

Métodos Formais para Especificação de Software

Carla Alessandra Lima Reis
UFPA
2006

Agenda

- Motivação
- Definição e Aplicações
- Exemplos
 - Método Algébrico
 - VDM
 - CSP
 - Redes de Petri
 - Gramáticas de Grafos
- Exmplo no Apsee
- Mitos/Fatos de métodos formais
- Referências

Motivação

- É muito difícil garantir a qualidade de software a partir de inspeções informais
 - Software muito grande
 - Inerente complexidade das linguagens e paradigmas de programação
 - Muita ênfase no teste do software
 - Inexistência de base semântica formal para:
 - verificar *a priori* o comportamento do sistema
 - realizar provas matemáticas acerca do produto (ex: ausência de *deadlock*)

Por que métodos formais?

- Lançador Ariane
 - Explodiu 40s após lançamento em 1996
- Therac-25
 - Entre 1985 e 1987, pelo menos 6 acidentes causaram mortes e ferimentos graves durante terapia de radiação
- Dezenas de outros acidentes
- Perda de controle do desenvolvimento e evolução de sistemas complexos



Motivação

- Métodos de Engenharia de Software
 - Diagrama Entidade-Relacionamento (ou diagrama de classes)
 - Formal, mas com expressividade limitada
 - SQL
 - Semântica fornecida através da álgebra relacional
 - Outros Modelos da Engenharia de Software (exemplo: UML)
 - Semântica descrita em língua natural
 - Idéia geral facilmente entendível por humanos
 - Ambigüidade e informalidade podem levar a interpretações múltiplas



Definição e Aplicações

- Métodos de Especificação Formal
 - Possuem sintaxe e semântica formais para a especificação dos dados, comportamento e funções de um sistema
 - Quando métodos formais são usados no desenvolvimento de um sistema, eles servem de base para a sua verificação formal
 - Permitem a descrição de sistemas complexos a partir de entidades abstratas (independentes de implementação)



Definição e Aplicações

- Métodos de Especificação Formal (cont.)
 - Utilização
 - Prototipação: (para sistemas de qualquer escala)
 - Especificação - em alto nível de abstração - do comportamento do sistema
 - Se suporte computacional estiver disponível, a especificação formal pode ser usada para gerar automaticamente protótipos e derivar programas em linguagens de programação
 - Especificação de sistemas críticos
 - Muito dinheiro ou vidas humanas envolvidas



Definição e Aplicações

- Classificações de métodos formais
 - Mais usual:
 - Orientados a Modelos (denotacional)
 - Orientados a propriedades (axiomática)
 - Orientados a comportamento
 - Híbridos
 - Classificação de [Lamsweerde, 02]
 - History-based
 - State-based
 - Transition-based
 - Functional
 - Operational

Definição e Aplicações

- Tipos de métodos formais
 - Orientados a Modelos
 - Descrevem o comportamento de um sistema a partir de estruturas matemáticas como tuplas, conjuntos, relações e funções
 - Adequados para especificação de estruturas de dados complexas com funções simples
 - Exemplos: VDM (Vienna Development Method) e Z

Definição e Aplicações

- Tipos de métodos formais
 - Orientados a Propriedades
 - Definem o comportamento a partir de propriedades, normalmente na forma de axiomas, as quais o sistema deve satisfazer
 - O projetista utiliza métodos orientados a propriedades para definir o conjunto mínimo de propriedades que um sistema deve satisfazer
 - Exemplos: OBJ, Larch, e especificações algébricas (em geral)

Definição e Aplicações

- Tipos de métodos formais
 - Orientados ao Comportamento
 - Define um modelo a partir da seqüência possível de estados
 - Usados na especificação de sistemas concorrentes, distribuídos e paralelos
 - Exemplos: Redes de Petri, Calculus of Communicating Systems (CCS), Communicating Sequential Processes (CSP), Diagramas de Transição de Estados, LOTOS (Language of Temporal Ordering Specifications)

Definição e Aplicações

- Classificação de [Lamsweerde, 02]
 - History-based
 - Usa lógica temporal para se referir aos estados passados, presentes e futuros.
 - As especificações referem-se a pontos no tempo, intervalos de tempo.
 - State-based
 - Caracterizam os estados do sistema em dado momento.
 - Invariantes e pré e pós condições restringem a aplicação de operações do sistema
 - Exemplos: Z, VDM ou B

Definição e Aplicações

- Classificação de [Lamsweerde, 02]
 - Transition-based
 - Representam as transições de um estado a outro.
 - Exemplos: Statecharts, PROMELA, ...
 - Functional specification
 - Representam um sistema como coleção estruturada de funções.
 - Especificação algébrica – OBJ, Larch
 - Higher-order functions
 - Operational specification
 - Coleção estruturada de processos.
 - Exemplo: Redes de Petri.

Aspectos críticos

- Por que poucos utilizam métodos formais?
 - Poucos métodos são apoiados por ferramentas CASE
 - O uso de métodos formais exige profissionais com boa formação matemática
 - Matemática Discreta
 - Teoria de Conjuntos
 - Teoria das Categorias
 - Notação usada por alguns métodos formais é considerada “muito matemática”
 - Ainda é dada muita ênfase na programação, ao invés das demais etapas do desenvolvimento de software

Importância e aplicabilidade de métodos formais

- Em vez de querer provar que o software está correto (o que em geral custa caro), focar na prevenção de defeitos através da escrita de especificações formais que ajudam a detectar e resolver ambigüidades, omissões e inconsistências desde cedo, e na utilização de técnicas de verificação mais automáticas, ainda que de âmbito limitado.

