Avaliação Automatizada de Sistemas Interactivos

José Creissac Campos

Departamento de Informática

Universidade do Minho

Seminários de Ciências da Computação

Mestrado/Curso de Especialização em Informática

Braga, 19 de Maio de 1997

Plano

1. Verificação (automatizada)

2. Sistemas Interactivos

3. Um Exemplo

Porquê Verificação?

- Complexidade dos sistemas desenvolvidos
  - Bug FDIV [Owre et al., 1996]
- Aplicações *Safety Critical*
  - Incidente com A300-600 - Nagoya
  - Acidente na apresentação de um AirBus
- Cultura das versões β não serve:

$$\left.\begin{array}{l}\circ\ \text{Sistemas complexos}\\ \circ\ \text{Aplicações } \textit{Safety-critical}\end{array}\right\} \longrightarrow \text{desenvolvimento rigoroso}$$

mas...

$$\text{Sistemas Complexos} \Longrightarrow \text{Especificações complexas}$$

$$\downarrow$$

Verificação Formal!

(tanto quanto possível automatizada)

Parte I

Verificação
(automatizada)

Como Fazer Verificação?

- Métodos Algoritmicos $\longrightarrow$ *Model Checking*

  $\rightarrow$ Sistemas Reactivos

  $\rightarrow$ Máquinas de Estados Finitos

  $\rightarrow$ Completamente automático (será?)

  $\rightarrow$ SMV (*Symbolic Model Verifier*), ...

- Métodos Dedutivos $\longrightarrow$ Demonstradores de Teoremas

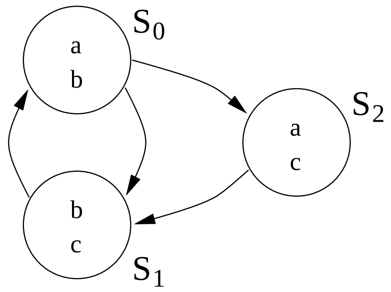
  $\rightarrow$ Genéricos

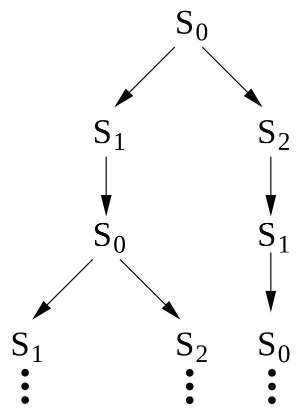
  $\rightarrow$ Dedução Assistida

  $\rightarrow$ PVS (*Prototype Verification System*), ...

O Que É *Model Checking*?

- Proposto por [Clarke et al., 1986]





- Lógica CTL (*Computational Tree Logic*)

$\rightarrow$ AG $\mathcal{P}$ - $\mathcal{P}$ é Universal (um Invariante) — AG (a $\vee$ b $\vee$ c)

$\rightarrow$ AF $\mathcal{P}$ - $\mathcal{P}$ é Inevitável — AF (b $\wedge$ c)

$\rightarrow$ EF $\mathcal{P}$ - $\mathcal{P}$ é Possível — EF (a $\wedge$ c)

$\rightarrow$ AX $\mathcal{P}$ - $\mathcal{P}$ no Próximo Estado — AX (c) / EX (b)

$\rightarrow$ A[$\mathcal{P}_1$U$\mathcal{P}_2$] - $\mathcal{P}_1$ até $\mathcal{P}_2$ — A[a U (b $\wedge$ c)]

SMV

```
MODULE main
VAR
  request : boolean;
  state : {ready,busy};
ASSIGN
  init(state) := ready;
  next(state) := case
                   state = ready & request : busy;
                   1 : {ready,busy};
                 esac;
SPEC
  AG(request -> AF state = busy)
```

SMV (ii)

```
MODULE main
VAR
  request : boolean;
  state : {ready,busy};
INIT
  state = ready
TRANS
  (state = ready & request) -> next(state) = busy
SPEC
  AG(request -> AF state = busy)
```

O Que São Demonstradores de Teoremas?

