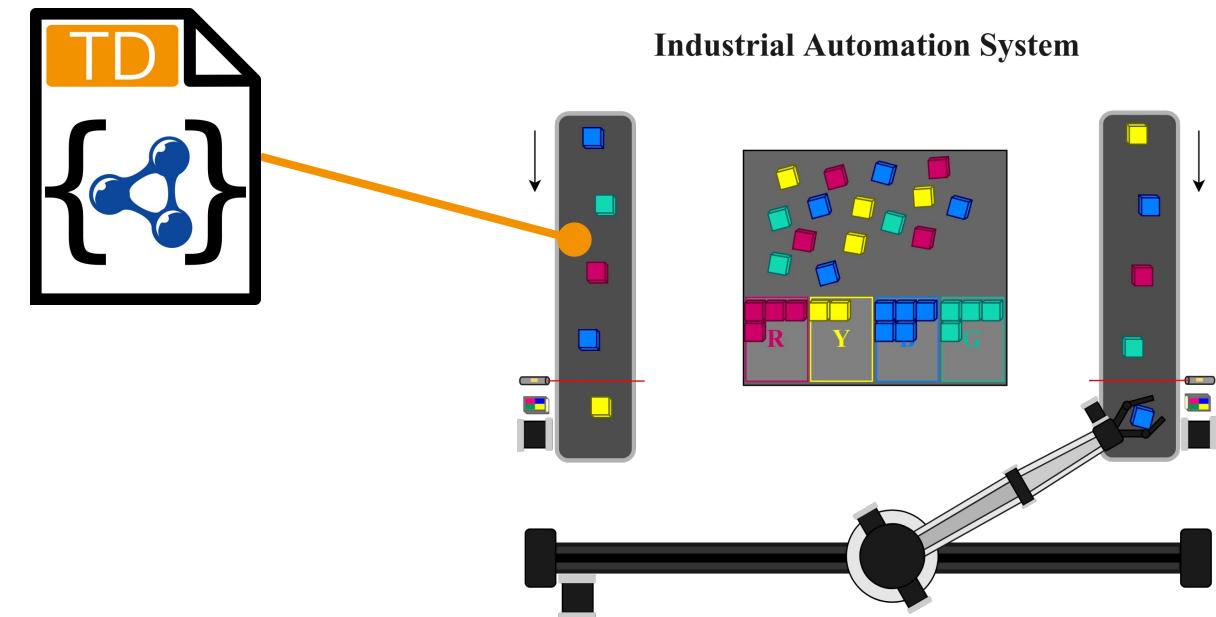
4. Conclusion

WoT Thing Description

Industrial Automation System



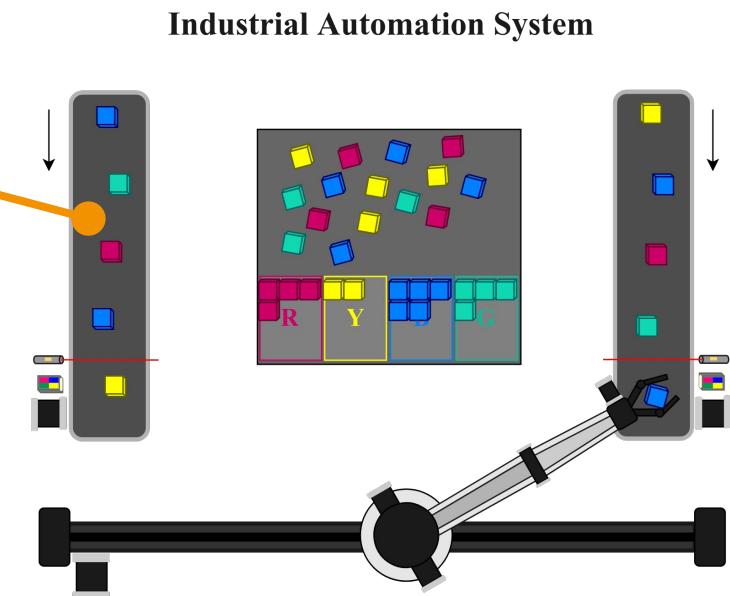
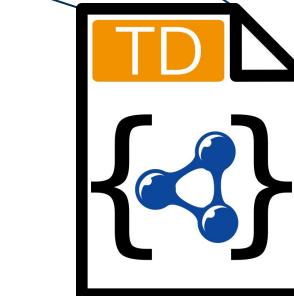
WoT Thing Description



WoT Thing Description

TD of Conveyor Belt

```
"name": "LeftConveyorBelt",
"@type": "conveyor-belt",
"properties": [
    "speed": {
        "type": "number",
        "minimum": -100.0,
        "maximum": 100.0
    },
    "status": {
        "type": "string",
        "enum": [ "on", "off" ],
    }
],
"actions": [
    "start": {...},
    "stop": {...},
],
"events": [
    "emergencystop": {...},
],
"security": [...]
```



1. Introduction

- 1.1. Motivation
- 1.2. WoT Thing Description
- 1.3. WoT System Description**
- 1.4. Problem Statement

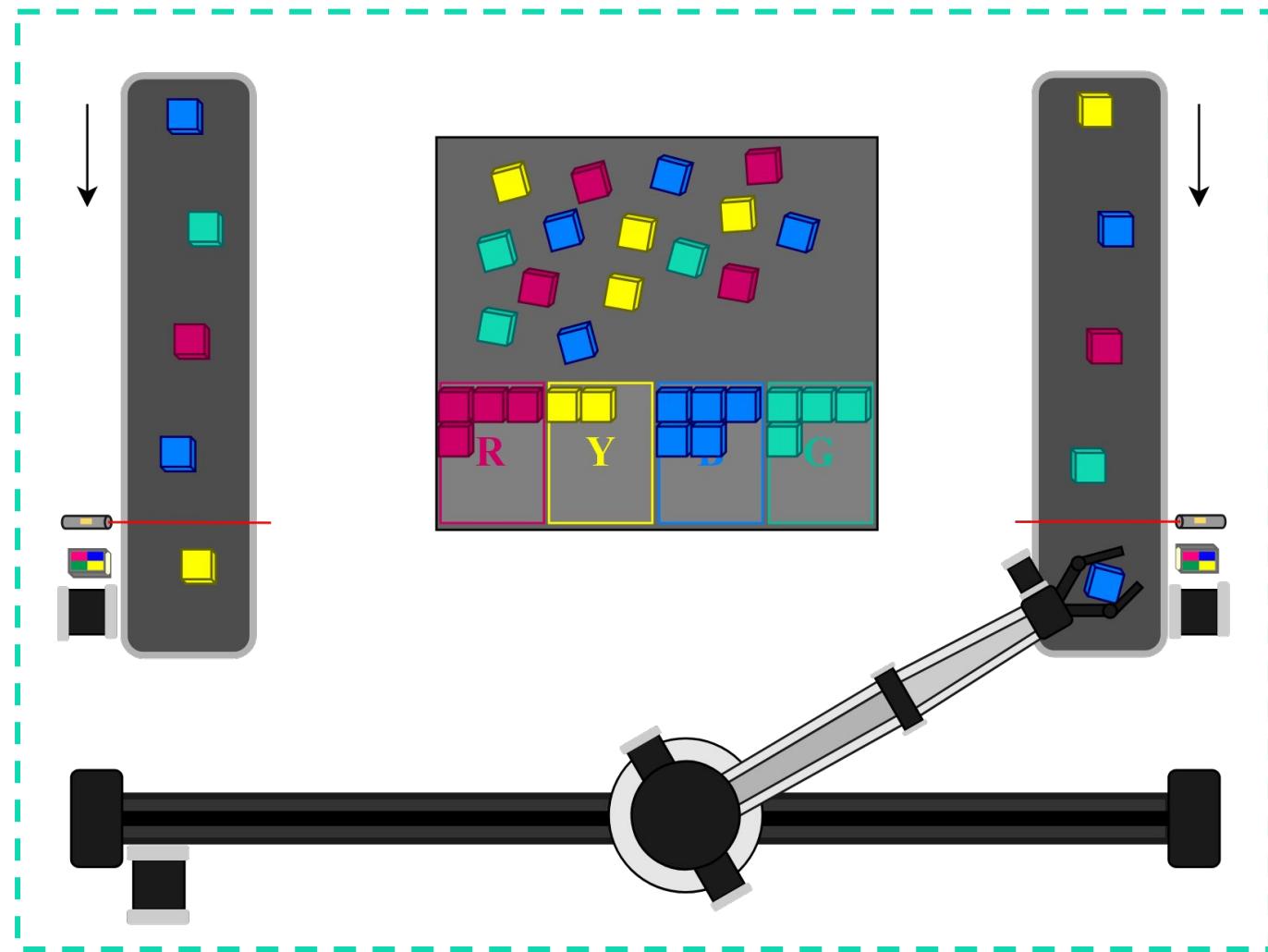
2. Approach

3. Evaluation

4. Conclusion

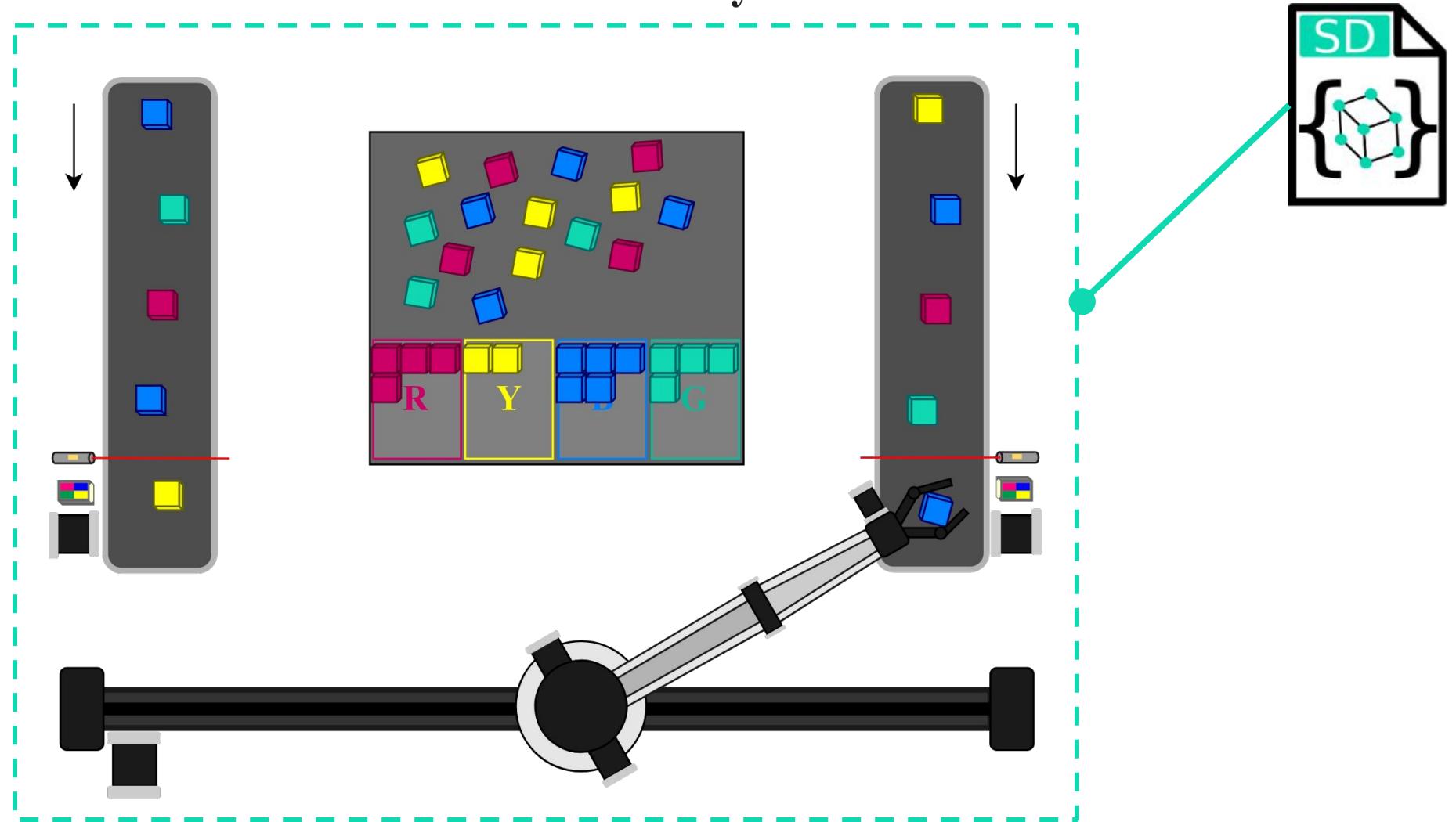
WoT System Description

Industrial Automation System



WoT System Description

Industrial Automation System



1. Introduction

- 1.1. Motivation
- 1.2. WoT Thing Description
- 1.3. WoT System Description
- 1.4. Problem Statement

2. Approach

3. Evaluation

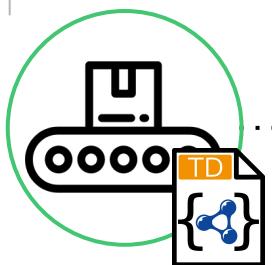
4. Conclusion

Problem Statement



Thing Description

Allows modelling simple and complex **standalone Things**

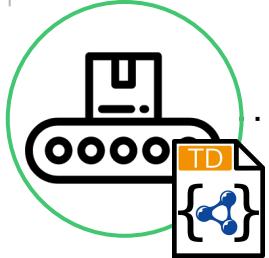


Problem Statement



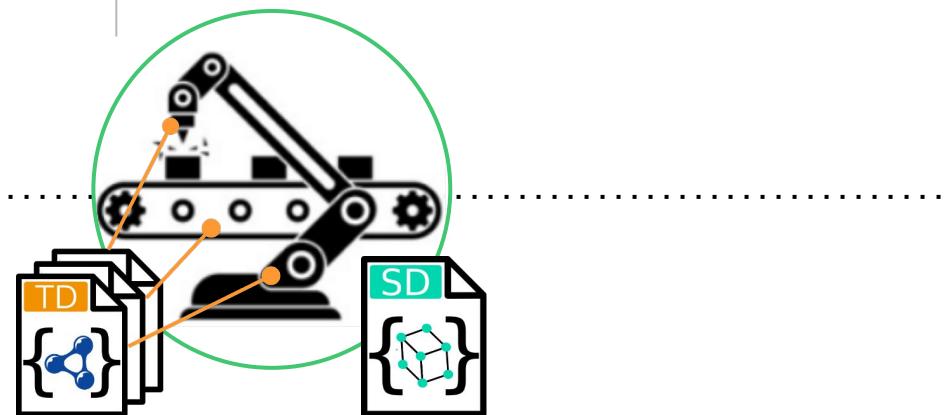
Thing Description

Allows modelling simple and complex **standalone Things**



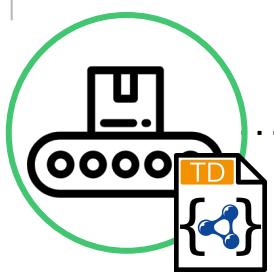
System Description

Allows describing interaction logic between Things, in order **to create WoT Systems**

