



Modélisation d'Entreprise et V&V

Complémentarité des
approches, des techniques et
intérêts de la V&V



V.Chapurlat
LGI2P- Laboratoire de Génie Informatique et
d'Ingénierie de Production - France

1

Objectifs non obligatoirement dans le bon ordre...

- ☞ **Etre ambitieux...** Quel est le réel intérêt et la nécessité de la vérification et de la validation (V&V) ?
- ☞ **Prendre un peu de recul...** Quelles sont les techniques classiques (de la moins formelle à la plus formelle) utilisées dans d'autres domaines avec des résultats reconnus ?
- ☞ **Proposer...** Comment introduire ces principes et ces techniques en Modélisation d'Entreprise ? Avec quels intérêts ou quels verrous ?
- ☞ **Démontrer que c'est faisable et important...** Des exemples de techniques...
- ☞ **Ecouter vos questions, idées, positions...**

2

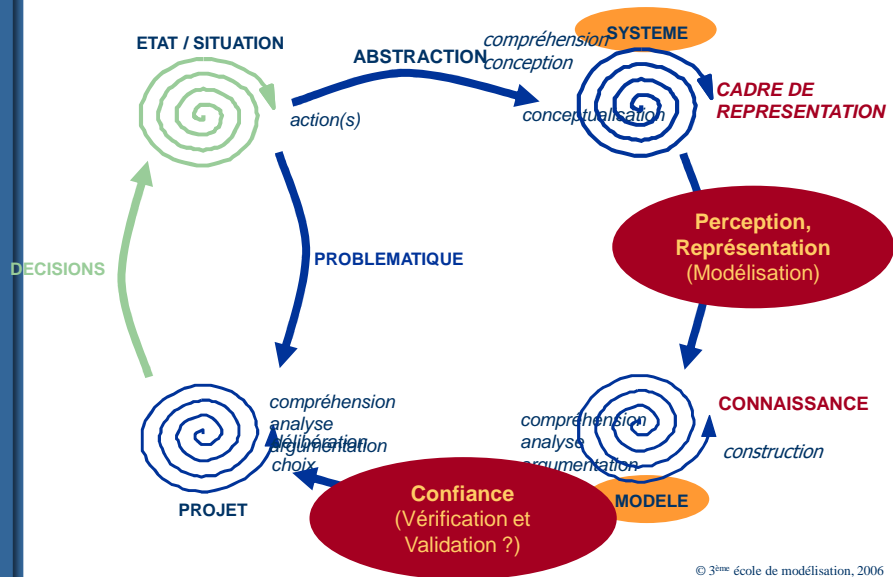
Introduction



V.Chapurlat
LGI2P- Laboratoire de Génie Informatique et
d'Ingénierie de Production - France

3

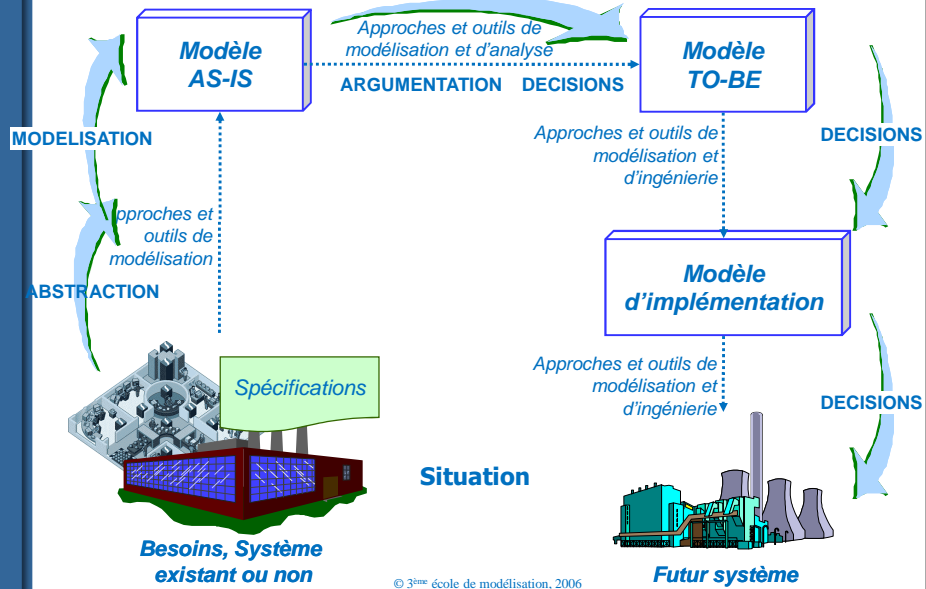
Idée / système / modèle(s) [Ecole de Modélisation Risque et ME, Arcachon 2007]