- Provas são realizadas da forma tradicional
- Sistema dedutivo: Axiomas + Regras de Dedução
- Exemplo:

Axiomas	Regras de Dedução
(1) $\forall_{x,y,z:S} \cdot x \bowtie y \wedge y \bowtie z \rightarrow x \bowtie z$ (2) $b \bowtie c$	$$\frac{P, P \rightarrow Q}{Q}$$

– Provar: $a \bowtie b \rightarrow a \bowtie c$

1.	$a \bowtie b$	premissa
2.	$a \bowtie b, b \bowtie c$	Axioma 2
3.	$a \bowtie b, b \bowtie c, a \bowtie b \wedge b \bowtie c \rightarrow a \bowtie c$	Axioma 1 com $x = a, y = b, z = c$
4.	$a \bowtie c$	Regra de dedução e 3.
5.	QED	

PVS

```
attributes: THEORY
  BEGIN
  Message: TYPE+
  IMPORTING optional_types[Message]
  attributes: TYPE = [# queue: list[Message],
                        current: optional[Message]
                     #]
  presentation: TYPE = [# current: optional[Message] #]

  rho: [attributes -> presentation] =
      lambda(s: attributes): (# current := current(s) #)

  q: VAR attributes
  extheorem: THEOREM current(q)=current(rho(q))
  END attributes
```

Conclusões

Model Checking

- Bons para controlo
- Necessitam de *down-scaling*
- Análise automatizada
- Bons para provas sobre propriedades temporais

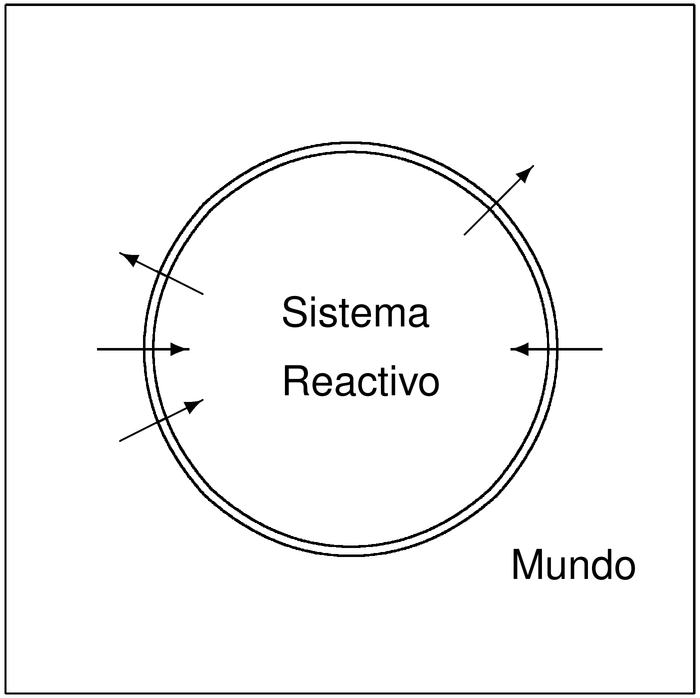
Demonstradores de teoremas

- Bons para estado
- Trabalham ao nível da especificação
- Análise Semi-automatizada
- Nem por isso
- Mais genéricos

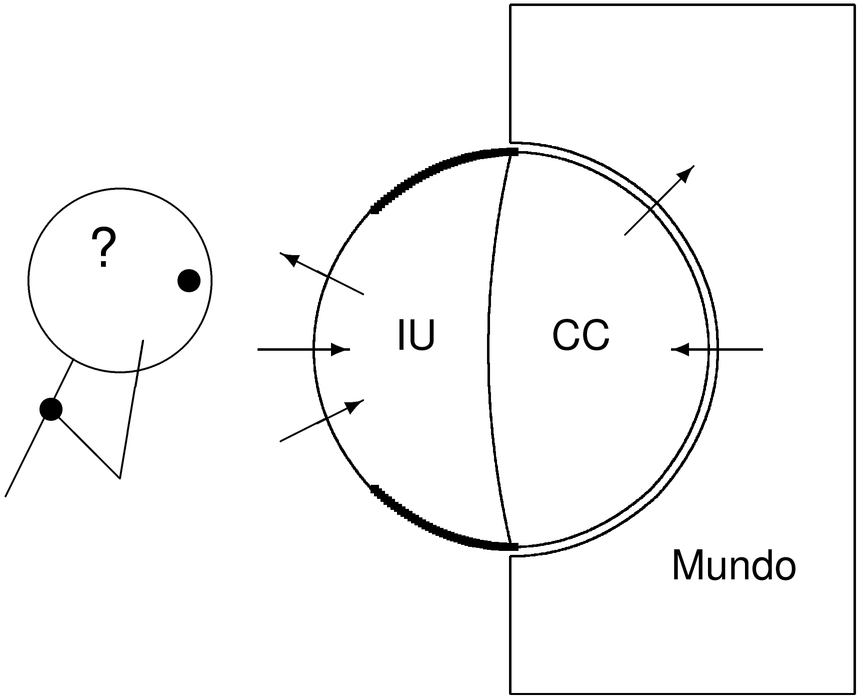
Parte II

Sistemas Interactivos

Porquê Sistemas Interactivos?

