QtModeling: Metamodelagem e Model-Driven Software Engineering com Qt5

Sandro S. Andrade

sandroandrade@kde.org / @andradesandro

FISL 15 – Fórum Internacional de Software Livre – Porto Alegre – Maio/2014

Introdução

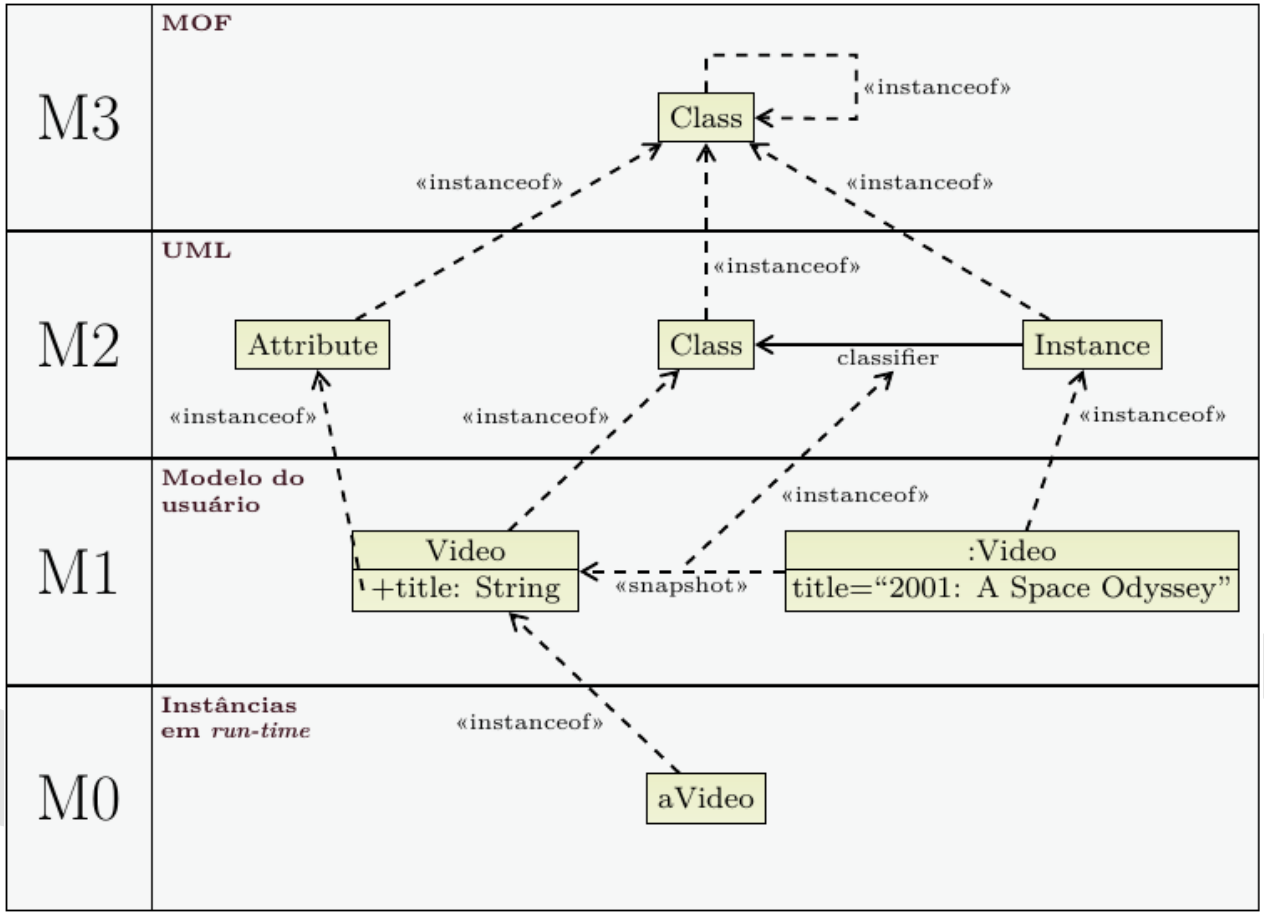
- Motivos para uso de modelos na Engenharia de Software:
  - Documentação
  - Geração automática de código / sincronização modelo-artefatos
  - Recuperação Arquitetural e Verificação de Conformidade
  - Gerência de Qualidade e Integração Contínua
  - Predição de Atributos de Qualidade / Simulação
  - Domain-Specific Languages (DSLs)
  - Models@run.time

Introdução

- Diferentes notações → diferentes propósitos → diferentes graus de formalidade:
  - Linguagem natural, gráficos informais, UML, Architecture Description Languages (ADLs)
- Linguagens de modelagem baseadas em metamodelos:
  - Metamodelo → Linguagem de Modelagem → Modelo
  - MOF → MOF → {UML, CWM, DuSE} → seu modelo

Introdução

- Exemplo de stack de metamodelagem:



Introdução

- Model-Driven Software Engineering (MDSE)
- Uma plataforma para MDSE deve ser capaz de:
  - Permitir a criação, manipulação e transformação de modelos descritos em diversas linguagens de modelagem
  - Permitir a criação de novas linguagens de modelagem (de domínio específico)
  - Viabilizar a integração do uso de modelos nas atividades que compõem o processo de desenvolvimento

Introdução

- Cenário exemplo 1:
  - Configuração de um exemplar de uma linha de produto (ex: novo sistema embarcado para smartphones) a partir do uso de modelos
  - Análise da arquitetura resultante (teste arquitetural antecipado)
  - Subsequente geração de código ou artefatos de integração

