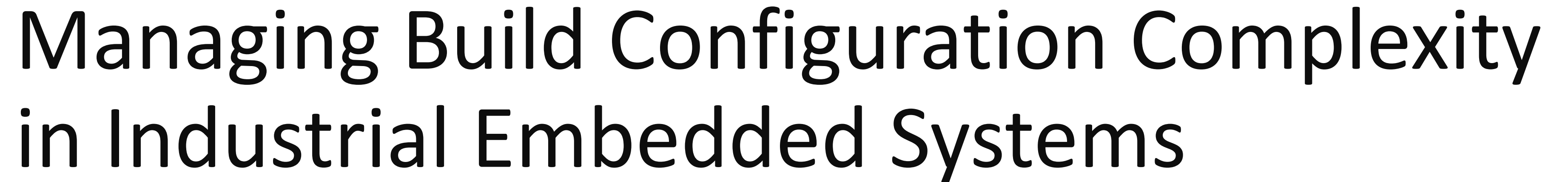
PRODUCTS & PLATFORMS
by HCL Technologies

HCL



Managing Build Configuration Complexity in Industrial Embedded Systems

- Dynamic Manipulation of Model Transformations using JavaScript

Mattias Mohlin and Elena Strabykina, HCL Technologies

MODELS 2018
CØPENHAGEN

CONTENT

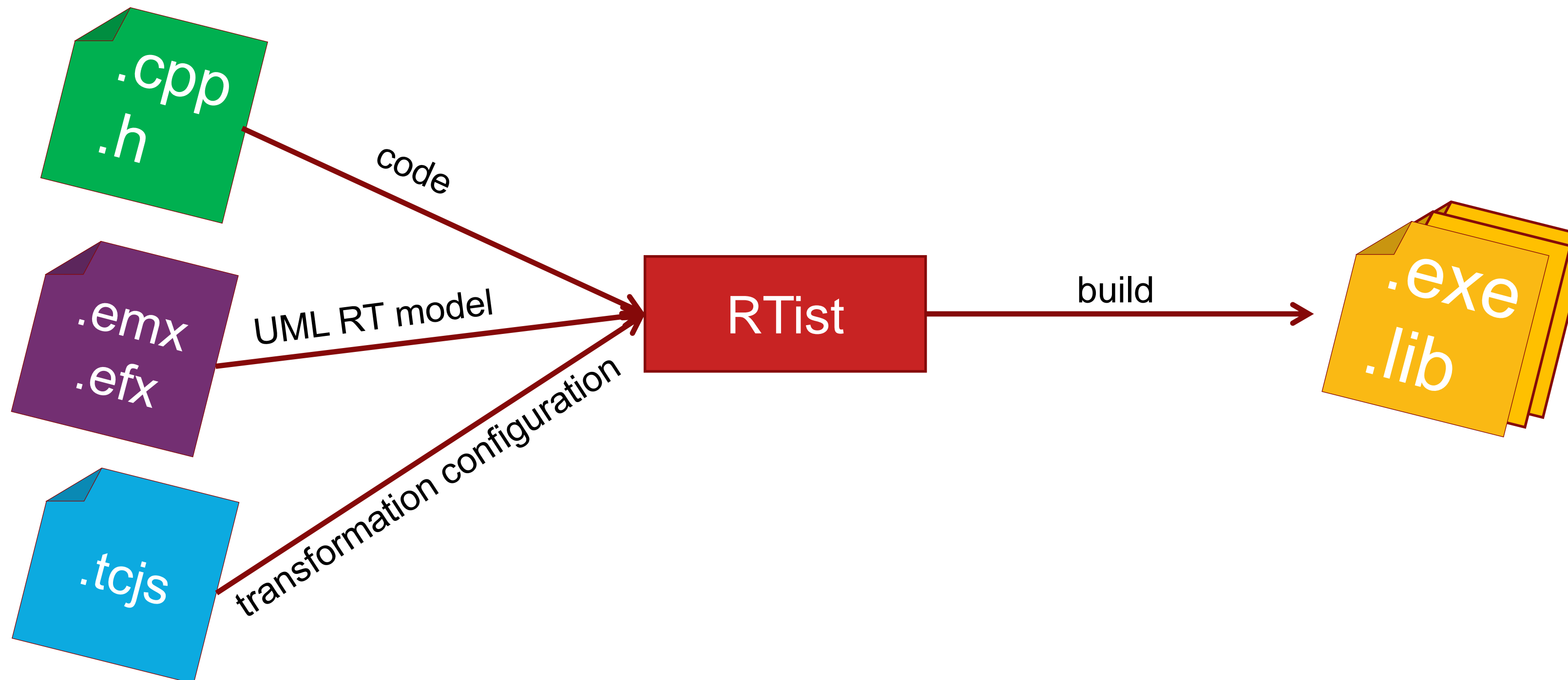
- UML-RT models and transformation configurations
- Build Variants
- Demo