4

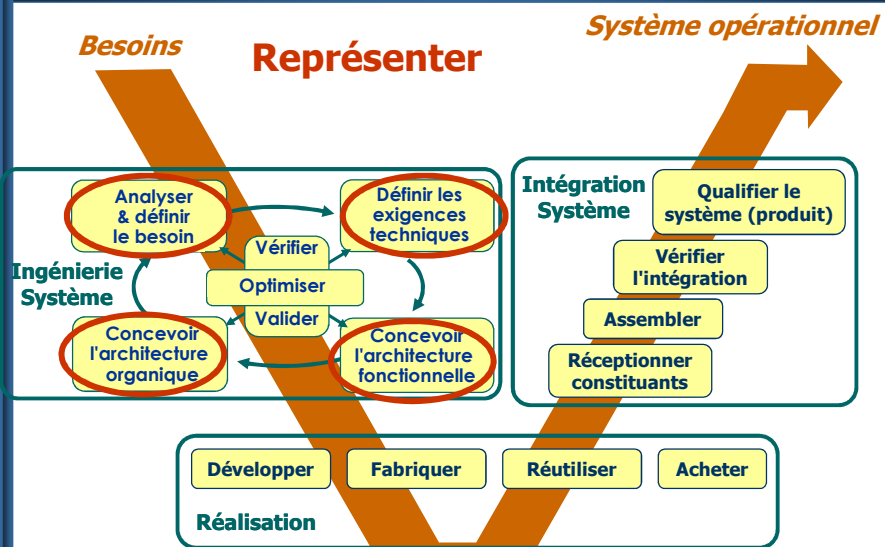
© 3^{ème} école de modélisation, 2006

En ME [Ecole de Modélisation Risque et ME, Arcachon 2007]



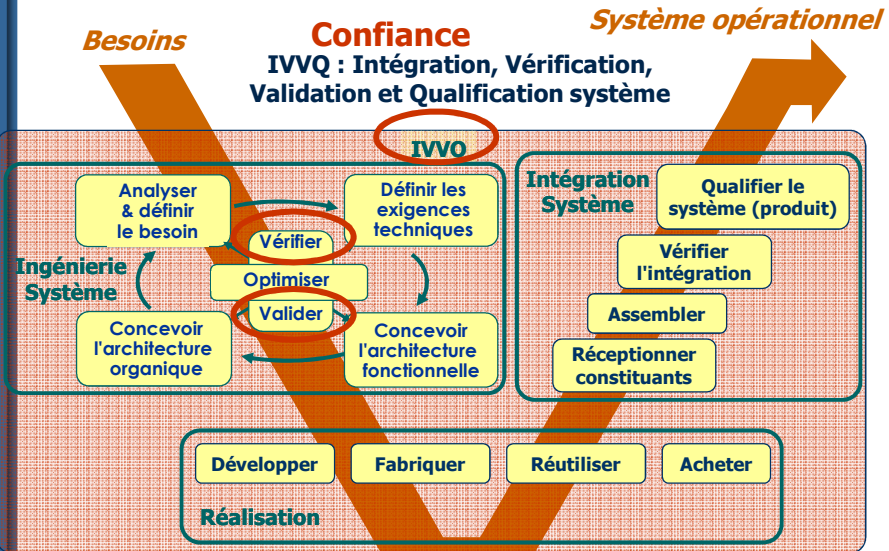
5

Un parallèle : Ingénierie Système



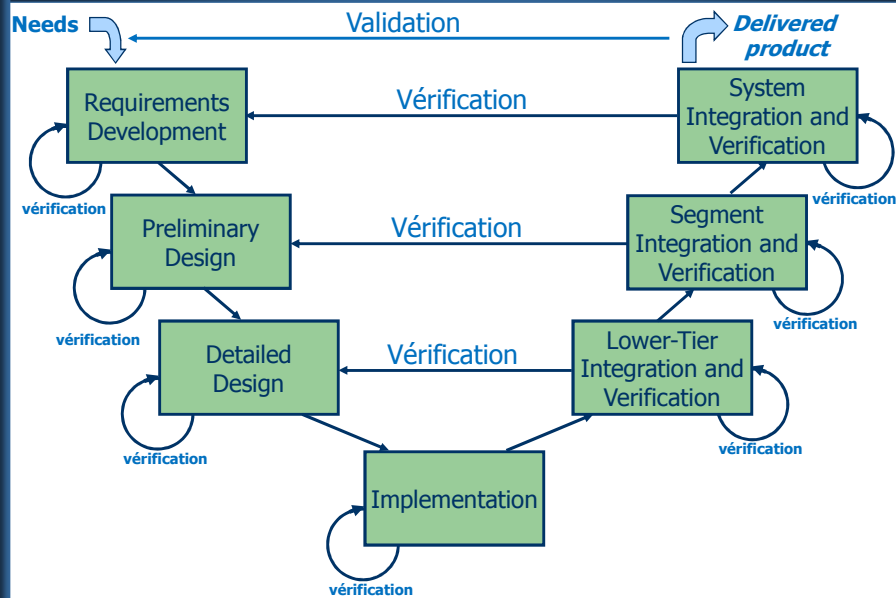
6

Un parallèle : Ingénierie Système



7

Un autre parallèle : le Génie Logiciel...



8

Mais aussi...

- ↙ Analyse et qualification de protocoles de communication temps réel, déterministes, ...
- ↙ Analyse d'assemblages de composants électroniques avant fabrication
- ↙ Simulation du fonctionnement de systèmes de production
- ↙ Vérification de l'assemblage structurel et fonctionnel de composants intégrés
- ↙ ...
- ↙ **Donc... Systèmes techniques... Quid des systèmes sociotechniques ?**

9

Quelle confiance ? V&V ? VVQC ? IVVQ ?