Método Algébrico

Especificação Algébrica

- Raciocínio necessário para definição é próximo ao paradigma de programação funcional
- Uma Especificação consiste de
 - Um conjunto de nomes de tipos (sorts)
 - Um conjunto de funções
 - Um conjunto de axiomas (semântica)

Exemplos

● Método Algébrico

○ Assinatura:

```
empty: → set
add: set, nat → set
is_empty: set → bool
is_member: set, nat → bool
```

○ Axiomas

```
is_empty(empty()) == true
is_empty(add(s,n)) == false
...
is_member(empty(), n) == false
is_member(add(s,j), n) == (n == j) or is_member(s, n)
```

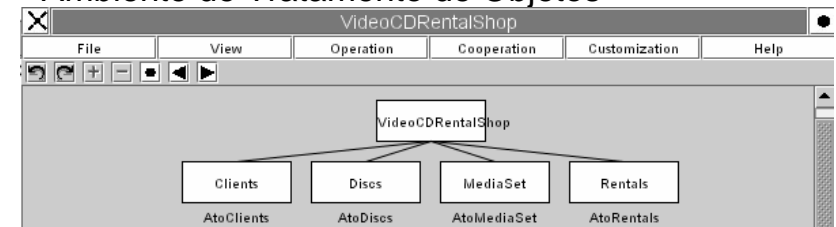
Exemplos

- Prosoft Algébrico
 - Desenvolvido na UFRGS e Uni-Stuttgart
 - Apoiado por um ambiente de desenvolvimento de software denominado Prosoft
 - Escrito em java
 - http://www.informatik.uni-stuttgart.de/ifi/bs/schlebbe/prosoft_doc/guide/
 - Evolução dos métodos algébricos
 - Troca de mensagens (no estilo Orientado a Objetos)

Exemplos

● Prosoft-Algébrico

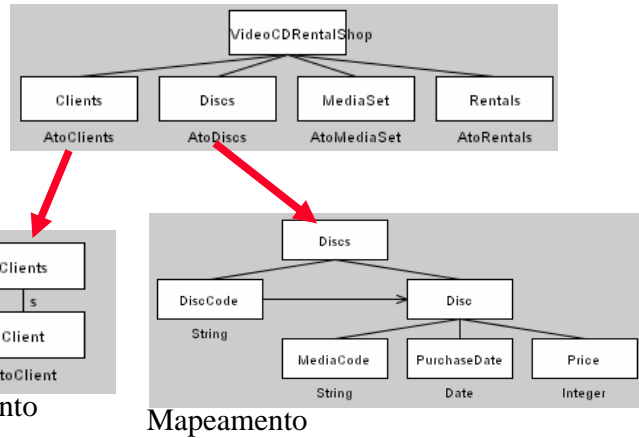
- Notação gráfica para definição de tipos hierárquicos de dados
- Componentes são denominados ATOs: Ambiente de Tratamento de Objetos



Registro

Nó-folha - Referência para Objetos de outras Classes ou para Tipos Primitivos

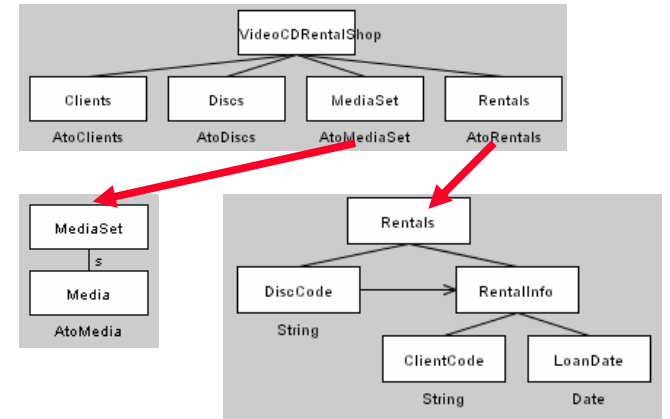
Métodos Formais



Conjunto

Mapeamento

Métodos Formais



Mapeamento

Métodos Formais

INTERFACES

include-client : Clients Client -> Clients
 exclude-client : Clients ClientCode -> Clients
 exist-client : Clients ClientCode -> Boolean

VARIAVEIS FORMAIS

clientcod : ClientCode
 client : Client
 clients : Clients

AXIOMAS


include-client(clients,client)
 = if not(exist-client(clients,ICS(AtoClient,get-clientcode,client)))
 then add(client,clients)
 else clients

exclude-client(add(client,clients),clientcod)
 = if clientcod = ICS(AtoClient,get-clientcode,client)
 then clients
 else add(client,exclude-client(clients,clientcod))

exist-client(add(client,clients),clientcod)
 = if clientcod = ICS(AtoClient,get-clientcode,client)
 then true
 else exist-client(clients,clientcod)

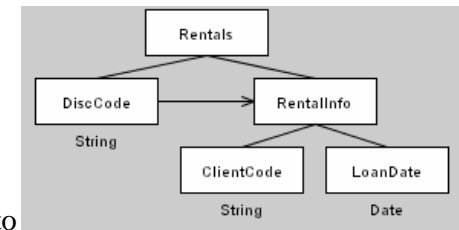
exist-client(emptyset,_) = false
 exclude-client(emptyset,_) = emptyset