Copyright © 2018 HCL Technologies Limited | www.hcltech.com
PRODUCTS & PLATFORMS by HCL Technologies
HCL

TRANSFORMING UML-RT MODELS TO C++

- A UML-RT model with contained C++ code is a complete specification of an application (e.g. embedded or IoT application)

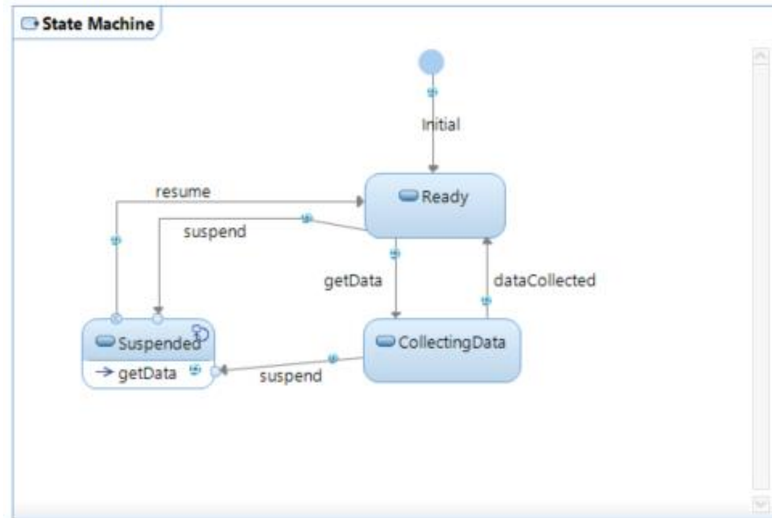
- A Transformation Configuration is a model describing how to transform the UML-RT model to a C++ executable or library



Copyright © 2018 HCL Technologies Limited | www.hcltech.com

UML-RT MODEL

- Behavior described by state machine diagrams and C++ code



Code View | Problems | Console | Search

C++ <Transition> - suspend

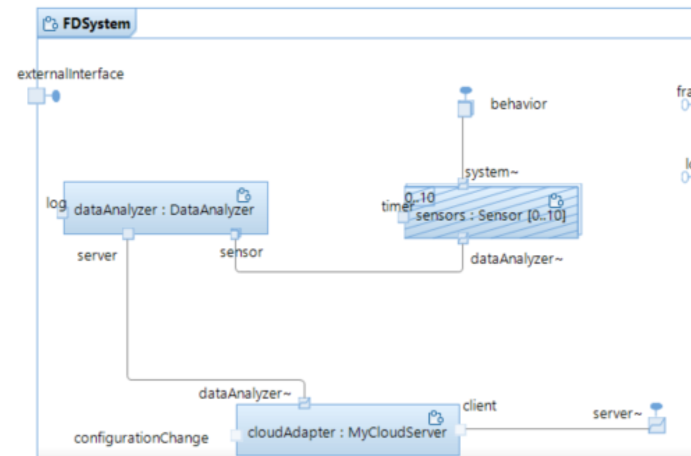
Showing code from the model suspend

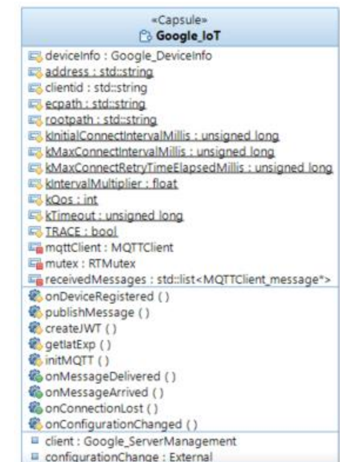
```cpp
printf("Suspending sensor #%d while it was collecting data\n", id);

// Sensor was suspended while collecting data. Cancel the data request.
timer.cancelTimer(tid);

// The sensor is suspended so return zero data
SensorData sd;
sd.sensorId = id;
int temp1 = rand() % 10;
int temp2 = rand() % 10;
sd.temperature = -1000; // Indicates suspended sensor
```