[Popkin 2003] : 'A model is a representation of concepts that:

- captures and communicates ideas
- can be changed
- can provide scenarios ('what if' and 'where do I fit in?')
- can be validated
- can be checked for rigor and robustness'

'The lack of confidence the actors may have on the obtained representations [...] due to a lack of convincing modeling tool, of really standardized method and formalisms and of verification and validation means' - EM as a tool', Verdal, Norway 1999

10

Définitions

Vérifier : Le modèle est bien construit et est cohérent du point de vue de son méta modèle

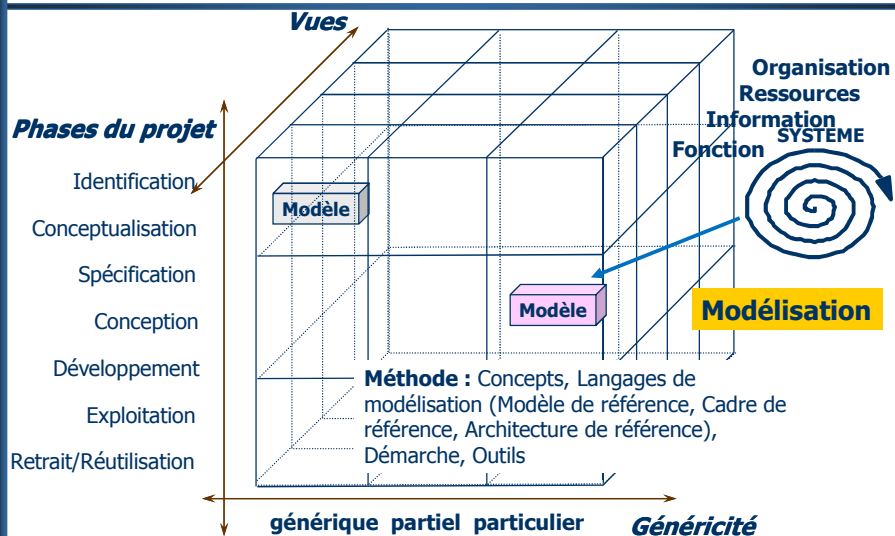
Valider : Le modèle est pertinent vis-à-vis des besoins de l'utilisateur et rend compte du phénomène réel

Qualifier : le modèle peut servir de base à la communication sans ambiguïtés ni interprétation parasite possible entre les activités du projet et/ou entre les acteurs d'un groupe de travail

Certifier : Le modèle est reconnu par une autorité compétente comme respectant un cadre normatif et pour servir de base à l'établissement d'un référentiel réutilisable et générique à un domaine

