Projeto GingaForAll

Apresentação: Msc. Diego Saraiva, Lucas Silva
Coordenação: Profa. Dra. Thaís Batista

Universidade Federal do Rio Grande do Norte
Programa de Pós-Graduação em Sistemas e Computação

Sumário I

Sumário II

Objetivos

GingaForAll

- Conceber uma linha de produto para o Ginga-CC orientada a aspectos e a modelos
- Prover ferramentas para gerência automática de variações
- Motivação
  - Suporte a diferentes plataformas de execução com diferentes capacidades de processamento, armazenamento e memória
  - Suporte a diferentes produtos para diferentes parcelas do mercado

Conceitos Básicos

Desenvolvimento de Software Orientado a Aspectos - DSOA

- Emprega a separação de conceitos transversais em todas as fases do ciclo de vida de um software, desde a elicitação de requisitos até a implementação.
- Conceitos transversais
  - São conceitos que encontram-se espalhados e entrelaçados pelo sistema
  - Não são bem modularizados adequadamente pela orientação a objetos
  - Aspectos facilitam a adição e a remoção de funcionalidades

Desenvolvimento dirigido a modelos - Model Driven Development

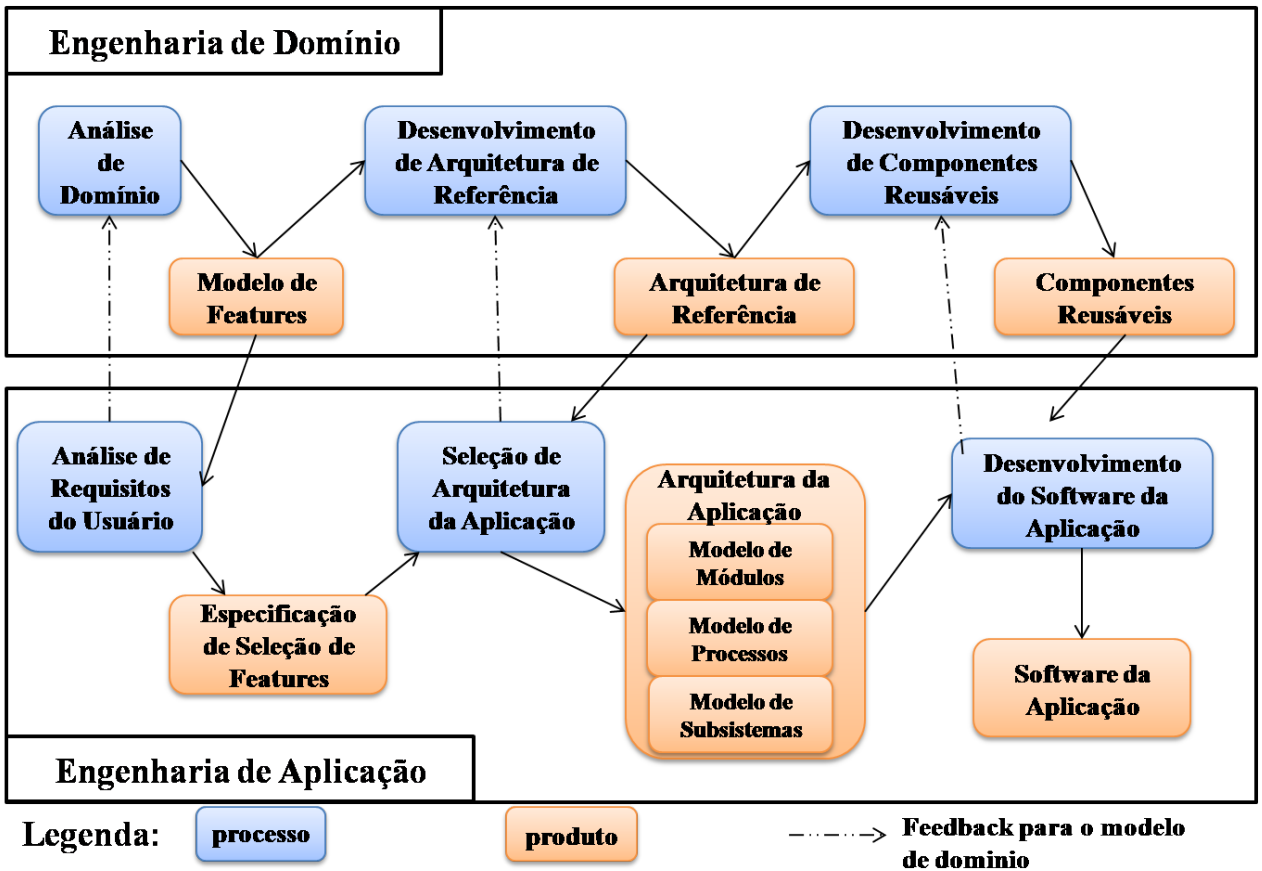
- Permite que um sistema possa ser modelado em diferentes níveis de abstração
- Reuso de conhecimento de domínio
- Utiliza padrões e linguagens especificadas pela OMG
- Aplicação de um conjunto de transformações entre modelos
  - Modelo para modelo
  - Modelo para texto

Linhas de Produto de Software

Uma Linha de Produto de Software (LPS) pode ser definida como: uma fámilia de sistemas que atende um determinado segmento de mercado [Clements e Northorp, 2001].

Objetivos

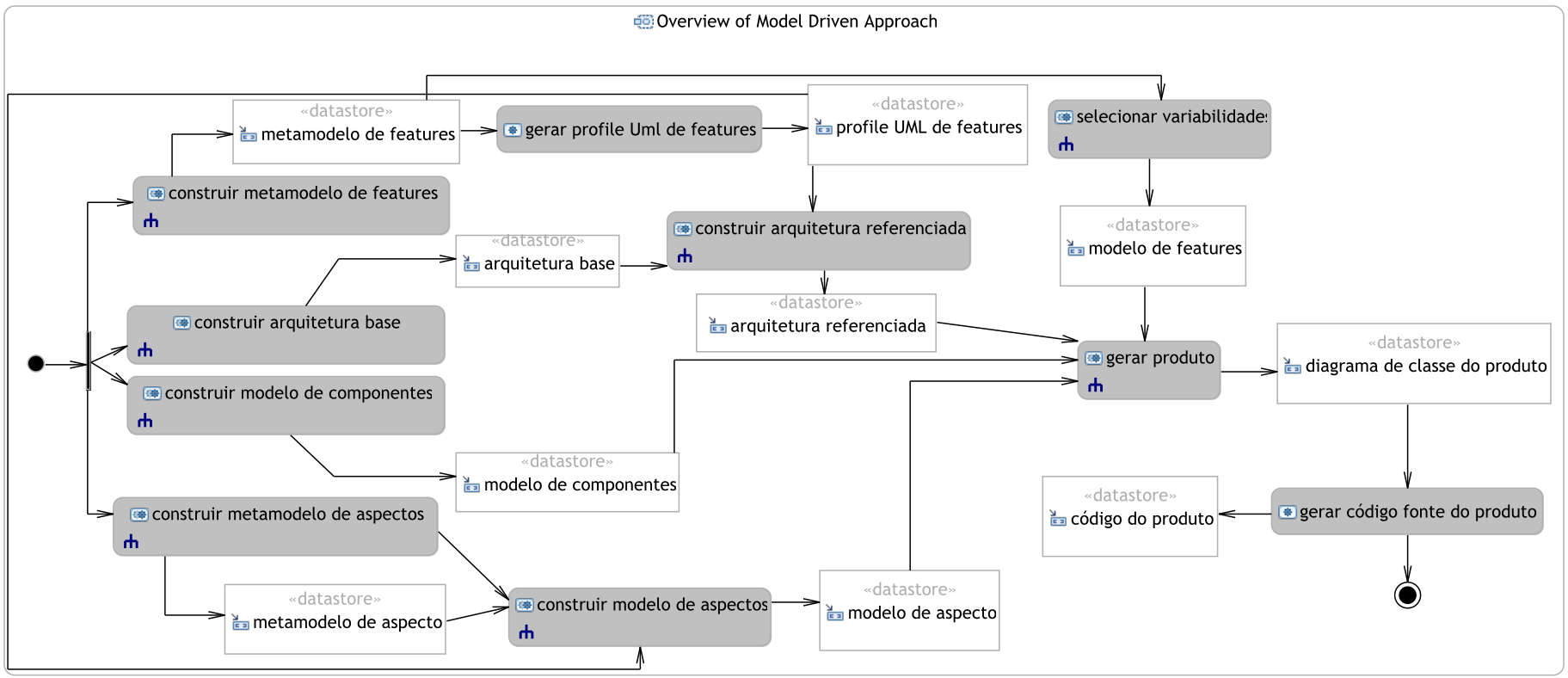
- Promover métodos, técnicas e ferramentas para produção em larga escala de família de produtos relacionados
- Permitir a gerência de variabilidades dos diferentes produtos da LPS


Engenharia de Domínio
Análise de Domínio
Desenvolvimento de Arquitetura de Referência
Desenvolvimento de Componentes Reusáveis
Modelo de Features
Arquitetura de Referência
Componentes Reusáveis
Análise de Requisitos do Usuário
Seleção de Arquitetura da Aplicação
Arquitetura da Aplicação
Modelo de Módulos
Modelo de Processos
Modelo de Subsistemas
Desenvolvimento do Software da Aplicação
Especificação de Seleção de Features
Software da Aplicação
Engenharia de Aplicação
Legenda:
processo
produto
Feedback para o modelo de dominio

Tecnologias utilizadas

Tecnologias utilizadas

- UML 2 - Unified Modeling Language
  - Modelagem dos profiles e modelos
- QVTo - Query/View/Transformation Operational
  - Transformações entre modelos
- EMF - Eclipse Modeling Framework
  - Criação de metamodelos e de editores de modelo
  - Ecore
- Acceleo
  - Transformações de modelo para texto


Overview of Model Driven Approach
«datastore»
metamodelo de features
gerar profile Uml de features
«datastore»
profile UML de features
selecionar variabilidades
construir metamodelo de features
construir arquitetura referenciada
«datastore»
modelo de features
«datastore»
arquitetura base
construir arquitetura base
«datastore»
arquitetura referenciada
gerar produto
«datastore»
diagrama de classe do produto
construir modelo de componentes
«datastore»
modelo de componentes
construir metamodelo de aspectos
«datastore»
código do produto
gerar código fonte do produto
«datastore»
metamodelo de aspecto
construir modelo de aspectos
«datastore»
modelo de aspecto

Visão Geral do Processo de Derivação GingaForAll

Atividades inicais do processo

Existem quatro atividades independentes que caracterizam o início do processo e podem ser realizadas em qualquer ordem:

- Construir metamodelo de features;
- Construir arquitetura base;
- Construir o metamodelo de aspectos
- Construir modelo de componentes.

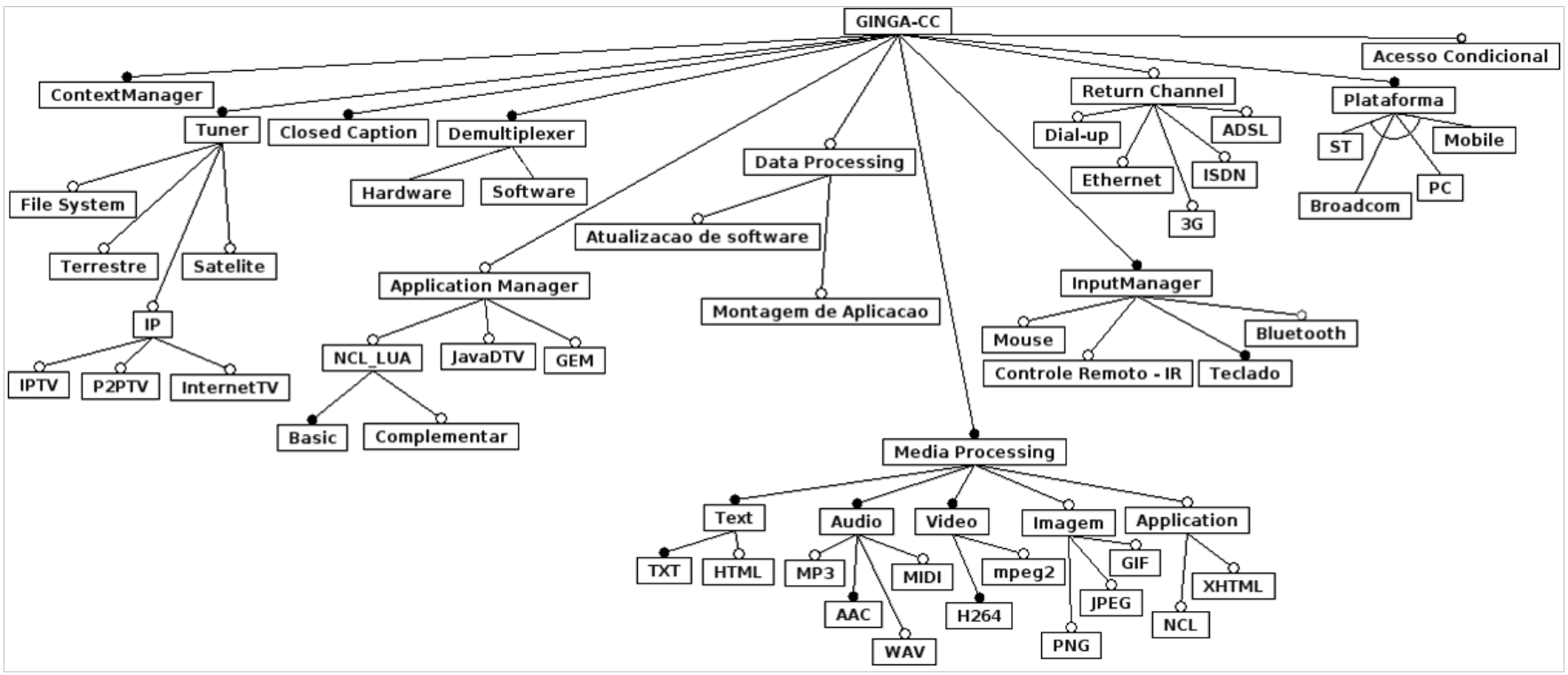
As demais atividades seguem o fluxo apresentado na figura passada.

Atividade Construir metamodelo de features

Definição

O metamodelo de features é um modelo que descreve todas as features possíveis do Ginga-CC com suas respectivas restrições e seus relacionamentos.