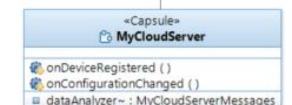
Effect | Guard

- Structure described by class and composite structure diagrams

- Stored in XML files with embedded C++ code snippets







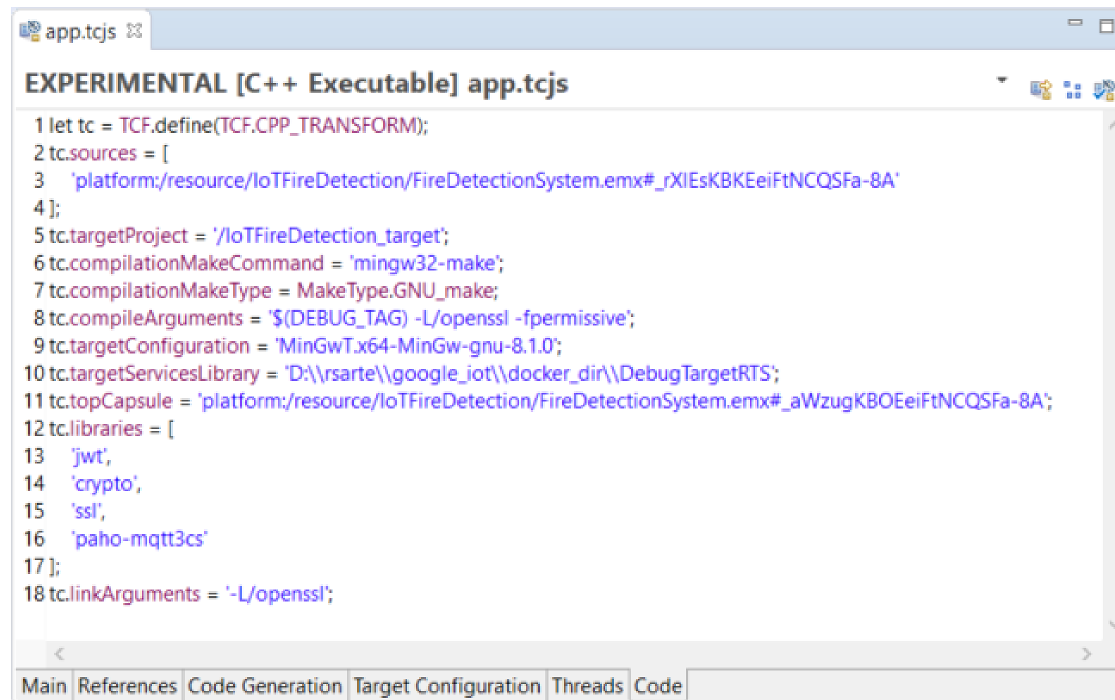
Copyright © 2018 HCL Technologies Limited | www.hcltech.com

PRODUCTS & PLATFORMS by HCL Technologies

HCL

TRANSFORMATION CONFIGURATION

- A model containing everything needed for transforming the UML-RT model into a C++ program and building an executable or library from it
- Stored in text files using the JavaScript language



Copyright © 2018 HCL Technologies Limited | www.hcltech.com

BUILDING MULTIPLE VARIANTS OF AN APPLICATION

- **Problem:** How to express variability in a transformation configuration?
  - Debug vs Release version
  - Different target platforms (OS, compiler etc.)
  - Instrumented builds (e.g. purify)
  - Static analysis (e.g. lint), etc...
- With static transformation configurations, you need one for each variant you want to build...
  => A huge number of transformation configurations to create and maintain!
  => Difficult for users to pick a consistent set of transformation configurations when building the model!
- Inheritance allows to break out common information in separate transformation configurations, but doesn't solve the problem (still a huge number of transformation configurations even if they all are small).
- We need dynamic transformation configurations where build properties can be manipulated programmatically!