Conjunto



Mensagens para outros objetos

Métodos Formais



Mapeamento

Exemplos

Métodos Formais

CLASSE
Rentals

INTERFACES

lend-disc : Rentals DiscCode ClientCode LoanDate -> Rentals
giveback-disc : Rentals DiscCode -> Rentals
is-lended : Rentals DiscCode -> Boolean
client-has-disc : Rentals ClientCode -> Boolean
calculate-bill : Rentals Discs DiscCode LoanDate -> Price

VARIAVEIS FORMAIS

ds : Discs
d : LoanDate
rentals : Rentals
ccod, clientcod : ClientCode
lendeddiscod, disccod : DiscCode

AXIOMAS

```
lend-disc(rentals,disccod,clientcod,today)
= if not(is-lended(rentals,disccod))
  then modify(disccod,reg-RentalInfo(clientcod,today),rentals)
  else rentals

giveback-disc(rentals,disccod)
= if is-lended(rentals,disccod)
  then restrict-with(rentals,add(disccod,emptyset))
  else rentals

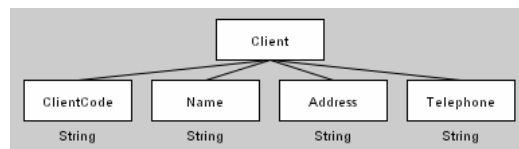
is-lended(rentals,disccod) = disccod belongs-to domain(rentals)

calculate-bill(modify(lendeddiscod,reg-RentalInfo(_,d),rentals),ds,disccod,today)
= if lendeddiscod = disccod
  then if days(d,today) = 0 then
    ICS(AtoDiscs,get-price,ds,<disccod>) else days(d,today)*
    ICS(AtoDiscs,get-price,ds,<disccod>)
  else calculate-bill(rentals,ds,disccod,today)
calculate-bill(emptymap,_,_,_) = 0

client-has-disc(modify(disccod,reg-RentalInfo(ccod,_),rentals),clientcod)
= if ccod = clientcod
  then true
  else client-has-disc(rentals,clientcod)
client-has-disc(emptymap,_) = false
```

mais

Métodos Formais



CLASSE
Client

INTERFACES

get-clientcode : Client -> ClientCode

VARIAVEIS FORMAIS

cli : Client

AXIOMAS

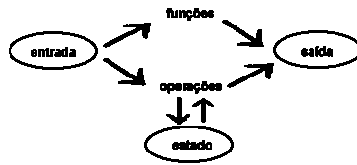
get-clientcode(cli) = select-ClientCode(cli)

Métodos Formais

Vienna Development Method

Métodos Baseados em Modelos

- Componentes de uma Especificação



- Um Modelo matemático é usado para descrever o estado do sistema.

Um Modelo Matemático para Pilhas

var

pilha, pilha' : seq[Int]

operações (procedimentos, métodos)

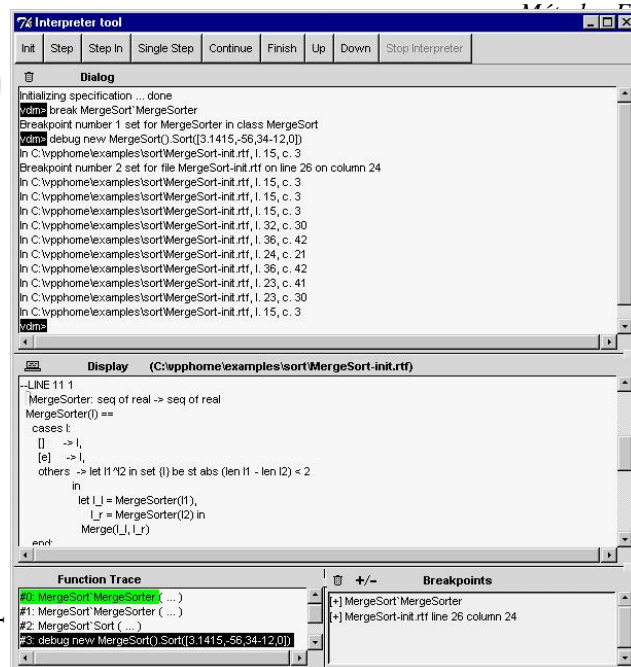
vazia = (pilha' = [])

push(i? : Int) = pilha'=[i?]^pilha

pop() = (pilha ≠ []) => pilha' = tail pilha

top(i! : Int) = (pilha ≠ []) => i! = head pilha

e_vazia (b! : Bool) = b! <=> (pilha = [])




Communicating Sequential Processes



Métodos formais para concorrência

- CSP
 - Tony Hoare, Oxford University
 - Um sistema é visto como um conjunto de processos seqüenciais (programas simples, sem concorrência) que executam de forma autônoma, os quais podem se comunicar
 - Baseado no conceito de evento
 - Alguma coisa que possa ser observada, é atômica e instantânea
 - Coleção de eventos que podem ocorrer com um processo P é dito alfabeto de P (αP)
 - Exemplos: atualização de uma variável, transferência de dados