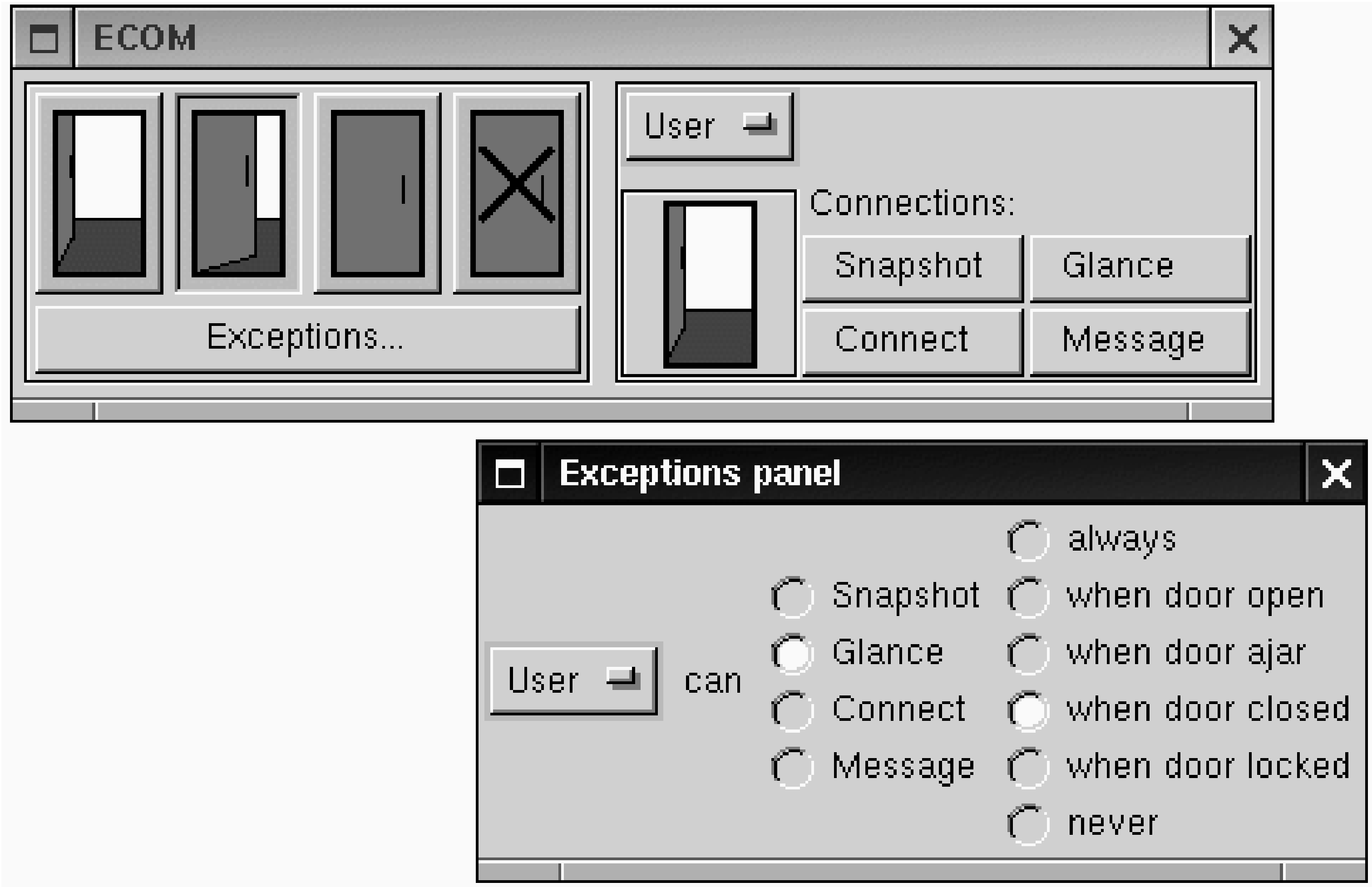


Porquê Sistemas Interactivos?