Introdução

- Cenário exemplo 2:
  - Uma arquitetura prescrita foi projetada pelo arquiteto
  - Constantes evoluções/manutenções podem degradar a arquitetura
  - Deseja-se integrar o uso de modelos nos scripts de Integração Contínua de modo a detectar antecipadamente possíveis desvios/erosões arquiteturais

Qt5

- Lançado oficialmente em Dezembro de 2012
- Qt-project.org
- Amplo conjunto de features:
  - Widgets, Multimídia, Network, Banco de Dados, OpenGL, Sensors, XML, dentre outras
- Plataformas suportadas:
  - Windows, Linux/X11, Max OS X, QNX, VxWorks, Android, iOS
- Nenhum suporte a uso de modelos (até então)

Qt5

QtModeling

- Conjunto de módulos add-on do Qt5 para MDSE
- Desenvolvido upstream
- http://qt-project.org/wiki/QtModeling
- git clone git://gitorious.org/qt/qtmodeling.git

QtModeling

- Objetivos:
  - Alta conformidade com os padrões do Object Management Group (OMG)
  - API Qt-ish para criação e manipulação de modelos:
    - Fácil compreensão
    - Boa produtividade
  - Excelente desempenho em múltiplas plataformas
  - Suporte por ferramenta (DuSE-MT):
    - Extensibilidade

QtModeling - API

```
QUmlModel *model = new QUmlModel("MyModel");
QUmlPackage *package = new QUmlPackage("Package1");
QUmlPrimitiveType *stringType = new QUmlPrimitiveType("String");
QUmlClass * class_ = new QUmlClass("Student");
class_->setAbstract(false);
QUmlProperty *property = new QUmlProperty(“name”);
property->setType(stringType);
property->setFinal(true);
... (cont.)
```

QtModeling - API

... (cont.)

```
package->addOwnedType(class_);
model->addPackagedElement(package);
model->addOwnedType(primitiveType);
// query model or
// perform 'well-formed/sanity/architectural conformance'
// checks
delete model;
```

QtModeling – Serialização XMI

```
QFile file("test.xmi");
if (!file.open(QFile::WriteOnly | QFile::Text)) {
    qDebug() << "Cannot write file !";
    return 1;
}
QXmiWriter writer(model);
writer.writeFile(&file);
file.close();
```

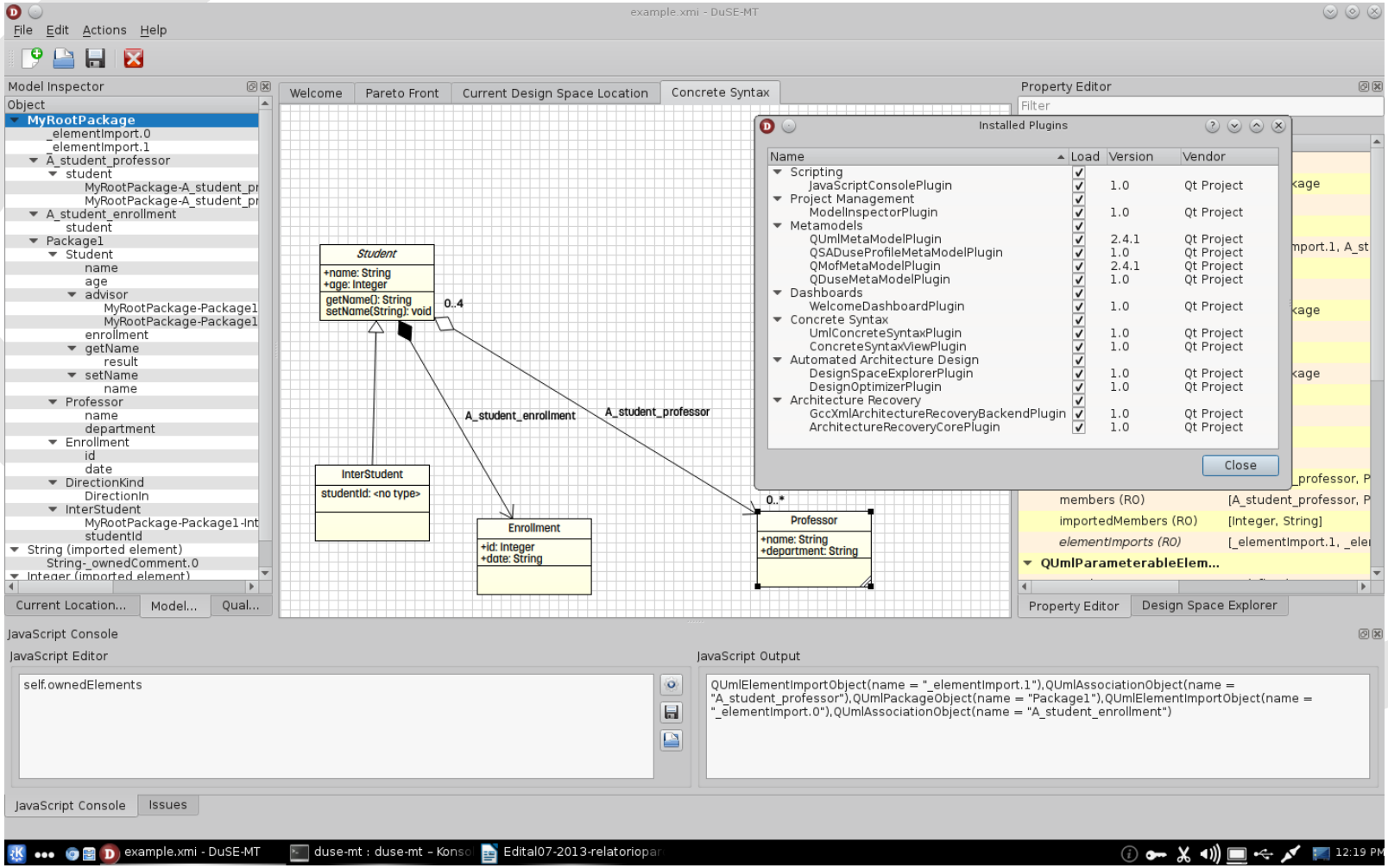
QtModeling

- Módulos disponíveis:
  - QtModeling
  - QtModelingWidgets
  - QtMof + metamodel plugin
  - QtUml + metamodel plugin
  - QtDuSE + metamodel plugin
  - QtSADuSEProfile + metamodel plugin