Exemplos

- CSP: exemplos
 - Relógio
 - $\alpha RELÓGIO = \{tick\}$
 - $RELÓGIO = (tick \rightarrow RELÓGIO)$
 - Robô
 - $\alpha ROBOT = \{forward, back\}$
 - $ROBOT = (forward \rightarrow ROBOT \mid back \rightarrow ROBOT)$
 - Máquina de venda de Chocolates e Toffee
 - $MVCT = \mu X. moeda \rightarrow (choc \rightarrow X \mid toffee \rightarrow X)$



Exemplos

- CSP: exemplos
 - Imagine um consumidor guloso, que deseja consumir chocolate ou toffee sem pagar, somente se não for possível a extração de doces sem pagamento é que ele se dispõe a pagar para comer um chocolate. O comportamento dele é representado por:
 - $GULOSO = (toffee \rightarrow GULOSO \mid choc \rightarrow GULOSO \mid moeda \rightarrow choc \rightarrow GULOSO)$
 - Contudo, a máquina MVCT não permite que sejam extraídos chocolates ou toffees sem o pagamento antecipado, como mostrado na especificação abaixo:
 - $MVCT = \mu X. moeda \rightarrow (choc \rightarrow X \mid toffee \rightarrow X)$
 - A interação entre os processos é:
 - $(GULOSO \mid MVCT) = \mu X. (moeda \rightarrow choc \rightarrow X)$
 (a interação entre dois processos corresponde a um processo que possua somente os eventos em comum a ambos)



Redes de Petri

Exemplos

● Redes de Petri

○ The World of Petri Nets

- <http://www.daimi.aau.dk/~petrinet/>

○ An Introduction to Petri Nets

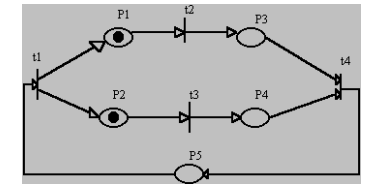
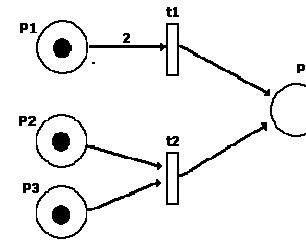
- http://www.daimi.au.dk/PetriNets/introductions/pn2000_introtut.pdf

○ Exemplos e Aplicações

- http://www.dei.isep.ipp.pt/~emt/infind/petri_b.pdf

Exemplo de rede de Petri

- Utilizamos um círculo para representar os locais, um retângulo para representar as transições, uma marca preta para representar as marcações e setas para representar os arcos. Os pesos dos arcos são representados diretamente sobre os arcos.
- Funcionamento: http://pdv.cs.tu-berlin.de/~azi/click_pn/click0.html



Gramáticas de Grafos

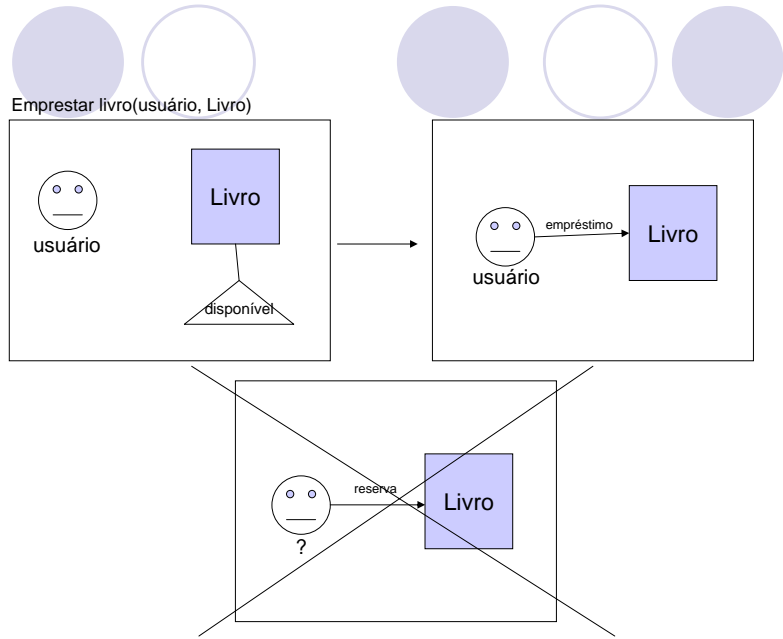
Métodos Formais

● Gramáticas de Grafos

- Desenvolvida em 1995 tendo como base semântica a Teoria das Categorias

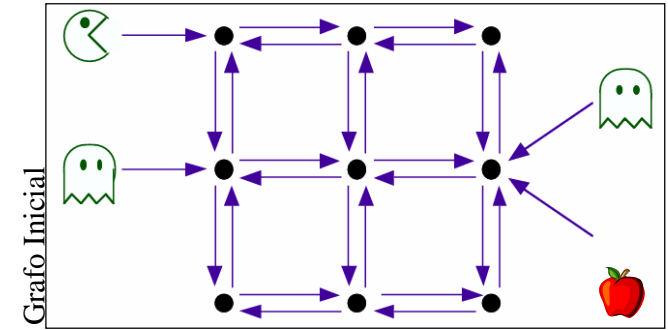
- É essencialmente uma Gramática, onde

- Nós são grafos e podem ter representação gráfica especial
- O paralelismo é implícito