Factor Humano complica as coisas!

Exemplo


ECOM
Exceptions...
User
Connections:
Snapshot
Glance
Connect
Message
Exceptions panel
User
can
Snapshot
Glance
Connect
Message
always
when door open
when door ajar
when door closed
when door locked
never

O Que É / Como Se Mede A Qualidade?

- Medidas Quantitativas
  - Velocidade
  - Resposta a acções
  - ...

    $\downarrow$

    Verdades objectivas

O Que É / Como Se Mede A Qualidade?

- Medidas Quantitativas
  - Velocidade
  - Resposta a acções
  - ...

  $\downarrow$

  Verdades objectivas

- Medidas Qualitativas
  - Facilidade de utilização
  - Adequação à tarefa
  - ...

  $\downarrow$

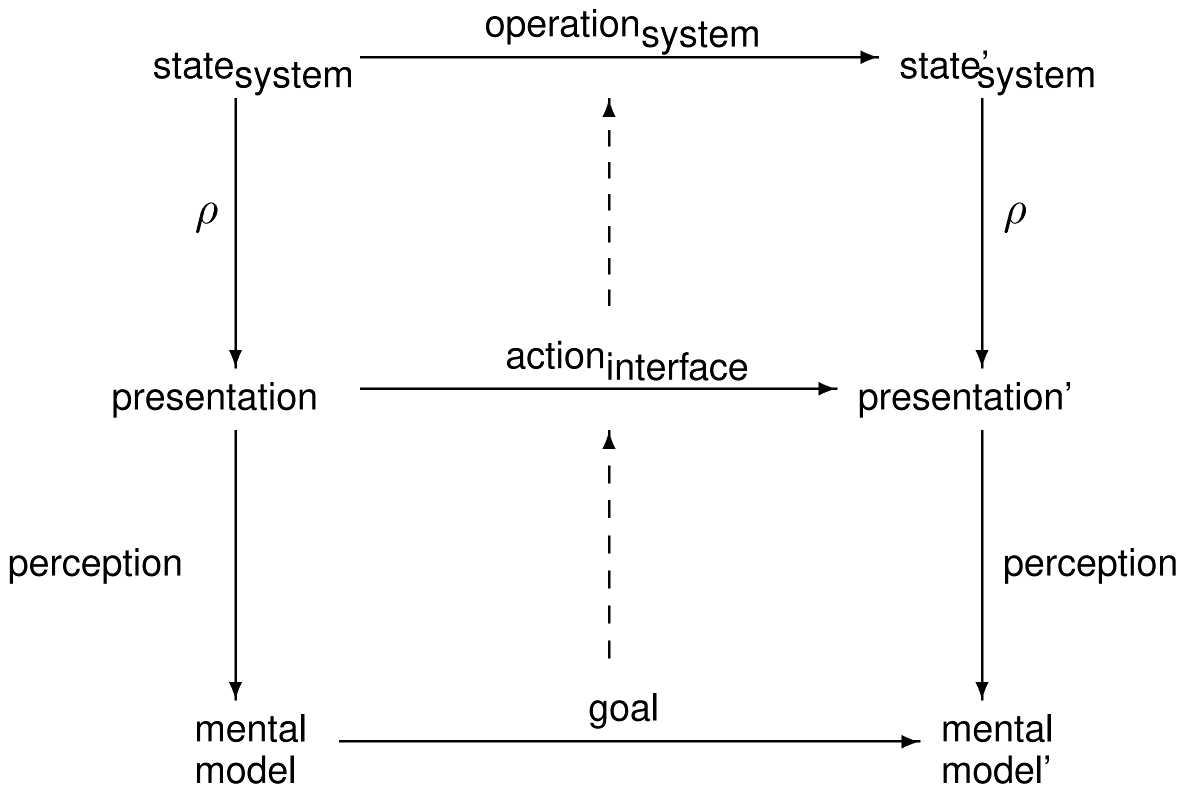
  "***Usabilidade***"

Usabilidade?

$$
\begin{tikzcd}
\text{state}_{\text{system}} \arrow[r, "\text{operation}_{\text{system}}"] \arrow[d, "\rho"'] & \text{state'}_{\text{system}} \\
\text{presentation} \arrow[r, "\text{action}_{\text{interface}}"] & \text{presentation'} \arrow[u, "R"']
\end{tikzcd}
$$

$$\text{action}_{\text{interface}} \circ \rho(\text{state}_{\text{system}}) = \rho \circ \text{operation}_{\text{system}}(\text{state}_{\text{system}}) \tag{1}$$

Usabilidade? (ii)



Modelos mentais?