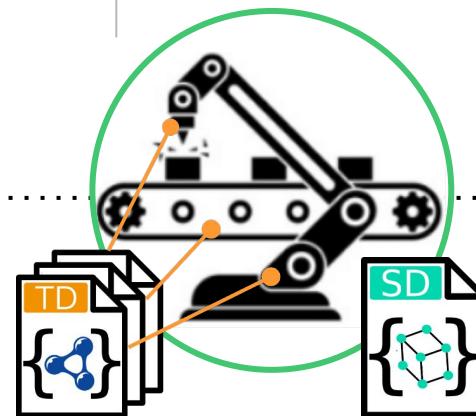


Problem Statement

 **Thing Description**
Allows modelling simple and complex **standalone Things**



 **System Description**
Allows describing interaction logic between Things, in order **to create WoT Systems**



 **Development Tools**
For **deployment, management, and monitoring** of WoT Systems



1. Introduction

2. Approach

2.1. Overview

2.2. Methodology

2.3. Execution Environment and Control

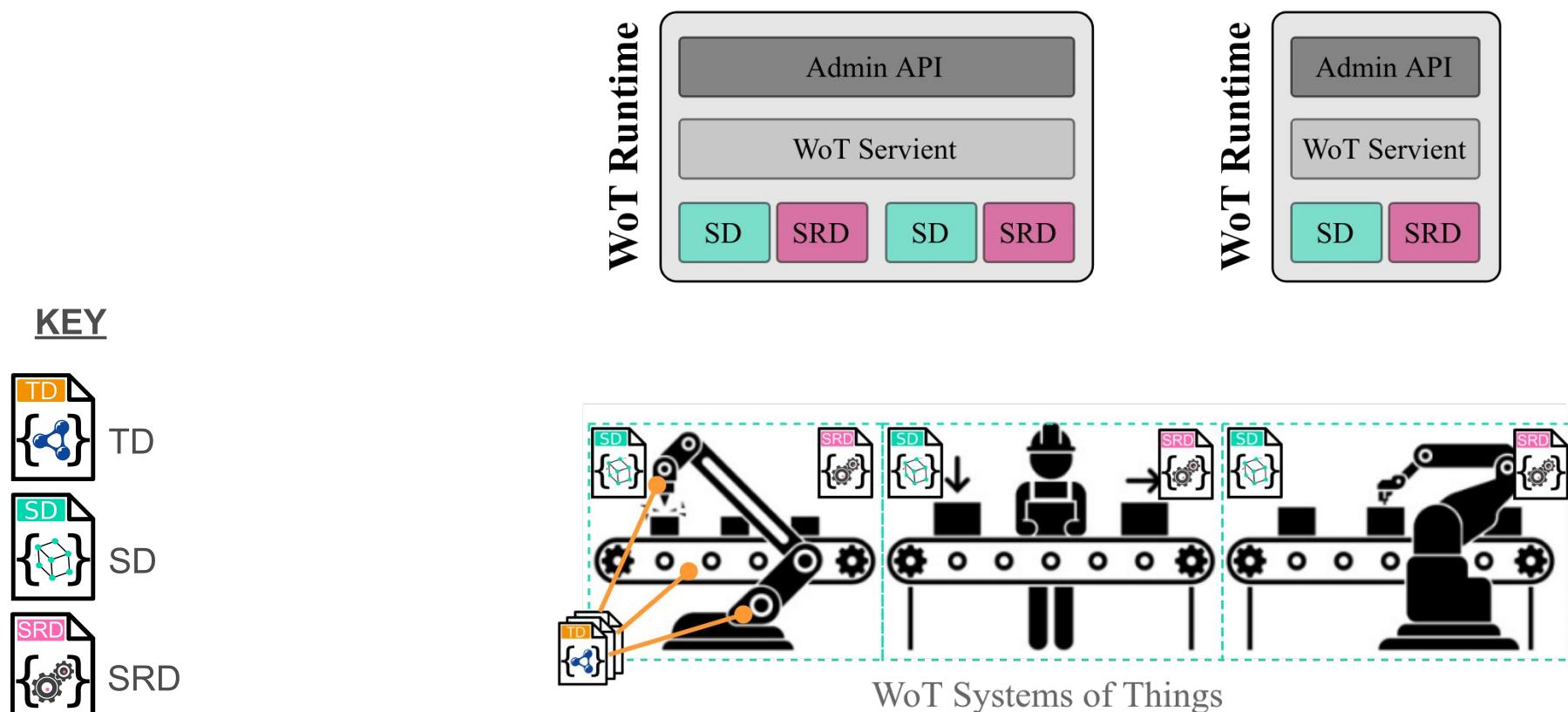
2.4. WoT Runtime UI

3. Evaluation

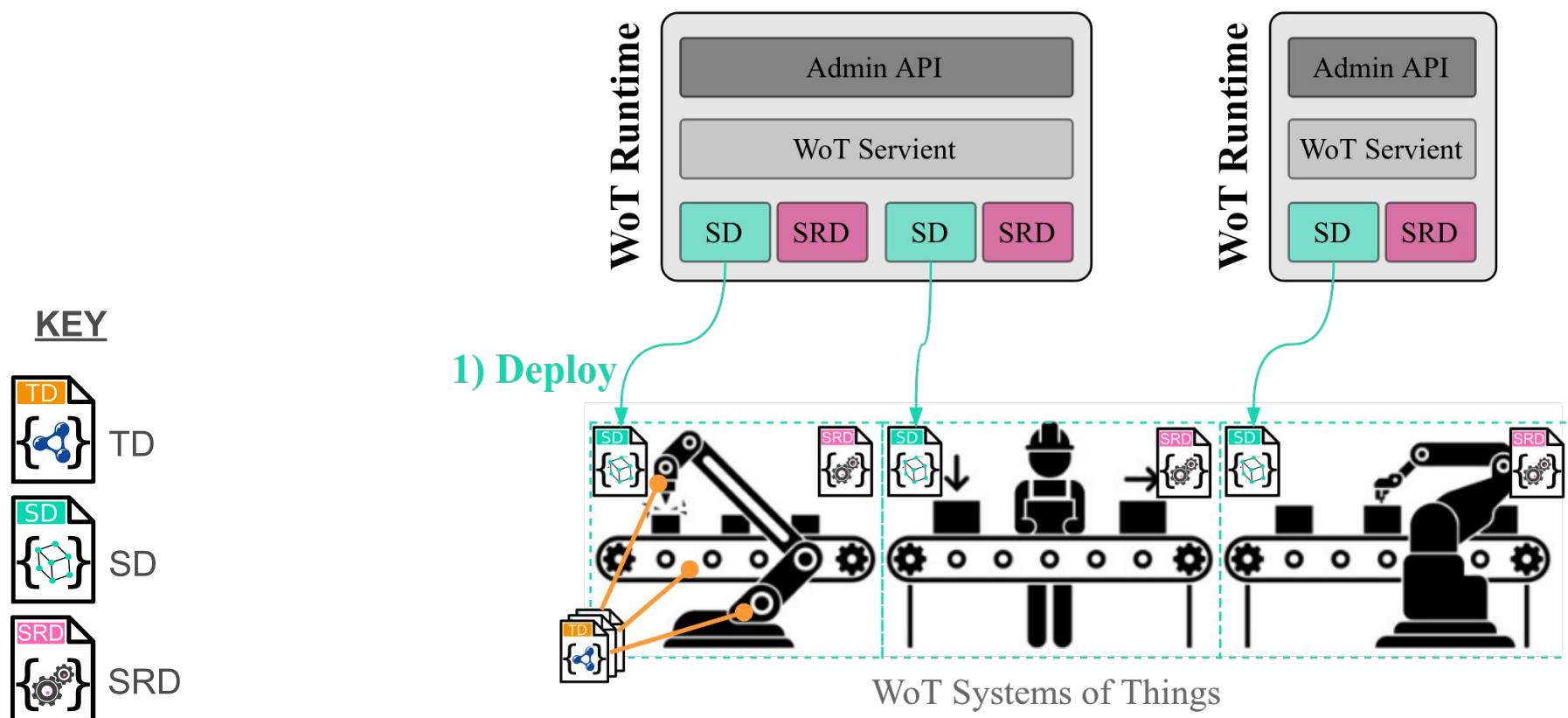
4. Conclusion

Approach

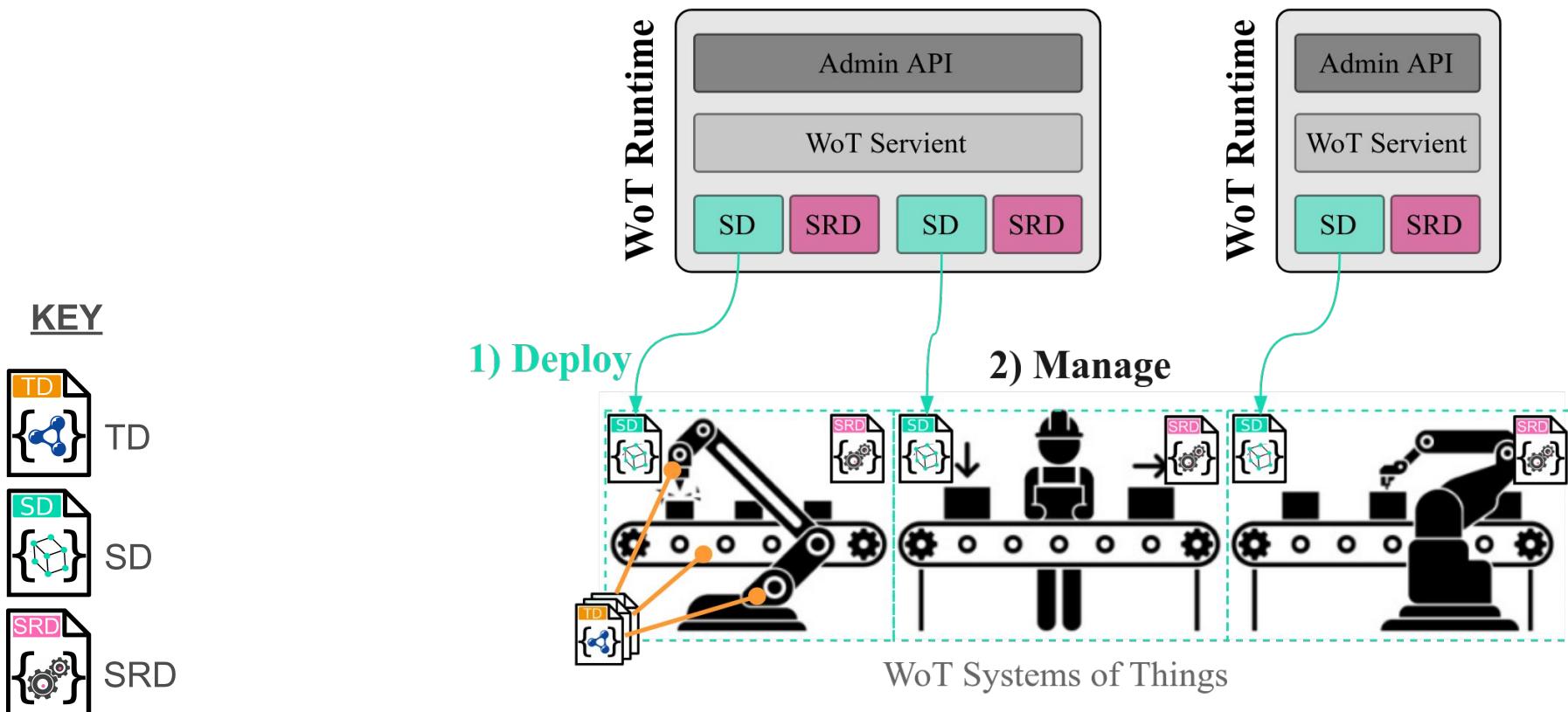
Overview



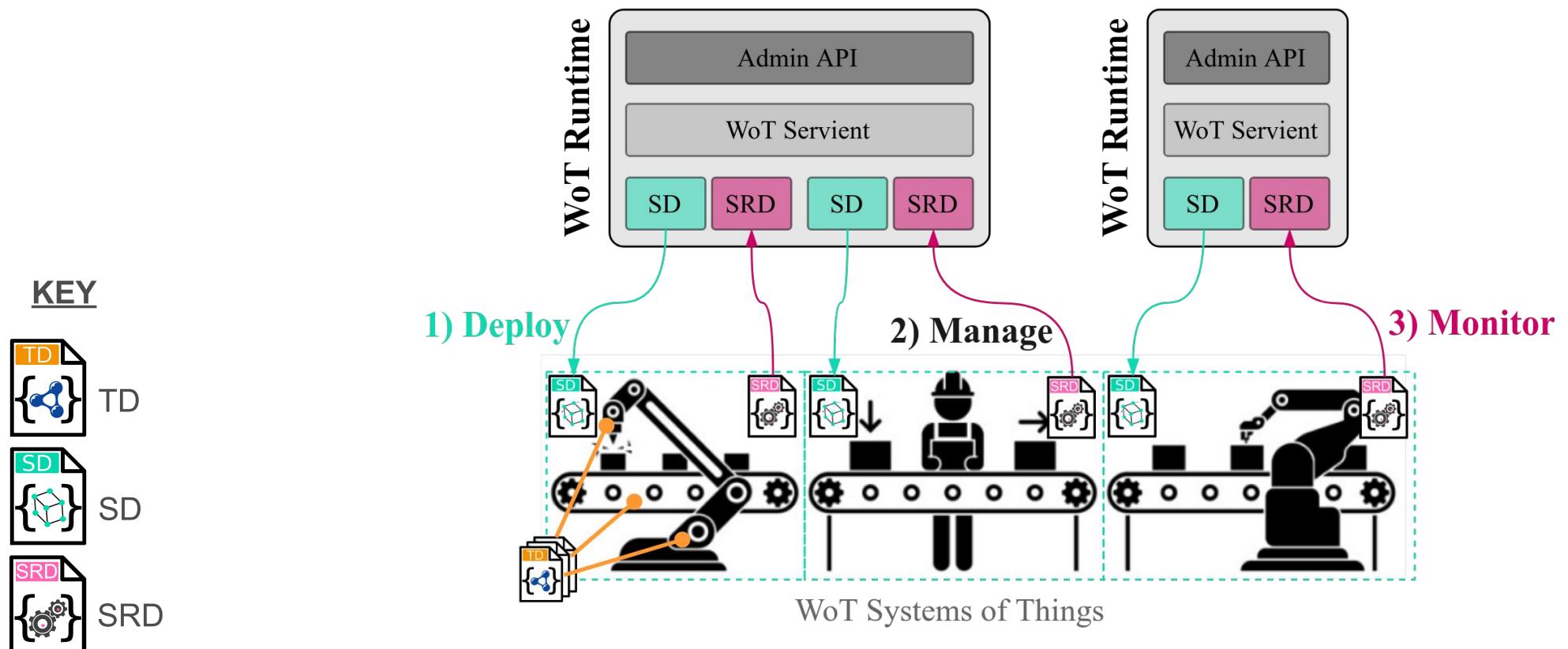
Overview



Overview

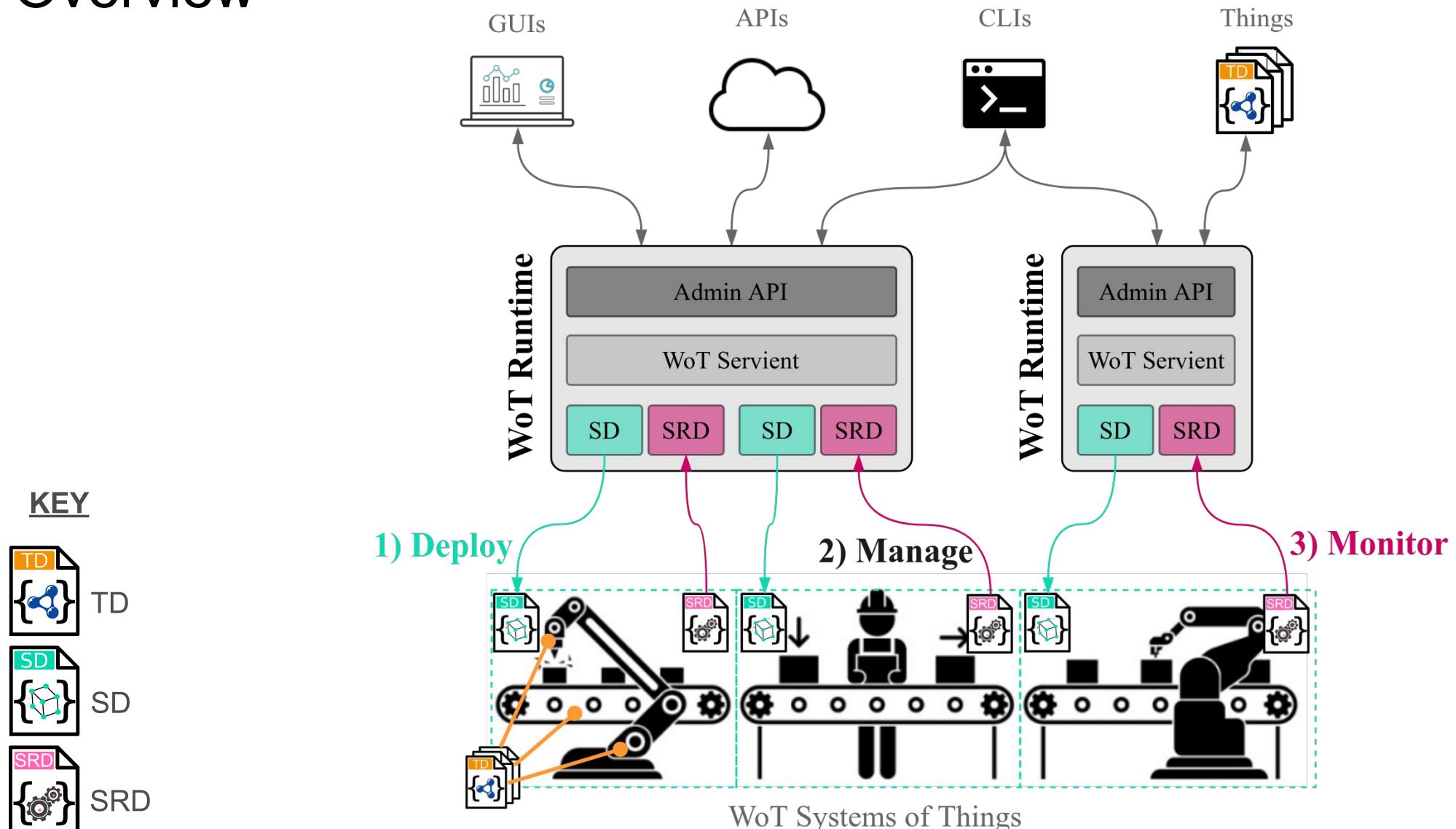


Overview



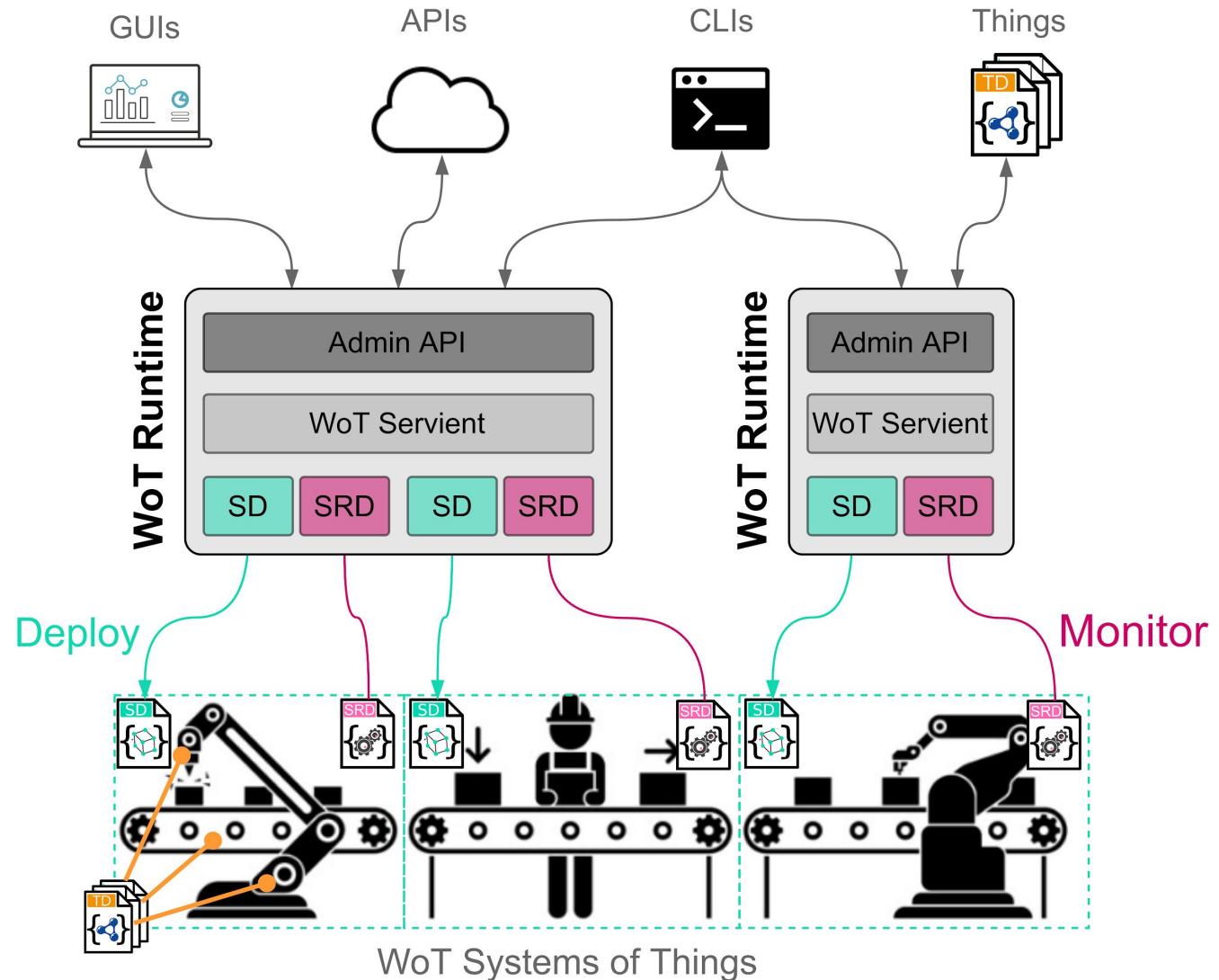
Approach

Overview



Approach

Overview



1. Introduction