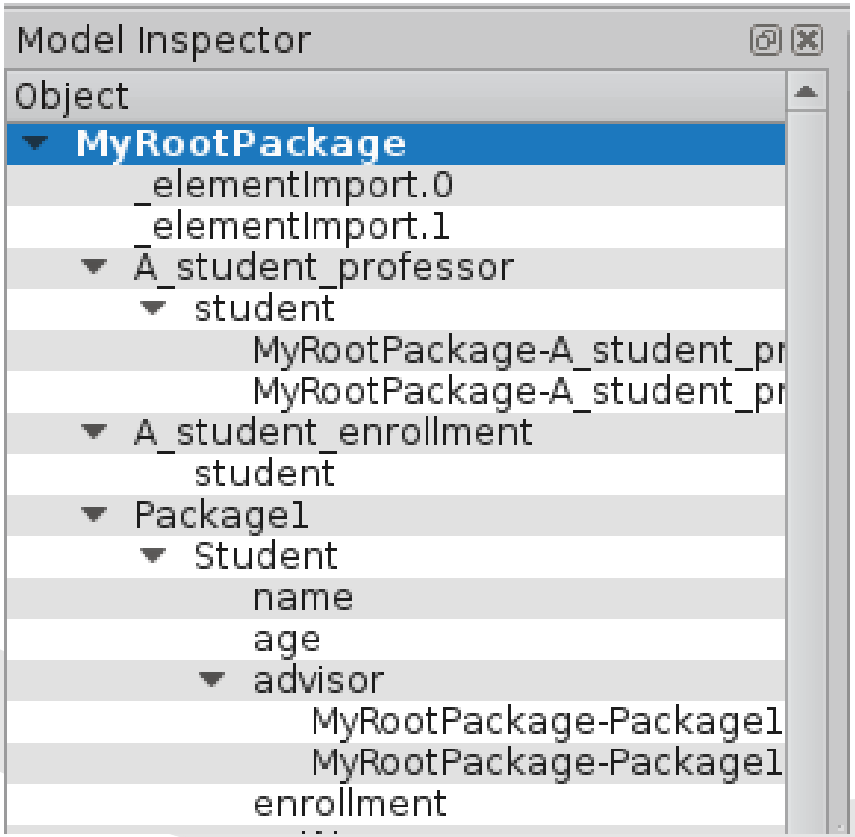
DuSE-MT

- Nasceu como um exemplo de uso do QtModeling
- Evoluiu para uma ferramenta de MDSE (duse.sf.net)
- Objetivos de design:
  - Metamodel-agnostic (metamodelos/linguagens são plugins)
  - Acesso e manipulação de modelos via linguagens interpretadas
  - Amplo editor de propriedades
  - Implementação declarativa de sintaxes concretas