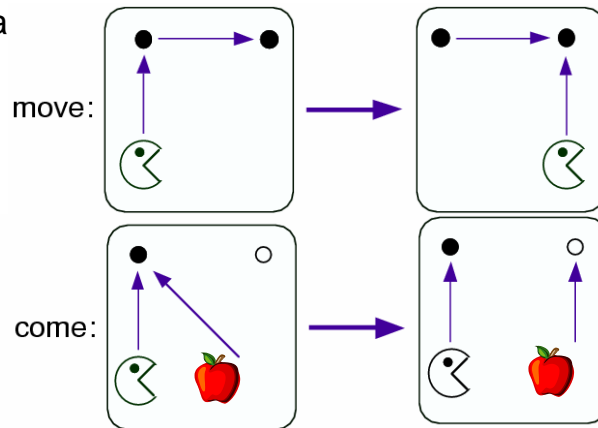
Exemplos

- Gramáticas de Grafos
- Exemplo: Pac-Man



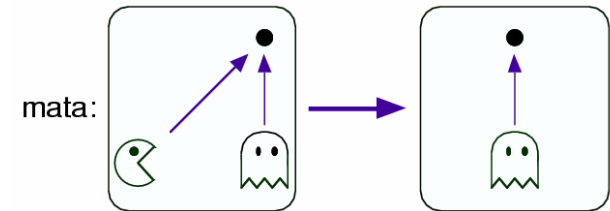
Exemplos

- Gramáticas de Grafos
- Regra



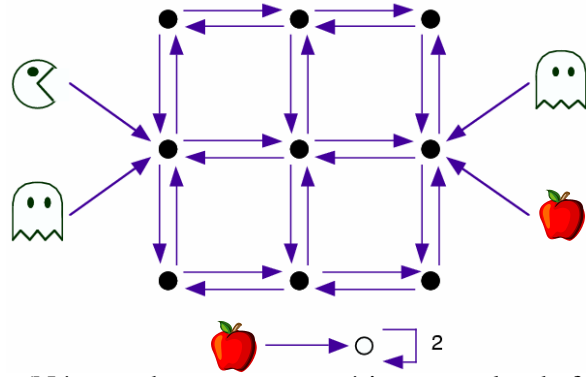
Exemplos

- Gramáticas de Grafos
- Regras (cont.)



Exemplos

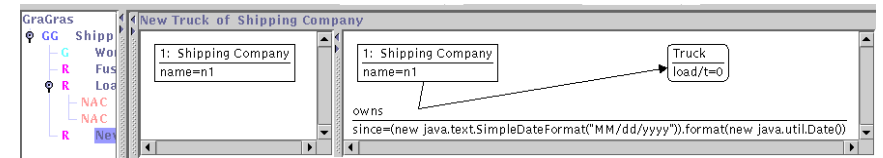
- Gramática de Grafos



(Número de maçãs necessário para pular de fase)

Futuro

- Ferramenta para simulação e geração de código: AGG (TU-Berlin)



Futuro

- Programação para Crianças

○ www.stagecast.com - programação p/ crianças



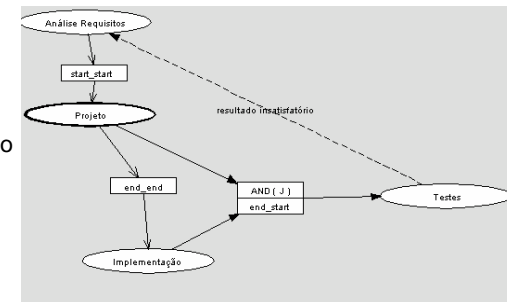
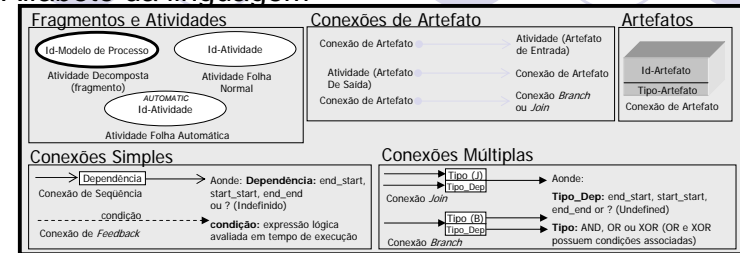
Exemplos Experiência própria: APSEE

