

RHAPSODY TAU INTEGRATION

1. INTRODUCTION

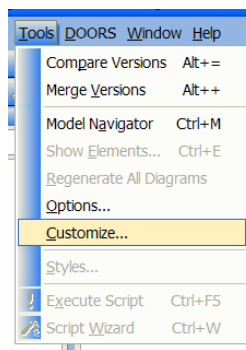
This document describes how to import models created in Tau into Rhapsody and export Rhapsody models to Tau.

2. GETTING STARTED

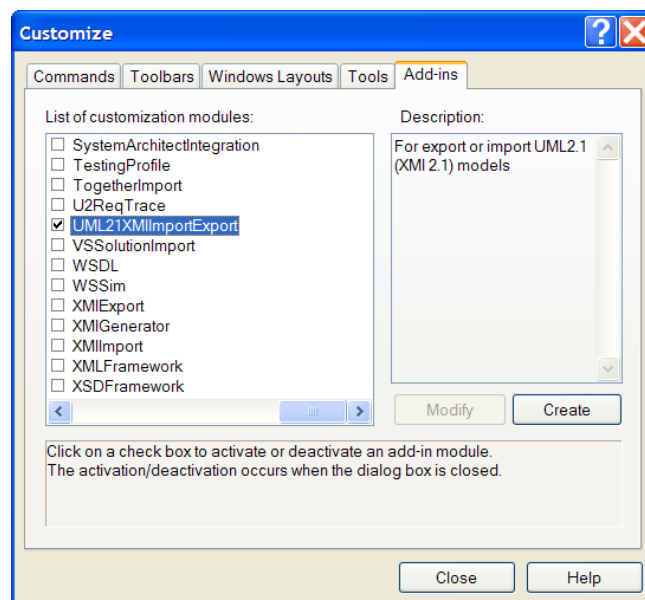
2.1. Importing from Tau

First of all, you have to export the Tau model to Rhapsody XMI from Tau:

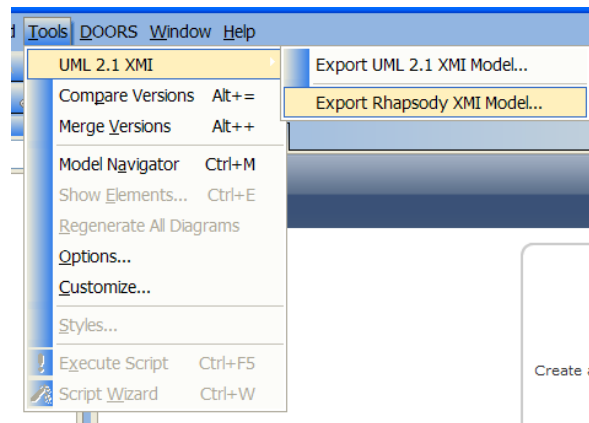
- Start Tau and create a new project, or use an existing project.
- Activate the addin through the menu *Tools > Customize...*



- Then in the *Add-ins* tab, check the *UML21XMIImportExport* addin and click on Close.



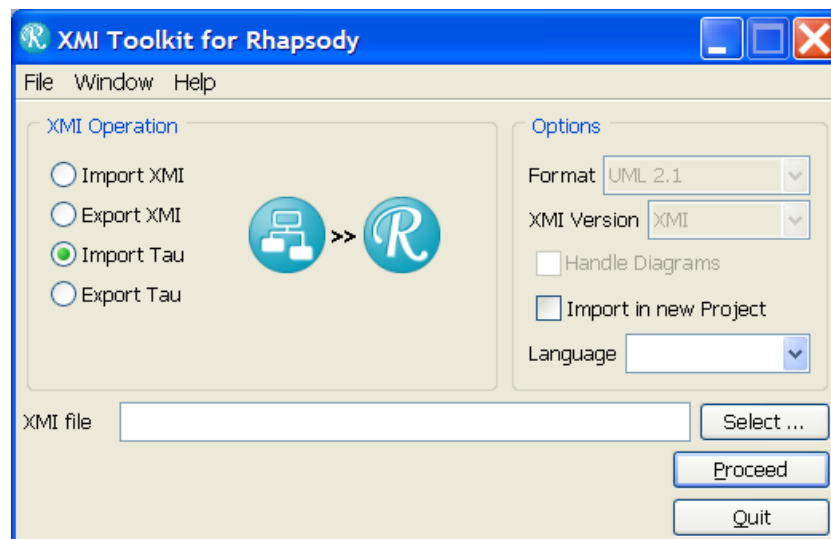
- Launch the export by selecting menu *Tools > UML 2.1 XMI > Export Rhapsody XMI Model...*