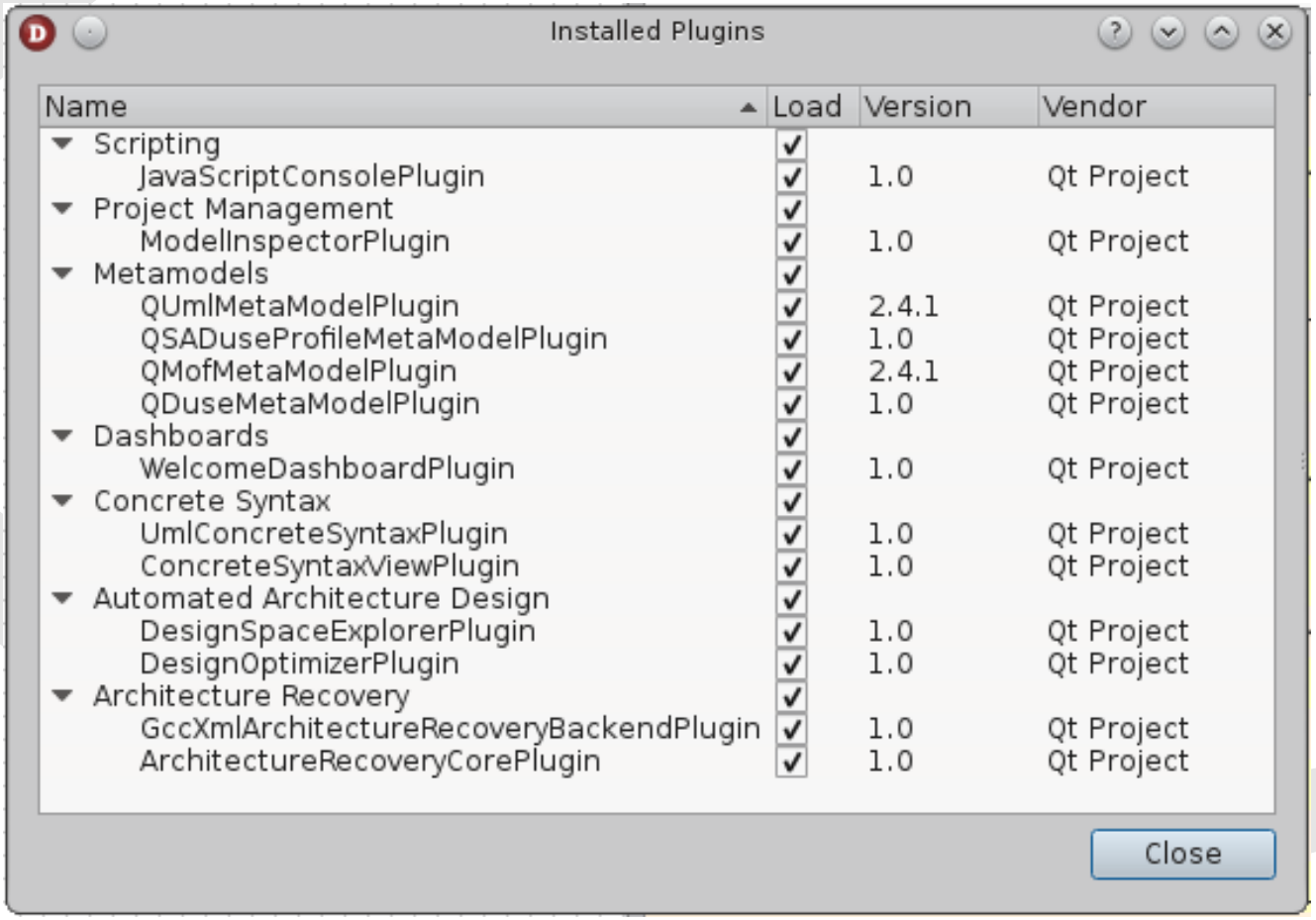
DuSE-MT


example.xmi - DuSE-MT
File Edit Actions Help
Model Inspector
Object
MyRootPackage
_elementImport.0
_elementImport.1
A_student_professor
student
MyRootPackage-A_student_pr
MyRootPackage-A_student_pr
A_student_enrollment
student
Package1
Student
name
age
advisor
MyRootPackage-Package1
MyRootPackage-Package1
enrollment
getName
result
setName
name
Professor
name
department
Enrollment
id
date
DirectionKind
DirectionIn
InterStudent
MyRootPackage-Package1-Int
studentId
String (imported element)
String-_ownedComment.0
Integer (imported element)
Current Location...
Model...
Qual...
Welcome
Pareto Front
Current Design Space Location
Concrete Syntax
Student
+name: String
+age: Integer
getName(): String
setName(String): void
0..4
A_student_enrollment
A_student_professor
InterStudent
studentId: <no type>
Enrollment
+id: Integer
+date: String
0..*
Professor
+name: String
+department: String
Property Editor
Filter
Installed Plugins
Name
Load
Version
Vendor
Scripting
JavaScriptConsolePlugin 1.0 Qt Project
Project Management
ModelInspectorPlugin 1.0 Qt Project
Metamodels
QUmlMetaModelPlugin 2.4.1 Qt Project
QSADuseProfileMetaModelPlugin 1.0 Qt Project
QMofMetaModelPlugin 2.4.1 Qt Project
QDuseMetaModelPlugin 1.0 Qt Project
Dashboards
WelcomeDashboardPlugin 1.0 Qt Project
Concrete Syntax
UmlConcreteSyntaxPlugin 1.0 Qt Project
ConcreteSyntaxViewPlugin 1.0 Qt Project
Automated Architecture Design
DesignSpaceExplorerPlugin 1.0 Qt Project
DesignOptimizerPlugin 1.0 Qt Project
Architecture Recovery
GccXmlArchitectureRecoveryBackendPlugin 1.0 Qt Project
ArchitectureRecoveryCorePlugin 1.0 Qt Project
Close
members (RO) [A_student_professor, P
importedMembers (RO) [Integer, String]
elementImports (RO) [_elementImport.1, _ele
QUmlParameterableElem...
Property Editor
Design Space Explorer
JavaScript Console
JavaScript Editor
self.ownedElements
JavaScript Output
QUmlElementImportObject(name = "_elementImport.1"),QUmlAssociationObject(name = "A_student_professor"),QUmlPackageObject(name = "Package1"),QUmlElementImportObject(name = "_elementImport.0"),QUmlAssociationObject(name = "A_student_enrollment")
JavaScript Console
Issues
example.xmi - DuSE-MT
duse-mt : duse-mt – Konsol
Edital07-2013-relatoriopar
12:19 PM