- Produto da análise dos requisitos da LPS
- Modelo de entrada: sem modelo de entrada
- Modelo de saída: metamodelo Ecore de features
- Justificativa: Definir as features da família de produtos, assim como especificar seus relacionamentos, restrições e dependências.

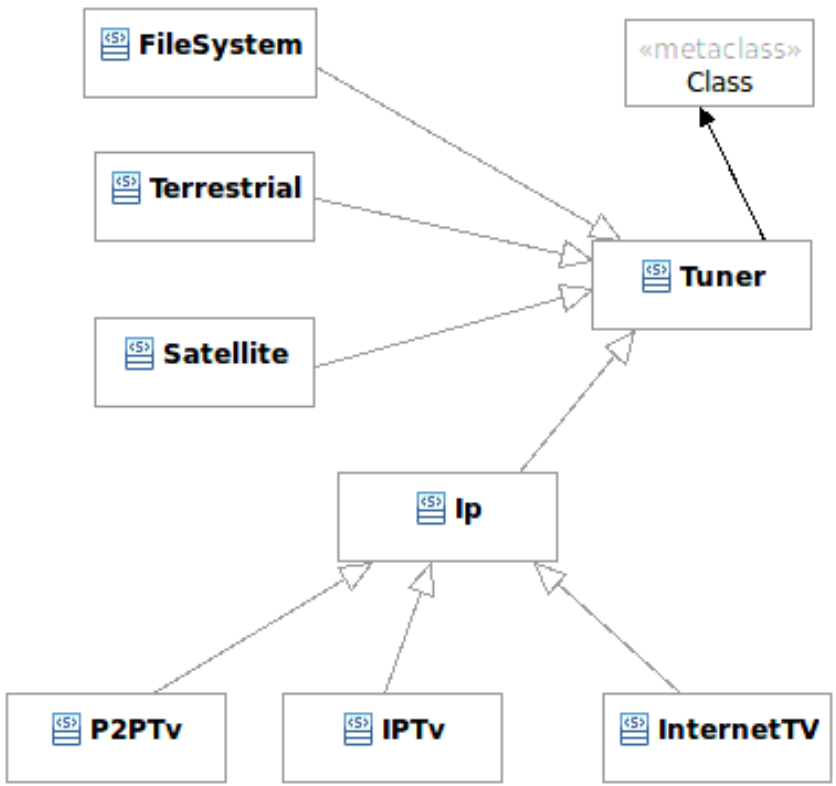

GINGA-CC
ContextManager
Tuner
Closed Caption
Demultiplexer
Data Processing
Return Channel
Acesso Condicional
Plataforma
File System
Terrestre
Satelite
IP
IPTV
P2PTV
InternetTV
Hardware
Software
Atualizacao de software
Montagem de Aplicacao
Application Manager
NCL_LUA
JavaDTV
GEM
Basic
Complementar
Dial-up
ADSL
Ethernet
ISDN
3G
ST
Mobile
PC
Broadcom
InputManager
Mouse
Bluetooth
Controle Remoto - IR
Teclado
Media Processing
Text
Audio
Video
Imagem
Application
TXT
HTML
MP3
MIDI
mpeg2
GIF
XHTML
AAC
H264
JPEG
NCL
WAV
PNG

Atividade Gerar profile Metamodelo UML de features

Definição

Um profile UML de features é composto de estereótipos para todas as features especificadas no metamodelo de features.

- Modelo de entrada: metamodelo de features
- Modelo de saída: Profile UML de features
- Dependência: Construir metamodelo de features
- Justificativa: Gerar automaticamente o profile UML de features, o qual permite anotar nos elementos da arquitetura base quais features um elemento está relacionado.

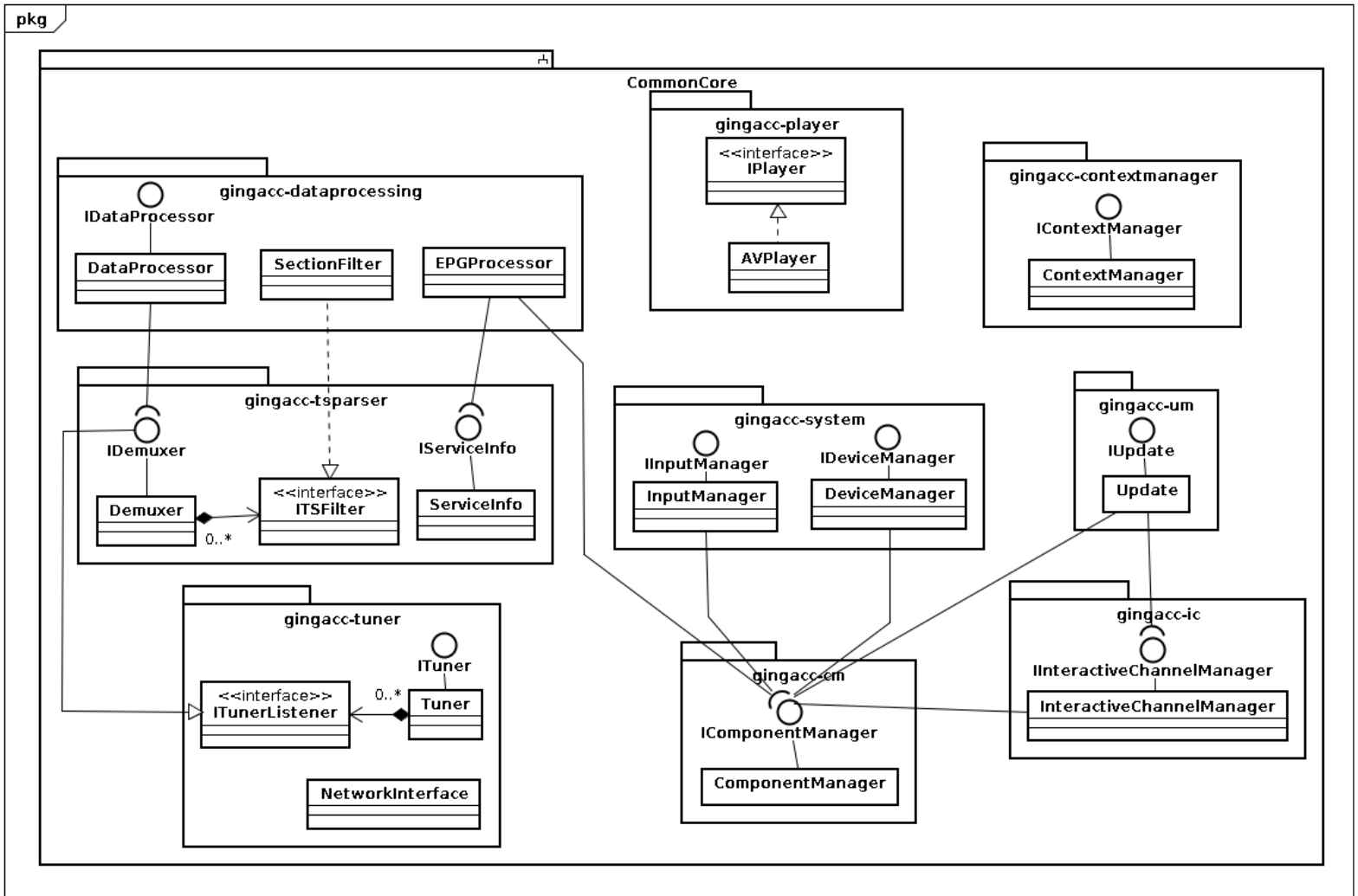

FileSystem
«metaclass»
Class
Terrestrial
Tuner
Satellite
Ip
P2PTv
IPTv
InternetTV

Atividade Construir arquitetura base

Definição

A atividade **Construir arquitetura base** consiste da modelagem de arquitetura base da LPS GingaForAll.

- Modelo de entrada: ausente;
- Modelo de saída: Diagrama de classes UML da arquitetura base
- Dependência: Sem dependência
- Justificativa: Definir a arquitetura base do Ginga-CC através da sua modelagem por meio de diagramas UML.

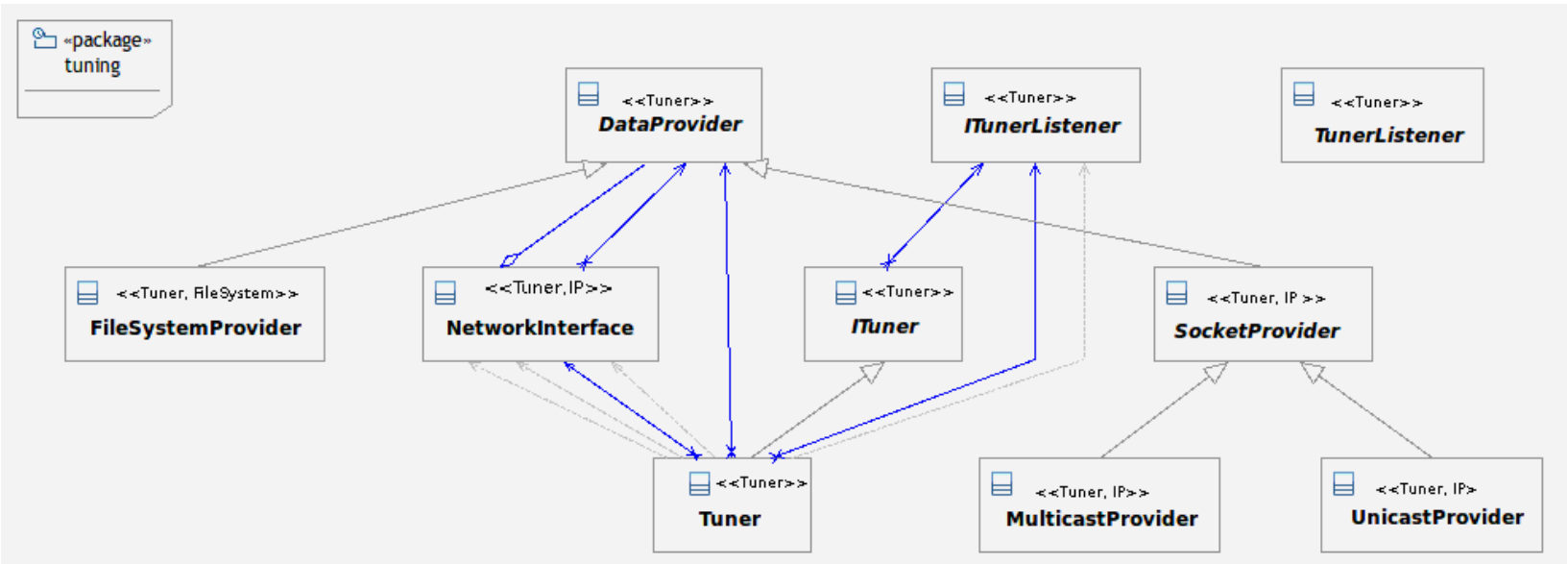

pkg
CommonCore
gingacc-player
<<interface>>
IPlayer
AVPlayer
gingacc-contextmanager
IContextManager
ContextManager
gingacc-dataprocessing
IDataProcessor
DataProcessor
SectionFilter
EPGProcessor
gingacc-tsparser
IDemuxer
Demuxer
0..*
<<interface>>
ITSFilter
IServiceInfo
ServiceInfo
gingacc-system
IInputManager
InputManager
IDeviceManager
DeviceManager
gingacc-um
IUpdate
Update
gingacc-tuner
ITuner
<<interface>>
ITunerListener
0..*
Tuner
NetworkInterface
gingacc-cm
IComponentManager
ComponentManager
gingacc-ic
IInteractiveChannelManager
InteractiveChannelManager

Construir arquitetura referenciada

Definição

Consiste em anotar (aplicar um estereótipo) os elementos da arquitetura base, usando o profile UML de features.

- Modelos de Entrada: Arquitetura base e profile UML de features
- Modelo de Saída: Diagrama de classes UML 2
- Dependência: metamodelo de features e arquitetura base.
- Justificativa: Anotar a arquitetura base com estereótipos indicando a quais features eles estão relacionadas.

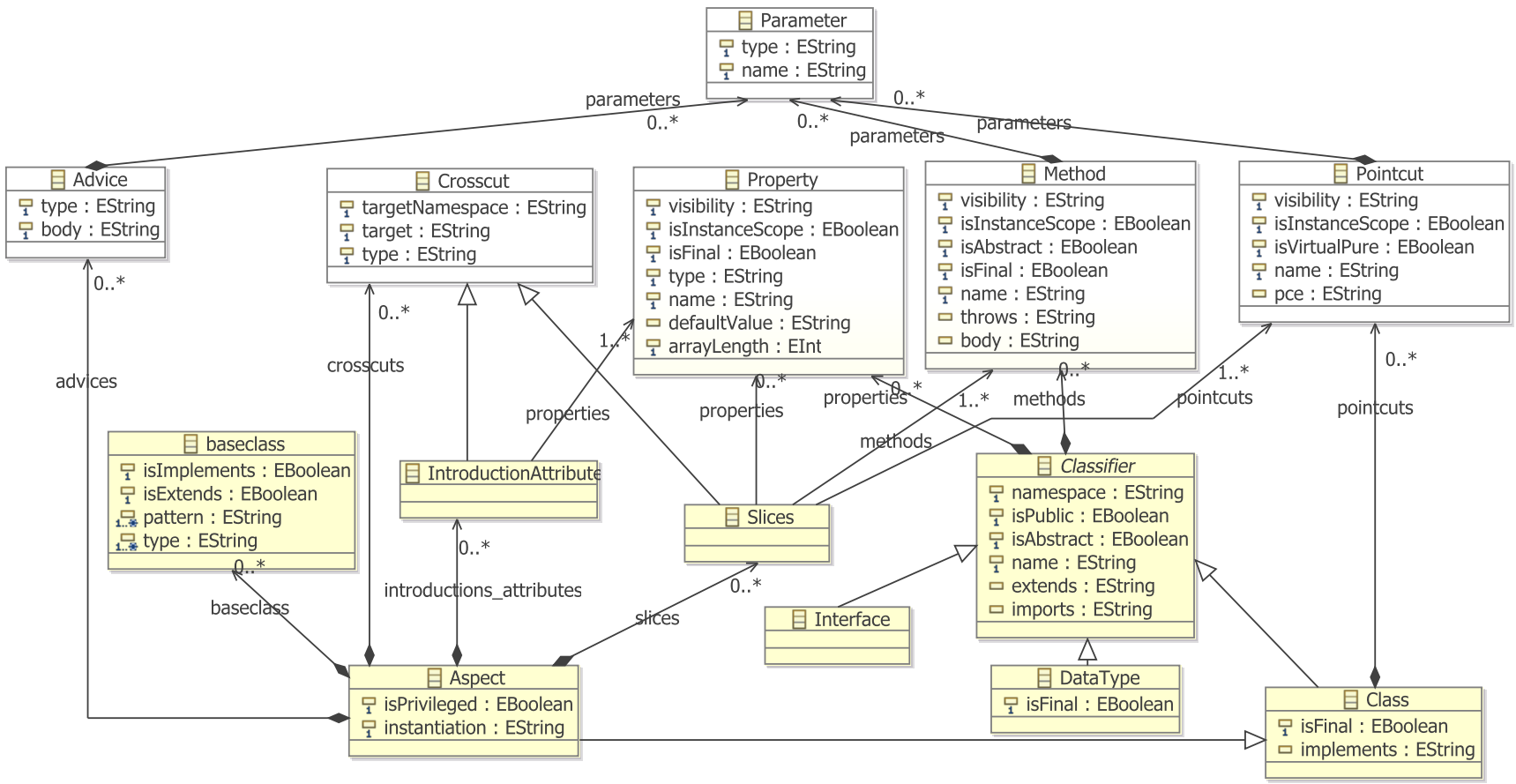

«package»
tuning
<<Tuner>>
DataProvider
<<Tuner>>
ITunerListener
<<Tuner>>
TunerListener
<<Tuner, FileSystem>>
FileSystemProvider
<<Tuner, IP>>
NetworkInterface
<<Tuner>>
ITuner
<<Tuner, IP >>
SocketProvider
<<Tuner>>
Tuner
<<Tuner, IP>>
MulticastProvider
<<Tuner, IP>
UnicastProvider

Construir metamodelo aspectos

Definição

consiste na especifição do metamodelo de aspecto a ser usado como base na modelagem de aspecto proposta.