2. Approach

2.1. Overview

2.2. Methodology

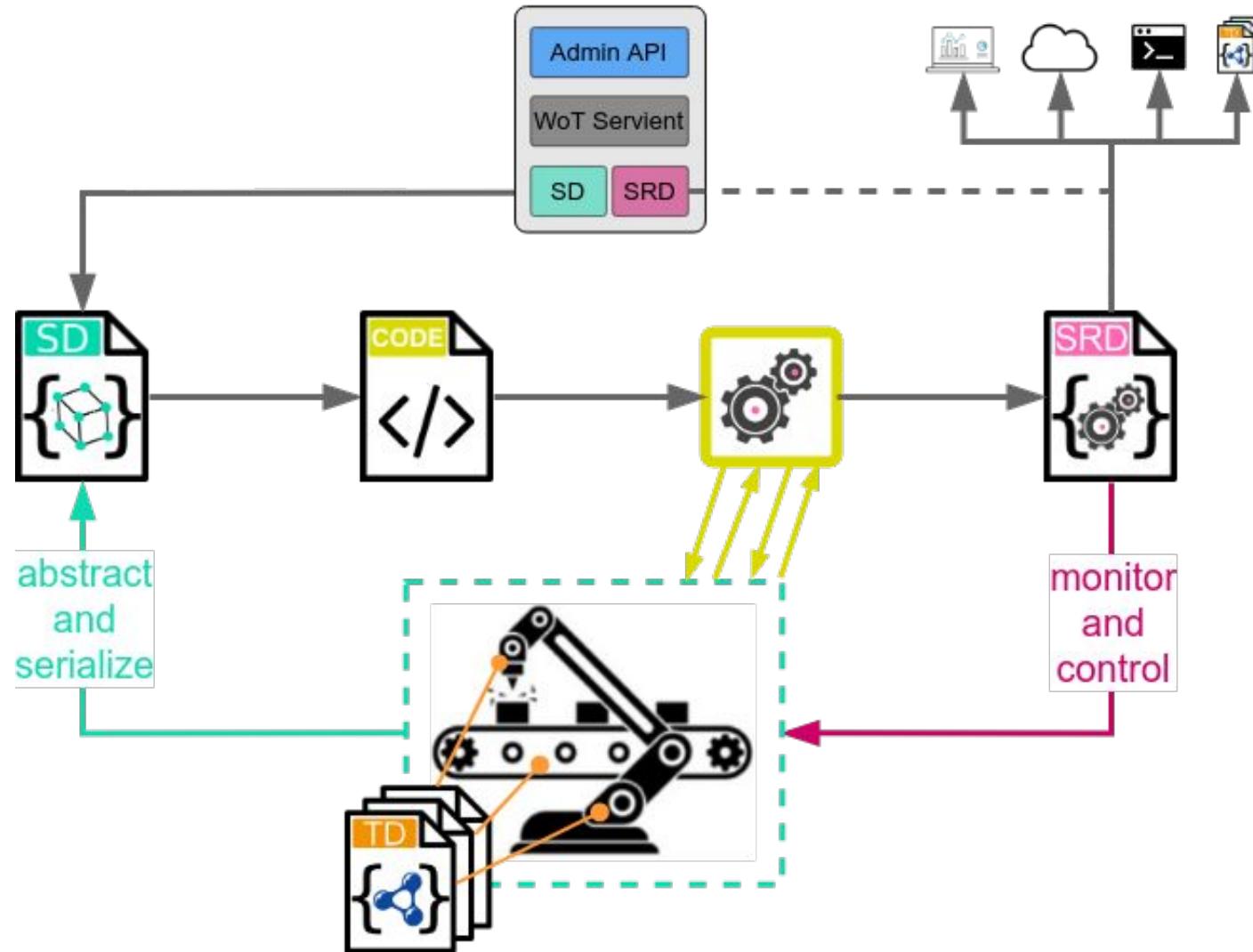
2.3. Execution Environment and Control

2.4. WoT Runtime UI

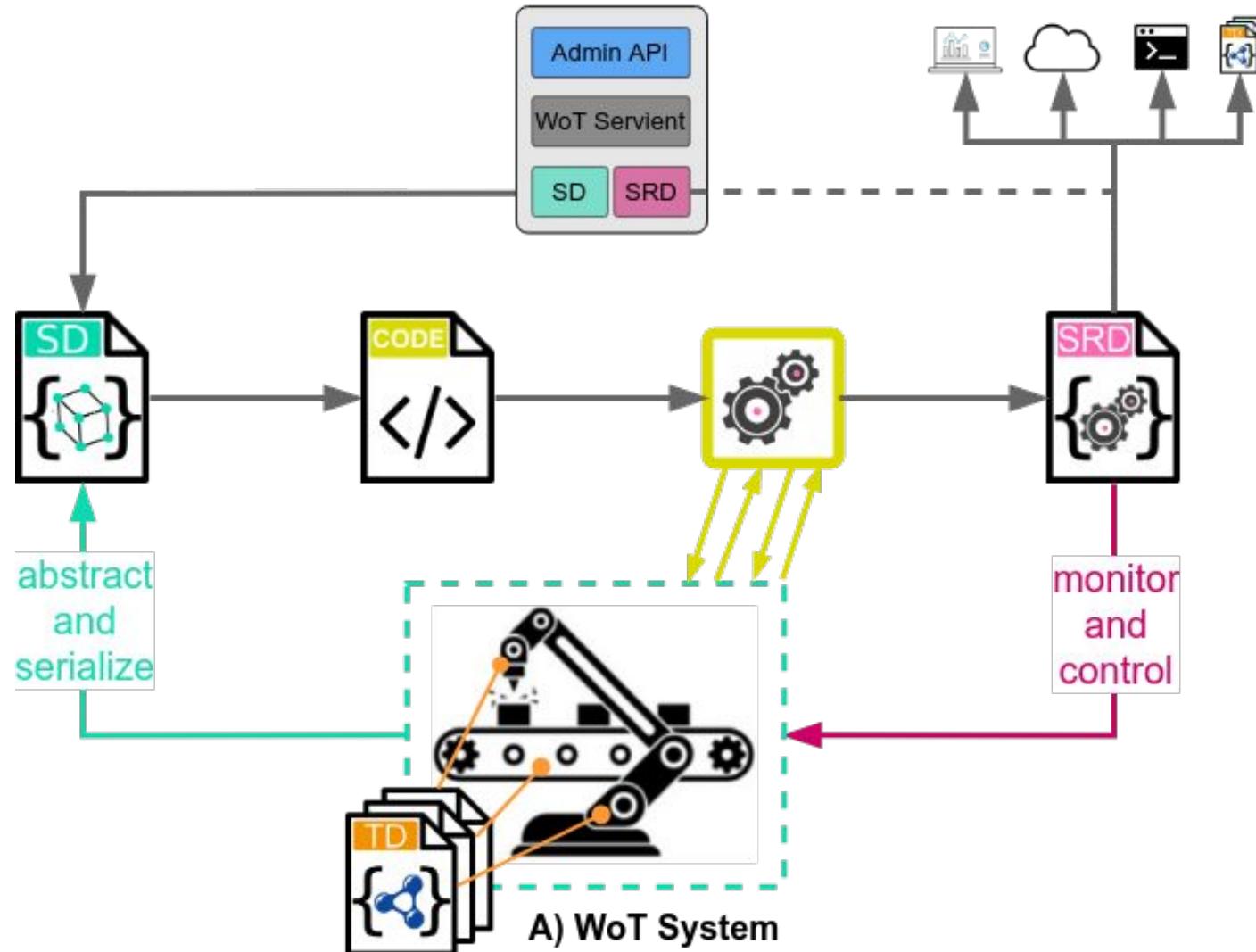
3. Evaluation

4. Conclusion

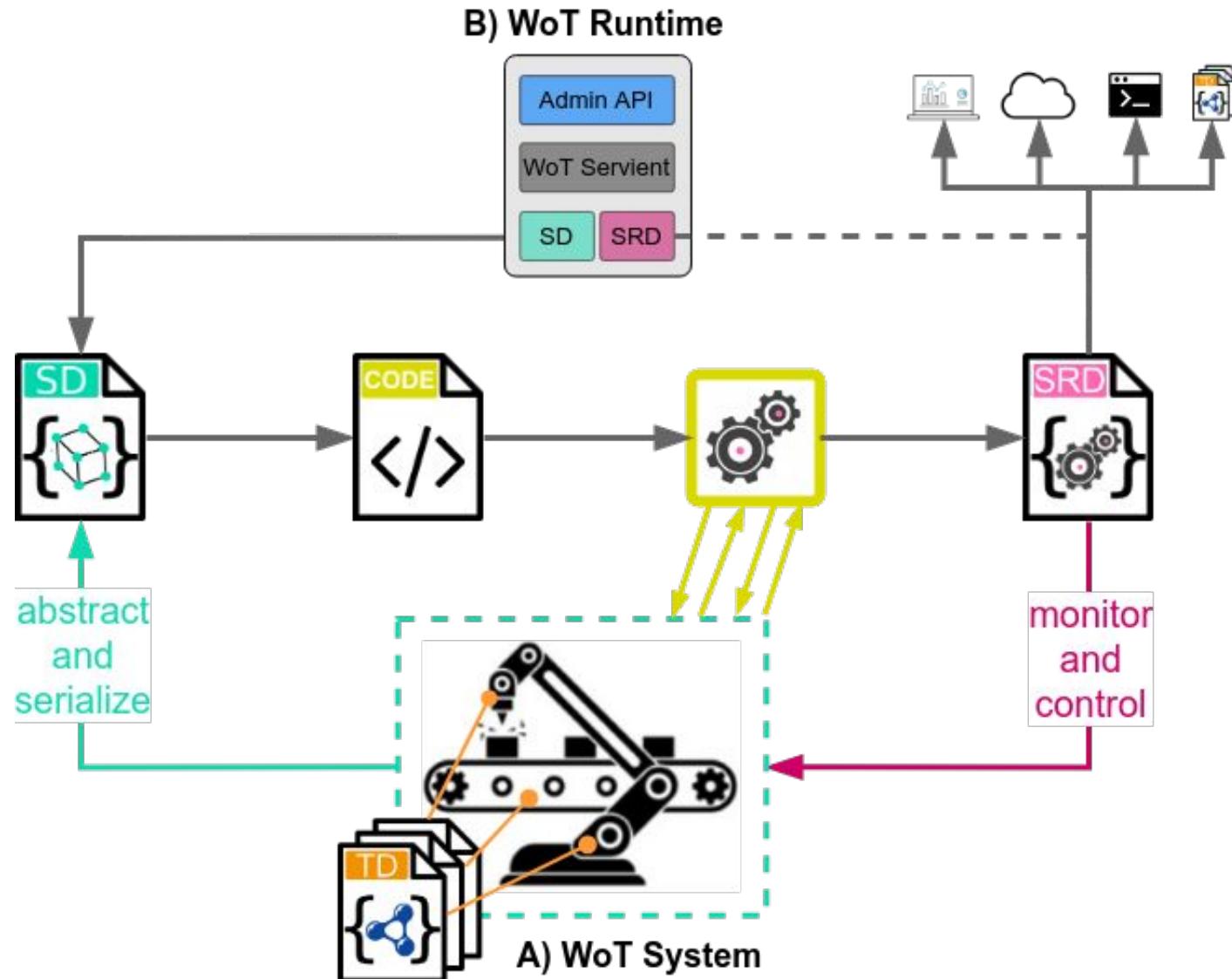
Methodology



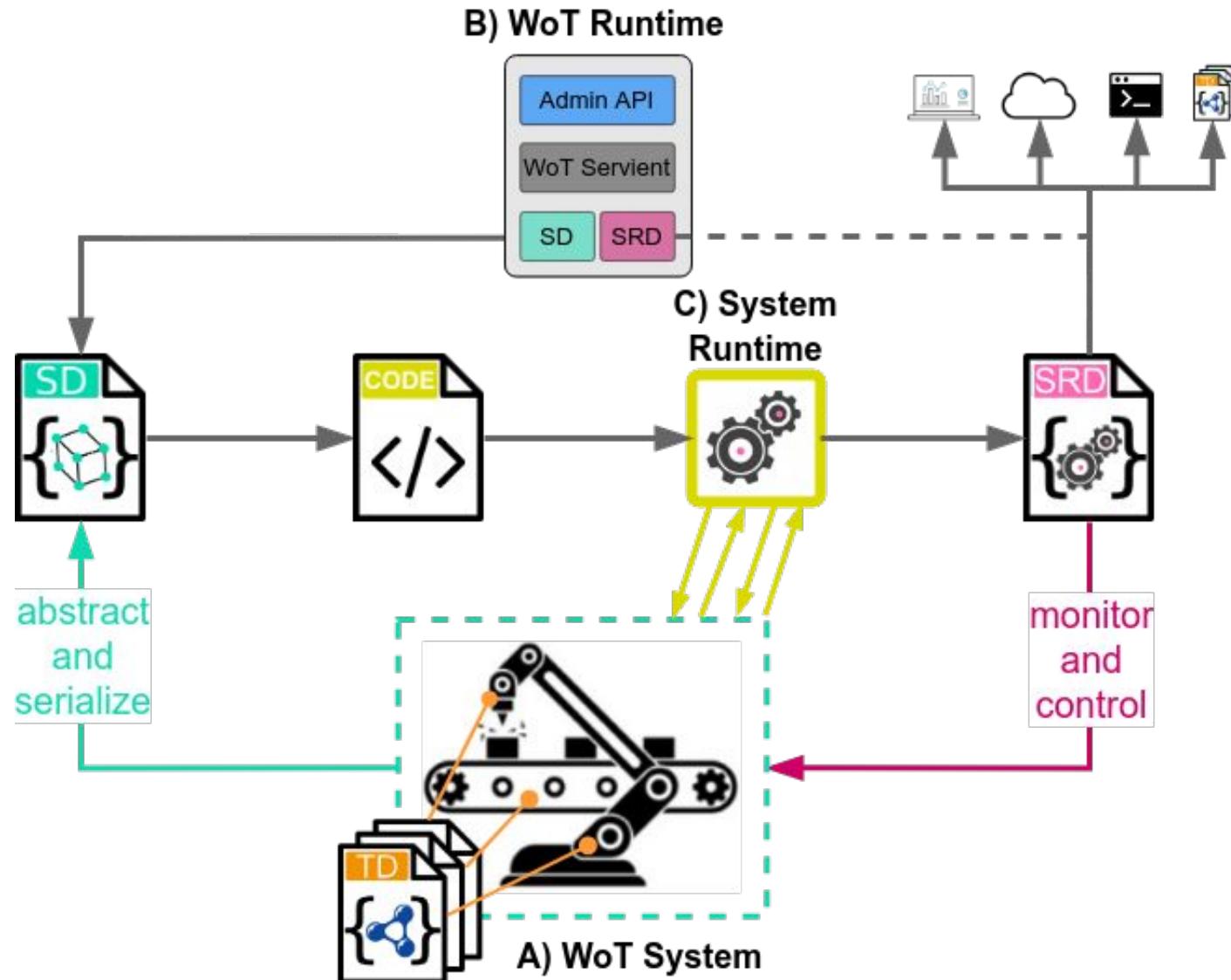
Methodology



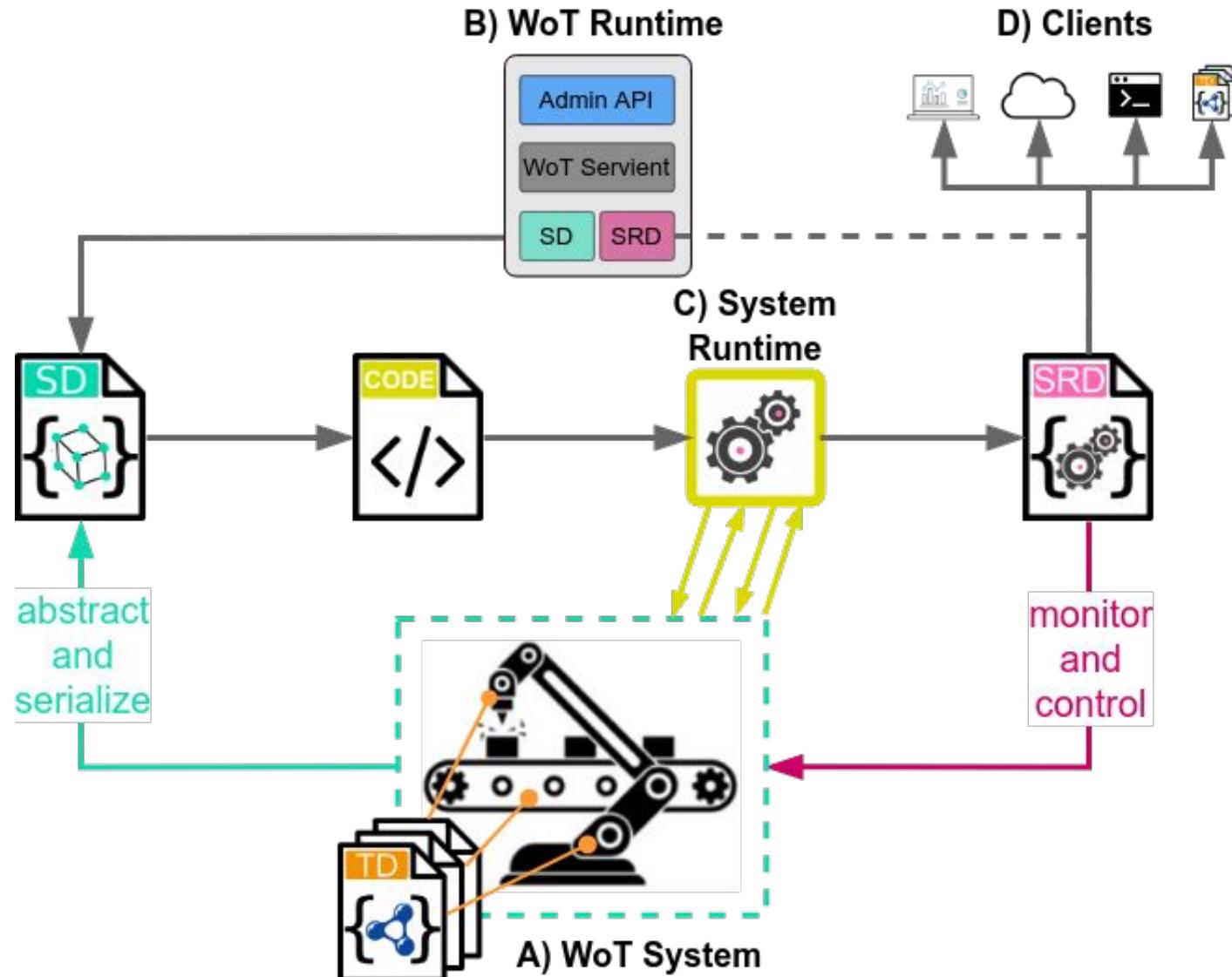
Methodology



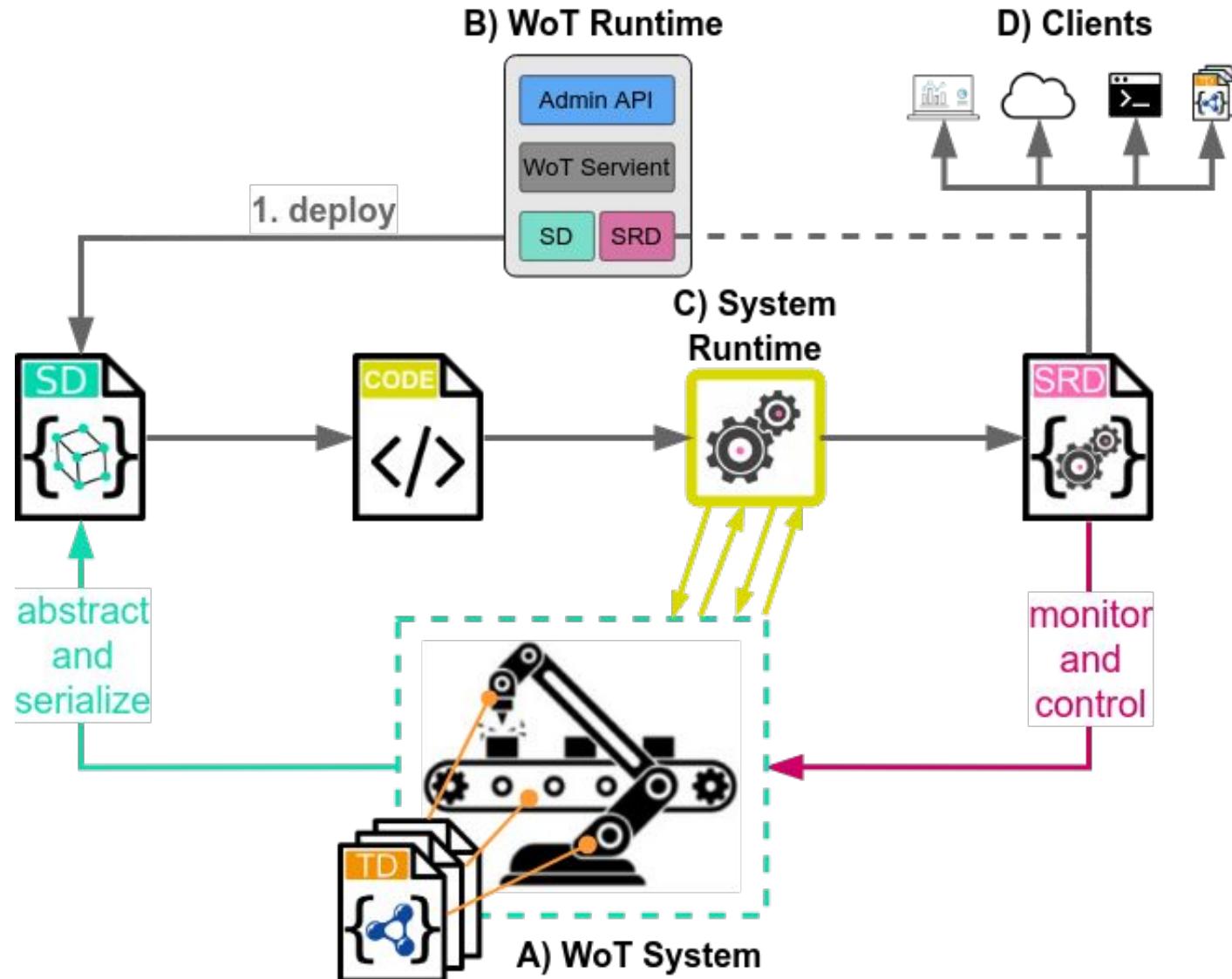
Methodology



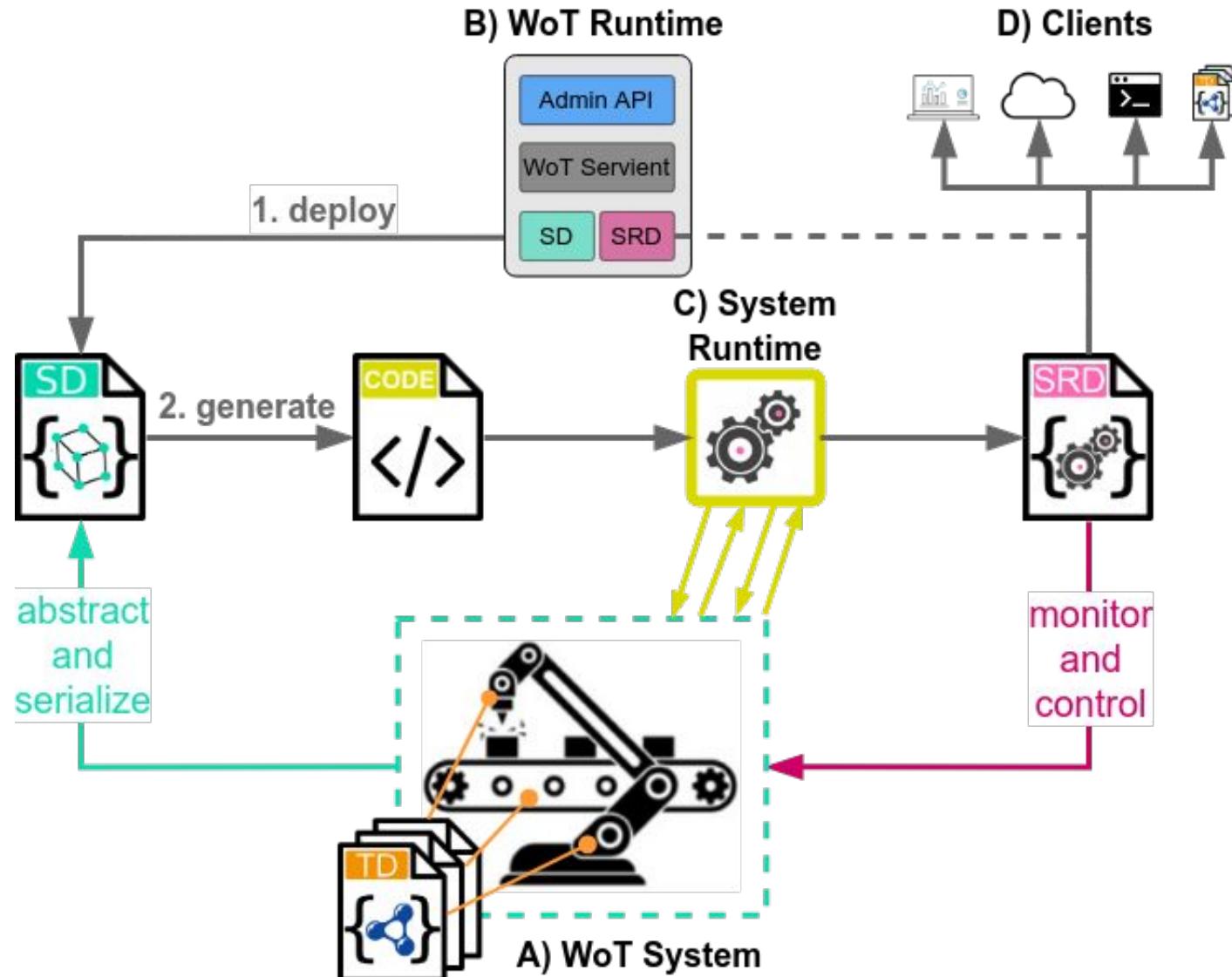
Methodology



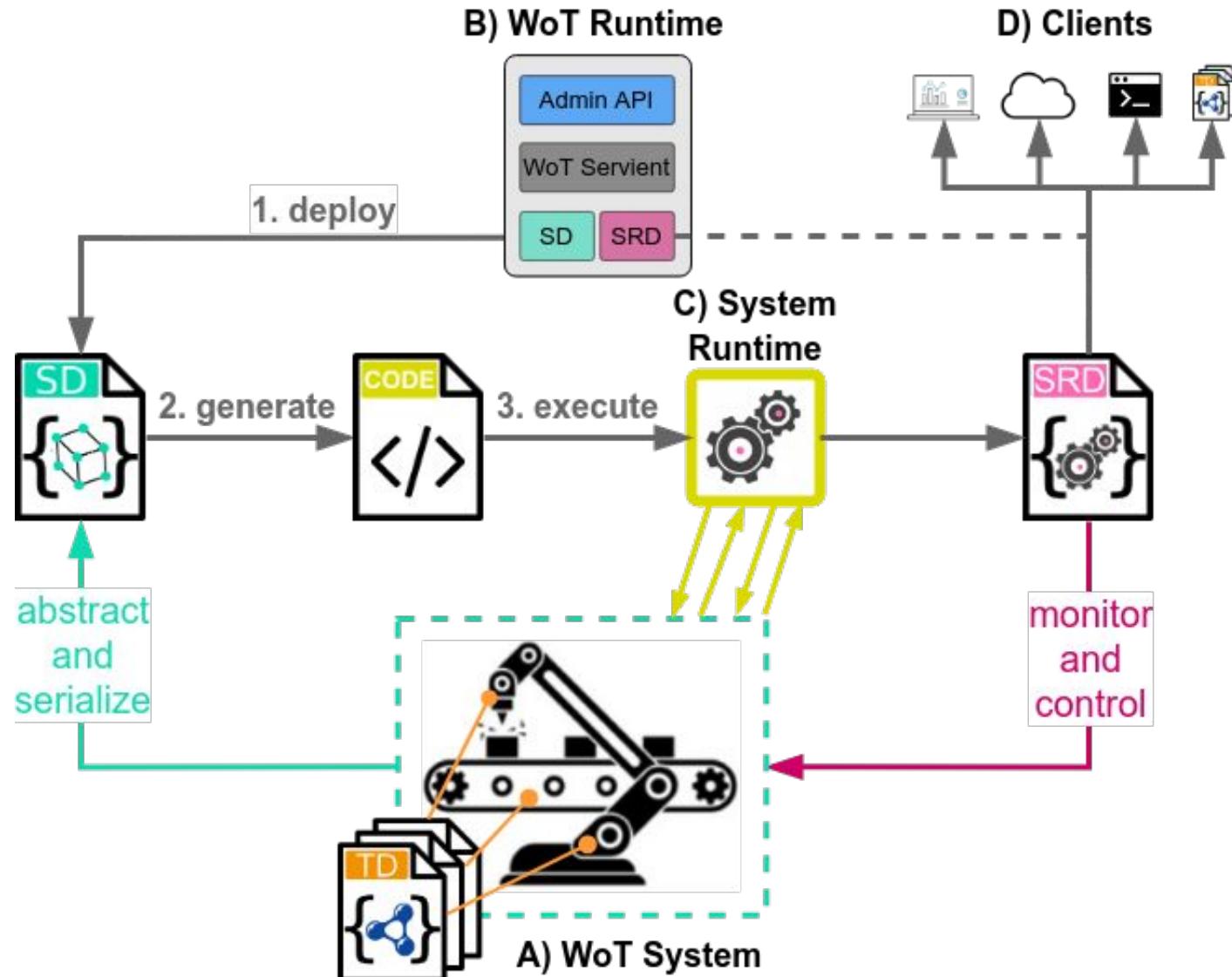
Methodology



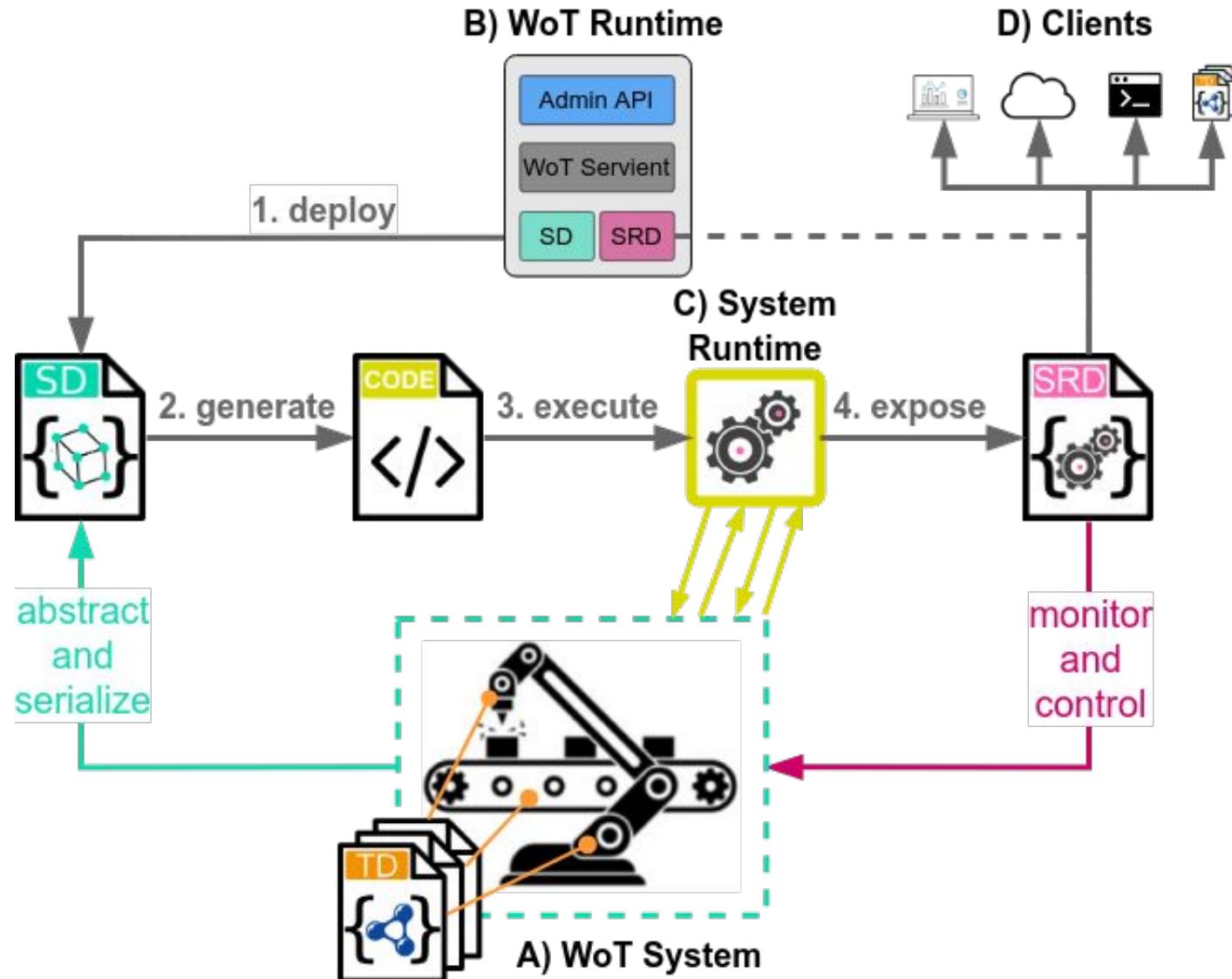
Methodology



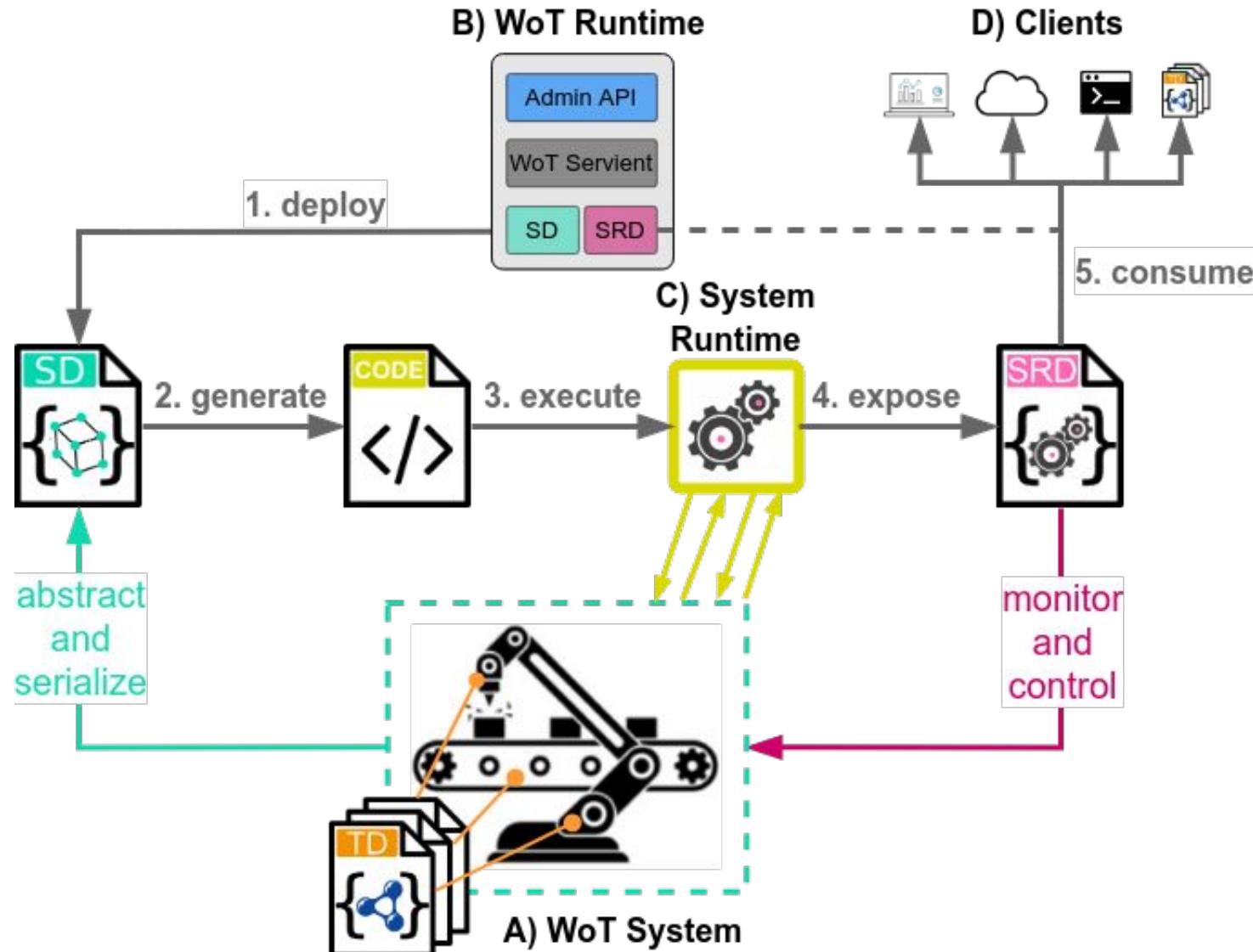
Methodology



Methodology



Methodology



1. Introduction

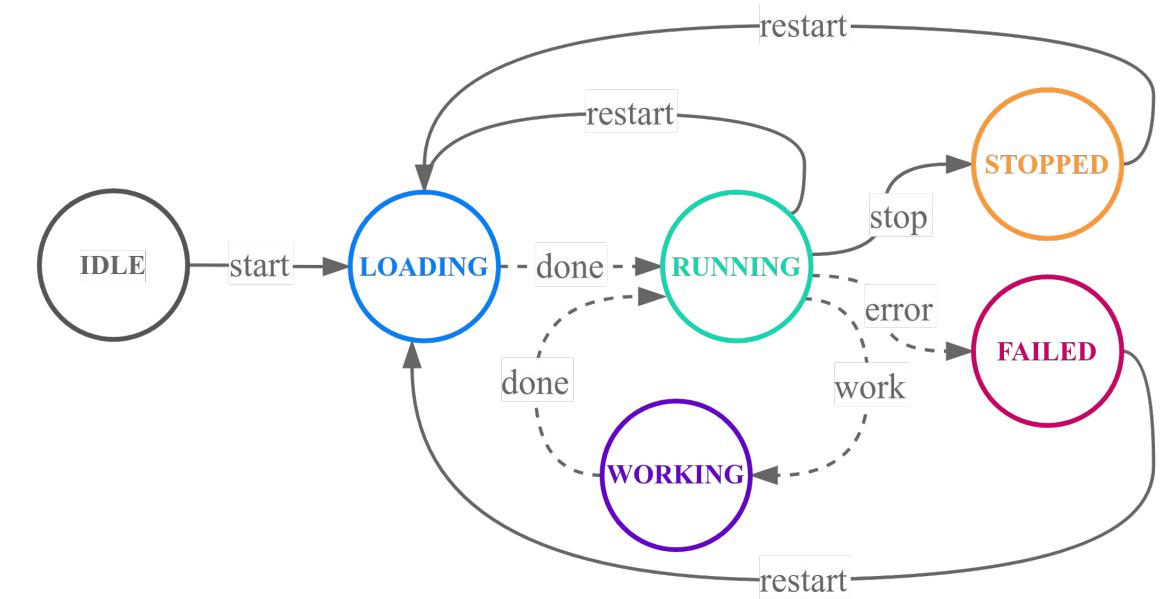
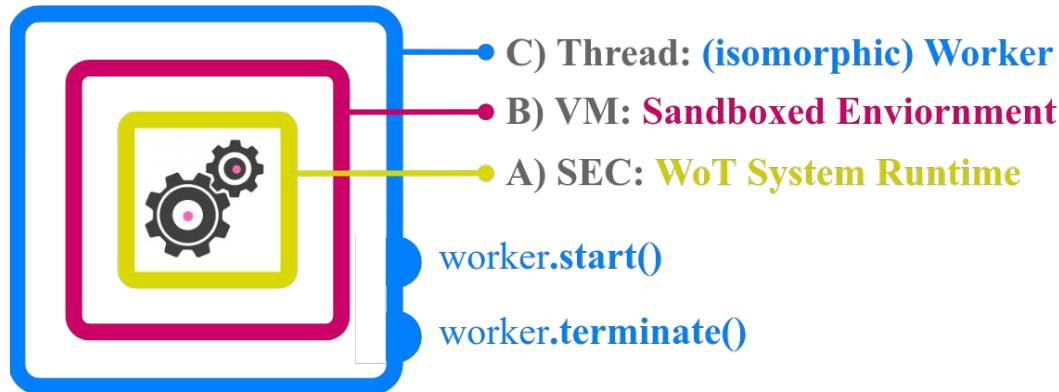
2. Approach

- 2.1. Overview
- 2.2. Methodology