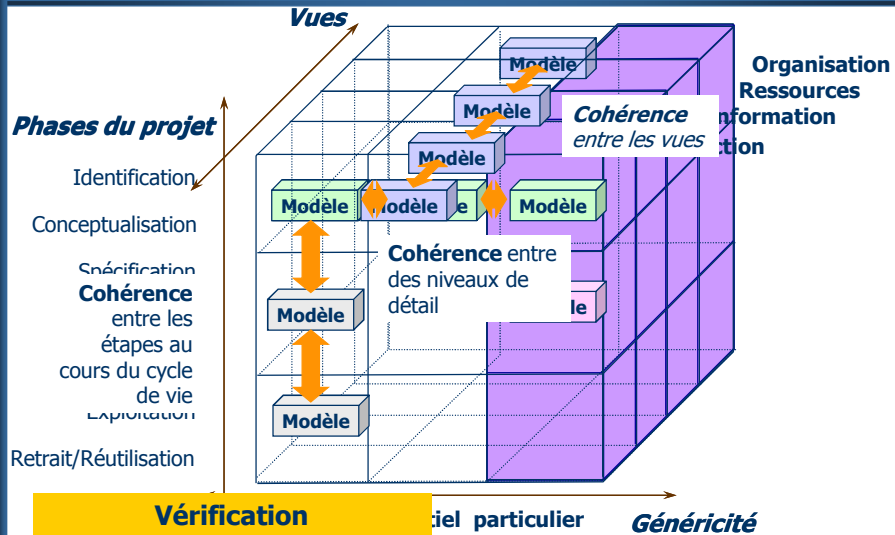
11

Positionnons-nous dans un cadre de référence



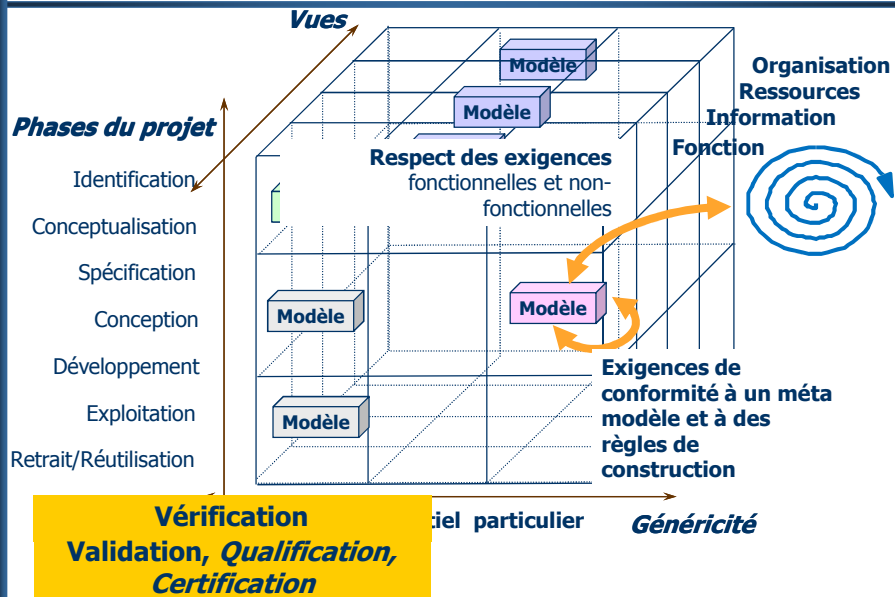
12

Positionnons-nous dans un cadre de référence



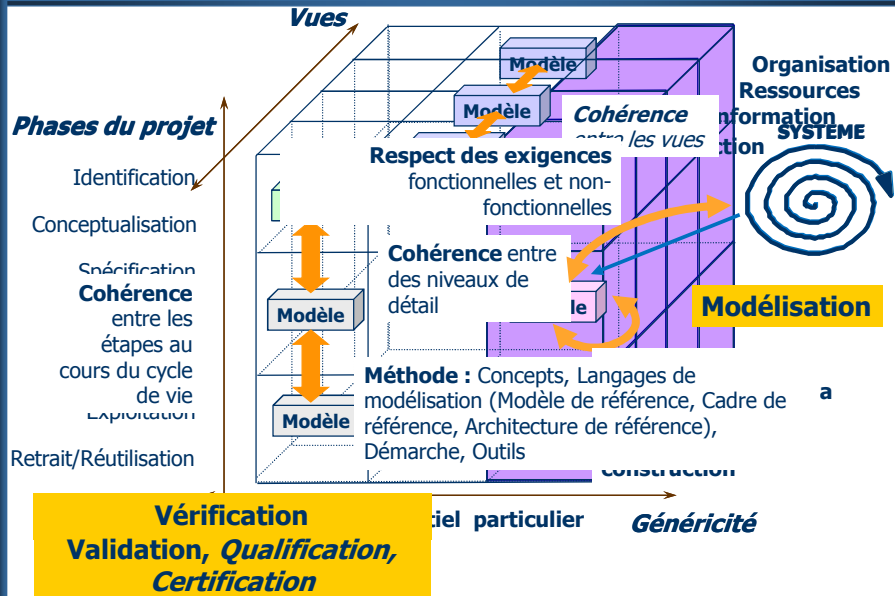
13

Positionnons-nous dans un cadre de référence

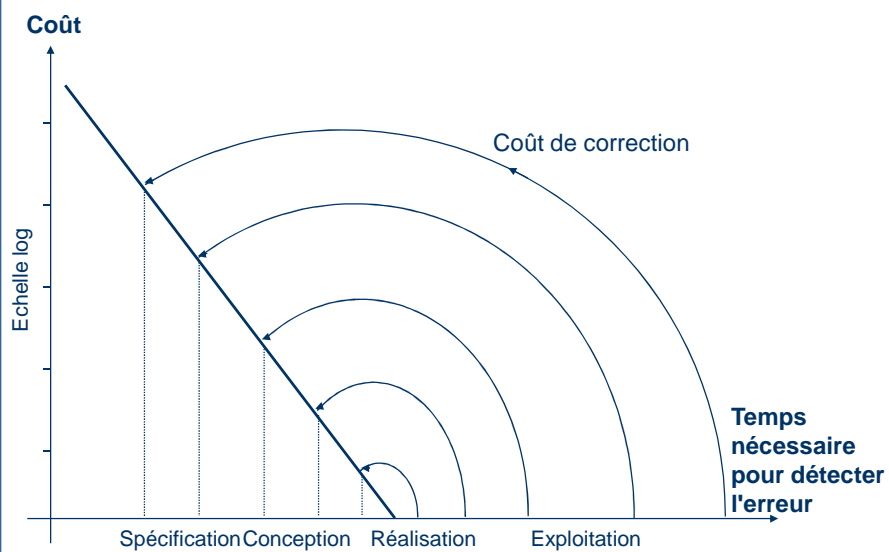


14

Positionnons-nous dans un cadre de référence



Pourquoi ? les erreurs détectées trop tard...



Pourquoi ?



V.Chapurlat
LGI2P- Laboratoire de Génie Informatique et
d'Ingénierie de Production - France

17

Pourquoi ? Intérêt... (1)

☞ Aide à la décision

- Une décision se justifie usuellement à partir de l'évaluation ou de l'interprétation de modèles du problème, du système cible, etc.
- Aide pour diagnostiquer certains faits (règles de construction formalisées, ...)

☞ Interopérabilité

- On peut faciliter ou soutenir les activités d'ingénierie nécessitant de régler des problèmes d'interopérabilité (structurelle, opérationnelle ou organisationnelle)

☞ Certification [EA 2003, ...]

- La certification passe par des représentations...

18

Pourquoi ? Intérêt... (2)

☞ **Management de la connaissance**

- La V&V nécessite souvent plus d'information que celle contenue dans le seul modèle pour être sûr d'atteindre l'objectif de qualité et d'exhaustivité nécessaire...

☞ **Système d'information** [MSVVEIS 2006, 2008, ...]

- Méthodologie
 - Analyse de modèles de processus, proposition de standards, ...
 - De fait : assister le passage d'un 'data driven environment' à un 'cooperative information and knowledge-driven environment'
- Technologie
 - Approches centrées composants
 - Approches centrées Services et SOA
 - Workflows, EAI, ERP, ...