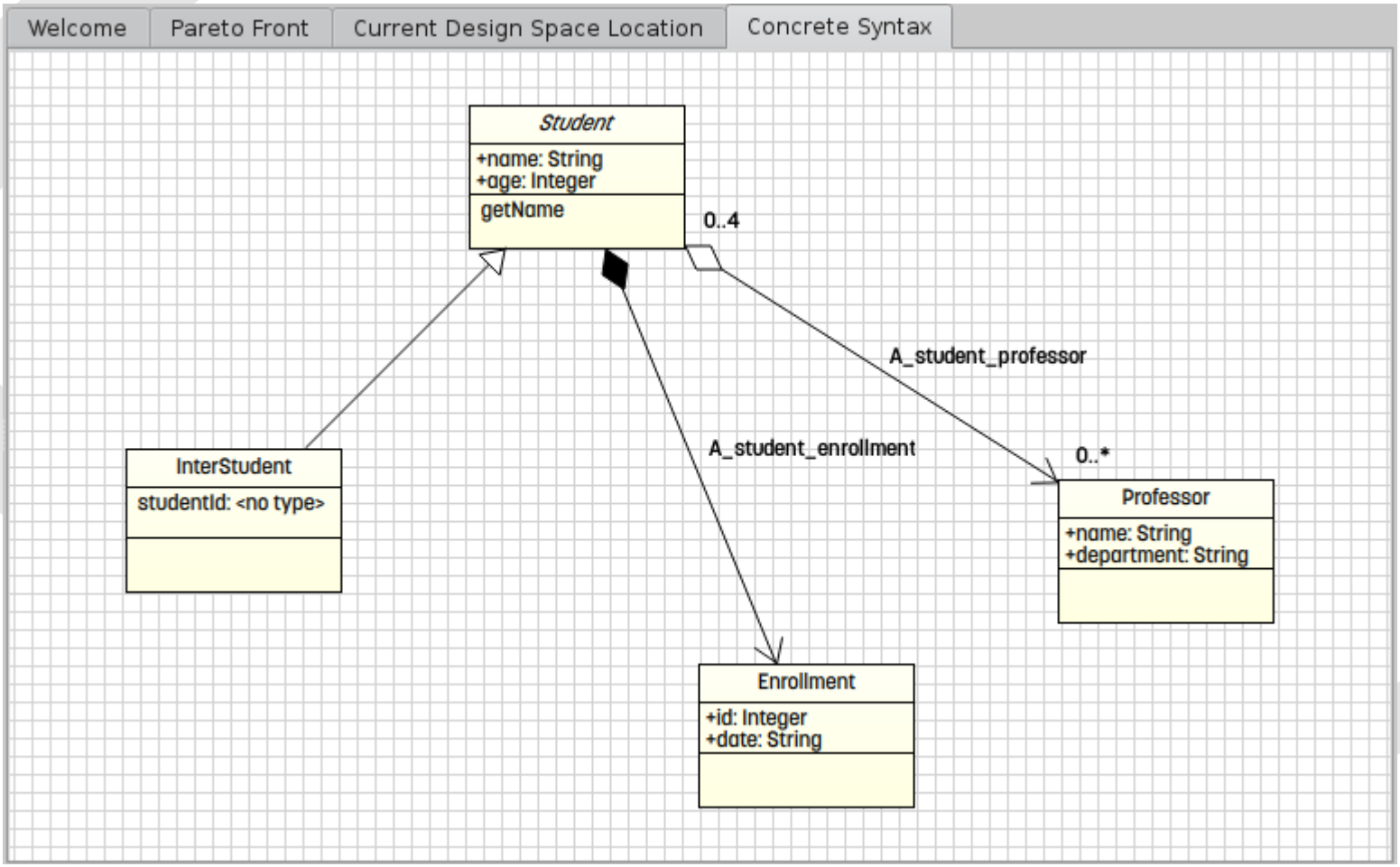
DuSE-MT – Model Inspector



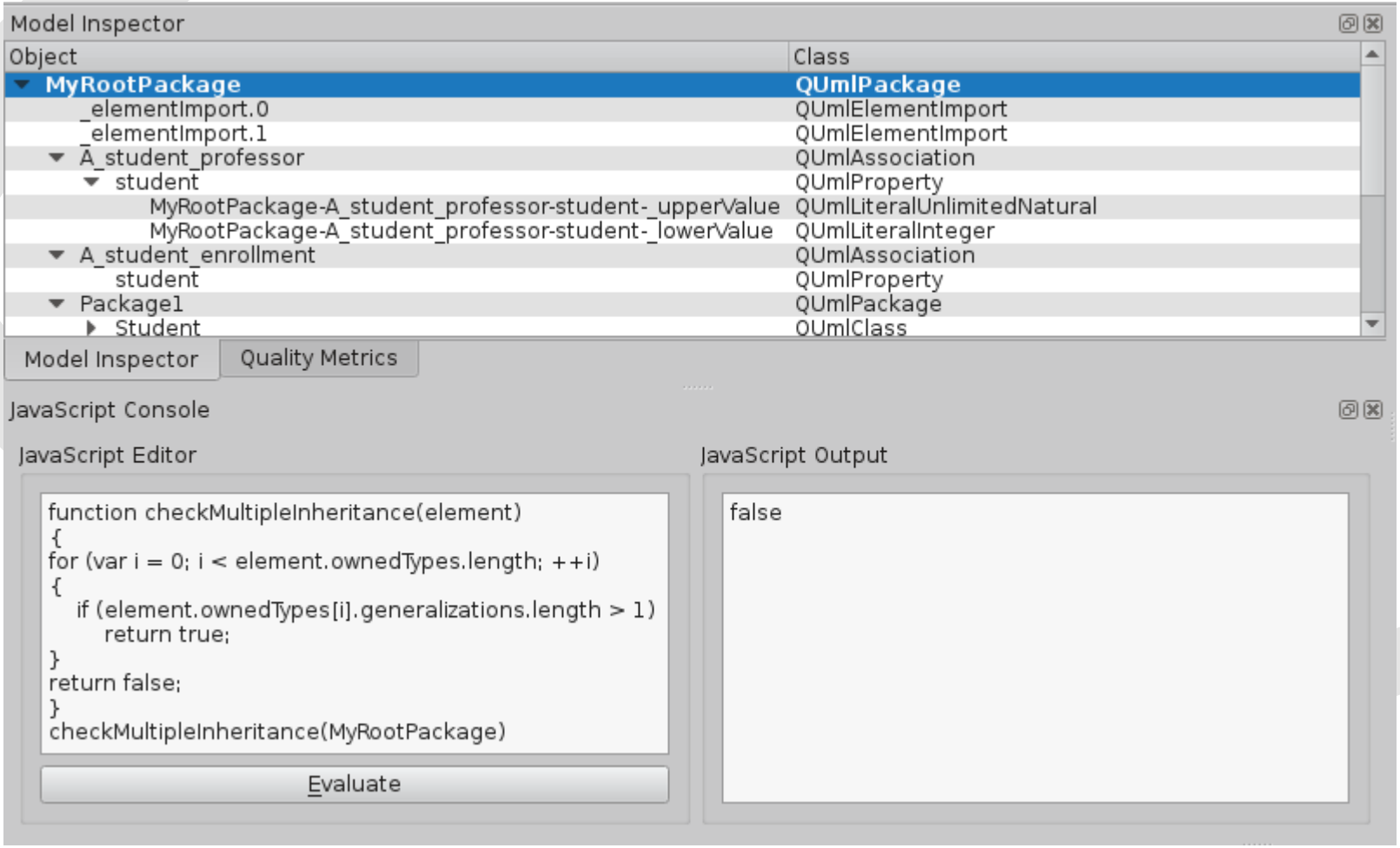
DuSE-MT – Plugins



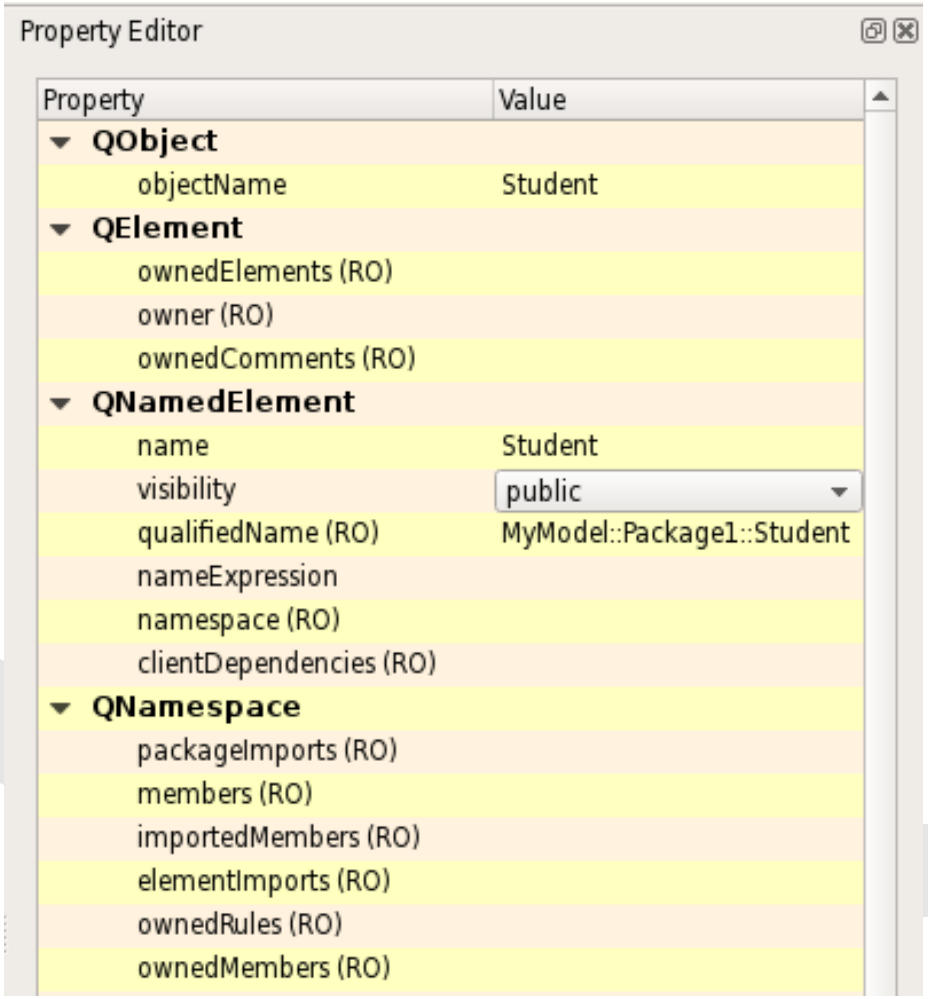
DuSE-MT – Sintaxe Concreta



DuSE-MT – Interpretador JavaScript



DuSE-MT – Property Editor



QtModeling - Desafios

- O metamodelo da UML possui 239 metaclasses
- Destas, 193 são classes concretas
- Muitas heranças múltiplas e dreaded diamonds
  - Estaria tudo ok se estivéssemos implementando em C++ (heranças virtuais) ...
  - ... mas QObject's não gostam de heranças múltiplas (muito menos virtuais)
  - Solução: QObject-free implementation + QObject-based wrappers

QtModeling - Desafios

- 91.83% do código do metamodelo UML foi gerado automaticamente:
  - OMG XMI → Perl Template Toolkit → QtModeling templates → QtUml/QtMof/Qt<your-language> implementation
- Exceções: propriedades derivadas e operações (sendo implementadas sob demanda)
- Futuro: Grantlee para suportar geração automática de implementações de metamodelos
  - Ex: definição de DSLs

QtModeling - Desafios

- A implementação atual possui suporte completo a:
  - Propriedades do tipo Derived (sob demanda) e Derived Union (já implementadas)
  - Propriedades opposite, subsetted e redefined
  - Operações (sob demanda)
  - Enumerações
  - Clonagem de elementos de modelagem
  - Memory ownership para agregações do tipo composite
  - Serialização XMI (metamodel-agnostic)