- Ligados à Psicologia Cognitiva
- Como Formaliza-los?
  - GOMS (Goals, Operations, Methods, Selection rules)
  - PUM (Programmable User Models)
  - ICS (Interacting Cognitive Subsystems)
- Separados do “Modelo do Sistema”

  $\downarrow$

  Syndetic Modelling [Duke et al., 1995] (modelo do sistema + modelo do utilizador)

Problema: Complexidade...

· dos modelos do utilizador.

GOMS - Goals, Operations, Methods, Selection rules

```
Goal: Locate Line
.     Choose Command                            ...(if not
.     [select Goal: use Line-feed Method            at line)
.                 .     Goal: specify Command
.             Goal: use Quote-String Method
.                 .     Goal: specify Command
.                 .     Goal: specify Argument
.             Goal: use Scroll Method
.                 .     Goal: specify Command
.                 .     Goal: specify Argument] ...(repeat
                                                    until
                                                    at line)
```

PUM - Programmable User Models

```
OBJECTS
file: addresses, contacts
disc: hard, floppy
folder: assay-demo

RELATIONS
file-on-disk(file, disk)
folder-on-disk(folder,disk)
file-in-folder(file,folder)   - detected by looking
file-on-desktop(file)         - detected by looking

OPERATIONS
operation move-file (file: F, folder: R)
  user-purpose:       file-in-folder(F, R) &
                      not file-on-desktop(F).
```

```
  filtering-precond: there is a D such that
                     file-on-disk(F, D) &
                     folder-on-disk(R, D)
  action:            drag-file(F, R)

DESIRED-STATE
  file-in-folder(contacts, assay-demo)

DEVICE-COMMANDS
  drag-file (file:F, folder: Dest)

INITIAL
  folder-on-disk(assay-demo, hard)
  file-on-desktop(contacts)
  file-on-disk(contacts,hard)
```

Abordagens de Engenharia de Software

- [Abowd et al., 1995] — AS + SMV
  - Abordagem baseada em sistema de produções
- [Paternò, 1995] — LOTOS + Logic Checker
  - Perde informação ao passar de LOTOS para LC
- [d'Ausbourg et al., 1996] — Lustre
  - Usa uma linguagem *Data Flow*
- [Bumbulis et al., 1996] — HOL
  - Analisa a arquitectura da interface
- Separadas do "modelo do utilizador" [Campos and Harrison, 1997]

Problema: Complexidade...

· dos sistemas a modelar.

Soluções

- Complexidade dos modelos do sistema:
  - Dividir para reinar

    $\downarrow$

    Multiplos Modelos

Soluções(ii)

- Complexidade dos modelos do sistema:
  - Dividir para reinar

    $\downarrow$

    Multiplos Modelos $\longrightarrow$ modelos à medida

    $\Downarrow$

    Possibilidade de ajustar tecnologia ao problema

Soluções(iii)

- Complexidade dos modelos do sistema:
  - Dividir para reinar

    $\downarrow$

    Multiplos Modelos $\longrightarrow$ modelos à medida

    $\Downarrow$

    Possibilidade de ajustar tecnologia ao problema
- Necessidade de modelos do utilizador:
  - **Não usar!**

    (explicitamente)

    $\downarrow$

    Modelos do utilizador são construidos com base em suposições acerca do comportamento dos utilizadores.