- Modelo de Entrada: Sem modelo de entrada
- Modelo de Saída: Metamodelo Ecore dos aspectos
- Dependência: Nenhuma
- Justificativa: Definir os elementos,relacionamentos,restrições e dependências dos elementos contidos no modelo de aspectos.

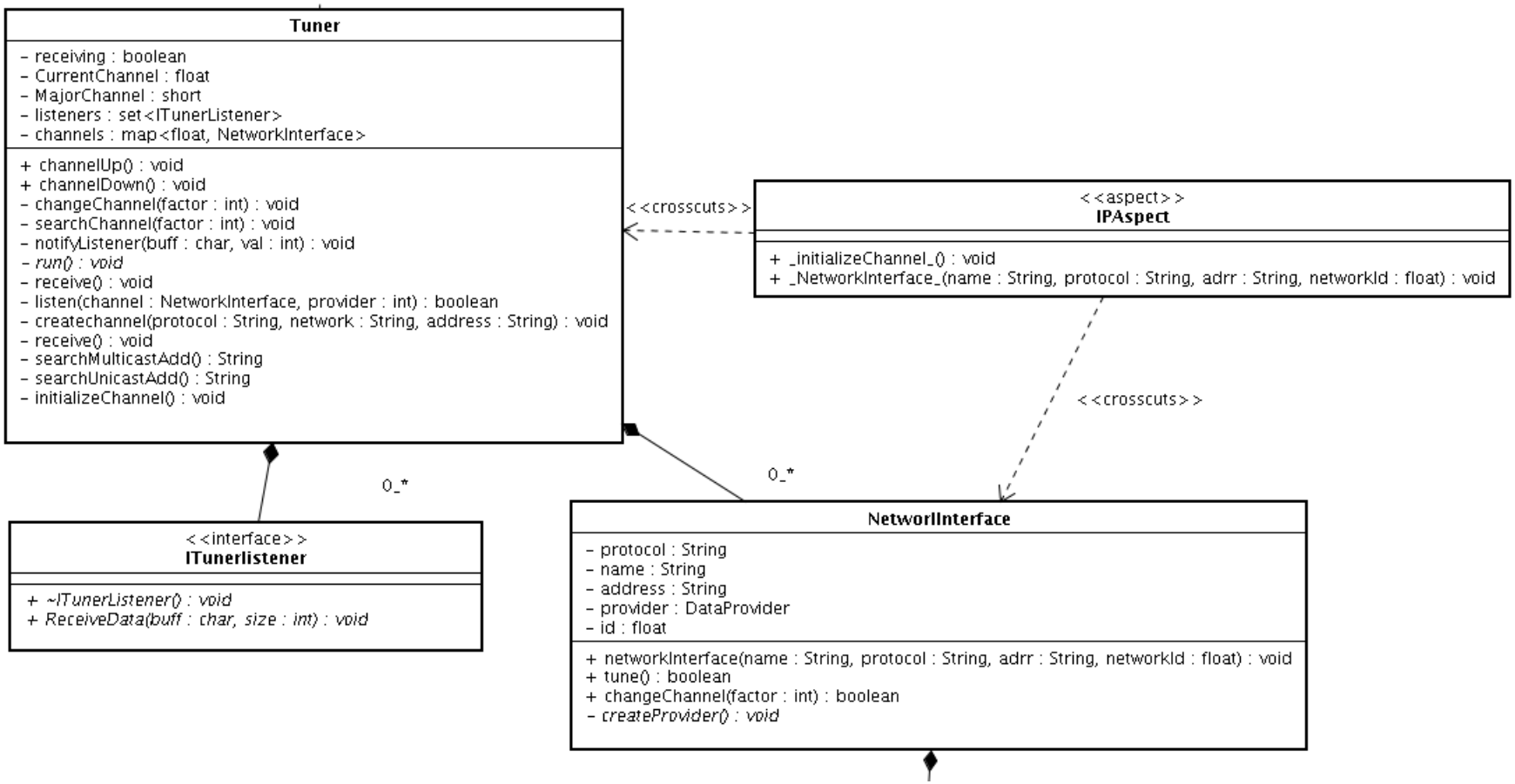

Parameter
type : EString
name : EString
parameters
0..*
0..*
parameters
0..*
parameters
Advice
type : EString
body : EString
Crosscut
targetNamespace : EString
target : EString
type : EString
Property
visibility : EString
isInstanceScope : EBoolean
isFinal : EBoolean
type : EString
name : EString
defaultValue : EString
arrayLength : EInt
Method
visibility : EString
isInstanceScope : EBoolean
isAbstract : EBoolean
isFinal : EBoolean
name : EString
throws : EString
body : EString
Pointcut
visibility : EString
isInstanceScope : EBoolean
isVirtualPure : EBoolean
name : EString
pce : EString
0..*
advices
0..*
crosscuts
1..*
properties
0..*
properties
0..*
properties
1..*
methods
0..*
pointcuts
0..*
pointcuts
baseclass
isImplements : EBoolean
isExtends : EBoolean
pattern : EString
type : EString
IntroductionAttribute
Slices
Classifier
namespace : EString
isPublic : EBoolean
isAbstract : EBoolean
name : EString
extends : EString
imports : EString
methods
0..*
introductions_attributes
0..*
baseclass
slices
0..*
Interface
Aspect
isPrivileged : EBoolean
instantiation : EString
DataType
isFinal : EBoolean
Class
isFinal : EBoolean
implements : EString

Atividade Criar modelo de Aspectos

Definição

Consiste em definir os pontos de corte dos aspectos através da criação do modelo de aspectos

- Modelo de Entrada: Metamodelo Ecore de aspectos e profile UML de features
- Modelo de Saída: Modelo aspectual em Ecore
- Dependência: metamodelo de features e de aspectos
- Justificativa: Definir os pontos de onde os de aspectos irão atuar e na arquitetura referenciada.

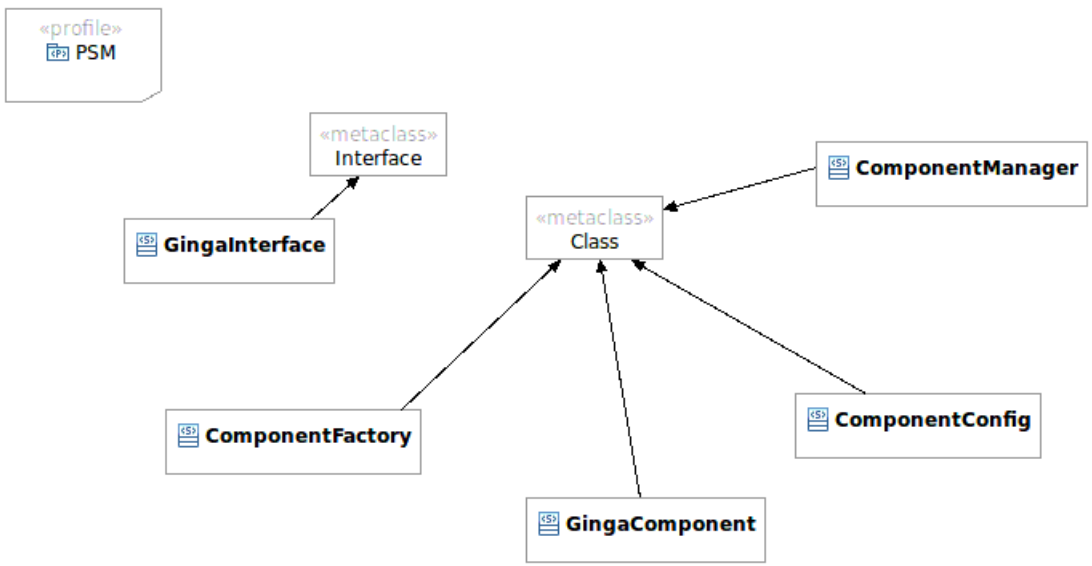

Tuner
- receiving : boolean
- CurrentChannel : float
- MajorChannel : short
- listeners : set<ITunerListener>
- channels : map<float, NetworkInterface>
+ channelUp() : void
+ channelDown() : void
- changeChannel(factor : int) : void
- searchChannel(factor : int) : void
- notifyListener(buff : char, val : int) : void
- run() : void
- receive() : void
- listen(channel : NetworkInterface, provider : int) : boolean
- createchannel(protocol : String, network : String, address : String) : void
- receive() : void
- searchMulticastAdd() : String
- searchUnicastAdd() : String
- initializeChannel() : void
<<crosscuts>>
<<aspect>>
IPAspect
+ _initializeChannel_() : void
+ _NetworkInterface_(name : String, protocol : String, adrr : String, networkId : float) : void
<<crosscuts>>
0_*
0_*
<<interface>>
ITunerlistener
+ ~ITunerListener() : void
+ ReceiveData(buff : char, size : int) : void
NetworIInterface
- protocol : String
- name : String
- address : String
- provider : DataProvider
- id : float
+ networkInterface(name : String, protocol : String, adrr : String, networkId : float) : void
+ tune() : boolean
+ changeChannel(factor : int) : boolean
- createProvider() : void

Atividade construir modelo de componentes

Definição

Essa atividade consiste na especifição do metamodelo de componentes a ser utilizado na modelagem da arquitetura base da LPS e de algumas variabilidades.

- Modelo de Entrada: Nenhum
- Modelo de Saída: Profile UML do modelo de componentes
- Dependência: Nenhuma
- Justificativa: Identificar os elementos da arquitetura do produto gerado que estão relacionados com o modelo de componentes e quais seus pápeis nele.

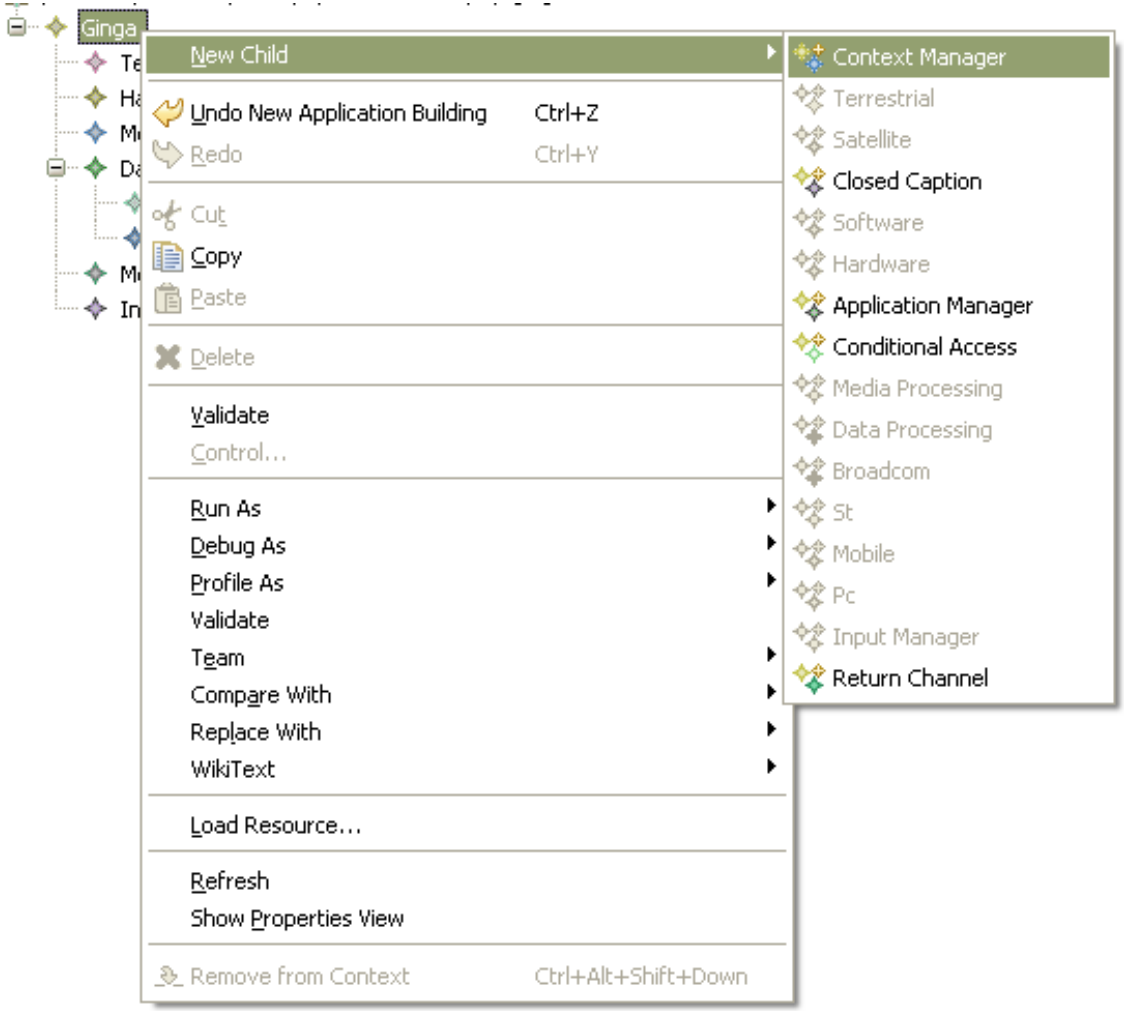

«profile»
PSM
«metaclass»
Interface
GingaInterface
«metaclass»
Class
ComponentManager
ComponentFactory
ComponentConfig
GingaComponent

Atividade Selecionar variabilidades

Definição

A atividade **Selecionar variabilidades** consiste da configuração do produto a ser gerado.