19

Pourquoi ? Intérêt... (3)

☞ **Gestion du risque** [NASA 2001, ERM CAS 2003, ...]

- Mieux formaliser et aider dans la détection ou l'émergence des situations de risques opérationnel, technique, humain, organisationnel, ...

☞ **Applicatif** : génération de modèles d'implémentation

- Code plus sûr, approches composants, meilleure adéquation aux besoins, temps réel ou non, ...

20

Pourquoi ? concluons... (4)

Mais rappelons que pour cela, il faut être apte à :

Vérifier

- Cohérence/Consistance des modèles de niveaux de détails différents
- Cohérence/Consistance des modèles issus de points de vue différents et/ou pour un usage prévu

Valider (si possible !) la pertinence du modèle avec le système

21

Comment ?



V.Chapurlat
LGI2P- Laboratoire de Génie Informatique et
d'Ingénierie de Production - France

22

Comment ? Dépend fortement ... (1)

☞ Du rôle de l'utilisateur

- Développeur ou vendeur d'outils (approches basées composants, configuration de modèles, interopérabilité, etc.)
- Ingénieur, chargé d'affaire, commercial (maintenance du modèle, ré utilisation de modèles anciens mais encore d'actualité si ils sont correctement paramétrés, etc.)
- Chercheur ?

☞ Du projet de l'utilisateur

- Durée possible, allouable à la V&V
- Outils de (d'aide à la) V&V
- Degré de criticité du projet, niveau d'implication de l'utilisateur...

23

Comment ? Dépend fortement ... (2)

☞ Des objectifs de l'utilisateur

- Veut-il seulement avoir confiance ?
 - C'est le 'bon' modèle
 - Il respecte les règles de description
 - Il correspond à son besoin
 - Il n'a pas fait d'erreur de modélisation, ...
- Veut-il pouvoir réutiliser ou maintenir des modèles ? (raffinement)
- Veut-il étudier une situation, un phénomène, la dépendance d'un système ?...

☞ Du niveau de compétence et des connaissances de l'utilisateur

24

Category	V&V Techniques		
Informal	Audit	Desk Checking	
	Face Validation	Inspections	
	Reviews	Turing Test	
	Walkthroughs		
Static	Cause-Effect Graphing	Control Analysis (calling structure; concurrent process; control flow; state transition)	
	Data Analysis (data dependency; data flow)	Fault/Failure Analysis	
	Interface Analysis (model interface; user interface)	Semantic Analysis	
	Structural Analysis	Symbolic Evaluation	
	Syntax Analysis	Traceability Assessment	
Dynamic	Acceptance Testing	Alpha Testing	
	Assertion Checking	Beta Testing	
	Bottom-Up Testing	Comparison Testing	
	Compliance Testing (authorization; performance; security; standards)	Debugging	
	Execution Testing (monitoring; profiling; tracing)	Fault/Failure Insertion Testing	
	Field Testing	Functional (Black-Box) Testing	
	Graphical Comparisons	Interface Testing (data; model; user)	
	Object-Flow Testing	Partition Testing	
	Predictive Validation	Product Testing	
	Regression Testing	Sensitivity Analysis	
	Statistical Techniques	Special Input Testing (boundary value; equivalence partitioning; extreme input; invalid input; real-time input; self-driven input; stress; trace-driven input)	
	Structural (White-Box) Testing (branch; condition; data flow; loop; path; statement)	Sub-model/Module Testing	
	Symbolic Debugging	Top-Down Testing	
		Visualization/Animation	
	Formal	Induction	Inference
		Logical Deduction	Inductive Assertions
Lambda Calculus		Predicate Calculus	
Predicate Transformations		Proof of Correctness	

27

Love G., Beck G. (2000) Model Verification and Validation for Rapidly Developed Simulation Models: Balancing Cost and Theory, see white paper of the Project Performance Corporation (<http://www.ppc.com>)

Comment ? Informel

- ☞ **Audit**
- ☞ **Examen détaillé (revue)**
- ☞ **Inspections**
- ☞ **Tests**
- ☞ **Validation dites « d'aspect »**

Nécessite une expertise par un collectif en supposant que les compétences présentes dans ce collectif suffisent...

28

Comment ? Statique

- ↙ **Analyse de données** (dépendance, flux)
- ↙ **Analyse de risques** (Diagramme causes-effets, AMDE, AFD, Diagramme nœud papillon, etc.)
- ↙ **Analyse des interfaces** (système/système, système/utilisateur, utilisateur/système/utilisateur, usages)
- ↙ **Analyse syntaxique / sémantique**
- ↙ **Évaluation symbolique**
- ↳ Pour une analyse structurelle / comportementale / fonctionnelle de l'entreprise au travers de « ses » modèles

**Automatisable en partie,
l'expertise restera de mise dans tous les cas...**

29

Comment ? Dynamique

- ↙ **Test** : acceptation, compliance, taux de service, sécurité, sûreté, performance
- ↙ **Simulation, émulation de modèle, animation de modèle, génération de traces d'exécution, ...**
- ↙ **Débogage assisté**

**Idem : automatisable en partie,
l'expertise restera de mise dans tous les cas...**

30

Comment ? Formel

- ↵ **Mettre en œuvre un système d'inférence** (règles de raisonnement dont on peut prouver la construction et tracer l'enchaînement) **pour déduire des conclusions à partir d'hypothèses formulées dans le modèle à vérifier**
- ↵ **Exécuter** (symboliquement ou non) **le modèle à vérifier pour pouvoir parcourir exhaustivement tous les comportements possibles de ce modèle**

Une modélisation selon une approche formelle est donc nécessaire...