- Select a location .xmi file and click on *Save*.

To import a Tau model into Rhapsody, use the XMI Toolkit (*Tools > Import XMI into Rhapsody*).

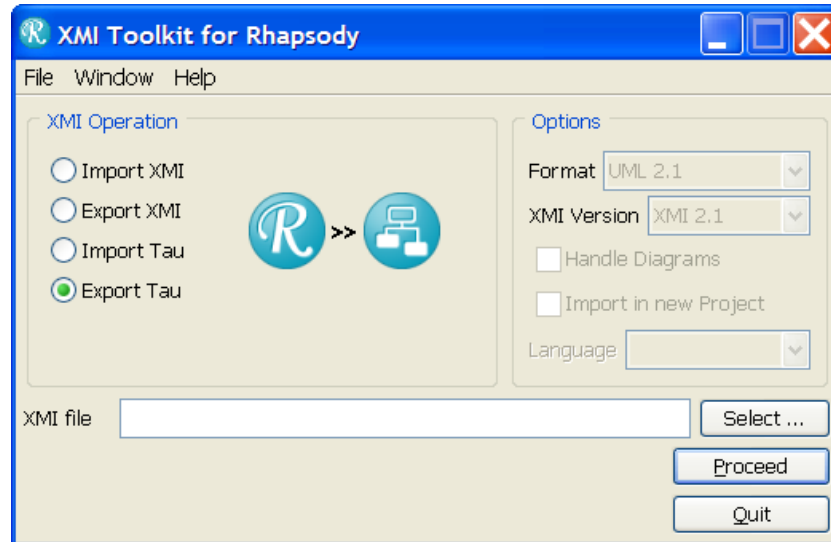
Select the *Import Tau* radio button, to import your Tau XMI file for Rhapsody into Rhapsody.



2.2. Exporting to Tau

To export a Rhapsody model into Tau, use the XMI Toolkit (*Tools > Export XMI into Rhapsody*).

Select the *Export Tau* radio button, to export your Rhapsody project in a XMI file for Tau.



3. RULES

The rules are grouped by diagram type so that users using a subset of the diagram types available in UML can concentrate on the rules they are more specifically interested in.

3.1. General

TAU	Rhapsody
Comment <ol style="list-style-type: none">1. text2. modelElement	Comment <ol style="list-style-type: none">1. specification2. anchoredElements
InformalConstraint <ol style="list-style-type: none">1. informalConstraintOwner2. first informalEntityFragment	Constraint <ol style="list-style-type: none">1. ConstraintsByMe2. Specification
TypeTemplateParameter <ol style="list-style-type: none">1. defaultType	TemplateParameter <ol style="list-style-type: none">1. unmanaged
Signature <ol style="list-style-type: none">1. formalTemplateParameter	Operation Classifier <ol style="list-style-type: none">1. templateParameters
Typed <ol style="list-style-type: none">1. Type2. first multiplicity = ClosedRange3. first multiplicity = OpenRange [no operator or “==”]4. first multiplicity = OpenRange [operator = “>=”]5. first multiplicity = OpenRange [operator = “>”]	Depends of each elements <ol style="list-style-type: none">2. multiplicity (String)