- 2.3. Execution Environment and Control**
- 2.4. Execution Control

3. Evaluation

4. Conclusion

Execution Environment and Control



1. Introduction

2. Approach

- 2.1. Overview
- 2.2. Methodology
- 2.3. Execution Environment and Control
- 2.4. WoT Runtime UI

3. Evaluation

4. Conclusion

WoT Runtime UI - System Dashboard

WoT Runtime

Hosts > Systems > colorSort

OVERVIEW LOGS METRICS TRACES

Properties

- system
- mashupLogic
- seqD
- codeTS
- baseTS
- code
- base
- logs
- traces
- hostOS
- memoryUsage
- resourceUsage
- uptime
- status
- statusSvg
- statechartSvg
- step
- Actions**
- start
- restart

resourceUsage

memoryUsage

logs

statechartSvg

The dashboard provides a comprehensive view of system health and performance. It includes real-time monitoring of CPU and memory usage, log history, and a detailed statechart diagram for the seqD component. The statechart shows states like IDLE, LOADING, RUNNING, WORKING, STOPPED, and FAILED, with various transitions and events.

WoT Runtime UI - System Dashboard (dark)

The screenshot displays the WoT Runtime UI System Dashboard (dark) interface. The top navigation bar includes tabs for SYSTEMS and THINGS, along with icons for file operations and search.

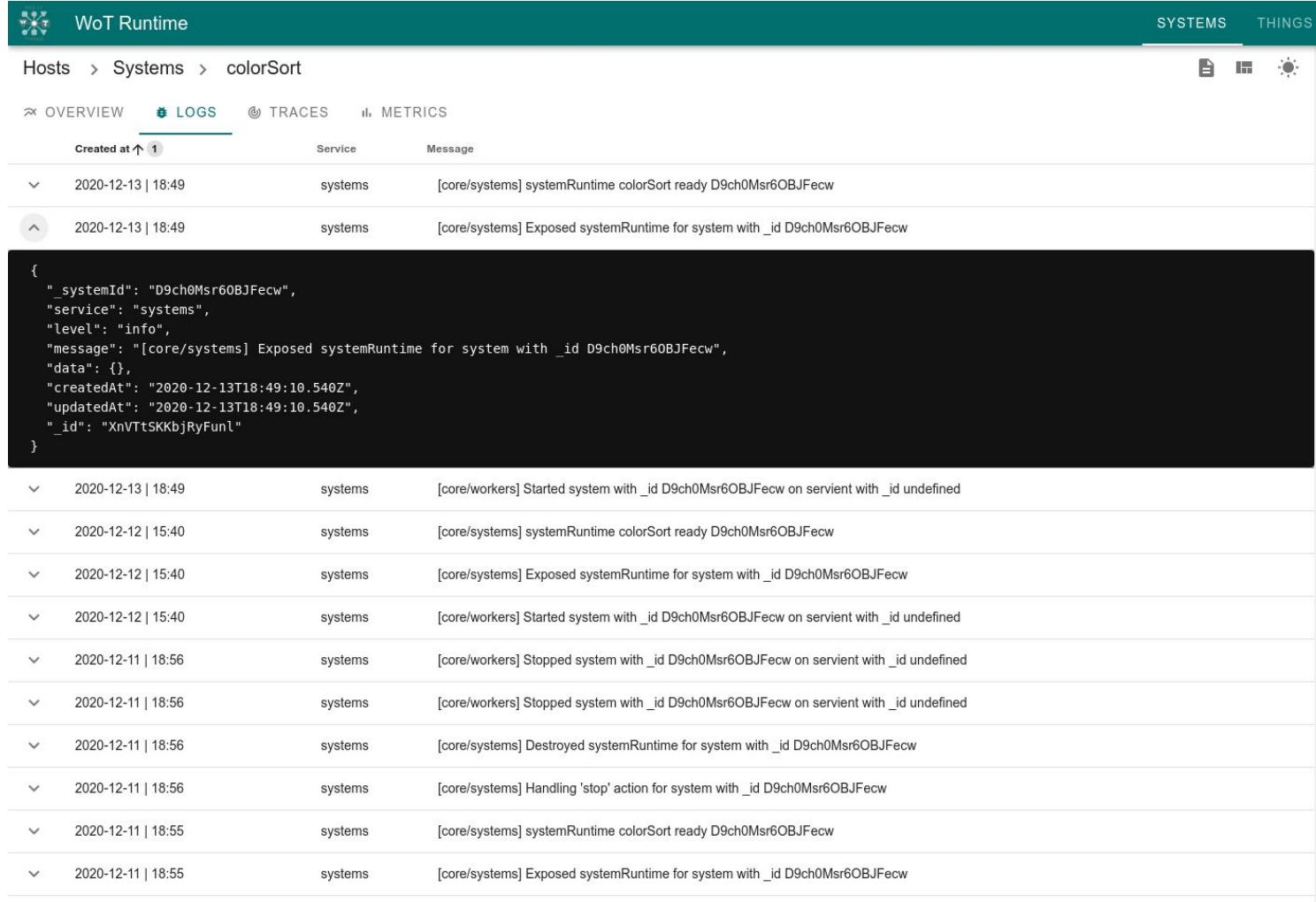
The left sidebar lists various system components and actions:

- Hosts > Systems > colorSort
- OVERVIEW, LOGS, METRICS, TRACES
- system: mashupLogic, seqD, codeTS, baseTS, code, base, logs, traces, hostOS, memoryUsage, resourceUsage, uptime, status, statusSvg, statechartSvg, step, Actions (start, restart, stop), Events

The main area contains several panels:

- statechartSvg:** A statechart diagram showing states: IDLE, LOADING, RUNNING, WORKING, FAILED, and STOPPED. Transitions include start from IDLE to LOADING, done from LOADING to RUNNING, done from LOADING to WORKING, stop from RUNNING to FAILED, error from WORKING to FAILED, and restart from FAILED back to LOADING.
- resourceUsage:** A chart titled "resourceUsage" showing CPU and memory usage over time. Legend includes: userCPUTime, systemCPUTime, maxRSS, sharedMemorySize, unsharedDataSize, unsharedStackSize, minorPageFault, majorPageFault, swappedOut, signalsCount, voluntaryContextSwitches, involuntaryContextSwitches.
- memoryUsage:** A chart titled "memoryUsage" showing memory usage over time. Legend includes: rss, swapTotal, swapUsed, external, arrayBuffers.
- codeTS:** A code editor panel showing TypeScript code related to WoT runtime logic.
- traces:** A panel showing trace logs with fields: path, type, method, ts, duration, end, createdAt, updatedAt, _id.