31

Comment ? synthétisons...

- ↵ **3 stratégies sont possibles**
- ↵ **Obtenir un modèle 'bon' et 'bien fait'**
 - Modélisation guidée
- ↵ **Exécution du modèle**
 - Simulation / Emulation
- ↵ **Etre sûr (et faire un effort supplémentaire) pour obtenir un modèle 'bon' et 'bien fait'**
 - Spécification et Preuve formelle

32

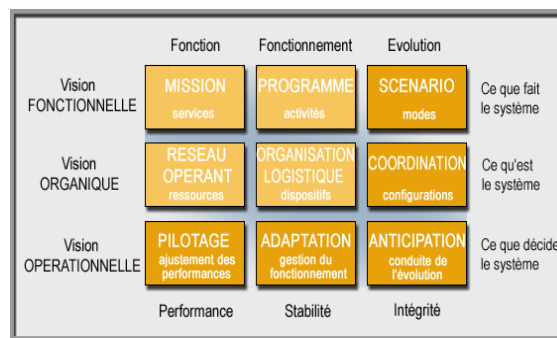
Stratégie employée (1/3) : modélisation « guidée »

- ☞ Modélisation à partir de modèles de référence (par exemple des modèles d'architecture, des constructs, des ontologies, des modèles de maturité, des normes et des standards...)
- ☞ Garantie au moins que certaines erreurs ou certains à priori sont absents
- ☞ Reste relativement trop informel pour atteindre un objectif de V&V

33

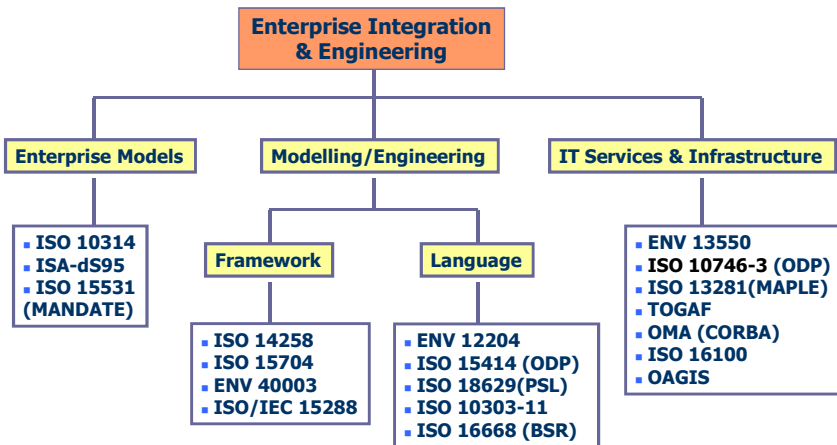
Modélisation guidée : cadres de modélisation

- ☞ Cadre de modélisation vu comme un guide de réflexion pour la modélisation
- ☞ Un exemple : le cadre de modélisation SAGACE [Benzaida 2007]



34

Modélisation guidée : contexte de normalisation



Et d'autres documents de référence :
standard de fait plutôt que normes pures !!!

© D.Chen – LAPS GRAI Bordeaux

35

Modélisation guidée : cadres de référence

☞ Exemples à discuter...

- ISO 14258 : Concepts & rules for enterprise models
- ISO 15704 : Requirements for Enterprise Reference Architecture and Methodologies
- ENV 40003 : Enterprise Integration - Framework for enterprise modelling (2002)
- ISO 15288 : Life-cycle management system

☞ Mais aussi venant d'autres domaines...

- ISO 15288
- ISO 1220-2005
- EIA 632, ...

36

Modélisation guidée : le 'standard' Zachman

	QUOI	COMMENT	OÙ	QUI	QUAND	POURQUOI
CONTEXTUEL	Liste des choses importantes	Liste des processus	Liste des emplacements	Liste des organisations	Liste des événements	Vision des opérations, buts, stratégies
CONCEPTUEL	Modèle d'information	Modèle de processus	Réseau logistique	Modèle de flux de travail	Calendrier principal	Plan opérationnel, modèle de rendement
LOGIQUE	Modèle logique des données	Architecture d'application	Architecture de distribution	Architecture d'interface humaine	Structure de traitement	Modèle de règles administratives
PHYSIQUE	Modèle physique des données	Conception de systèmes	Architecture de systèmes	Architecture de présentation	Structure de contrôle	Conception des règles
MISE EN ŒUVRE	Définition des données	Programme	Architecture des réseaux	Architecture de la sécurité	Définition du moment propice	Spécification des règles
OPÉRATIONS	Données	Services	Réseaux	Personnes	Horaires	Règles

37

© John Zachman

Stratégie employée (2/3) : exécution du modèle : simulation / émulation

☞ Démarche très outillée, reconnue à tous points de vue, recherche et pratique sur le domaine en avance de phase

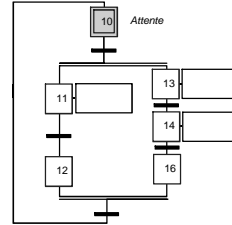
☞ Très pratiquée et reconnue en V&V mais :