3.2. Package & Class Modeling

TAU	Rhapsody
Session <ol style="list-style-type: none">1. ownedMember	Project
Package <ol style="list-style-type: none">1. ownedMember2.<ul style="list-style-type: none">• <i>Attributes</i>• <i>Operations</i>	Package <ol style="list-style-type: none">1. depends of type2.<ul style="list-style-type: none">• Variables• Functions
Package <<profile>> <ol style="list-style-type: none">1. ownedMember	Profile
Association <ol style="list-style-type: none">1. isDerived2. associationEnd	N/A <ol style="list-style-type: none">1. N/A2. Relation
Dependency	Dependency
Dependency <<import>> <ol style="list-style-type: none">1. client = Namespace2. supplier = Package	<<import>> Dependency

Generalization 1. child 2. parent	Generalization 1. derivedClass 2. baseClass
Generalization 1. child != Interface 2. parent = Interface	Generalization 1. derivedClass 2. baseClass
Class	Class
Interface	Class <<Interface>>
Attribute 1. aggregation = « reference » 2. aggregation = « aggregate » 3. aggregation = « composite » 4. ownerScope = « classifier » 5. ordering = « ordered » 6. isDerived 7. changeability = frozen 8. unNavigableOwner	Attribute 1. unmanaged 2. unmanaged 3. N/A 4. isStatic 5. unmanaged 6. N/A 7. isConstant 8. always naviguable (never linked to any association)
Operation 1. ownerScope = « classifier » 2. inlineMethod 3. inlineMethod = Interaction 4. generalization.parent 5. parameter	Operation 1. isStatic=true 2. unmanaged 3. unmanaged 4. N/A 5. arguments
Parameter 1. ordering = « ordered »	Argument (in an Operation) 1. N/A
DataType without literals	Type
DataType with literals 1. literal	Type (kind = enumeration) 1. enumerationLiterals
Literal 1. expression	EnumerationLiteral 1. value
Choice	Class
Timer	Class
Save	Class
Syntype	Type
Signal 1. parameter	Event <<Signal>>
SignalList	Event<<Signal>>
Collaboration	Collaboration
Port 1. isBehaviorPort 2. realized & required	Port 1. isBehavioral 2. providedInterfaces & requiredInterfaces

3.3. UseCase Modeling

TAU	Rhapsody
Attribute <<actor>> 1. namespace	Actor 1. owner

Dependency <<include>> 1. client = UseCase 2. supplier = UseCase	Dependency <<include>> 1. client = UseCase 2. supplier = UseCase
Dependency <<extend>> 1. client = UseCase 2. supplier = UseCase	Dependency <<extend>> 1. client = UseCase 2. supplier = UseCase
Performance 1. source 2. target	Association (2 Relations)

3.4. Activity Modeling

TAU	Rhapsody
AcceptEventNode	AcceptEventAction
ActionNode	Action
ActivityDecisionNode 1. expression	Connector (connectorType = Condition)
ActivityEdge 1. activityPartition	Transition 1. Not partition in RHP, but hierarchy
ActivityFinalNode	State(LocalTermination)
ActivityImplementation 1. activityPartition	Flowchart 1.
ActivityPartition 1. subPartition	unmanaged
ConnectorNode 1. label	Junction Connector
FlowFinalNode	Termination State
ForkNode	Connector (connectorType= Fork)
InitialNode	State (isRoot = true)
ObjectNode	State (stateType = objectflow)
SendSignalNode 1. inlineActivity.inlineMethod.action. expression.called	SendSignalAction

3.5. Scenario Modeling

TAU	
Interaction 1. participant	Collaboration (linked to a SequenceDiagram) 1. ClassifierRole
LifeLine 1. involvedFragment 2. selector = IndexExpr 3. selector = Ident 4. selector = any other expression 5. type	ClassifierRole 1. unmanaged 2. unmanaged 3. unmanaged 4. unmanaged 5. formalClassifier
InteractionOperand	InteractionOperand
CombinedFragment 1. operator = « alt »	InteractionOperator 1. interactionType

