CHESS

CHESS Runtime Monitoring - Quick Guide

May 2020

1 Table of Contents

2 Document history

Date	Changes
May 2020	First version

3 Introduction

The CHESS tool includes trace analysis and back propagation support.

CHESS allows to derive threads timing information for software components, such as execution time, activation, blocking time, etc. by analysis of log traces, and to propagate the results back to the modelling environment, in terms of annotations.

The trace must provide information about threads scheduling; it must be a textual file with entries in the following format:

<timestamp> <eventId> <instance_name>_<port_name_providing_the_operation>_<operation_name>

where

<eventId> = 0 (running), 1 (ready), 2 (blocked), 3 (sleep), 4 (wakeup)

so the typical states in which a thread can be wrt the scheduling flow.

The trace can start with a number indicating the CPU speed factor (default is 1, if not available). Values obtained from the trace will be divided by the CPU speed factor and considered in micro seconds.

For more information see [2].

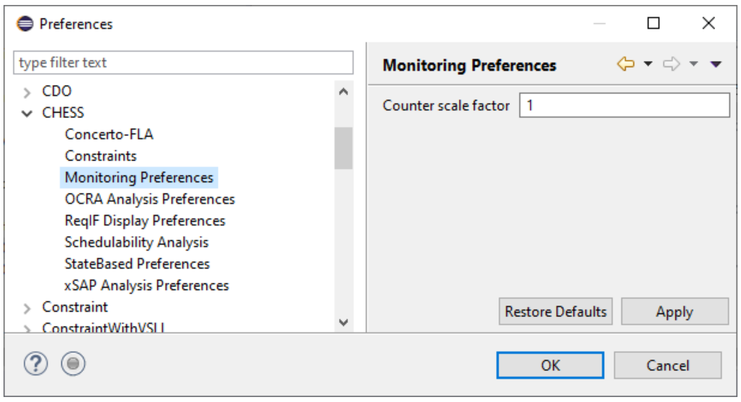
The runtime support may also be used to analyse the execution time of sw components operations, in case of the system implantation has no tasks.

4 Usage

Assumption: component implementations have been modelled in the Component View. The SW system has been modelled as composite component with internal parts typed as component implementations. CHRtSpecifications have been used (at least one) to decorate components implementation operations, in particularly as sporadic or cyclic ones. The instance model of the SW system component has been generated (via the BuildInstance command). See the “CHESS SW Component Guide” for further details on the Sw Components methodology.

Preferences

Use the page below to set the counter scale factor: this factor will be used to scale the values derived from the traces (the value from the traces will be divided by the counter scale factor)

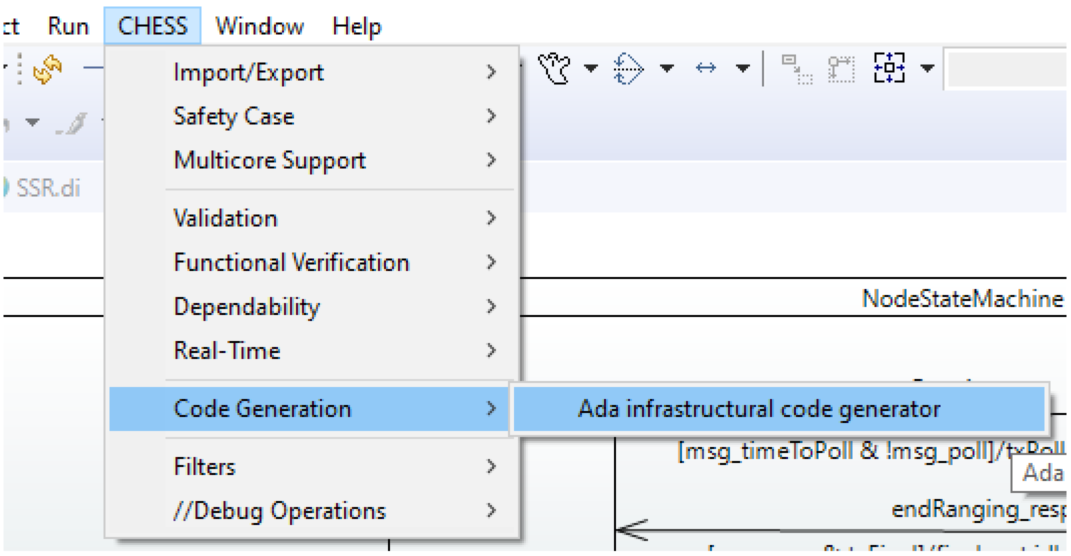


Monitoring Preferences

Counter scale factor 1

Scenario 1 (as implemented in [2])

1. Invoke code generation (see figure below): a dedicated analysis context named *CodeGen<current date>* is automatically generated in the RTAnalysisView, together with the PSM[1] (in the PSMView). The file *CodeGen<current date>_monitoring.xml* is also generated with the list of designed tasks (according to the available CHRtSpecifications)



2. Once the trace .xml file is available, right click on it and select CHESS->Monitoring Trace Analysis. This step generated the *traces* folder and the .xml (“analysisContext name” + “_monitoring.xml”) file to be used for the back propagation
3. right click the generated .xml file and select CHESS->Monitoring BackPropagation to import the results.
4. Open CHESS, go to the PSM analysis context, check the SaStep of the <<SaEndToEndFlow>> corresponding to the monitored operation, in particular its execTime

[1] The PSM in CHESS is a Package stereotyped as <<PSMPackage>>, available in the PSMView. The PSMPackage stereotype comes with an attribute referring the analysis context which has been used to invoke the command which generated the PSM itself. Indeed the PSM is automatically generated, from the invocation of scheduling analysis and code generation.