DuSE-MT - Desafios

- Amplo uso de reflexão computacional, de modo a não depender de um metamodelo particular
- QtScript para viabilização de um Interpretador JavaScript para acesso e manipulação de modelos:
  - Criação de regras de boa-formação pelo próprio usuário
  - Plugins escritos em JavaScript (ou Python, Ruby, etc com o KDE Kross)
- Novos models e views do Qt para acesso aos elementos e propriedades de um elemento

DuSE-MT - Desafios

- Sintaxe concreta:
  - QML to the rescue
  - Porque QML:
    - Linguagem declarativa (boa produtividade, boa compreensão)
    - Execução suportada por hardware (OpenGL)
    - Amplamente utilizada para construção de GUIs/UXs em diversos form factors
    - Tecnologia de preferência para construção de novos widgets para o KDE Plasma

DuSE-MT - Desafios

- Exemplo de sintaxe concreta em QML:

```
import QtQuick 2.0
UmlElement {
    UmlSlot { id: nameSlot; anchors.top: parent.top
        Text { id: label, text: element.name,
                anchors.centerIn: parent
        }
    }
... (cont.)
```

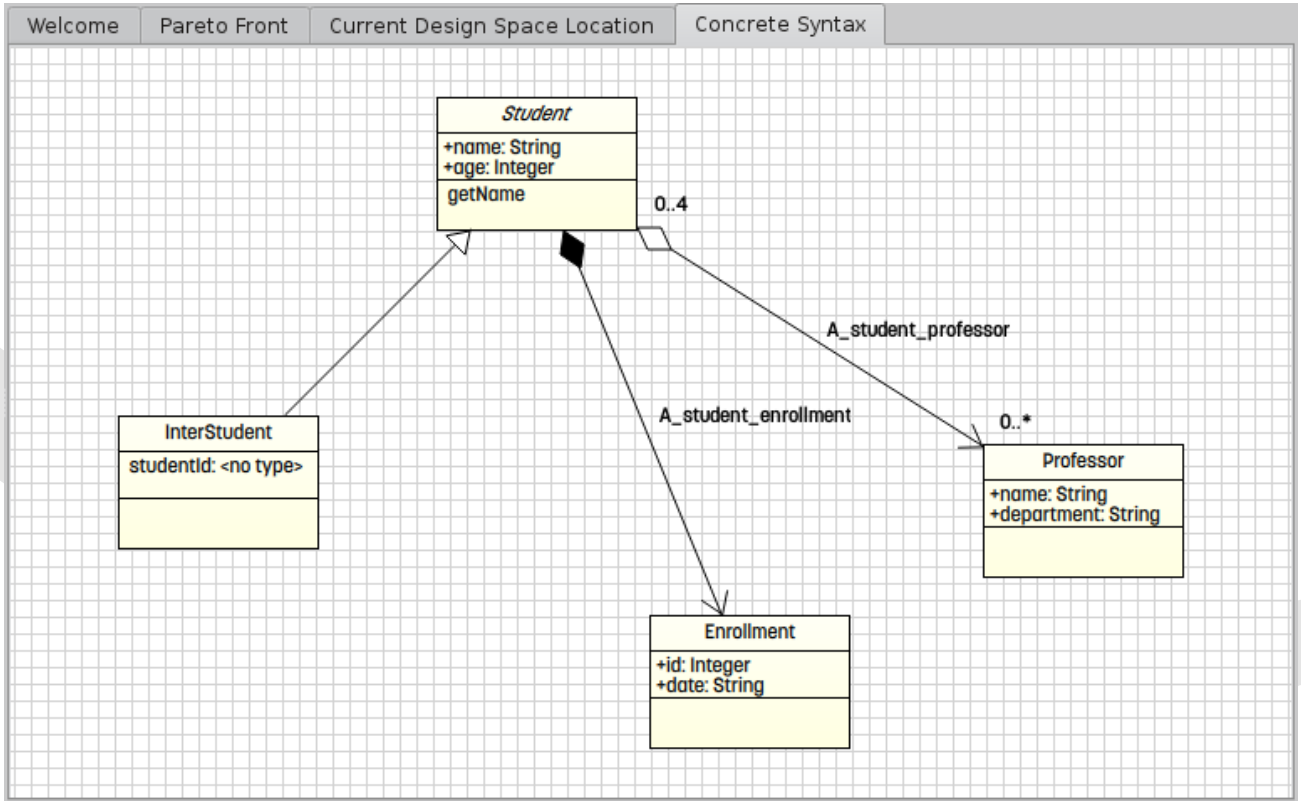
DuSE-MT - Desafios

- Exemplo de sintaxe concreta em QML:

```
UmlSlot { id: attributeSlot
    anchors { top: nameSlot.bottom; topMargin: -1 }
    height: (parent.height - nameSlot.height)/2
    ListView { model: element.ownedAttributes
                    anchors { fill: parent; margins: 4 }
      delegate: Text {
        text: visibility(modelData.visibility) + modelData.name + ": " +
              modelData.type
      }
```