Execução de Processos no APSEE

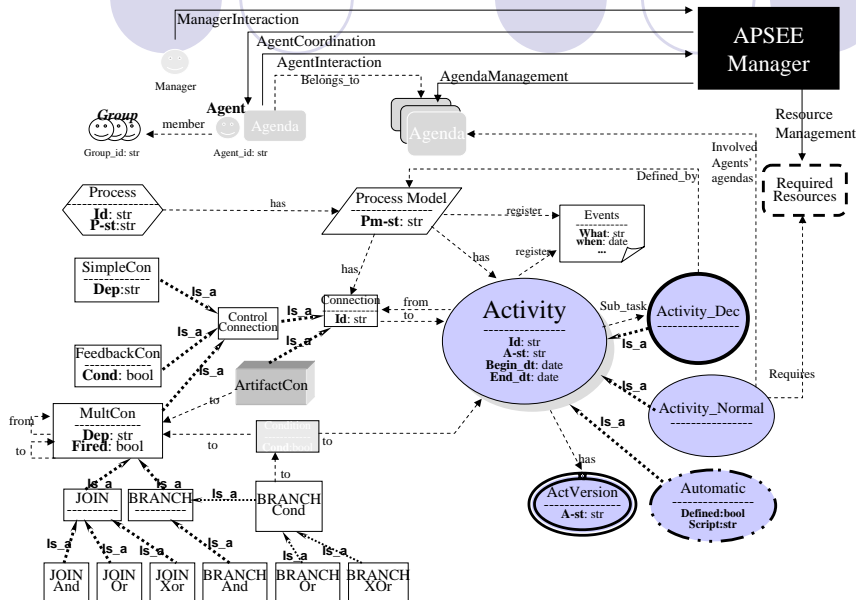
- Uso de Gramática de Grafos para definir a execução de processos
 - Motivação:
 - Devido a complexidade do meta-modelo
 - Devido às transformações complexas que envolvem a execução de processos
 - Proposta de uma linguagem visual
 - A modelagem do sistema é feita em termos de grafos e regras de transformação de grafos
 - Um modelo de processo de software na APSEE-PML é um grafo
 - A execução desse processo é modelada através de regras
 - Abordagem utilizada:
 - Definição de um grafo-tipo (com todos os tipos de arcos e nodos do sistema)
 - Definição de regras de derivação que especificam a transformação do processo

Execução de Processos no APSEE

- Alfabeto da linguagem

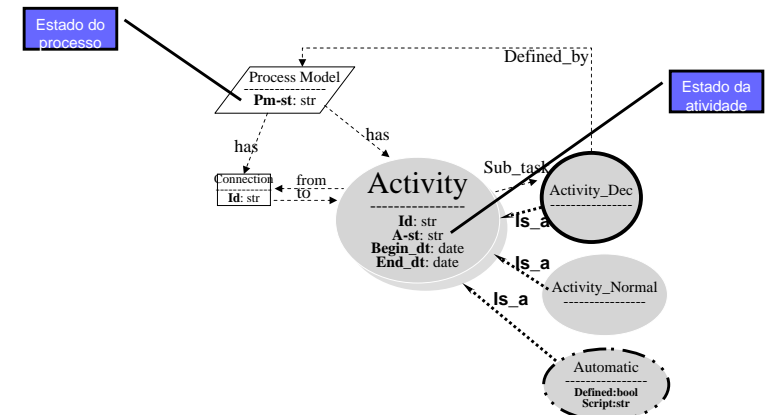


Grafo-Tipo do meta-modelo APSEE



Grafo-Tipo do meta-modelo APSEE

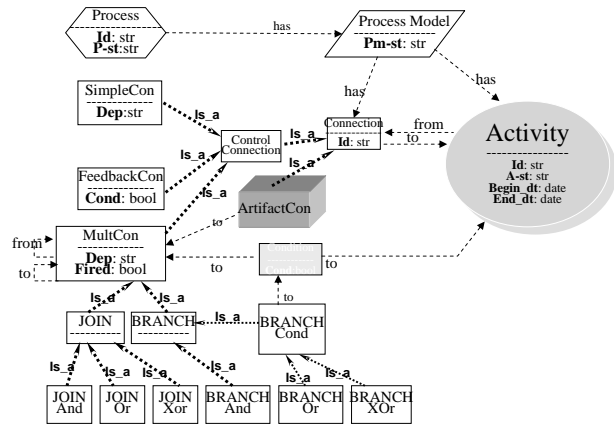
Parte essencial – Modelo de Processo



Um modelo de processo é composto por conexões e atividades; Atividades podem ser decompostas, normais ou automáticas

Grafo-Tipo do meta-modelo APSEE

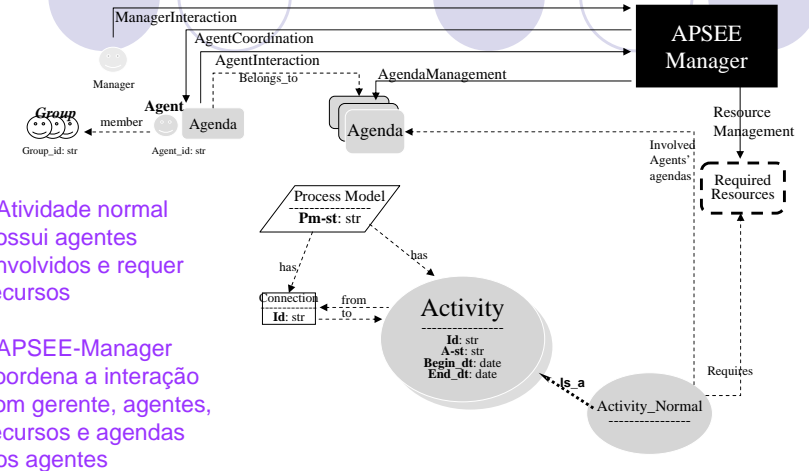
Conexões entre atividades



- Conexões podem ser de controle ou de artefato.
- As conexões de controle São: simples, feedback e múltiplas
- Join e Branch são conexões múltiplas

Métodos Formais

Grafo-Tipo do meta-modelo APSEE



- Atividade normal possui agentes envolvidos e requer recursos
- APSEE-Manager coordena a interação com gerente, agentes, recursos e agendas dos agentes

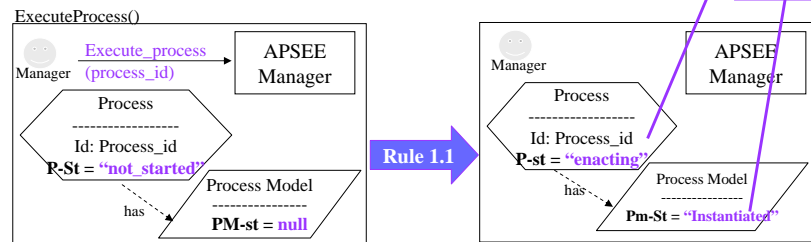
Integração com o modelo organizacional
E interação com usuários

Métodos Formais

Algumas regras de execução

Qualquer regra somente pode ser aplicada se o lado esquerdo ocorre no grafo (processo).