- Réservé à certains objectifs de modélisation (évaluation de performances, test de scénarios)
- Nécessaire interprétation des résultats
 - subjectivité potentielle du modelleur
 - niveau d'expertise du modelleur
 - scénarios / situations non testés
- Le modèle lui-même peut s'avérer inepte à la base avant même de l'utiliser : pertinence des résultats
- Le langage de modélisation doit être muni d'une sémantique opérationnelle précisant les règles d'évolution
 - déterminisme comportemental
 - hypothèses temporelles

38

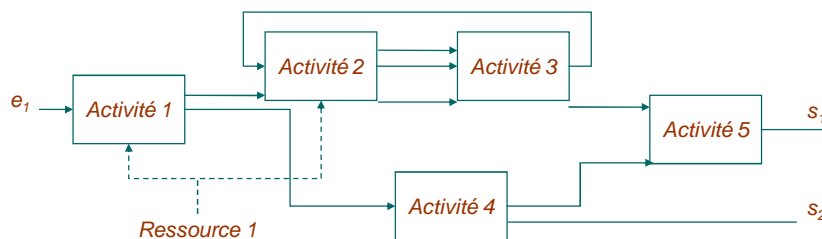
Sémantique opérationnelle : un exemple...

- ☞ **Éléments du langage** : Étape / Transition, Lien, Action, Réceptivité
- ☞ **Graphe bi-partite** : alternance
- ☞ **Structure** : parallélisme, choix, reprise, etc.
- ☞ **Point de vue** : contrôle/commande
- ☞ **Règle d'évolution**
 - Principe d'activation d'une étape
 - 5 règles d'évolution classiques + 1 règle de forçage + 1 règle d'activation des macros-étapes
- ☞ **Permet de vérifier** : respect des conditions de réinitialisation, détection de cas de parallélisme interprété, erreur de sorties simultanées, etc.



39

Sémantique opérationnelle : autre exemple...



- ☞ Comment interpréter ce modèle sans ambiguïtés ?
- ☞ Comment chaque élément du langage se comporte-t-il ?
- ☞ Quelles sont les hypothèses temporelles (synchrone, asynchrone, de type événement discret, etc.) ?

40

Stratégie employée (3/3) : preuve formelle

- ⌘ **Attention** : *une méthode est formelle si 'ses règles de manipulations de symboles sont basées sur la forme (de ces symboles) et non sur leur sens (leur sémantique)'*
- ⌘ **Raisonnement rigoureux menant à la démonstration de « propriétés »**

Caractéristique, exigence devant être satisfaite par le système ou un modèle de ce système

N'ayez crainte, nous allons y revenir...

41

Stratégie employée (3/3) : preuve formelle

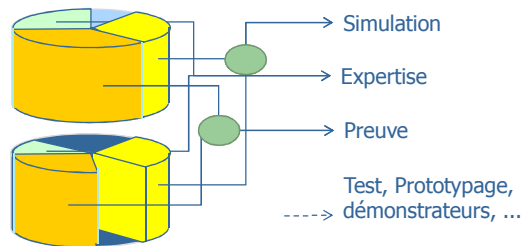
- ⌘ **Difficile à appliquer à des langages de modélisation trop peu formels ou simplement formalisés**
 - Relativement plus long à mettre en œuvre
 - Réservé à des 'pratiquants'
- ⌘ **Donc moins utilisée en V&V mais excellence reconnue car :**
 - **Exhaustivité de la preuve** : pas de scénarios 'oubliés'
 - **Interprétation rapide** : la preuve s'appuie sur la forme du modèle et non sur la sémantique qui est sans ambiguïté possible
 - La **certification** de modèle devient possible
 - L'effort supplémentaire de modélisation peut s'avérer très bénéfique dans la **connaissance / compréhension** du système modélisé

42

Comment ? oui, mais...

☞ Modélisation d'Entreprise ► Ingénierie d' (en) Entreprise

- Aide à la décision, au pilotage, ...
- Evaluation de performance
- Communication, partage d'avis, argumentation, négociation
- Implémentation, ...



43

Comment ? oui, mais...

☞ Modélisation d'Entreprise ► Ingénierie d' (en) Entreprise

- Aide à la décision, au pilotage, ...
- Evaluation de performance
- Communication, partage d'avis, argumentation, négociation
- Implémentation, ...

☞ Hypothèse générale

- Contexte MDE (Model Driven Engineering), IDM (Ingénierie Dirigée par les Modèles) ou encore MBSE (Model based System Engineering) : tout est modèle...
- Besoins non formalisés / notions de V&V absente en ME (sauf implémentation...)
- Quelles techniques utiliser alors ?
 - Modélisation guidée : OK
 - Simulation : OK
 - Preuve : pourquoi pas ?

44

Stratégie choisie : la preuve formelle

Spécification d'exigences sous
forme de propriétés et preuve
formelle



V.Chapurlat
LGI2P- Laboratoire de Génie Informatique et
d'Ingénierie de Production - France

45

Il faut ...