- Modelo de Entrada: Metamodelo de features
- Modelo de Saída: Modelo de Ecore de features (configuração do produto)
- Dependência: Metamodelo de features
- Justificativa: Realizar a configuração do produto.


Ginga
New Child
Undo New Application Building Ctrl+Z
Redo Ctrl+Y
Cut
Copy
Paste
Delete
Validate
Control...
Run As
Debug As
Profile As
Validate
Team
Compare With
Replace With
WikiText
Load Resource...
Refresh
Show Properties View
Remove from Context Ctrl+Alt+Shift+Down
Context Manager
Terrestrial
Satellite
Closed Caption
Software
Hardware
Application Manager
Conditional Access
Media Processing
Data Processing
Broadcom
St
Mobile
Pc
Input Manager
Return Channel

Atividade Gerar produto

Definição

Consiste em realizar o weaving do modelo de aspectos e da arquitetura referenciada.

- Modelo de Entrada: Modelo Ecore de features e de aspectos, profile UML de componentes e da arquitetura referenciada.
- Modelo de Saída: Arquitetura do produto - UML
- Dependência: Variabilidades, modelo de Aspectos e arquitetura referenciada
- Justificativa: Obter o produto selecionado pelo engenheiro de aplicação.

Atividade Gerar código fonte do produto

Definição

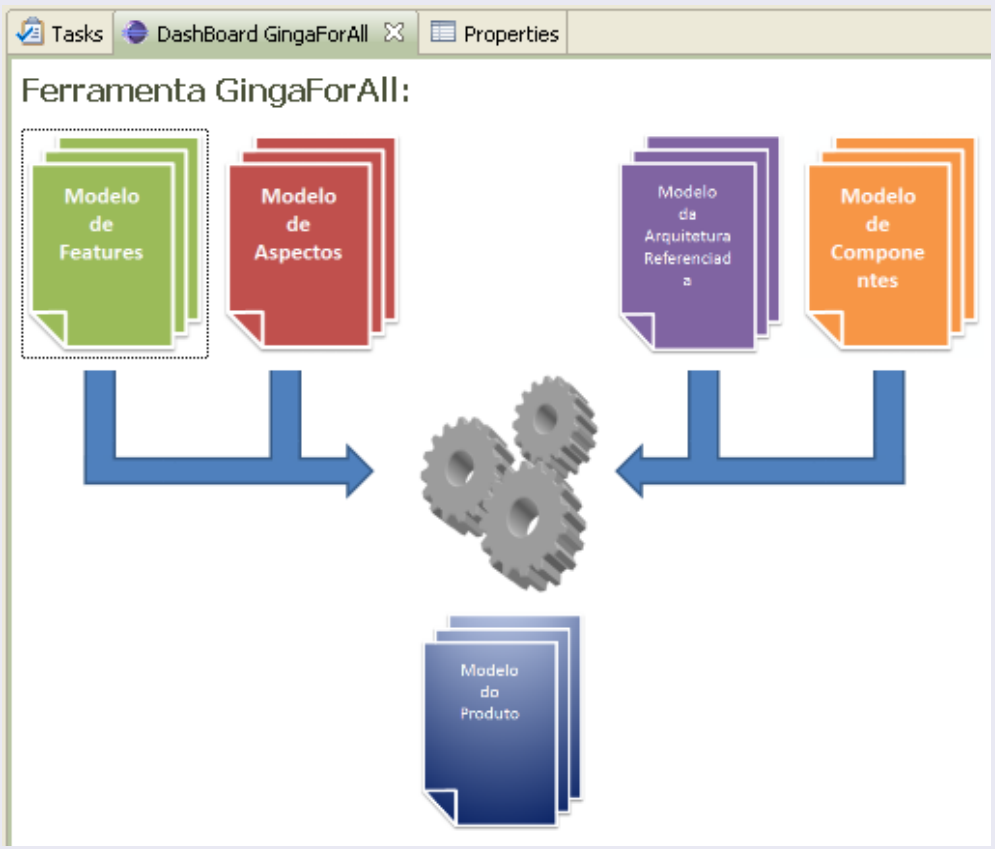
A Atividade de **Gerar código fonte do produto** realiza a geração de código fonte em C++ a partir do diagrama de classe gerado pela atividade gerar produto.

- Modelo de Entrada: Diagrama de classe UML 2 da arquitetura do produto.
- Modelo de Saída: Código fonte do produto
- Dependência: Gerar produto
- Justificativa: Obter o código fonte do produto selecionado pelo engenheiro de aplicação.

Visão Geral

1. Instanciação do processo GingaForAll
2. Plugin do Eclipse
   - Ambiente completo para desenvolvimento da abrodagem MDA
   - Plataforma aberta de desenvolvimento
   - Grande comunidade de usuários e desenvolvedores
3. Gerência das variabilidades da LPS GingaCC
4. Dar suporte a geração de seus produtos

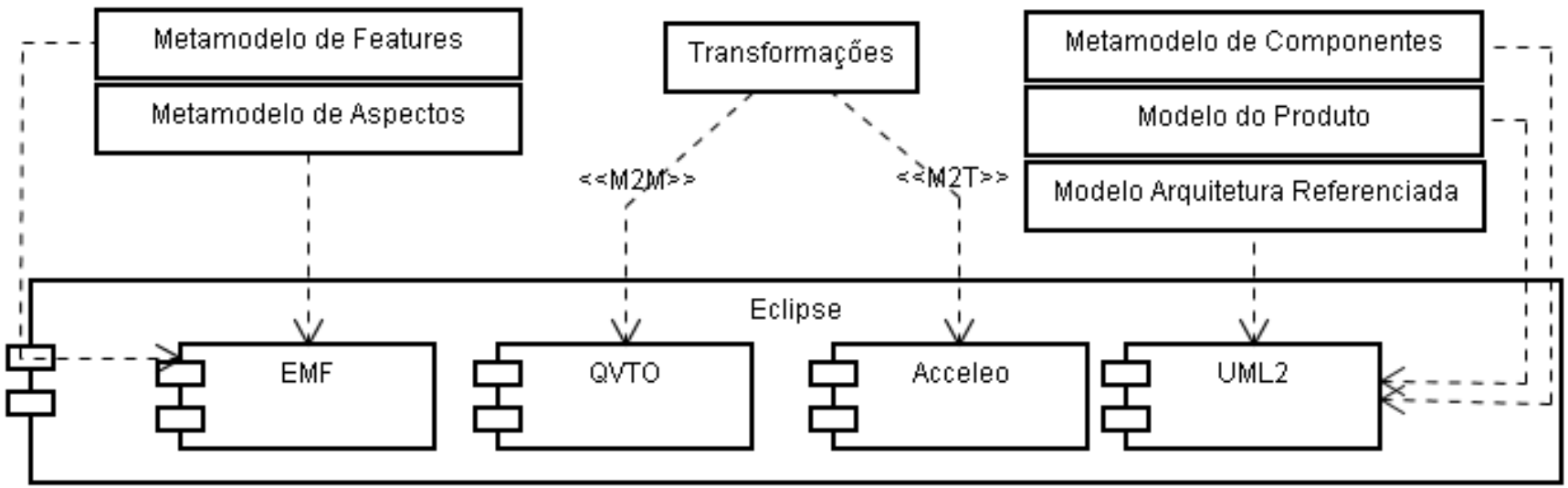
Ferramenta GingaForAll



Tecnologias utilizadas

- UML 2 - Unified Modeling Language
  - Modelagem dos profiles e modelos
- QVTo - Query/View/Transformation Operational
  - Transformações entre modelos
- EMF - Eclipse Modeling Framework
  - Criação de metamodelos e de editores de modelo
  - Ecore
- Acceleo
  - Transformações de modelo para texto