Copyright © 2018 HCL Technologies Limited | www.hcltech.com
PRODUCTS & PLATFORMS
HCL

BUILD VARIANTS

- **Solution:** Allow the transformation configuration to be dynamically manipulated at build time

Build Variant – a set of transformation configuration properties that are specific for a certain type of build, described in a separate JavaScript file.

default_settings.js

```
let globals = TCF.globals();
globals.makeArguments = '$ARG_GLOBAL';
```

Main_configuration.js

```
let tc = TCF.define(TCF.CPP_TRANSFORM);
tc.sources = ['platform:/resource/CM/CPPModel.emx#_uysz8NQ3EeexPbULy_rI8g'];
tc.genUserCodeQualifiers = globals.makeArguments + 'MY_ARGUMENT';
tc.type = CppTransformType.Executable;
```

target.js

```
tc.makeType = 'Library_makeType';
tc.TargetRTS = '${RSA_RT_HOME}/C++/TargetRTS';
```

flags.js

```
tc.compileArguments = 'new_compile_arguments';
```

validate_override.js

```
let ext_TC = TCF.load('platform:/resource/Lib/lib.tc');
tc.prerequisites = ext_TC.prer.add('platform:/resource/LibraryProject1/tc/Lib2.js');
```

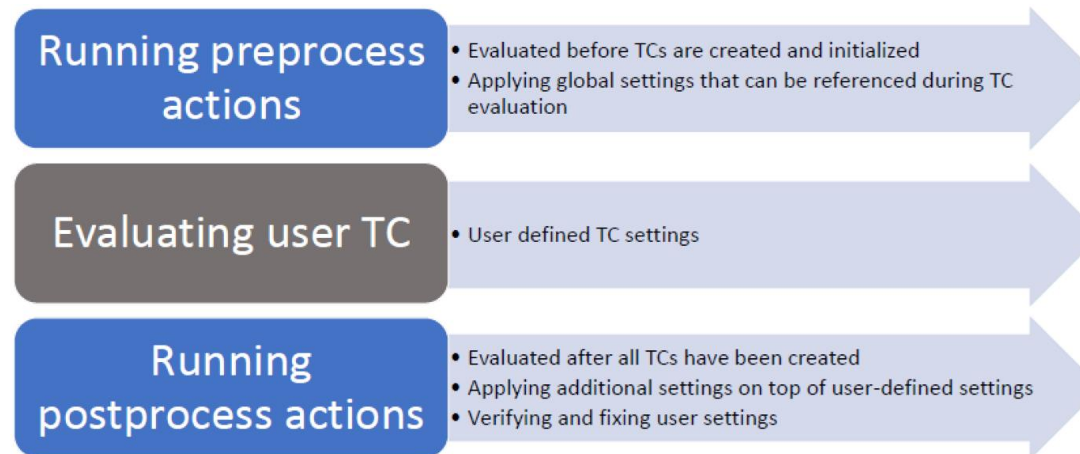

Copyright © 2018 HCL Technologies Limited | www.hcltech.com

PRODUCTS & PLATFORMS by HCL Technologies

HCL

IMPLEMENTATION OF BUILD VARIANTS APPROACH

- Each build variant script can be called one or two times, by defining one or both of these functions
  - `function preProcess(<args>)`
    Called before evaluation of the transformation configuration. Set global properties that can be referenced later.
  - `function postProcess(topTC, allTCs, <args>)`
    Called after evaluation of the transformation configuration. Verify and override user-defined properties.



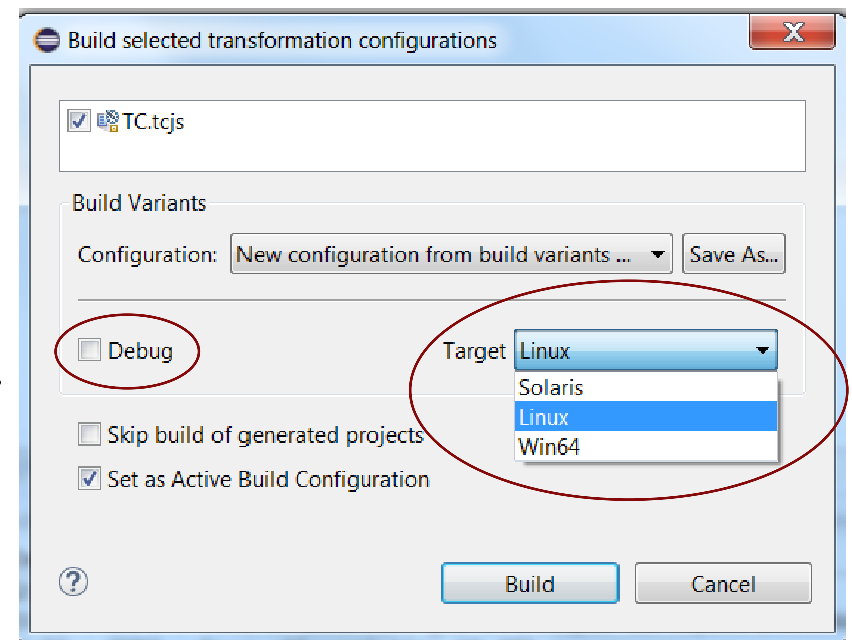
- Custom arguments can be passed from the build variants script (allows to reuse the same build variant script for multiple choices)