2. operator = « assert » 3. operator = « break » 4. operator = « loop » 5. operator = « neg » 6. operator = « opt » 7. operator = « par » 8. operator = « region » 9. operator = « seq » 10. operator = « strict »	
CombinedFragment 1. operator = « consider » 2. operator = « ignore »	unmanaged
CompoundActionOccurrence	ConditionMark
ContinuationFragment	unmanaged
InteractionOccurrence 1. operation.inlineMethod	InteractionOccurrence 1. referenceSequenceDiagram
Message 1. constraint = MessageConstraint 2. constraint = any other constraint 3. data.expression 4. data.instanceOf = Signal 5. data.instanceOf = Operation	Message 1. unmanaged 2. unmanaged 3. unmanaged
Destroy	DestructionEvent
MessageReceive	MessagePoint
MessageSend	MessagePoint
InstanceCreateReceive	unmanaged
InstanceCreateSend	unmanaged
InstanceDelete	unmanaged
OperationCallReceive	unmanaged
OperandCallSend	unmanaged
OperationReplyReceive	unmanaged
OperationReplySend	unmanaged
GuardedConstraint 1. guard	unmanaged
ElseConstraint	unmanaged
LoopConstraint 1. minNbrOfIterations 2. maxNbrOfIterations 3. hiddenStereotypeInstance contains « AsteriskLoopConstraint »	unmanaged
Gate	unmanaged

3.6. Behavioral Modeling

TAU	Rhapsody
DecisionAction 1. expression 2. container transition	Connector (isConditionConnector)
DecisionAnswerTransition 1. first range	Transition

2. action	
JoinAction 1. joinedLabel.action	
LabelTransition	Transition
MultiState	State
NextStateAction 1. nextStateKind = « history » 2. nextStateKind = « deepHistory » 3. nextStateKind = « normal »	N/A 1. Connector (ConnectorKind = History) 2. Connector (ConnectorKind = History) 3. N/A
ReturnAction	State(LocalTermination)
SimpleStateMachine 1. state 2. transition	Statechart 1. rootstate made from the region
StartTransition 1. action	Transition (isDefaultTransition=1) 1. action
State 1. inlineStateMachine 2. doAction 3. entryAction 4. exitAction	State 1. nestedStatechart 2. unmanaged 3. entryAction 4. exitAction
StateMachine 1. parameter	Statechart 1. unmanaged
StopAction	Connector (isTerminateConnector = true)
Trigger 1. definition = Signal 2. definition = Operation	Trigger 1. (isEvent) 2. depends of the Operation
TimeTrigger 1. when	Trigger 1. (isTimeout) body
TriggeredTransition 1. asteriskTrigger = true 2. action 3. guard	Transition 1. trigger

3.7. Component Modeling

TAU	Rhapsody
Class <<component>>	Class
Connector 1. connectorEnd	Link 1. from/to
ConnectorEnd 1. port 2. part	Link 1. fromPort/toPort 2. from/to

3.8. Deployment Modeling

Tau	Rhapsody
Artifact 1. namespace	Module (+ nestedClass)
Class <<deploymentSpecification>>	Module

Class <<node>>	Class
Class <<executionEnvironment>>	Class
Dependency <<manifest>> 1. supplier 2. client = Artifact	unmanaged
Dependency <<deploy>> 1. supplier = Node or ExecutionEnvironment 2. client = Artifact	unmanaged

3.9. Extensibility

Tau	Rhapsody
Extension 1. stereotype 2. range 3. extended	N/A Stereotype with ofMetaClass
Stereotype	Stereotype/tag

3.10. Graphical mapping from Tau to Rhapsody

Tau	Rhapsody
ClassDiagram	ObjectModelDiagram
ComponentDiagram	ComponentDiagram
PackageDiagram	ObjectModelDiagram
UseCaseDiagram	UseCaseDiagram