Arquitetura da Ferramenta



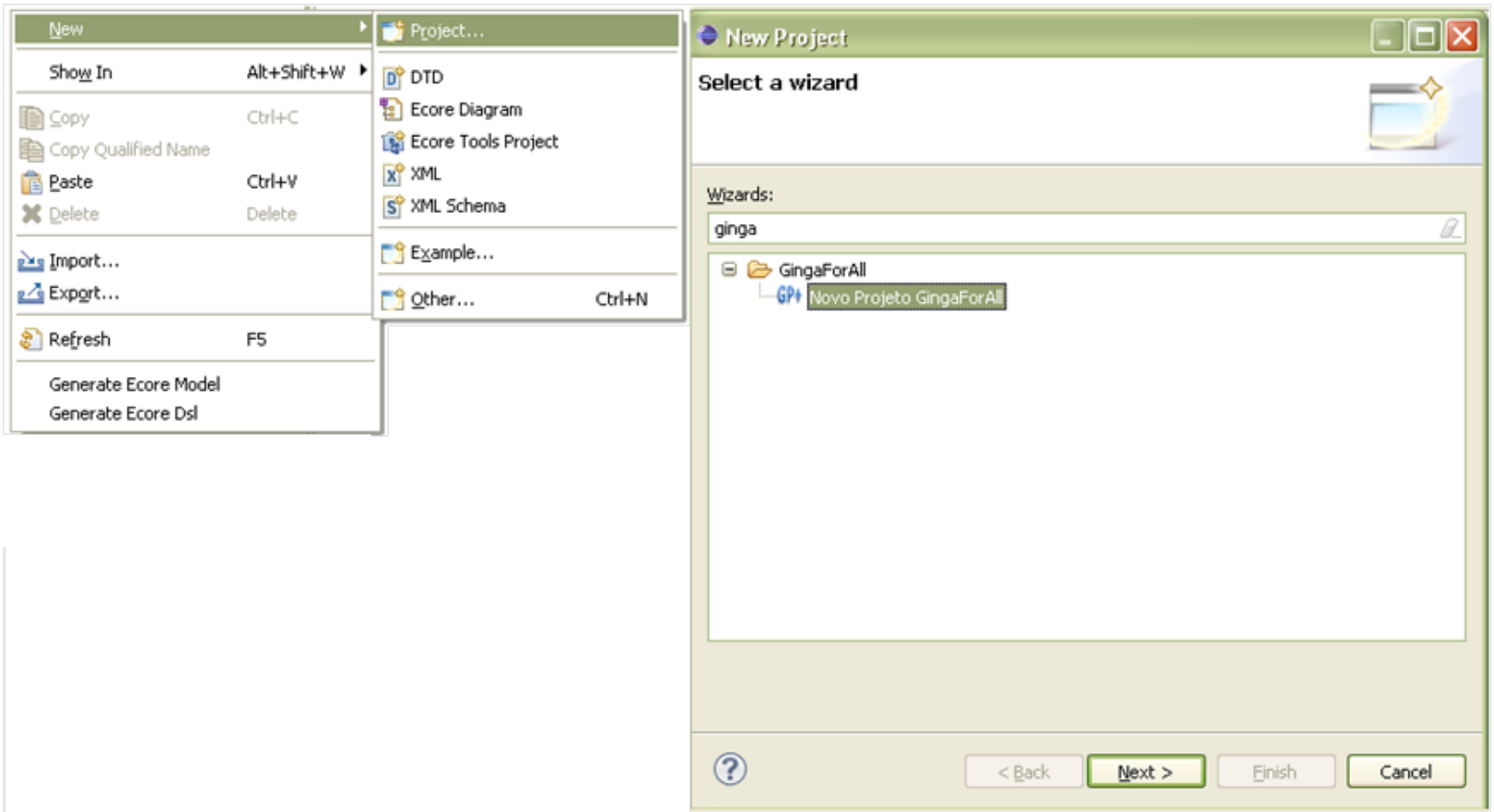
Implementação

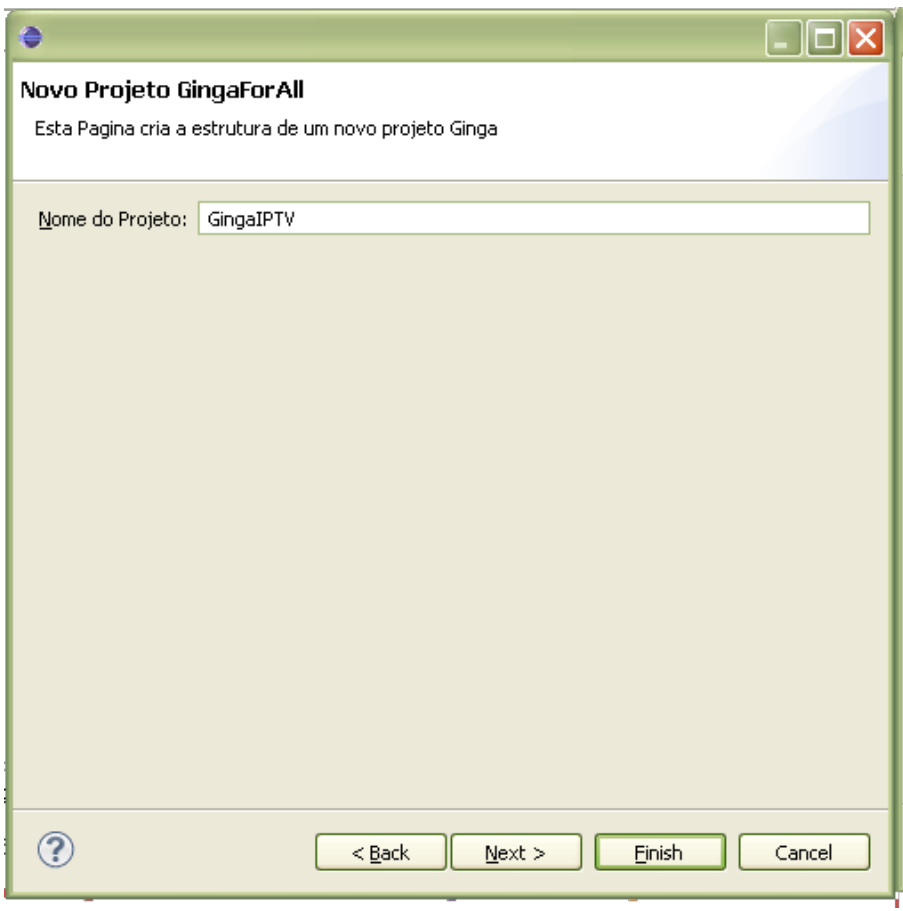
Requisitos

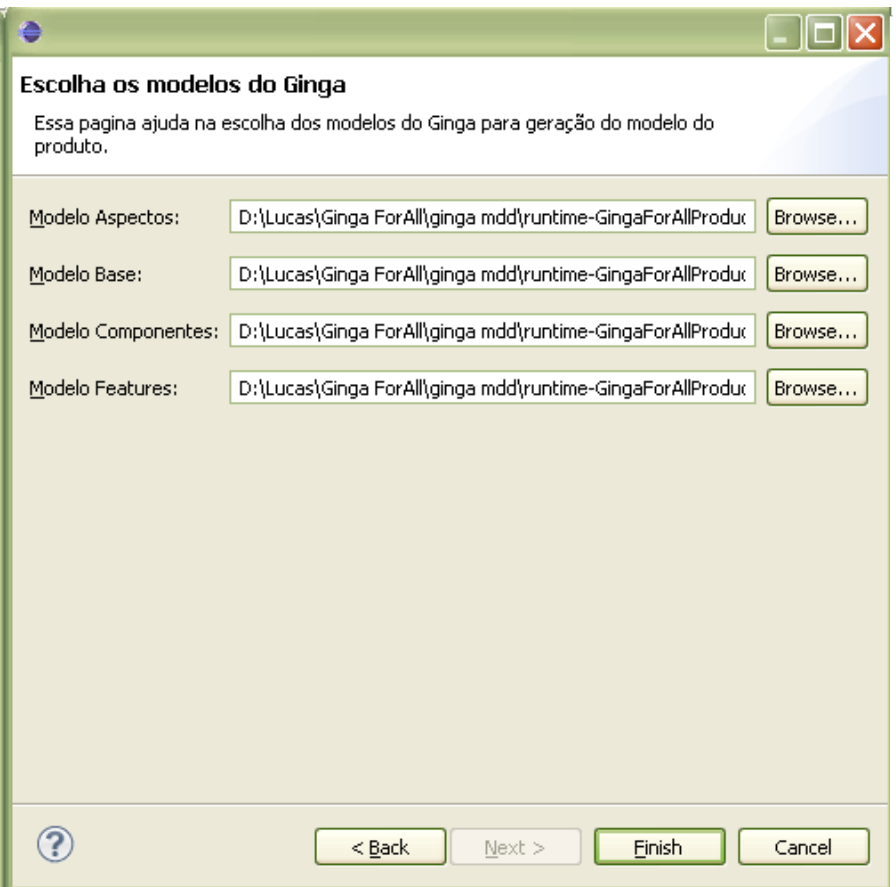
- Criar um novo projeto do tipo GingaForAll,
- CRUD (Create/Read/Update/Delete) dos modelos baseados na estrutura do Ginga,
- Gerar um modelo do produto a partir dos modelos de entrada,
- Gerar Código Fonte a partir do modelo do produto

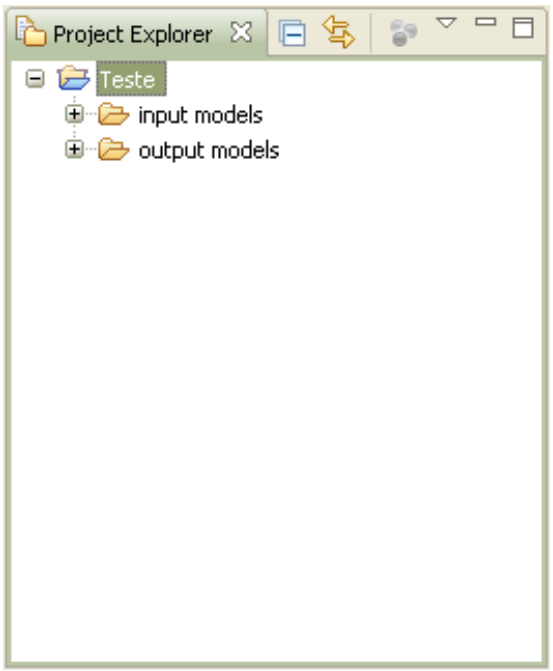
I - Criar um novo projeto do tipo GingaForAll

- Wizard para a criação de um novo projeto que possibilite:
- A escolha do nome do projeto,
- Importar modelos.
- Gerar Estrutura de diretórios personalizada


New
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Import...
Export...
Refresh F5
Generate Ecore Model
Generate Ecore Dsl
Project...
DTD
Ecore Diagram
Ecore Tools Project
XML
XML Schema
Example...
Other... Ctrl+N
New Project
Select a wizard
Wizards:
ginga
GingaForAll
Novo Projeto GingaForAll
< Back
Next >
Finish
Cancel


Novo Projeto GingaForAll
Esta Pagina cria a estrutura de um novo projeto Ginga
Nome do Projeto:
GingaIPTV
< Back
Next >
Finish
Cancel


Escolha os modelos do Ginga
Essa pagina ajuda na escolha dos modelos do Ginga para geração do modelo do produto.
Modelo Aspectos:
Modelo Base:
Modelo Componentes:
Modelo Features:
D:\Lucas\Ginga ForAll\ginga mdd\runtime-GingaForAllProduc
Browse...
< Back
Next >
Finish
Cancel