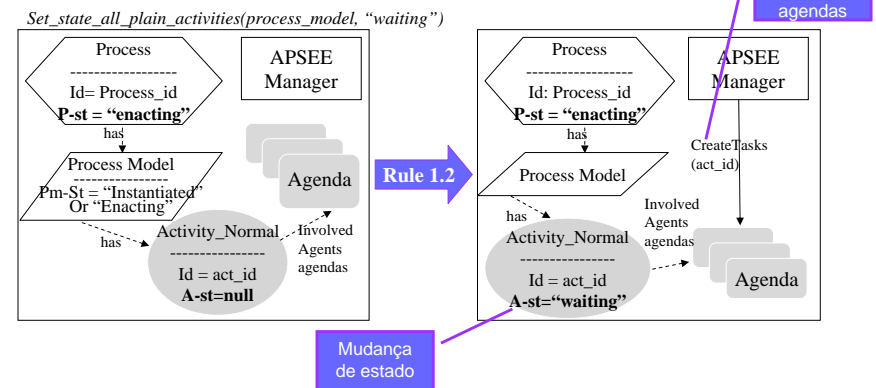
1a. Regra: Gerente solicita execução de processo
Neste caso, a regra é aplicada quando o gerente solicita execução do processo `process_id` que está no estado `not_started` (não iniciado)



Métodos Formais

Algumas regras de execução

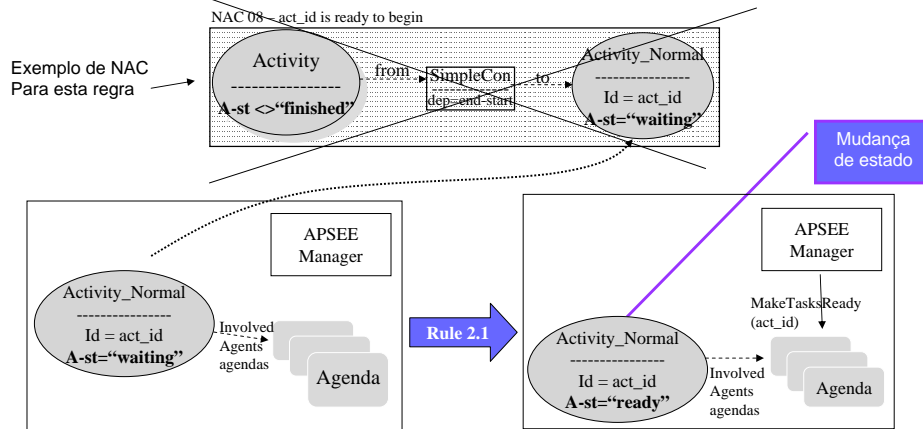
2a. Regra: Inicializando o estado das atividades folha do processo
Neste caso, a regra é aplicada para todas as atividades normais de um processo iniciado que ainda não entraram em execução. A atividade passa para o estado `waiting` e o Apsee-Manager insere a atividade nas agendas dos agentes envolvidos



Exemplo de regra com condição negativa

3a. Regra: Transição de Waiting para Ready:

A regra pode ser aplicada para atividades normais no estado **waiting**, porém somente se a condição negativa não está presente. A condição negativa mostrada impede que a atividade passe para o estado **ready** se depende de uma atividade que não terminou

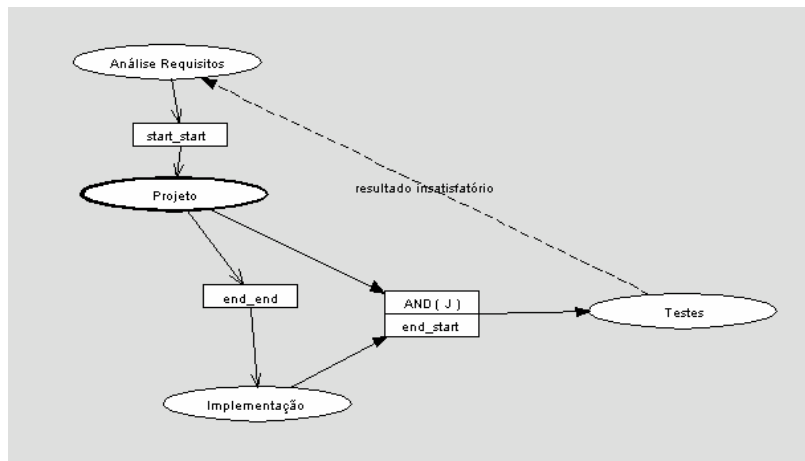


Utilização do formalismo na execução do processo

- Atualmente, 373 regras estão especificadas
 - 170 para execução de processos
 - 203 para garantia da consistência durante a criação de processos e para modificações dinâmicas no processo
- O formalismo foi usado para derivar a implementação do sistema