Soluções (iv)

- Complexidade dos modelos do sistema:
  - Dividir para reinar

    $\downarrow$

    Multiplos Modelos $\longrightarrow$ modelos à medida

    $\Downarrow$

    Possibilidade de ajustar tecnologia ao problema

- Necessidade de modelos do utilizador:
  - **Não usar!**

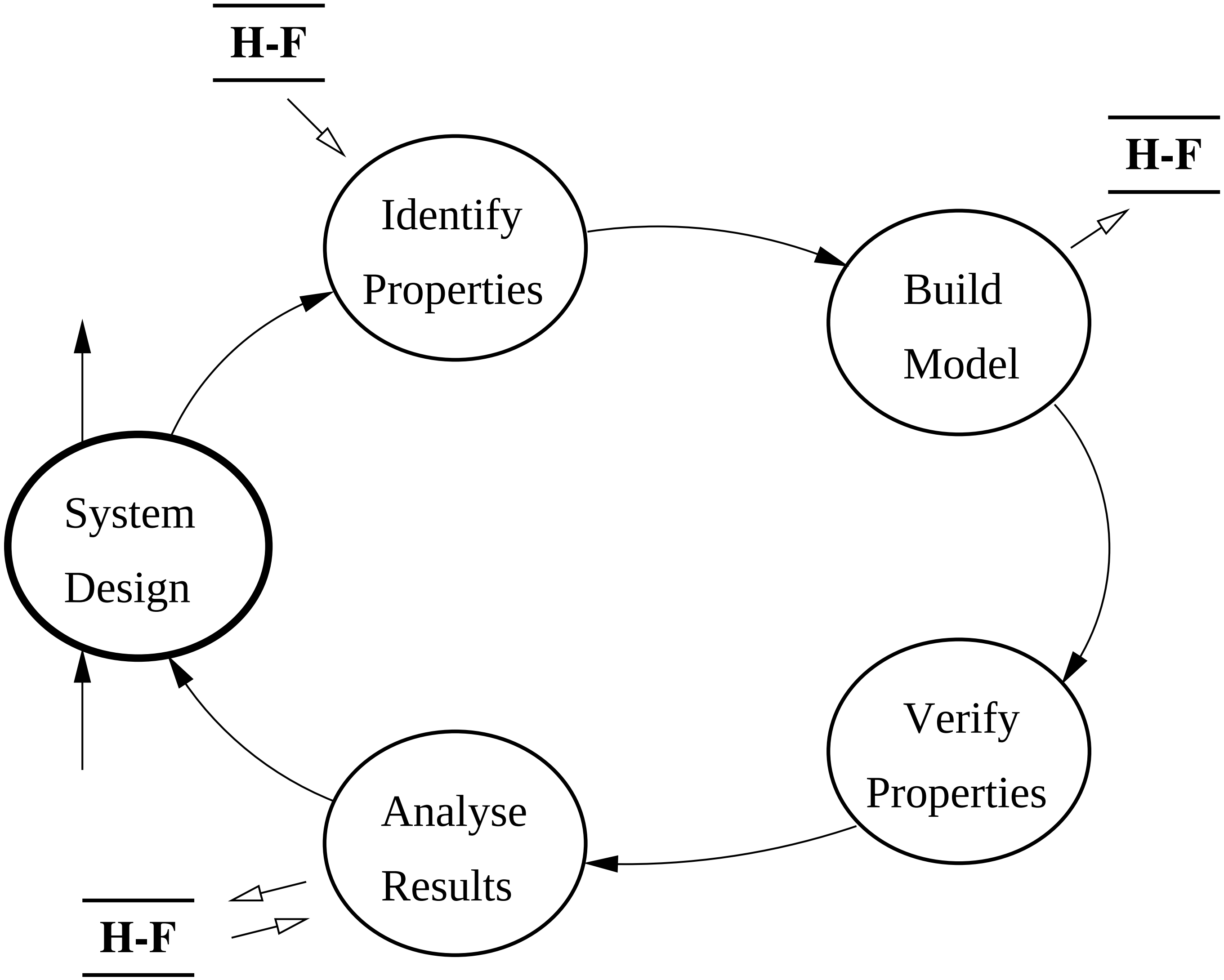
    (explicitamente)

    $\downarrow$

    Modelos do utilizador são construidos com base em suposições acerca do comportamento dos utilizadores.