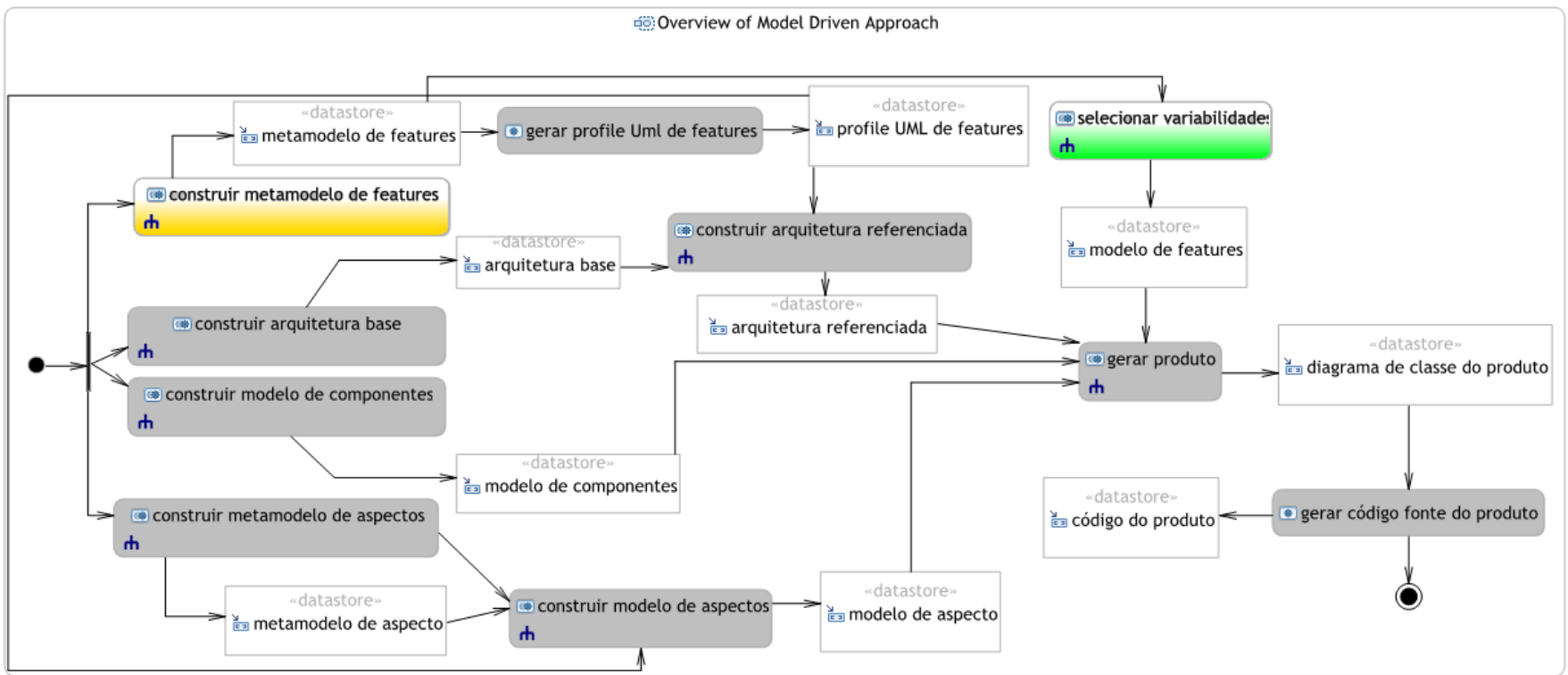

Project Explorer
Teste
input models
output models

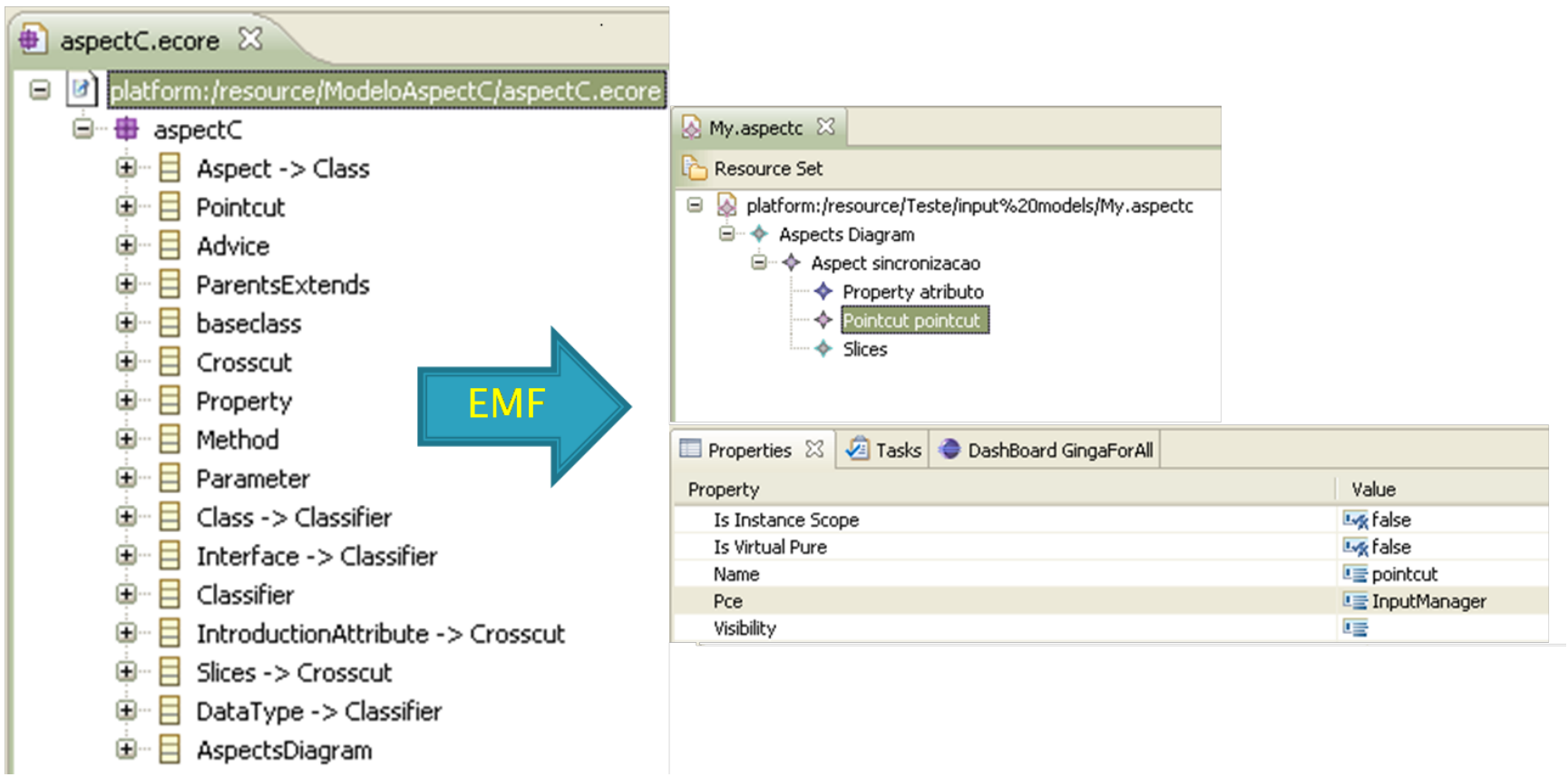
II - CRUD dos modelos

Os CRUD dos modelos baseados na estrutura do Ginga

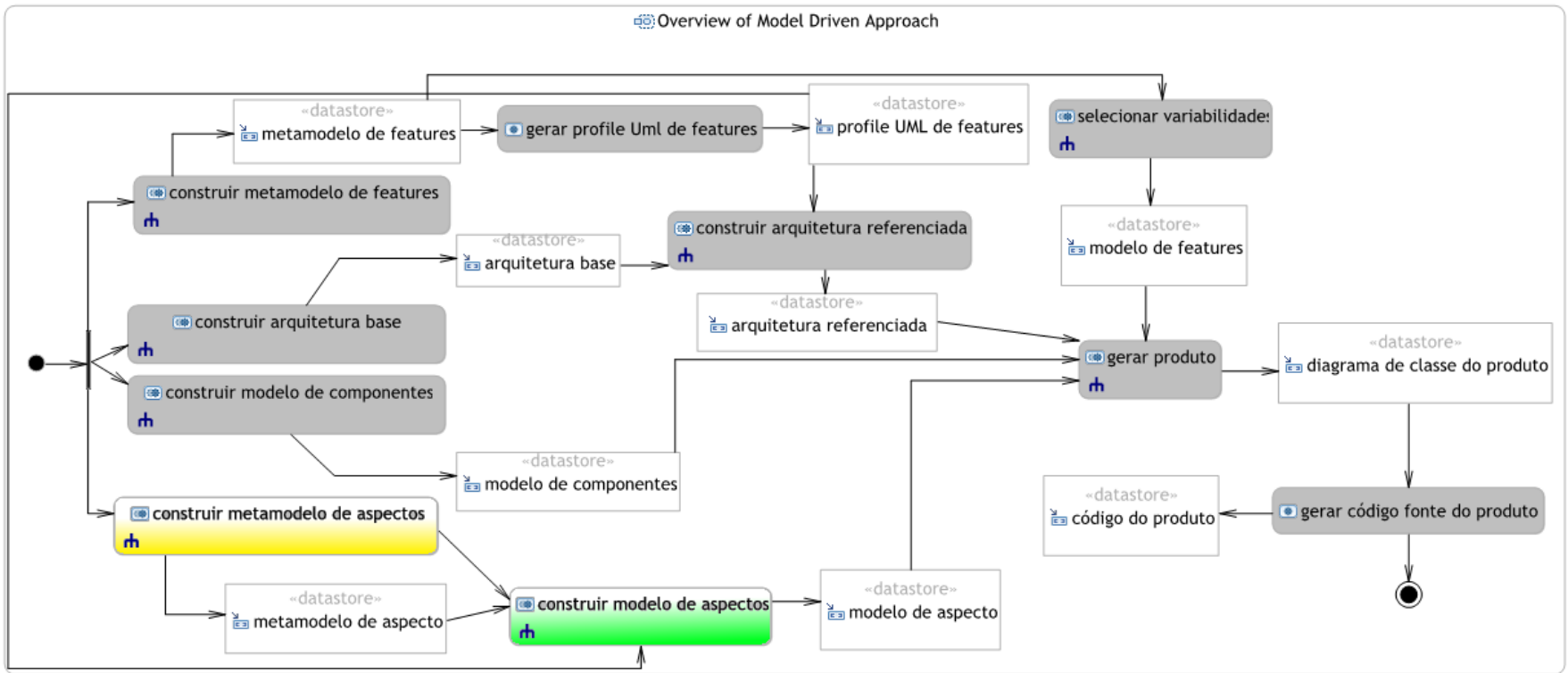
- Uso do EMF para a criação dos editores dos modelos
- Modelo Features e o Modelo de aspectos
- Uso do UML2 para criação dos modelos:
- Modelo de componentes, Modelo da arquitetura referenciada e Modelo do produto

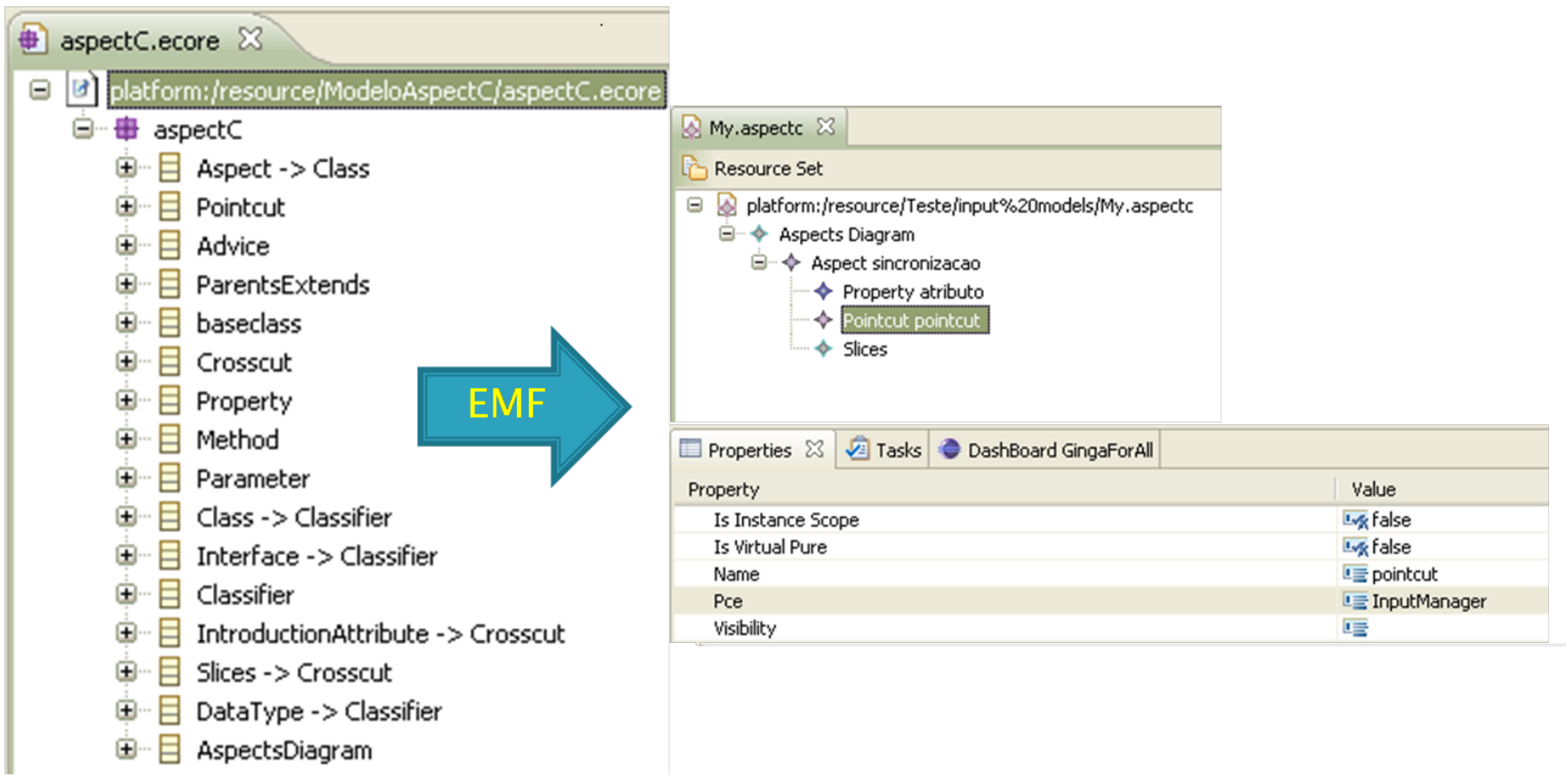
Modelo de Features



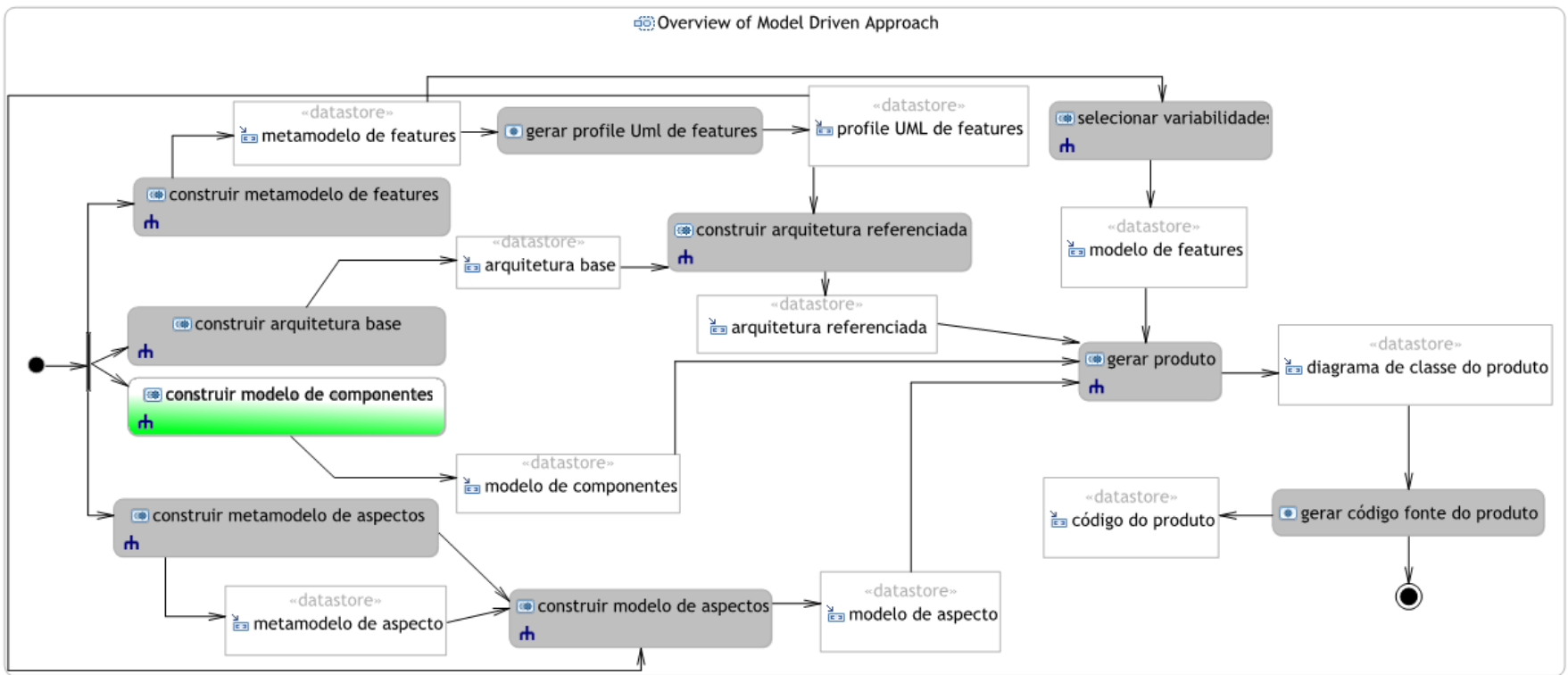

aspectC.ecore
platform:/resource/ModeloAspectC/aspectC.ecore
aspectC
Aspect -> Class
Pointcut
Advice
ParentsExtends
baseclass
Crosscut
Property
Method
Parameter
Class -> Classifier
Interface -> Classifier
Classifier
IntroductionAttribute -> Crosscut
Slices -> Crosscut
DataType -> Classifier
AspectsDiagram
EMF
My.aspectc
Resource Set
platform:/resource/Teste/input%20models/My.aspectc
Aspects Diagram
Aspect sincronizacao
Property atributo
Pointcut pointcut
Slices
Properties
Tasks
DashBoard GingaForAll
Property
Value
Is Instance Scope
false
Is Virtual Pure
false
Name
pointcut
Pce
InputManager
Visibility

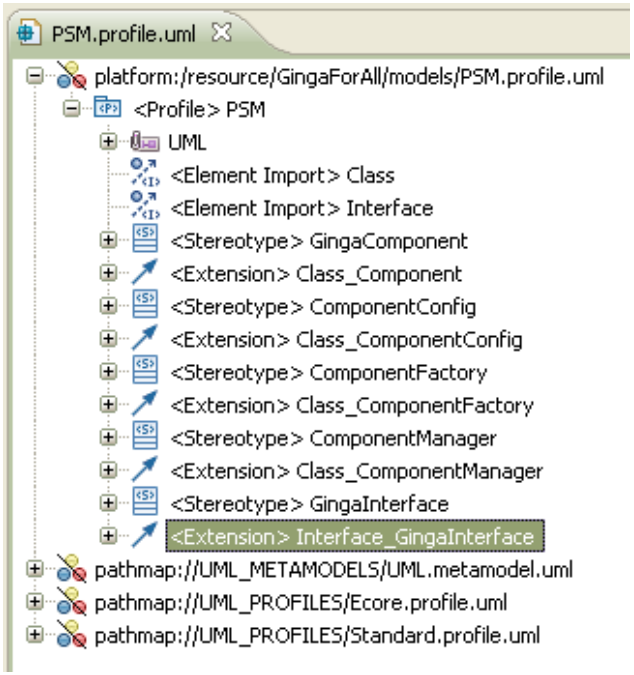
Modelo de aspectos



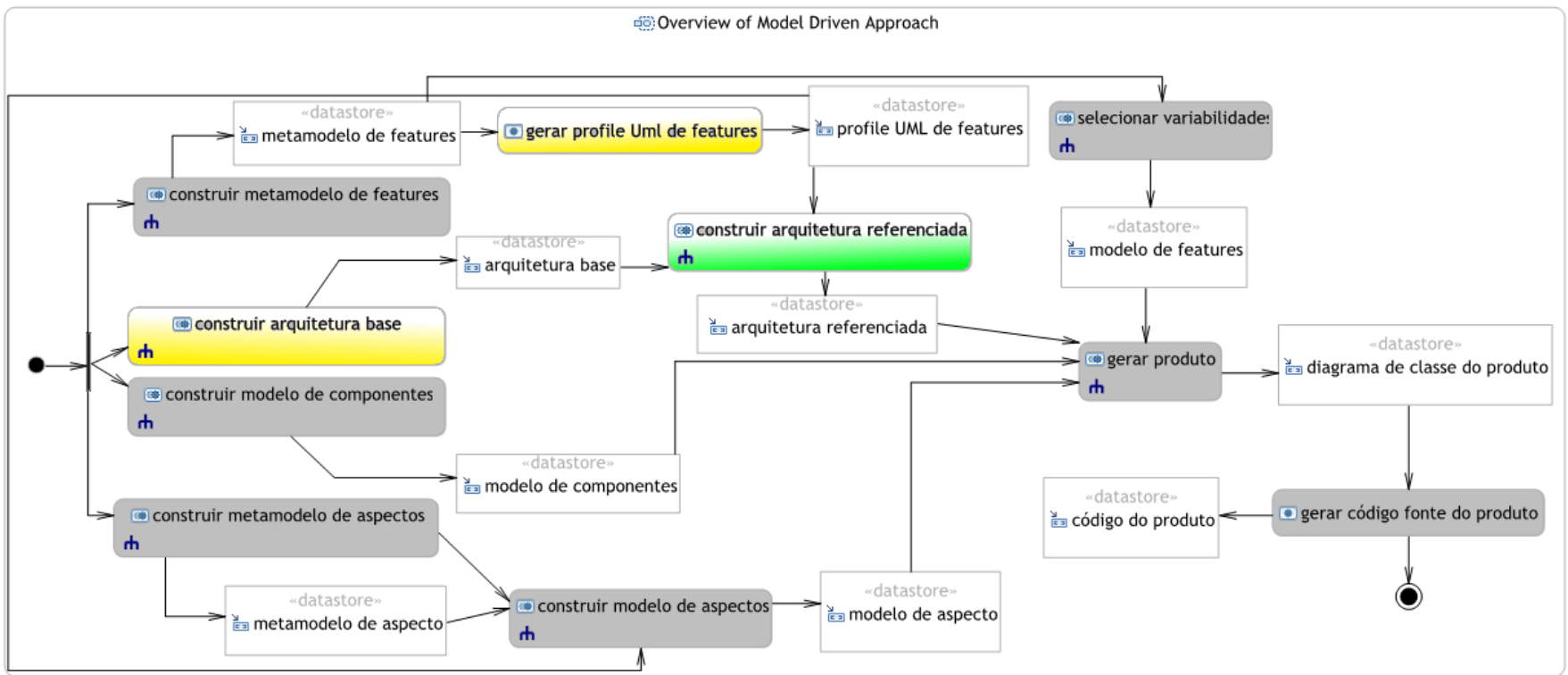

aspectC.ecore
platform:/resource/ModeloAspectC/aspectC.ecore
aspectC
Aspect -> Class
Pointcut
Advice
ParentsExtends
baseclass
Crosscut
Property
Method
Parameter
Class -> Classifier
Interface -> Classifier
Classifier
IntroductionAttribute -> Crosscut
Slices -> Crosscut
DataType -> Classifier
AspectsDiagram
EMF
My.aspectc
Resource Set
platform:/resource/Teste/input%20models/My.aspectc
Aspects Diagram
Aspect sincronizacao
Property atributo
Pointcut pointcut
Slices
Properties
Tasks
DashBoard GingaForAll
Property
Value
Is Instance Scope
false
Is Virtual Pure
false
Name
pointcut
Pce
InputManager
Visibility