DuSE-MT - Desafios

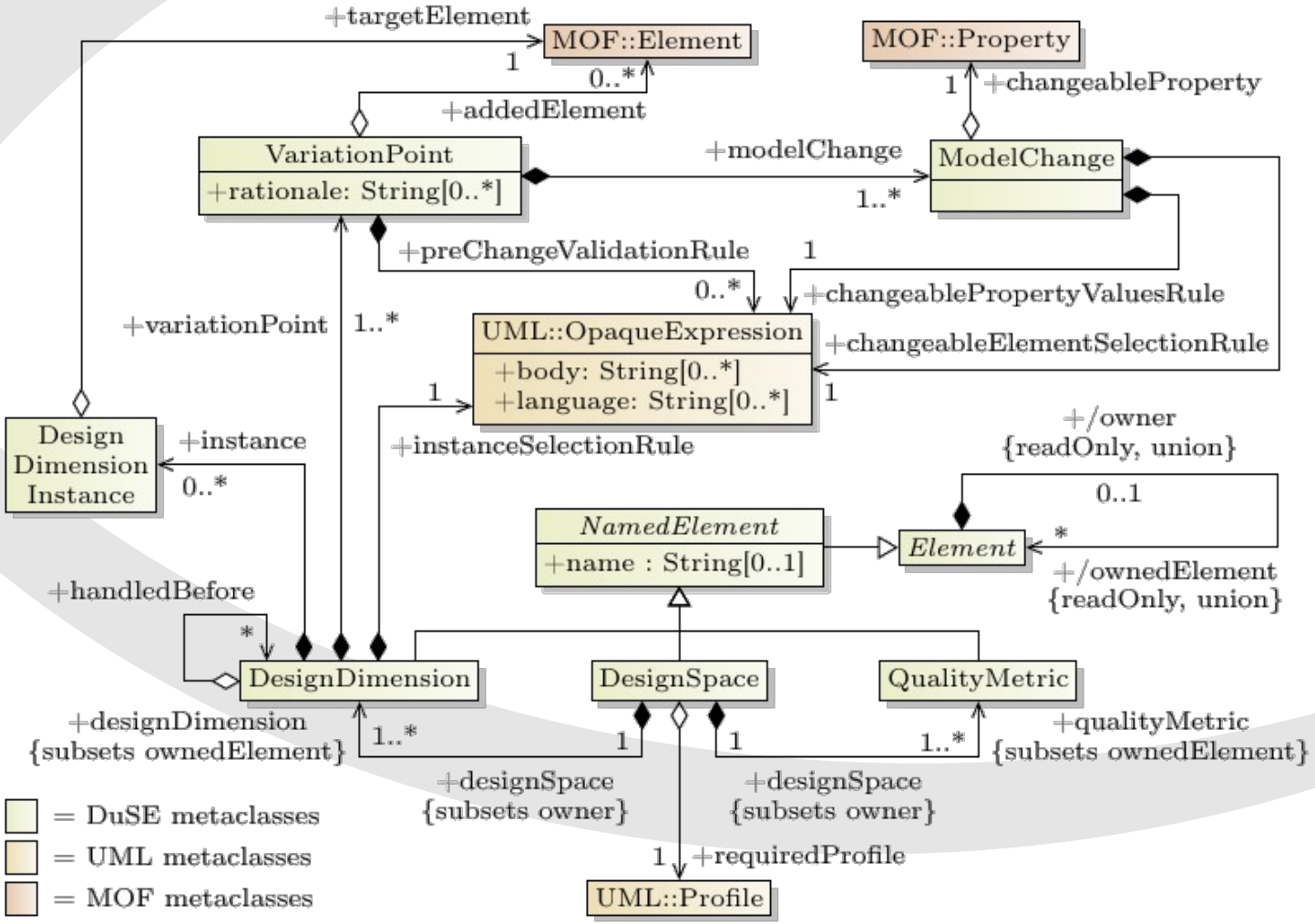
- Resultado:



DuSE

- Processo para projeto arquitetural automatizado de arquiteturas de software
- Linguagem para criação de design spaces arquiteturais específicos de domínio
- Objetivos:
  - Representar conhecimento refinado de design de uma maneira mais sistemática e estruturada
  - Suportar o (re)design automatizado de arquiteturas
  - Suportar otimização multi-objetivo de arquiteturas

DuSE - metamodelo



DuSE - otimização

- Otimizando arquiteturas de software:
  - Trade-offs entre atributos de qualidade são inevitáveis
  - Otimização multi-objetivo com articulação a posteriori de preferências é uma boa forma de abordar o problema
- Novo módulo: QtOptimization
  - Influenciado pelo JMetal
  - Disponibiliza o algoritmo evolucionário para otimização multi-objetivo NSGA-II
  - Retorna um Pareto-front de arquiteturas (near-)optimal

DuSE - aplicações

- SA:DuSE
  - Design Space concreto, especificado em DuSE, que captura as principais dimensões de design no domínio de Sistemas Self-Adaptive
  - Aplicado para gerar automaticamente arquiteturas de controladores para arquiteturas MapReduce e WebServers
- SO:DuSE (em andamento)
  - Idem, para Sistemas Self-Organizing

QtModeling – o futuro

- Comunidade em formação:
  - 1 professor, 3 alunos de pós-graduação, 2 bolsistas PIBIC, 2 alunos de graduação
  - 2 contribuidores iniciantes na Europa e Asia
- Ainda não temos um release :(

QtModeling – o futuro

- Trabalhos futuros:
  - Framework para geração flexível de código
  - Redes sociais para modelagem de software
  - Estilos arquiteturais, recuperação, métricas, model@RT (em andamento)
  - Boost no Umbrello ?
  - Suporte a MDSE no QtCreator e KDevelop ?

QtModeling – o futuro

- O QtModeling é um projeto Qt
- Contribuições são sempre bem-vindas
- Há muito a ser feito !!!

Obrigado ! Perguntas ?

Sandro S. Andrade

sandroandrade@kde.org / @andradesandro

FISL 15 - Fórum Internacional de Software Livre - Porto Alegre - Maio/2014