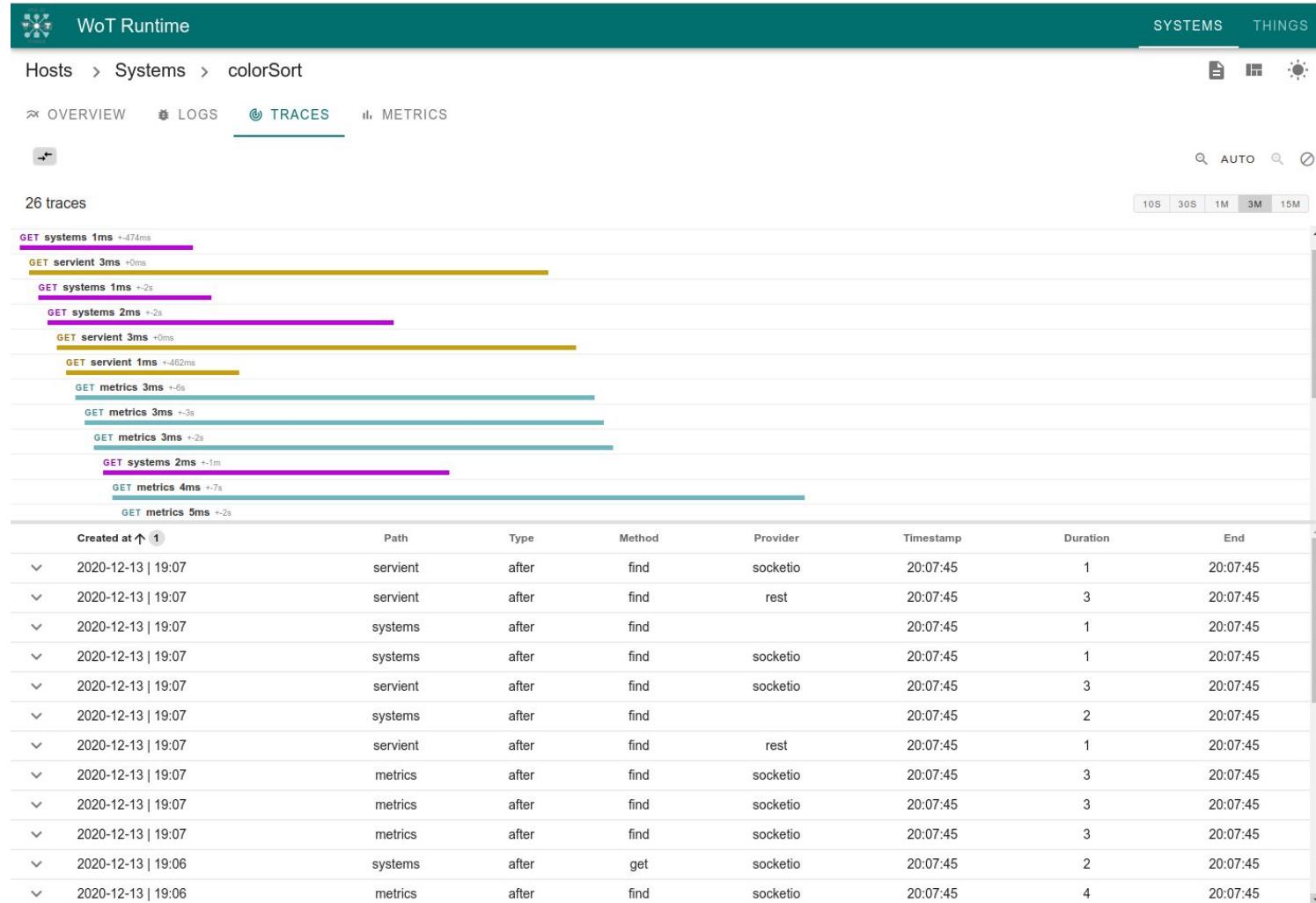
WoT Runtime UI - Logs



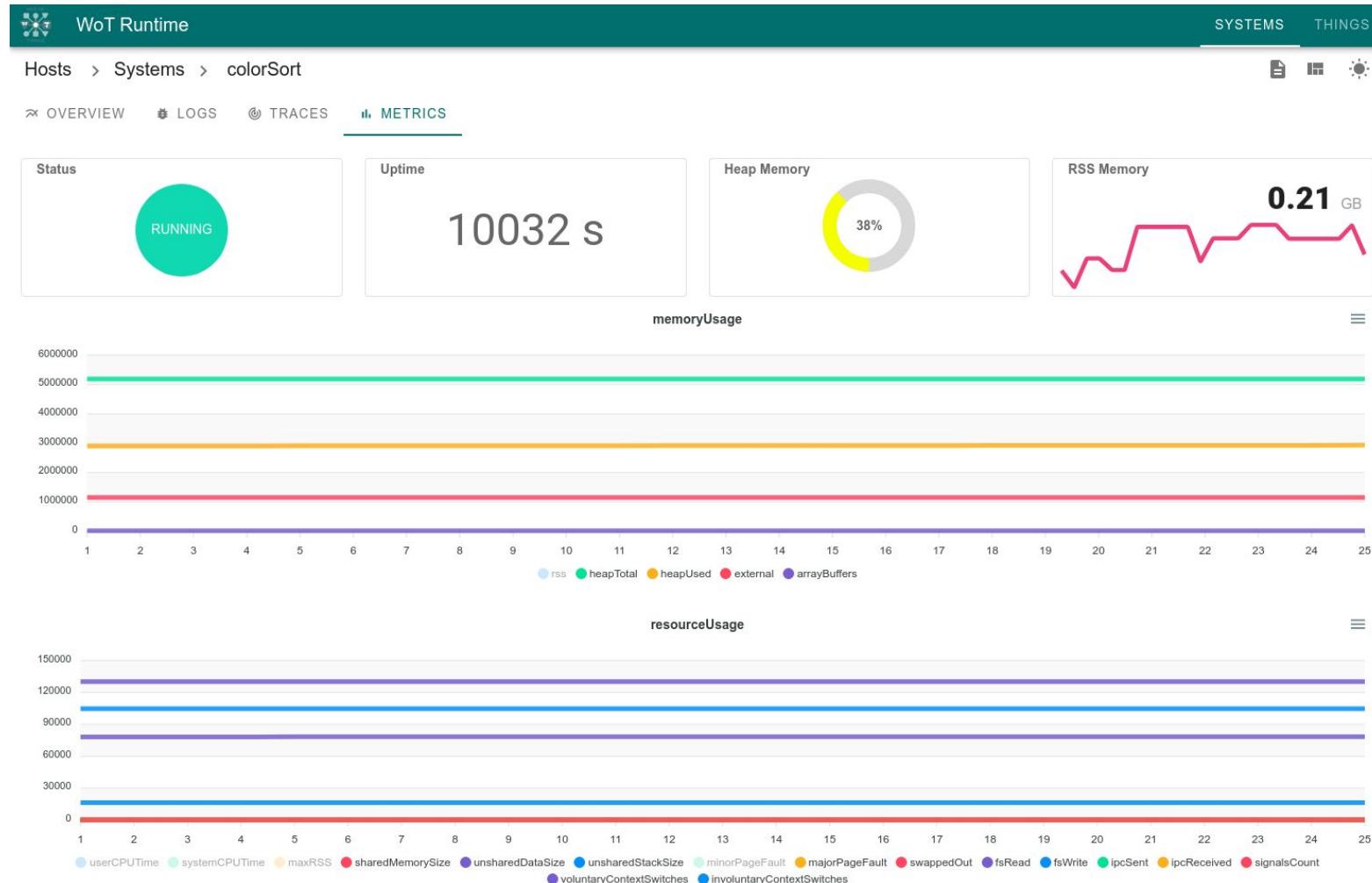
The screenshot shows the WoT Runtime UI interface. At the top, there's a navigation bar with icons for hosts, systems, and things, followed by tabs for Systems and Things. Below the navigation is a breadcrumb trail: Hosts > Systems > colorSort. Underneath the breadcrumb, there are four tabs: OVERVIEW, LOGS (which is selected), TRACES, and METRICS. The main area displays a table of log entries. The columns are 'Created at' (sorted by ascending time), 'Service', and 'Message'. The first two log entries are expanded, showing their raw JSON content. The third log entry is highlighted with a black box, revealing its detailed structure.

| Created at | Service | Message |
|--|---------|--|
| 2020-12-13 18:49 | systems | [core/systems] systemRuntime colorSort ready D9ch0Msr6OBJFecw |
| 2020-12-13 18:49 | systems | [core/systems] Exposed systemRuntime for system with _id D9ch0Msr6OBJFecw |
| { "_systemId": "D9ch0Msr6OBJFecw", "service": "systems", "level": "info", "message": "[core/systems] Exposed systemRuntime for system with _id D9ch0Msr6OBJFecw", "data": {}, "createdAt": "2020-12-13T18:49:10.540Z", "updatedAt": "2020-12-13T18:49:10.540Z", "_id": "XnVTtSKKbjRyFunl" } | | |
| 2020-12-13 18:49 | systems | [core/workers] Started system with _id D9ch0Msr6OBJFecw on servient with _id undefined |
| 2020-12-12 15:40 | systems | [core/systems] systemRuntime colorSort ready D9ch0Msr6OBJFecw |
| 2020-12-12 15:40 | systems | [core/systems] Exposed systemRuntime for system with _id D9ch0Msr6OBJFecw |
| 2020-12-12 15:40 | systems | [core/workers] Started system with _id D9ch0Msr6OBJFecw on servient with _id undefined |
| 2020-12-11 18:56 | systems | [core/workers] Stopped system with _id D9ch0Msr6OBJFecw on servient with _id undefined |
| 2020-12-11 18:56 | systems | [core/workers] Stopped system with _id D9ch0Msr6OBJFecw on servient with _id undefined |
| 2020-12-11 18:56 | systems | [core/systems] Destroyed systemRuntime for system with _id D9ch0Msr6OBJFecw |
| 2020-12-11 18:56 | systems | [core/systems] Handling 'stop' action for system with _id D9ch0Msr6OBJFecw |
| 2020-12-11 18:55 | systems | [core/systems] systemRuntime colorSort ready D9ch0Msr6OBJFecw |
| 2020-12-11 18:55 | systems | [core/systems] Exposed systemRuntime for system with _id D9ch0Msr6OBJFecw |

WoT Runtime UI - Traces



WoT Runtime UI - Metrics



1. Introduction

- 1.1. Motivation
- 1.2. WoT Thing Description
- 1.3. WoT System Description
- 1.4. Problem Statement

2. Approach

- 2.1. Overview
- 2.2. Methodology
- 2.3. Execution Environment
- 2.4. Execution Control

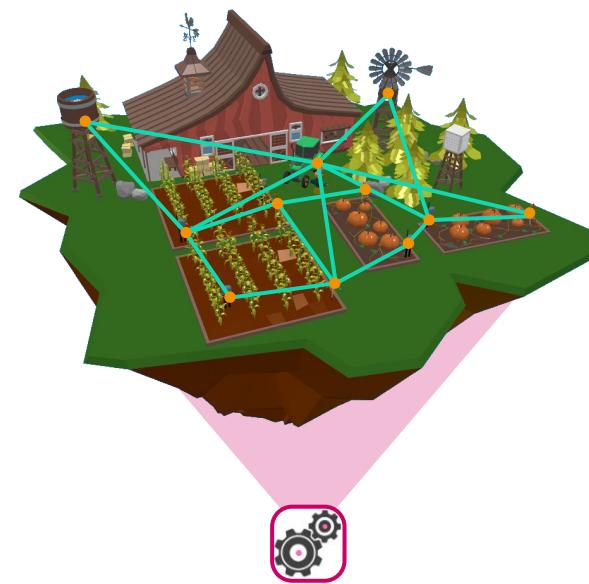
3. Evaluation

- 3.1. Results

4. Conclusion

Results

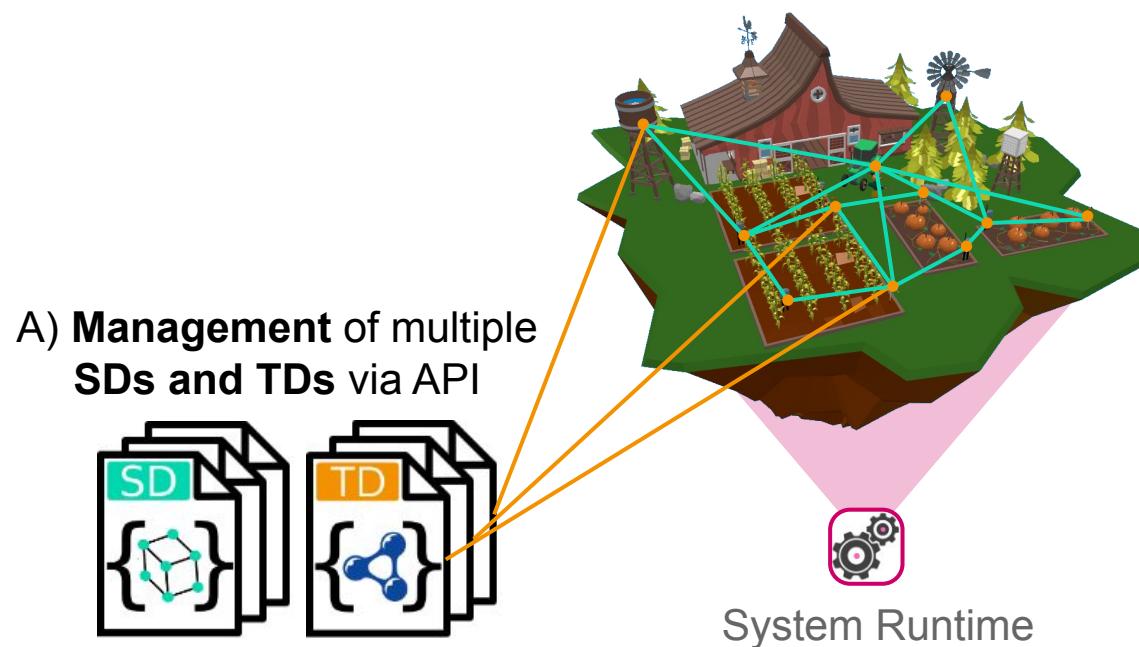
Smart Farm Simulation Use Case



System Runtime

Results

Smart Farm Simulation Use Case



Results

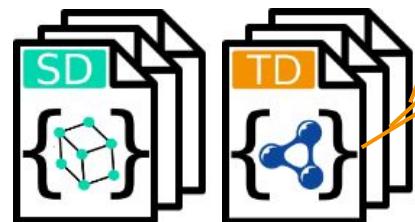
Smart Farm Simulation Use Case

B) Runtime monitoring via logs, metrics and traces

| | |
|------------------------------|--------|
| [warn] DeprecationWarning | Yellow |
| [error] ReferenceError | Red |
| [info] Started system 239... | Blue |
| [info] Created SRD for sy... | Blue |
| [error] NetworkError | Red |