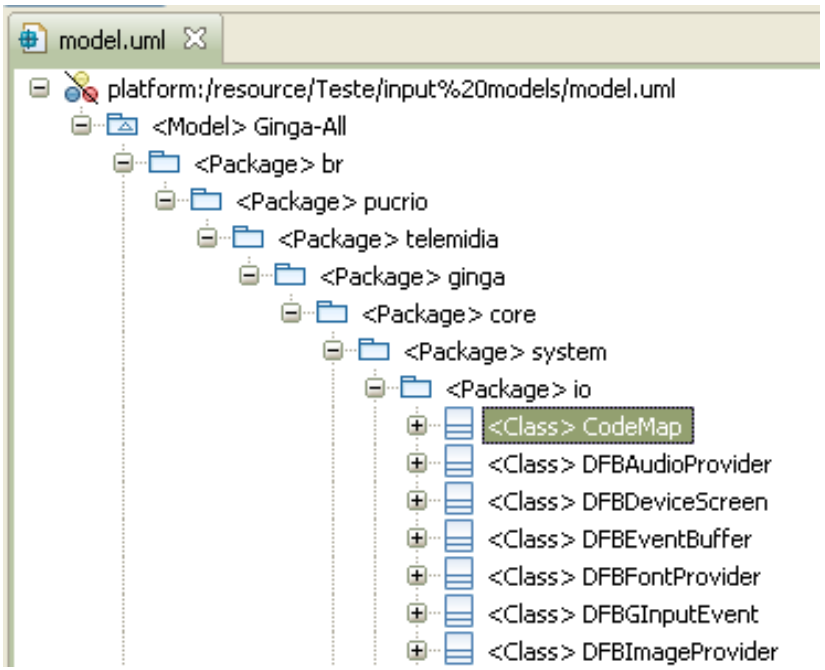
Modelo de Componentes




PSM.profile.uml
platform:/resource/GingaForAll/models/PSM.profile.uml
<Profile> PSM
UML
<Element Import> Class
<Element Import> Interface
<Stereotype> GingaComponent
<Extension> Class_Component
<Stereotype> ComponentConfig
<Extension> Class_ComponentConfig
<Stereotype> ComponentFactory
<Extension> Class_ComponentFactory
<Stereotype> ComponentManager
<Extension> Class_ComponentManager
<Stereotype> GingaInterface
<Extension> Interface_GingaInterface
pathmap://UML_METAMODELS/UML.metamodel.uml
pathmap://UML_PROFILES/Ecore.profile.uml
pathmap://UML_PROFILES/Standard.profile.uml

Modelo da Arquitetura Referenciada





- 78 classes
- 19.825 LOCs

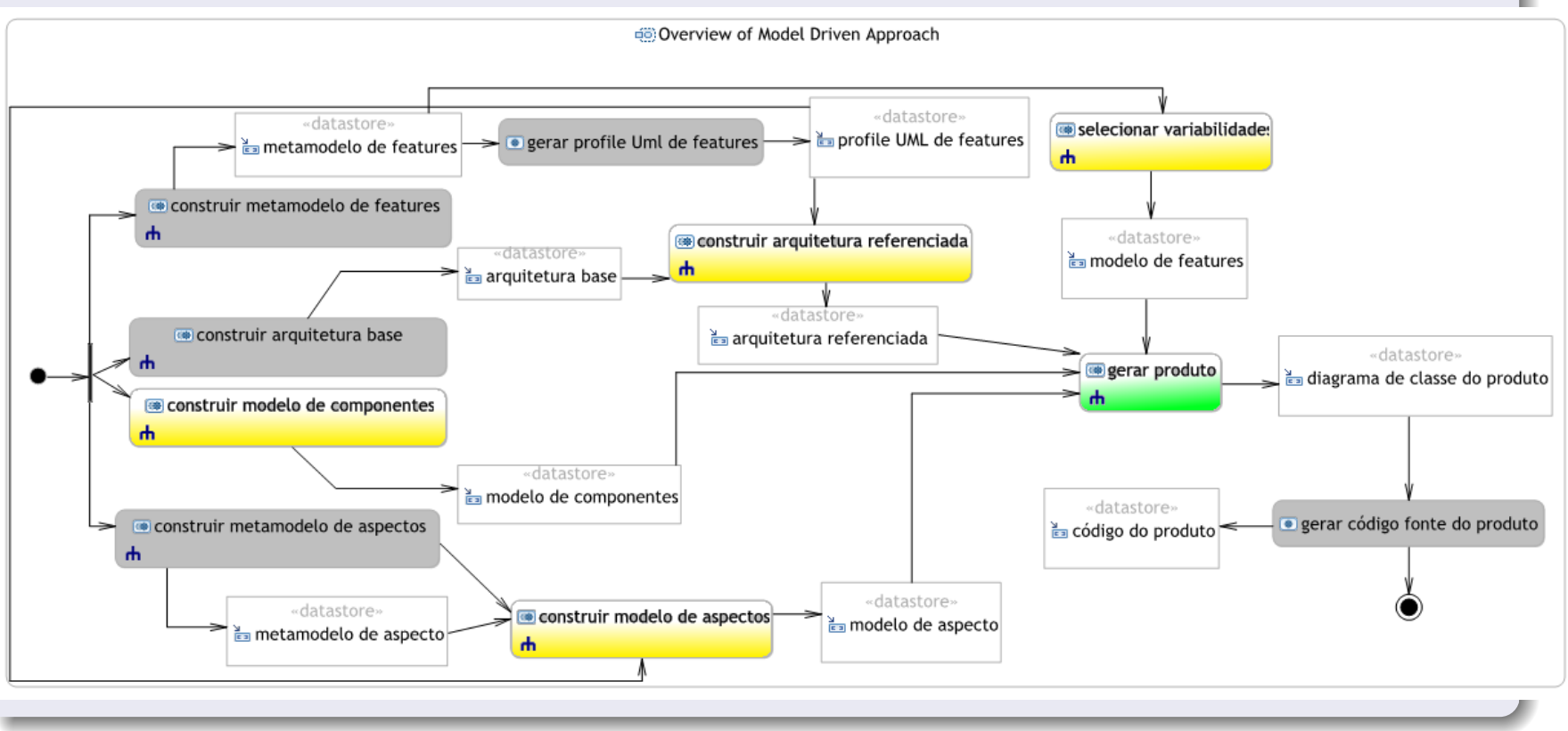
Modelo do Produto

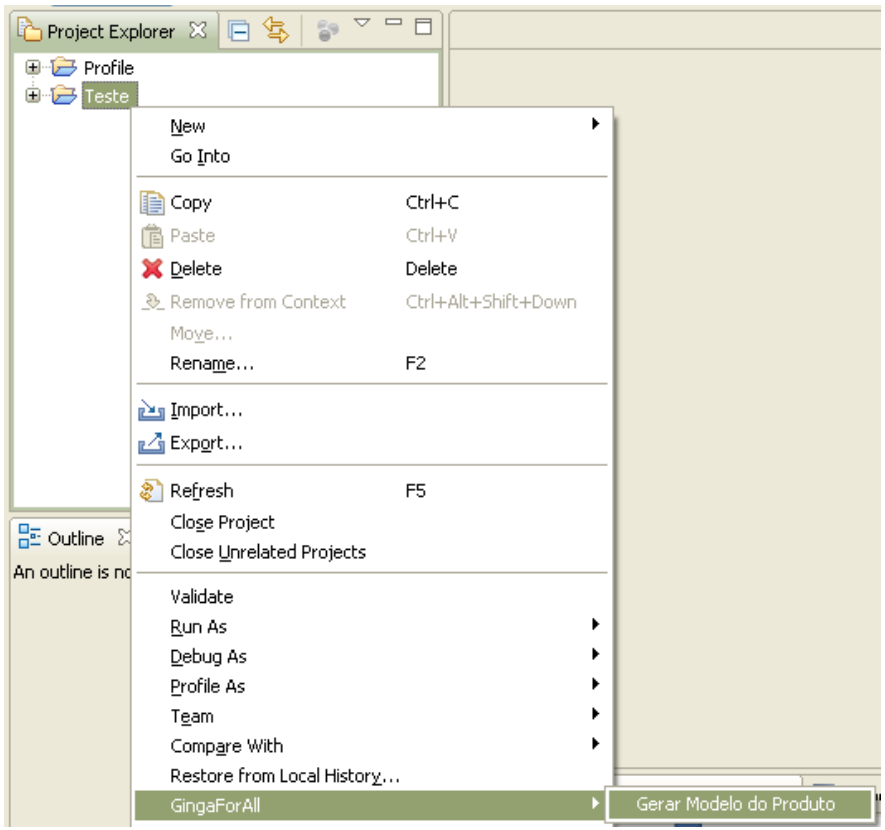
Geração de Produtos

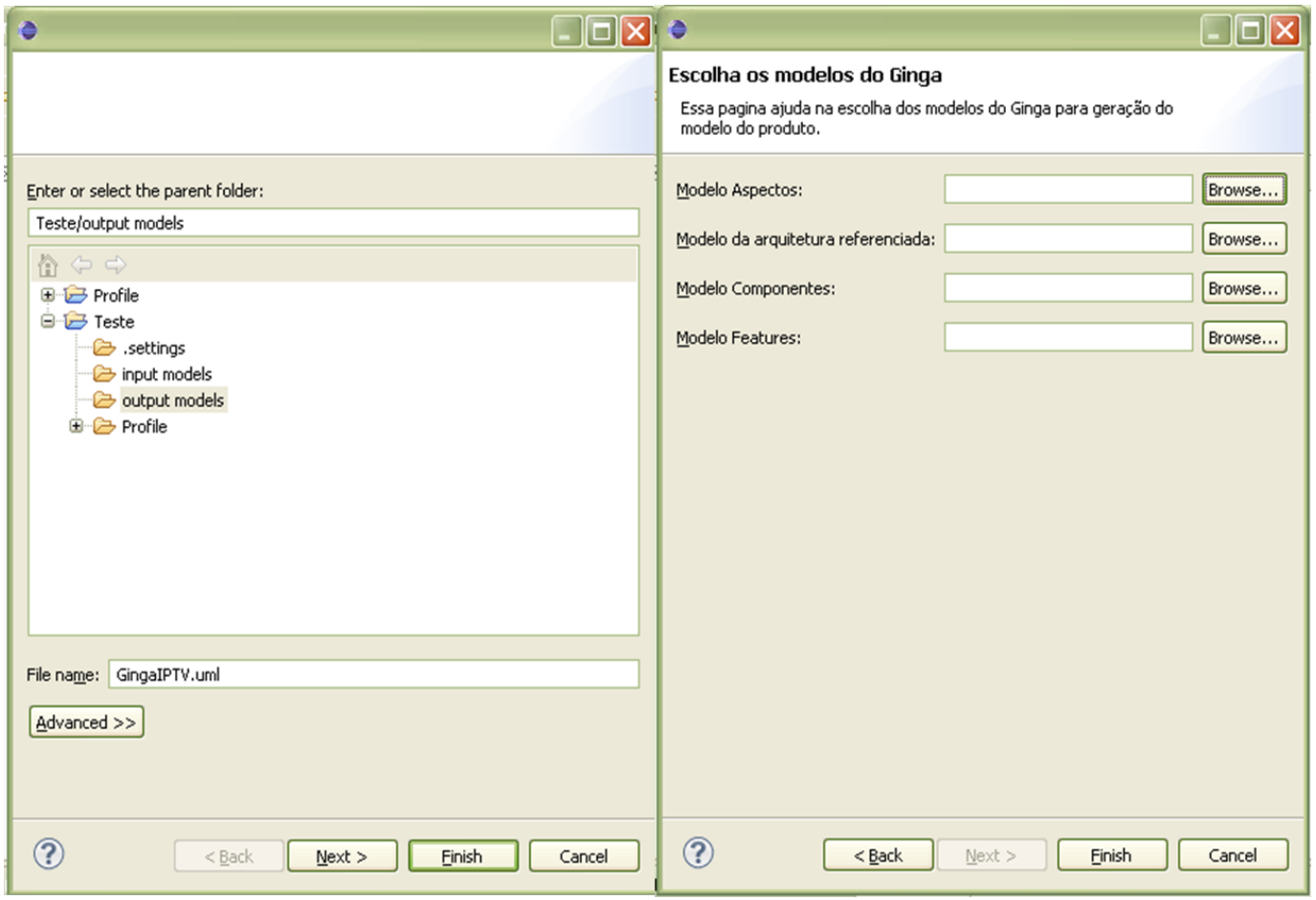
Geração de um modelo do produto a partir dos modelos de entrada