    $\Downarrow$

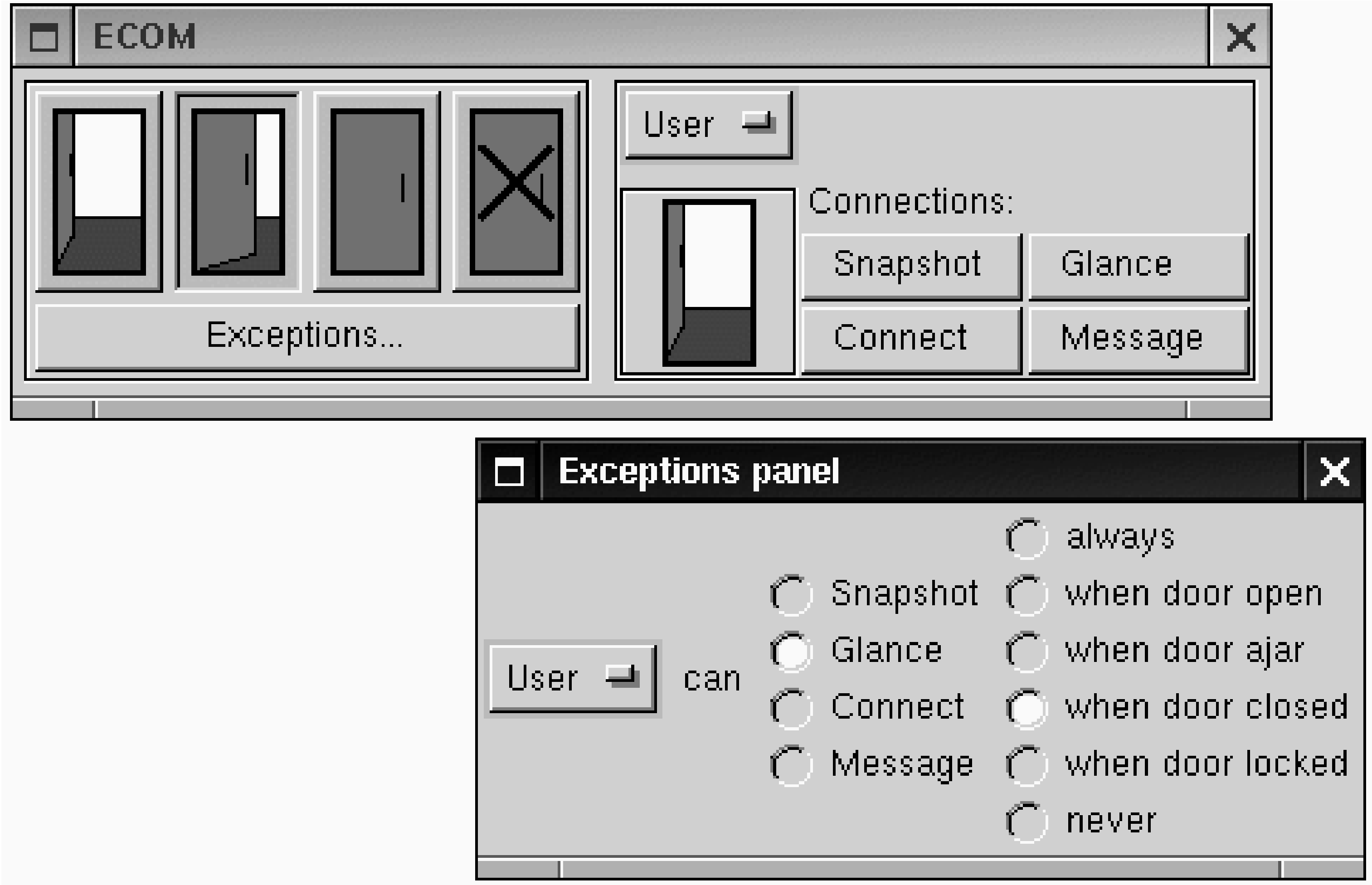
    Incorporar essas suposições nas propriedades a verificar!


H-F
Identify Properties
H-F
Build Model
System Design
Verify Properties
Analyse Results
H-F

Parte III

Exemplos

ECOM (revisitado)



Analisar Previsibilidade

Consegue um utilizador prever se tentar uma ligação vai ter sucesso?

$\downarrow$

Desenvolver modelo (para que ferramenta?)