Copyright © 2018 HCL Technologies Limited | www.hcltech.com
PRODUCTS & PLATFORMS
HCL

INTEGRATION WITH IDE

- A build variants declaration file (JavaScript) describes the high-level choices that cause the variability
- The script renders a dynamic user interface to allow the user to make the choices when building
  - Checkbox for single choice
  - Drop-down menu for multiple choice
- Each choice is mapped to a build variant script to be applied to the transformation configuration when it is built

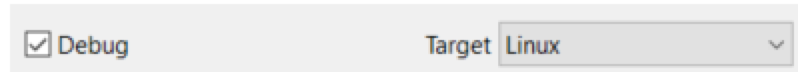
```
let debug = { name: 'Debug', script: 'debug.js', control : { kind: 'checkbox' },
  defaultValue : false, description: 'Build for debugging‘ };

let target = { name: 'Target',
  alternatives: [
    {name: 'Solaris', script: 'Target.js', args: [ 'Solaris' ], description: 'Settings for Solaris target platform'},
    {name: 'Linux', script: 'Target.js', args: [ 'Linux' ], defaultValue: true, description: 'Settings for Linux target platform'},
    {name: 'Win64', script: 'Target.js', args: [ 'Windows' ], description: 'Build settings for Windows 64bit'}
  ]
}

function initBuildVariants(tc) {
  BVF.add(debug, target)
}
```



Copyright © 2018 HCL Technologies Limited | www.hcltech.com
PRODUCTS & PLATFORMS
HCL

BUILD CONFIGURATIONS AND BATCH BUILDS

- Each combination of choices made in the build variants user interface is called a build configuration and can be represented textually
  - For example: “Debug; Target=Linux” is equivalent to these UI settings



- The build configuration string can be specified as an argument to the model compiler when performing a batch build (e.g. `--buildConfig=“Debug; Target=Linux”`)

Copyright © 2018 HCL Technologies Limited | www.hcltech.com

DEMO

 Copyright © 2018 HCL Technologies Limited | www.hcltech.com

EVALUATION RESULTS

Build Variants approach was evaluated on *N* model projects and *K* target platforms:

- got *K* times less number of transformation configurations for maintenance (now it is not required to create separate transformation configurations for each platform);
- less maintenance effort when adding new target platform (instead of *N* new files with new settings we need to update only 2 files with Build Variants declaration and Build Variants implementation);
- managed to decrease the size of SCM repository where models are stored;
- removed inconsistency between end-user builds invoked from the tool and backend builds invoked from automatic testing system;
- reported errors and warning messages during validation of input TCs.

Copyright © 2018 HCL Technologies Limited | www.hcltech.com

PRODUCTS & PLATFORMS
HCL

THANK YOU AND JOIN THE HCL TEAM

- DevOps, Agile and Models (MAD @ Models)
  - Industry speakers
- Modeling Tools Restarted –meet HCL

Join the HCL team at the Modeling Tools Restarted session on Tuesday and at our exhibit booth on Wednesday!

We also invite you to participate in the following sessions featuring speakers from HCL:

SUNDAY, OCTOBER 14, 2018	**W5 EXE:** Workshop on Executable Model
MONDAY, OCTOBER 15, 2018	**W7 MDETools:** Workshop on Model Driven Engineering Models, Agile & DevOps @ Models
TUESDAY, OCTOBER 16, 2018	Model-Based System Engineering (MBSE) **W13 PAINS:** Pains in Model-Driven Engineering Practice **Modeling Tools Restarted – Meet HCL!** *Come hear more about who HCL is and what we're doing around our Modeling tools. Hear from one of our modeling customers, participate in a Tutorial and enjoy a conversation with other conference colleagues during a small social.*
WEDNESDAY, OCTOBER 17, 2018	Industry Day

WX-TC3074-AT7645

PRODUCTS & PLATFORMS

HCL

Copyright © 2018 HCL Technologies Limited | www.hcltech.com