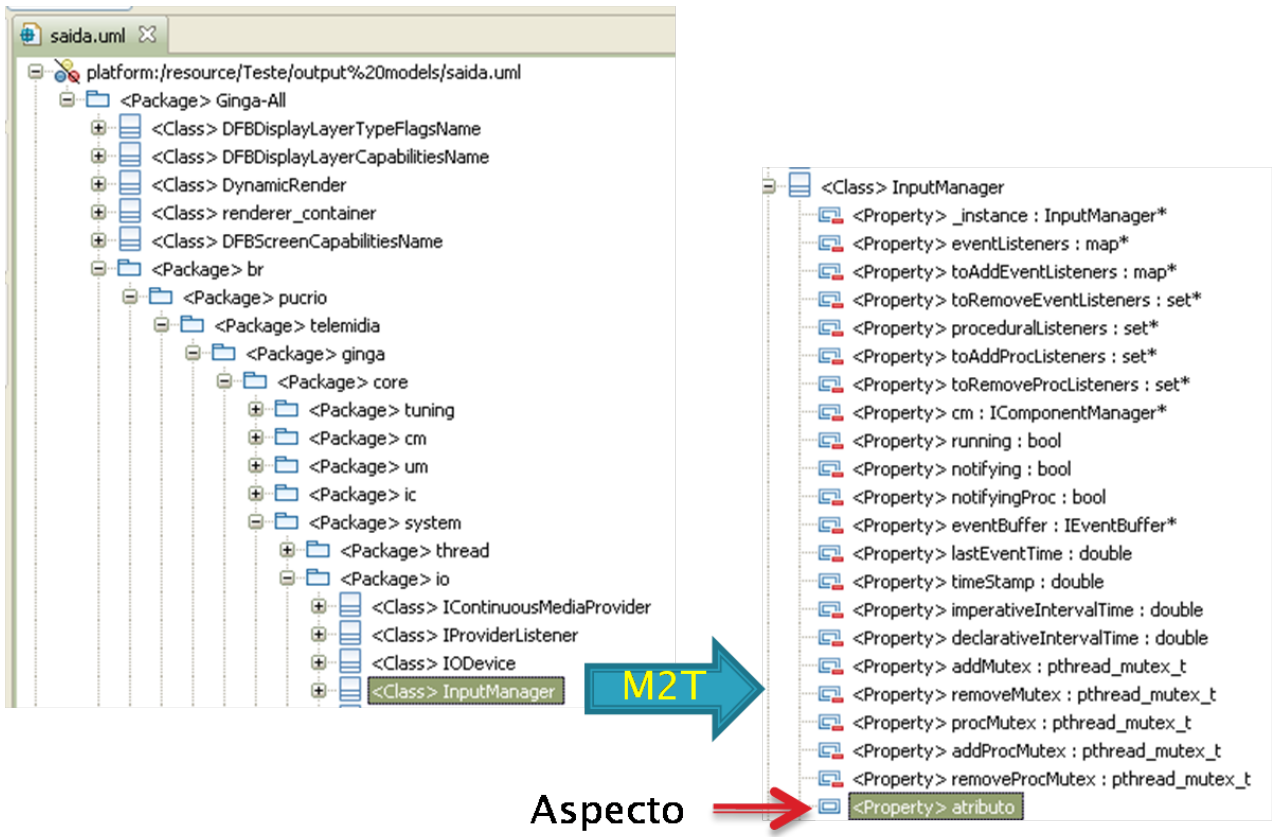
- Uso do QVTO para transformação M2M
- Modelo do Produto UML2

III - Geração do produto




Project Explorer
Profile
Teste
Outline
An outline is no
New
Go Into
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Move...
Rename... F2
Import...
Export...
Refresh F5
Close Project
Close Unrelated Projects
Validate
Run As
Debug As
Profile As
Team
Compare With
Restore from Local History...
GingaForAll
Gerar Modelo do Produto


Enter or select the parent folder:
Teste/output models
Profile
Teste
.settings
input models
output models
Profile
File name:
GingaIPTV.uml
Advanced >>
< Back
Next >
Finish
Cancel
Escolha os modelos do Ginga
Essa pagina ajuda na escolha dos modelos do Ginga para geração do modelo do produto.
Modelo Aspectos:
Modelo da arquitetura referenciada:
Modelo Componentes:
Modelo Features:
Browse...
< Back
Next >
Finish
Cancel

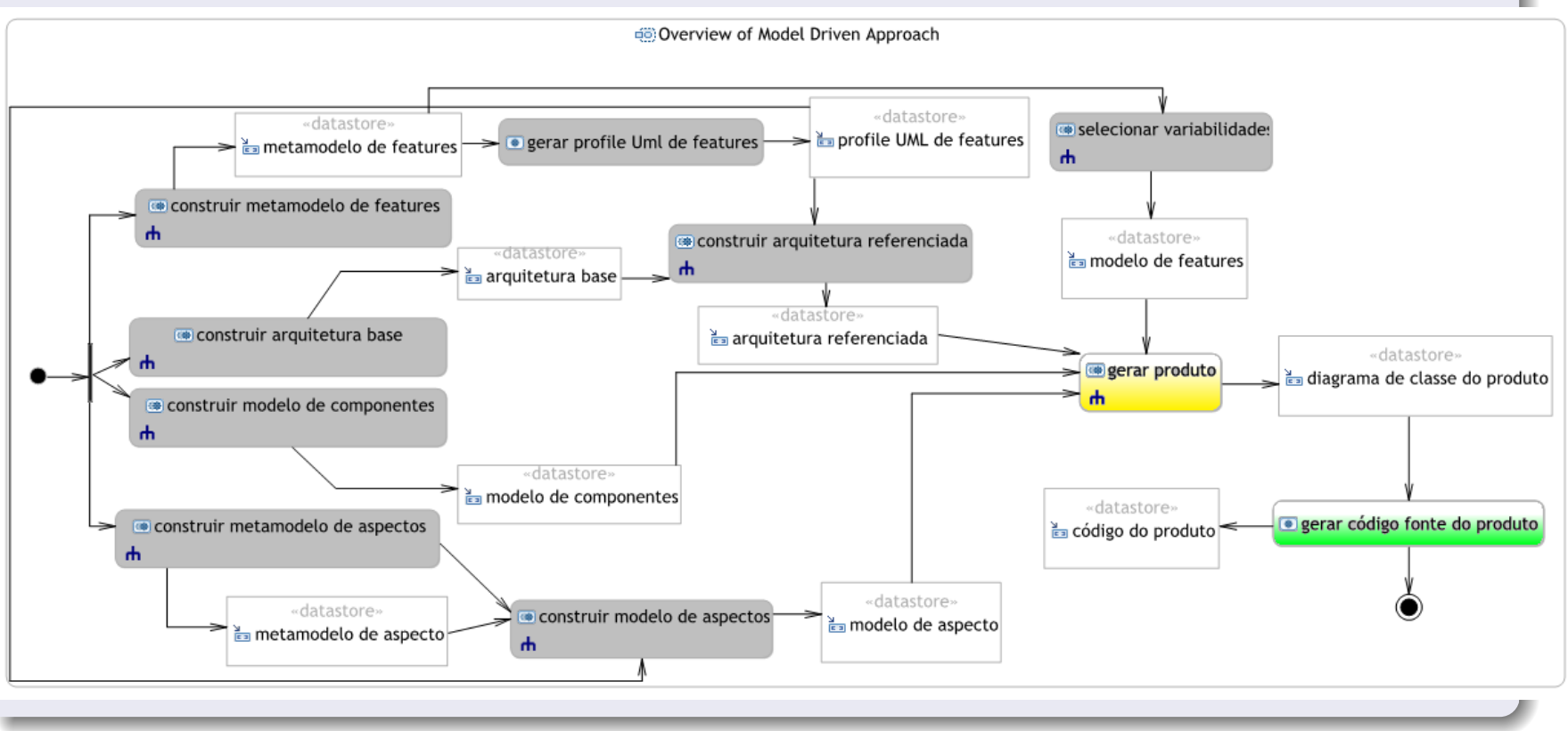

saida.uml
platform:/resource/Teste/output%20models/saida.uml
<Package> Ginga-All
<Class> DFBDisplayLayerTypeFlagsName
<Class> DFBDisplayLayerCapabilitiesName
<Class> DynamicRender
<Class> renderer_container
<Class> DFBScreenCapabilitiesName
<Package> br
<Package> pucrio
<Package> telemidia
<Package> ginga
<Package> core
<Package> tuning
<Package> cm
<Package> um
<Package> ic
<Package> system
<Package> thread
<Package> io
<Class> IContinuousMediaProvider
<Class> IProviderListener
<Class> IODevice
<Class> InputManager
M2T
Aspecto
<Class> InputManager
<Property> _instance : InputManager*
<Property> eventListeners : map*
<Property> toAddEventListeners : map*
<Property> toRemoveEventListeners : set*
<Property> proceduralListeners : set*
<Property> toAddProcListeners : set*
<Property> toRemoveProcListeners : set*
<Property> cm : IComponentManager*
<Property> running : bool
<Property> notifying : bool
<Property> notifyingProc : bool
<Property> eventBuffer : IEventBuffer*
<Property> lastEventTime : double
<Property> timeStamp : double
<Property> imperativeIntervalTime : double
<Property> declarativeIntervalTime : double
<Property> addMutex : pthread_mutex_t
<Property> removeMutex : pthread_mutex_t
<Property> procMutex : pthread_mutex_t
<Property> addProcMutex : pthread_mutex_t
<Property> removeProcMutex : pthread_mutex_t
<Property> atributo

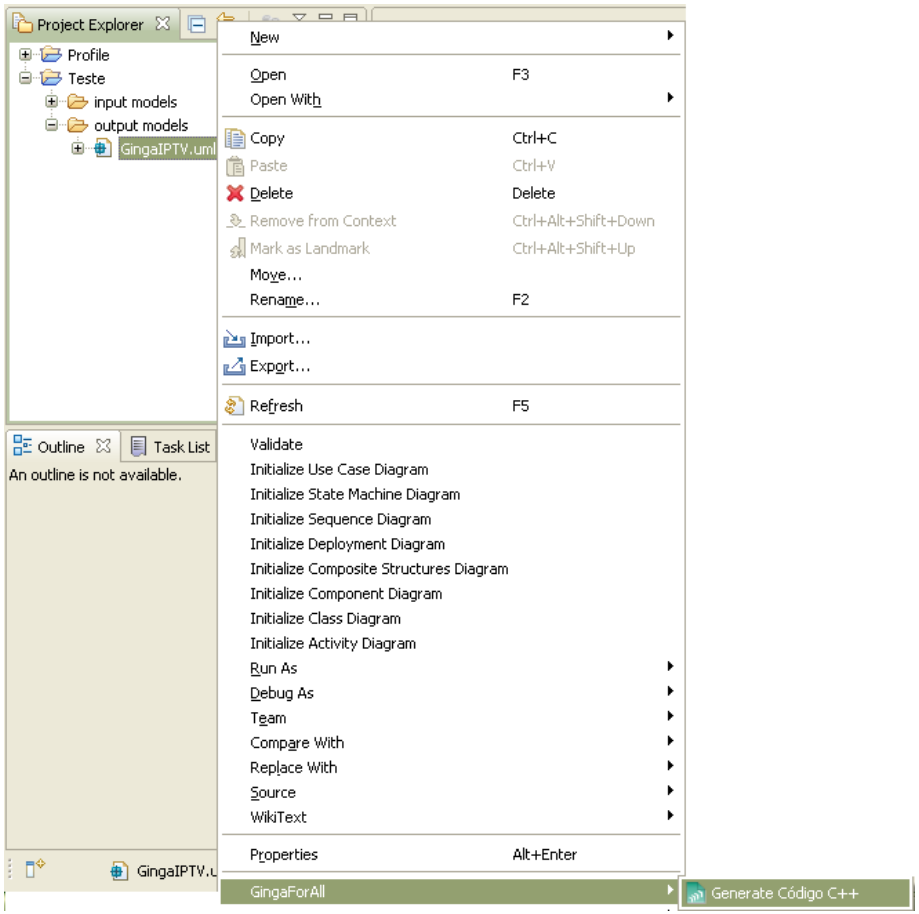
Gerar Código Fonte a partir do modelo do produto

Uso do Acceleo 2 etapas:

- Criação dos Templates: ".mtl"
- Integração com a Ferramenta
- Modelo do Produto UML2

IV - Geração do Código do Produto




Project Explorer
Profile
Teste
input models
output models
GingaIPTV.uml
Outline
Task List
An outline is not available.
New
Open F3
Open With
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Mark as Landmark Ctrl+Alt+Shift+Up
Move...
Rename... F2
Import...
Export...
Refresh F5
Validate
Initialize Use Case Diagram
Initialize State Machine Diagram
Initialize Sequence Diagram
Initialize Deployment Diagram
Initialize Composite Structures Diagram
Initialize Component Diagram
Initialize Class Diagram
Initialize Activity Diagram
Run As
Debug As
Team
Compare With
Replace With
Source
WikiText
Properties Alt+Enter
GingaForAll
Generate Código C++

```
namespace br {
namespace pucrio {
namespace telemidia {
namespace ginga {
namespace core {
namespace system {
namespace io {
namespace InputManager {
#include <InputManager.h>
    //Construtores
        InputManager::~InputManager(){}
        InputManager::InputManager(){}
    // getters e setters
    public:
        bool InputManager::getRunning(){
            return running;
        }
        void InputManager::setRunning(bool running){
            this->running = running;
        }
        map* InputManager::getToAddEventListeners(){
            return toAddEventListeners;
        }
        void InputManager::setToAddEventListeners(map* toAddEventListeners){
            this->toAddEventListeners = toAddEventListeners;
        }
        pthread_mutex_t InputManager::getAddProcMutex(){
            return addProcMutex;
        }
```

Conclusão

O processo de Derivação GingaForAll

- O processo é constituído por um conjunto de atividades sistemáticas que são utilizadas para automaticamente transformar e refinar os modelos que definem a LPS GingaForAll.
- MDD + SPL + Aspectos

Benefícios do processo de derivação

- Gerencia automatica das variabilidades do Ginga
- Geração de esqueleto de código para diferentes produtos Ginga

Site do projeto

- www.dimap.ufrn.br/gingaforall