$\downarrow$

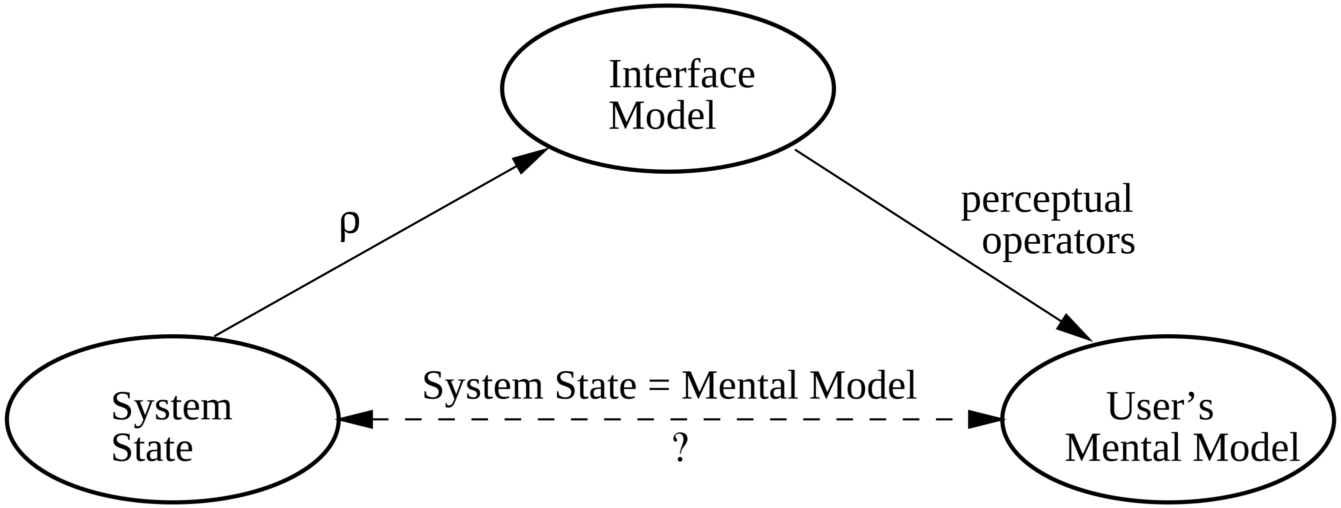
Formalizar propriedade

$\downarrow$

Fazer verificação

`(Demonstração da utilização do SMV)`

Analisar Adequação da interface



(Demonstração da utilização do PVS)

Conclusões

Não esquecer o utilizador!

References

[Abowd et al., 1995] Abowd, G. D., Wang, H.-M., and Monk, A. F. (1995). A formal technique for automated dialogue development. In *Proceedings of the First Symposium of Designing Interactive Systems - DIS'95*, pages 219–226. ACM Press.

[Bodart and Vanderdonckt, 1996] Bodart, F. and Vanderdonckt, J., editors (1996). *Design, Specification and Verification of Interactive Systems '96*, Springer Computer Science. Springer-Verlag/Wien.

[Bumbulis et al., 1996] Bumbulis, P., Alencar, P. S. C., Cowan, D. D., and Lucena, C. J. P. (1996). Validating properties of component-based graphical user interfaces. In [Bodart and Vanderdonckt, 1996], pages 347–365.

[Campos and Harrison, 1997] Campos, J. C. and Harrison, M. D. (1997). Formal verification of interactive systems: A review. In [Harrison and Torres, 1997], pages 109–124.

[Clarke et al., 1986] Clarke, E. M., Emerson, E. A., and Sistla, A. P. (1986). Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263.

[d'Ausbourg et al., 1996] d'Ausbourg, B., Durrieu, G., and Roché, P. (1996). Deriving a formal model of an interactive system from its UIL description in order to verify and to test its behaviour. In [Bodart and Vanderdonckt, 1996], pages 105–122.

[Duke et al., 1995] Duke, D., Barnard, P., Duce, D., and May, J. (1995). Syndetic modelling. Technical Report ID/WP49, Amodeus-2 project. citar antes DukeDBM98.

[Harrison and Torres, 1997] Harrison, M. D. and Torres, J. C., editors (1997). *Design, Specification and Verification of Interactive Systems '97*, Springer Computer Science. Eurographics, Springer-Verlag/Wien.

[Owre et al., 1996] Owre, S., Rushby, J., and Shankar, N. (1996). Analyzing tabular and state-transition requirements specifications in PVS. Technical Report CSL-95-12, SRI International Computer Science Laboratory.

[Paternò, 1995] Paternò, F. D. (1995). *A Method for Formal Specification and Verification of Interactive Systems*. PhD thesis, Department of Computer Science, University of York.