A) Management of multiple SDs and TDs via API



System Runtime

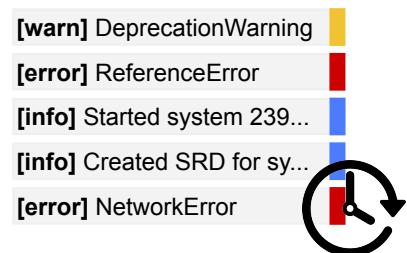
Results

Smart Farm Simulation Use Case

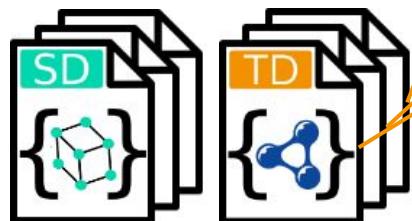
C) Mashup **monitoring and control** via auto-generated GUI



B) Runtime monitoring via logs, metrics and traces



A) Management of multiple SDs and TDs via API

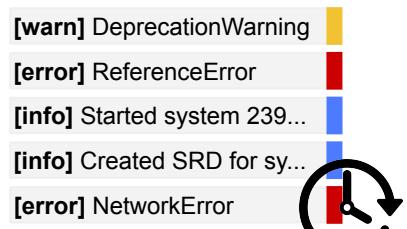


System Runtime

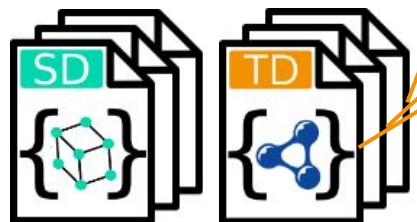
Results

Smart Farm Simulation Use Case

B) Runtime monitoring via logs, metrics and traces

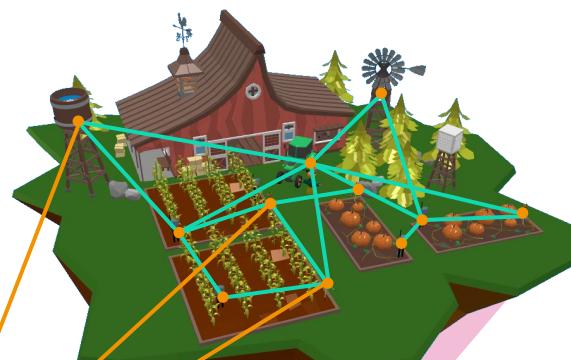
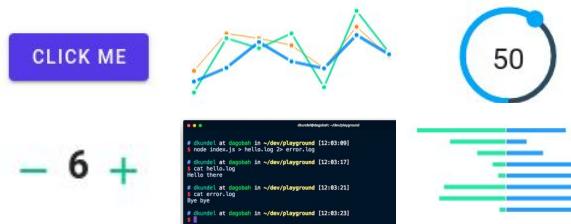


A) Management of multiple SDs and TDs via API

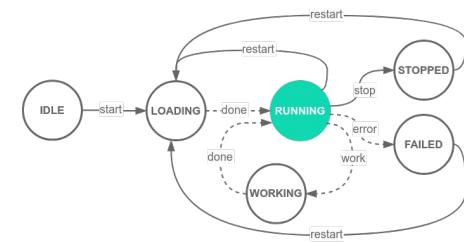


System Runtime

C) Mashup monitoring and control via auto-generated GUI

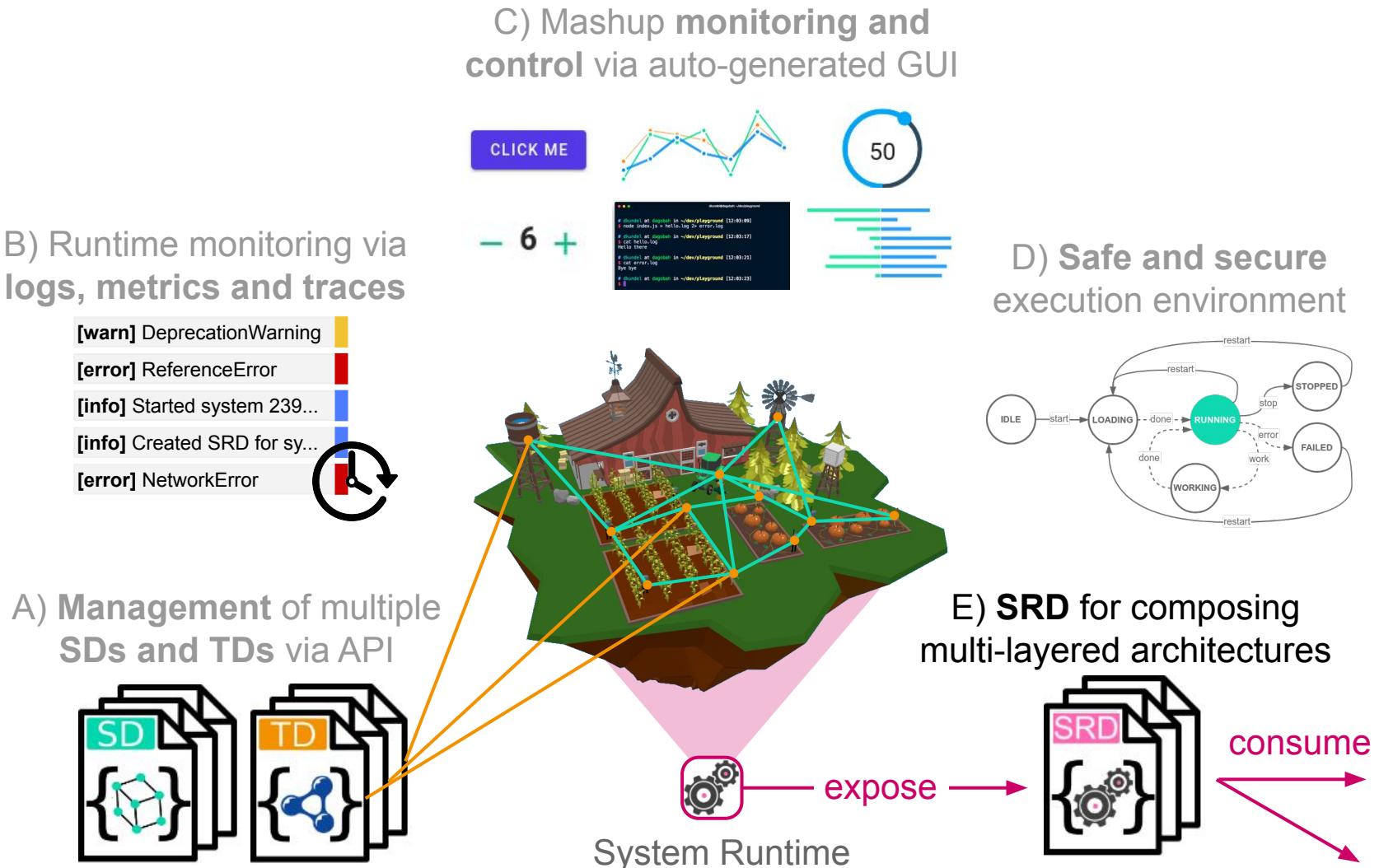


D) Safe and secure execution environment



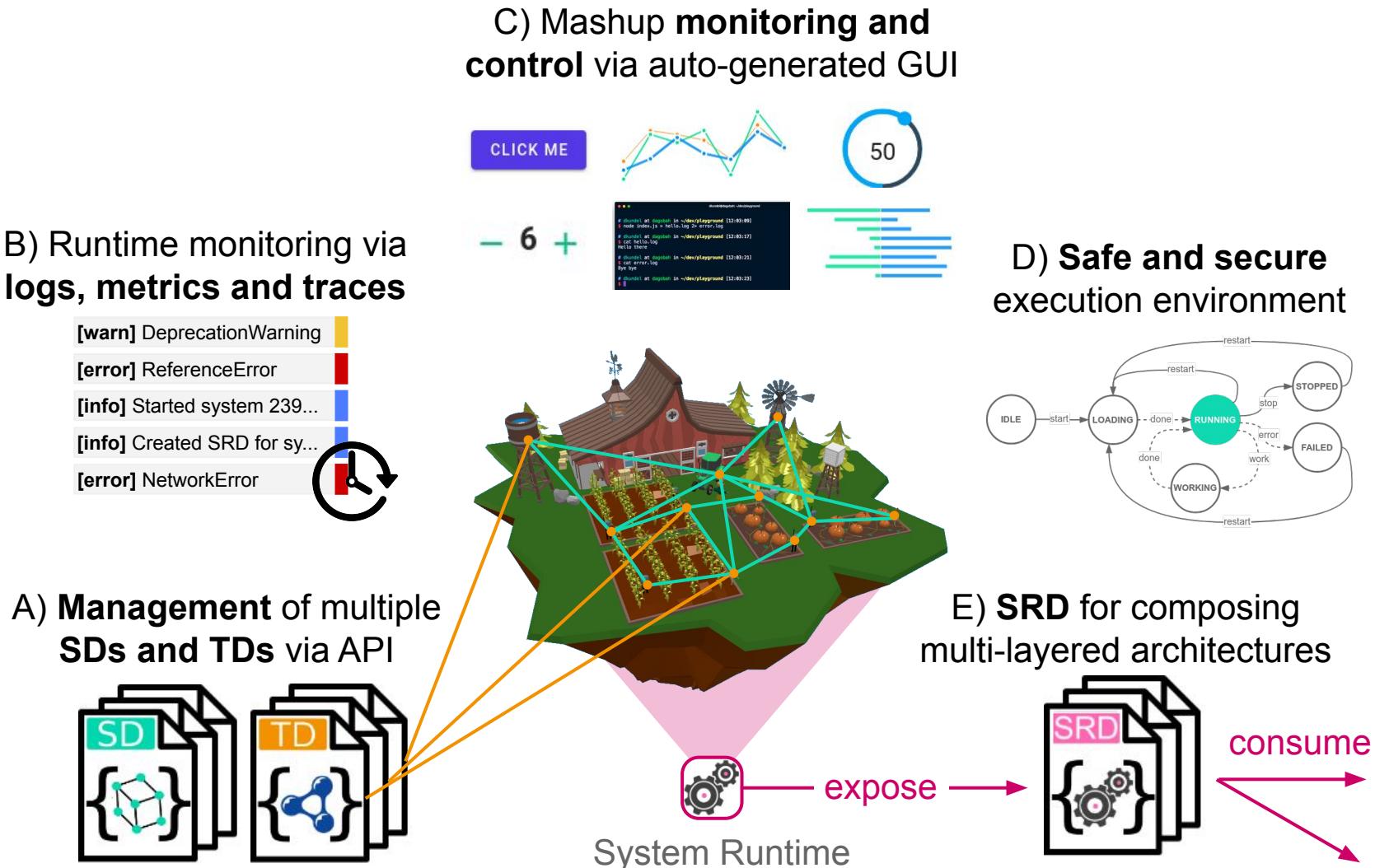
Results

Smart Farm Simulation Use Case



Results

Smart Farm Simulation Use Case



1. Introduction

- 1.1. Motivation
- 1.2. WoT Thing Description
- 1.3. WoT System Description
- 1.4. Problem Statement

2. Approach

- 2.1. Overview
- 2.2. Methodology
- 2.3. Execution Environment
- 2.4. Execution Control

3. Evaluation

- 3.1. Results

4. Conclusion

Conclusion

In conclusion:

- The WoT Runtime Framework introduces an additional WoT building block for:
 - remote deployment of SDs into safe execution Environments,
 - management of WoT System Runtimes,
 - and monitoring and control through the auto-generated SRD

Conclusion

In conclusion:

- The WoT Runtime Framework introduces an additional WoT building block for:
 - remote deployment of SDs into safe execution Environments,
 - management of WoT System Runtimes,
 - and monitoring and control through the auto-generated SRD
- The SRD allows composing multi-layered WoT architectures

Conclusion

In conclusion:

- The WoT Runtime Framework introduces an additional WoT building block for:
 - remote deployment of SDs into safe execution Environments,
 - management of WoT System Runtimes,
 - and monitoring and control through the auto-generated SRD
- The SRD allows composing multi-layered WoT architectures
- The approach is evaluated in two use cases
 - An industrial automation scenario
 - and a smart farm simulation

demonstrating how the proposed solution works in practice

- The WoT Runtime framework lays the basis for future in distributed WoT architectures

Thank you for your attention