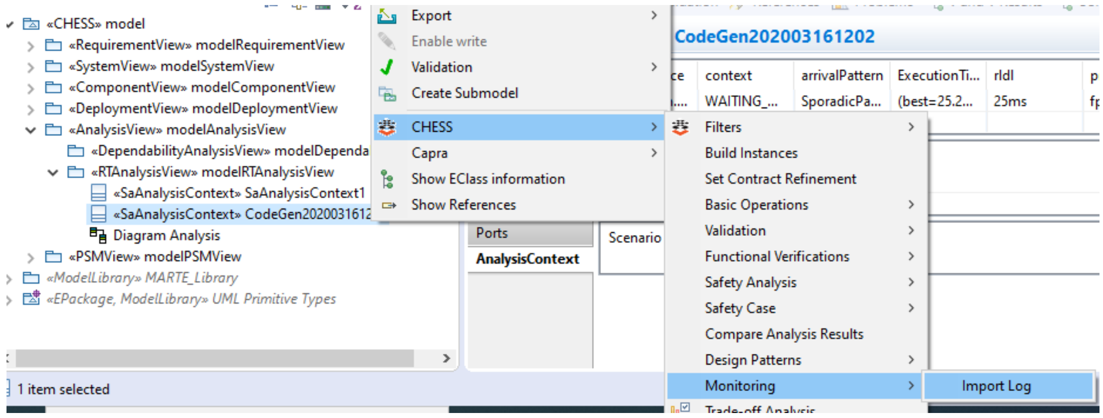
5 Usage in case no tasks are defined

The runtime support may be used to analyse the execution time of an operation in case of no scheduling system/no tasks: in this case the analysis of traces with event of type 3 and 4 only are used (4 is used as a *start operation* event and 3 as an *end operation* event).

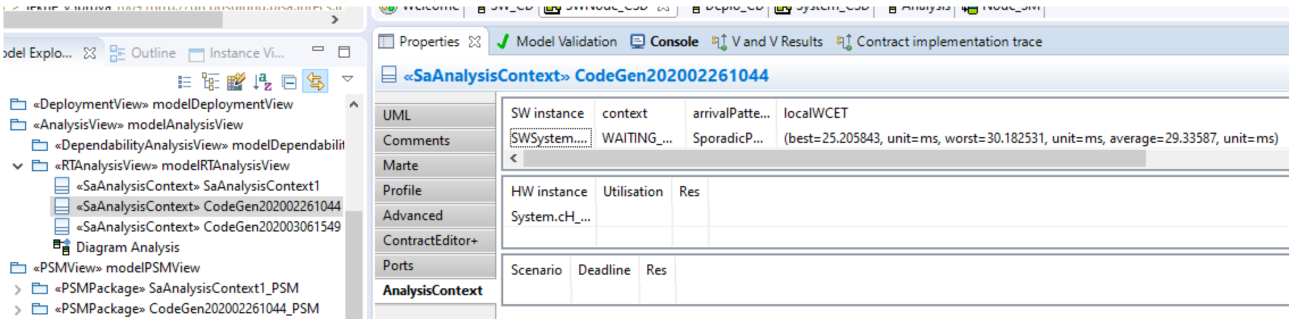
Note: **only WCET back propagated values can be considered significant!!!** This is due to how the log is generated at runtime; in particular the timestamp counter is frequently restarted and this has implication on the way e.g. arrival time is currently calculated.

Scenario 2 (provided for systems without task activation)

1. See step 1 of scenario 1
2. Right click the analysis context generated from point 1 and select CHESS→Monitoring→Import Log. Use the wizard to select the log file (no restriction on the extension of the log file is applied)



3. The tool automatically execute the analysis and the back propagation of the results on the current model. The tool assumes that the *monitoring.xml* file with the list of the thread is available under the *src-gen_model directory*, as generated by the code generation step
4. Select the analysis context generated from point 1 and check the AnalysisContext tab of the properties view to see the imported results about tasks (WCETs, response time, blocking time)


odel Explo...
Outline
Instance Vi...
«DeploymentView» modelDeploymentView
«AnalysisView» modelAnalysisView
«DependabilityAnalysisView» modelDependabilit
«RTAnalysisView» modelRTAnalysisView
«SaAnalysisContext» SaAnalysisContext1
«SaAnalysisContext» CodeGen202002261044
«SaAnalysisContext» CodeGen202003061549
Diagram Analysis
«PSMView» modelPSMView
«PSMPackage» SaAnalysisContext1_PSM
«PSMPackage» CodeGen202002261044_PSM
Properties
Model Validation
Console
V and V Results
Contract implementation trace
«SaAnalysisContext» CodeGen202002261044
UML
Comments
Marte
Profile
Advanced
ContractEditor+
Ports
AnalysisContext

SW instance	context	arrivalPatte...	localWCET
SWSystem....	WAITING_...	SporadicP...	(best=25.205843, unit=ms, worst=30.182531, unit=ms, average=29.33587, unit=ms)

HW instance	Utilisation	Res
System.cH_...		

Scenario	Deadline	Res

6 References

[1] https://www.polarsys.org/chess/publis/CHESSToolset_UserGuide.pdf.

[2] CONCERTO, «D4.10 – Run-time Monitoring mechanisms for multicore systems – Toolset Final Version», www.concerto-project.org/, 2015.