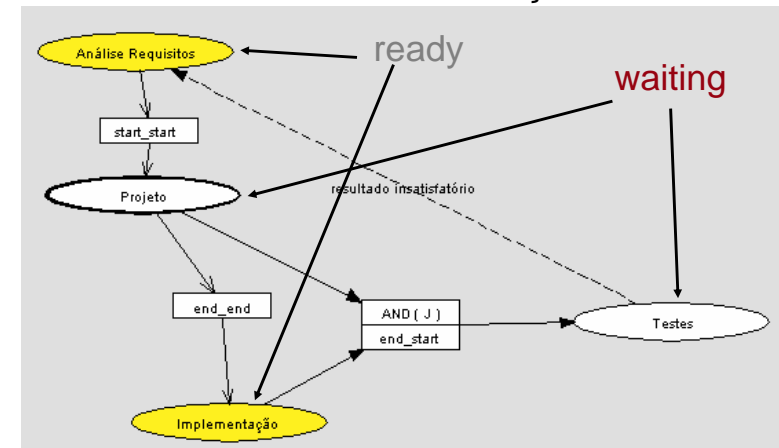
Exemplo de Execução no APSEE

Processo exemplo antes da execução



Exemplo de Execução no APSEE

Gerente solicita execução



Exemplo de Execução no APSEE

Agente envolvido já pode iniciar atividade

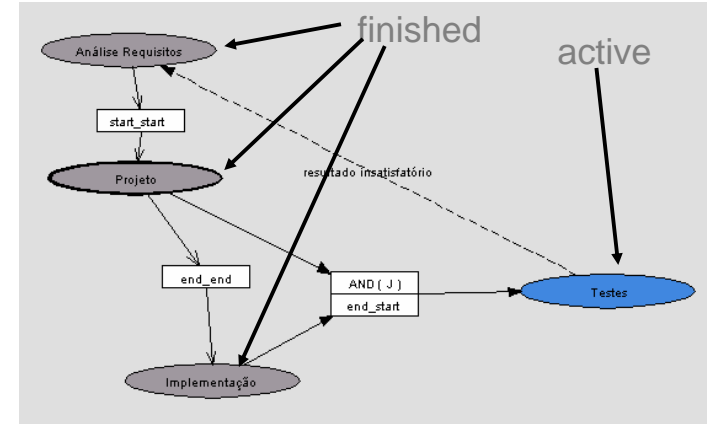
The screenshot shows the 'TaskAgenda' application window. On the left, a small diagram shows a task flow with 'Análise Requisitos' highlighted in yellow. The main window displays a table of tasks:

ID	State	Priority	Begin Date	End Date
Análise Requisitos	Ready	0		

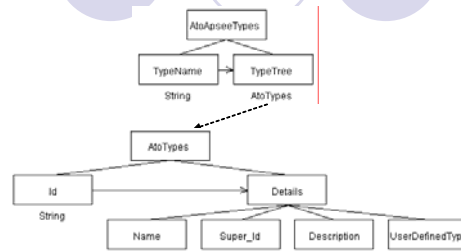
Below the table, there are fields for 'Delegated From', 'Delegated To', and 'Working Hours' (0.0). A 'Task Log' section shows two entries: '[2002-01-20 11:00:47] - Added' and '[2002-01-20 11:00:47] - Made ready'.

Exemplo de Execução no APSEE

Após várias derivações...
Atividade Testes está ativa



Especificação algébrica no APSEE



Super_type_of: Types, String, String → t
 Super_type_of(types, id1, id2) =
 Direct_super(types, id2) = id1 or
 Super_type_of(types, id1, direct_super(types, id2))

Direct_Super: Types, String → String
 Direct_Super(modify(id, (s, Super_id s), r), id1) =
 If id1 = id
 Then s
 Else Direct_Super(r, id1)

Obs: essa função simplesmente obtém o identificador do superior direto de id1. O mapeamento é consumido até encontrar o id procurado. *

Mitos/Fatos de Métodos formais

1. Métodos formais são úteis para encontrar erros mais cedo no desenvolvimento
2. São úteis para fazer você pensar mais sobre o sistema a construir
3. São úteis para qualquer tipo de aplicação, não somente as críticas
4. São baseados em especificações matemáticas, que são mais fáceis de entender do que programas
5. Podem diminuir o custo de desenvolvimento
6. Podem ajudar os clientes a entender o sistema
7. Estão sendo usados com sucesso em projetos industriais

Referências

- Lamsweerde, Axel. Formal Specification: A Roadmap. Proceedings of the conference on The Future of Software Engineering, Finkelstein, A. (ed.) ACM Press, 2002.
- Deharbe, David et al. Introdução a métodos formais: Especificação, Semântica e Verificação de Sistemas Concorrentes. Revista de Informática teórica e aplicada (RITA), Vol VII, Num. 1. Setembro, 2000.
- Hall, Anthony. Seven Myths of Formal Methods. IEEE Software, September, 1990.



Métodos Formais para Especificação de Software

Carla Alessandra Lima Reis

clima@ufpa.br

www.cultura.ufpa.br/clima