Langage de modélisation du système
Langage de modélisation de propriétés
Système de preuve

- ☞ **Langage de modélisation formel du système (ou formalisable)**
 - La sémantique est sans ambiguïté : il suffit alors de manipuler des équations à la syntaxe donnée
- ☞ **Langage de modélisation formel des propriétés**
 - Langage relevant d'une logique formelle si possible isomorphe ou identique au langage de modélisation du système
- ☞ **Système de preuve formelle des propriétés sur le modèle du système**
 - À la main : pour assister un opérateur
 - Semi-automatique
 - Automatique

46

Langage de modélisation du système

→ Langage de modélisation du système
Langage de modélisation de propriétés
Système de preuve

Intuitivement

- Un **vocabulaire** : des signes, un formalisme
- Un ensemble de règles qui permettent d'écrire des **formules** : une **grammaire**
- Un ensemble d'**axiomes**, c'est-à-dire de formules reconnues comme vraies (acceptées comme indiscutables)
- Un ensemble de **règles d'inférence** pour transformer une formule en une autre et obtenir ainsi des théorèmes à nouveau applicables...

Un système formel est :

- **constant** : on ne peut pas démontrer une formule et son contraire (principe dit 'du tiers exclu')
- **complet** : pour toute formule du système formel, il existe un processus de transformation qui permet de prouver qu'elle est vraie ou fausse

47

Propriété : c'est quoi ?

• Langage de modélisation du système
→ Langage de modélisation de propriétés
• Système de preuve

Un et un seul des signaux envoyés par les capteurs de position de la cabine dans la cage doit être actif à un instant donné [PARISSIS et al. 96]

Le modèle est (doit être) réinitialisé (réinitialisable) et vivant [PROTH et al, 95]

La quantité d'eau recyclée par la pompe doit rester comprise entre 0,4m³/s et 0,5m³/s [TRAVE-MASSUYES et al, 97]

Le processus d'assemblage doit pouvoir démarrer dès que l'on atteint le seuil de stockage en entrée Smin [projet Schneider 1999]

Un principe possible i.e. une propriété générique

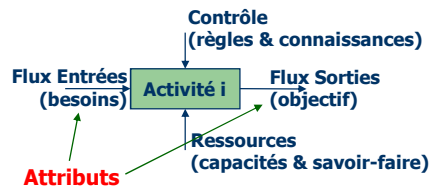
- Toute activité d'un processus transforme un ou plusieurs attributs de chacune de ses entrées dans le but de fournir les sorties désirées correspondant à sa mission dans une finalité donnée
- **Attention** : le type de transformation (Temps, Espace et/ou Forme) implique certaines relations entre les entrées et les sorties (énergie/énergie, matériel/énergie, énergie/information, etc.) [Féliot, Bond Graphs, ...]

48

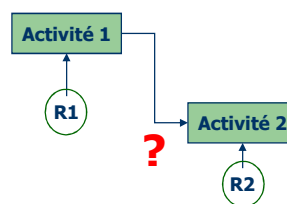
Propriété : exemples

- Langage de modélisation du système
- ➔ Langage de modélisation de propriétés
- Système de preuve

- ⊗ Le type de transformation (**temps, espace, forme**) d'une activité implique certaines **relations** entre ses **entrées** et ses **sorties**



- ⊗ Les ressources de deux activités séquentielles ont des caractéristiques spécifiques tel que le **placement géographique**



49

Propriété

- Langage de modélisation du système
- ➔ Langage de modélisation de propriétés
- Système de preuve

- ⊗ **Sûreté** : 'Quelque chose de mauvais ne doit pas se produire'
- ⊗ **Vivacité** : 'Quelque chose de bon doit fatalement arriver'
- ⊗ **Sécurité** : 'Quelque chose doit toujours arriver' ou 'Son contraire ne doit jamais arriver'
- ⊗ **Atteignabilité** : 'Une situation sera toujours atteinte tôt ou tard'
- ⊗ **Statiques / Dynamiques** : Avec ou sans prise en compte d'une échelle de temps physique ou logique
- ⊗ **Fonctionnelles** (ce que doit faire le système) / **Non fonctionnelles** (le niveau de performance qu'il doit atteindre, la sécurité qu'il doit offrir, la sûreté de fonctionnement, etc.)

50

Propriété

- Langage de modélisation du système
- ➔ Langage de modélisation de propriétés
- Système de preuve

☞ Exprimée au moyen d'un langage formel

- Logique propositionnelle, logique des prédicats
- Logique temporelle (linéaire, arborescente, ...) : LTL, PLTL, CTL, TCTL, RTL, ...)
- PSL : Property Specification Language [PSL 2004]
- LUSP : Langage Unifié de Spécification de Propriété [Lamine et al. 2001], ...
- Pi-Calcul, Mu-Calcul, Lambda-Calcul, Algèbres (dédiées, tropicales, ...), ...

☞ Exemple: « le processus d'assemblage doit pouvoir démarrer dès que l'on atteint le seuil S_{min} »

P1 : AG ((seuil < S_{min}) \Rightarrow
next(state('ProcessusAssemblage') = P));

51

Raisonner

- Langage de modélisation du système
- Langage de modélisation de propriétés
- ➔ Système de preuve

☞ Raisonner = obtenir des énoncés (propositions) à partir d'autres énoncés

☞ L'obtention d'une nouvelle proposition par des prémisses est appelée déduction

- Pas d'autre(s) information(s) que celle(s) contenue(s) dans la prémisse plus les règles, les axiomes et les théorèmes du système formel ne sont nécessaires

☞ La preuve d'une proposition doit être exhaustive et ne contenir aucune ambiguïté en s'appuyant sur :

- Les axiomes, les théorèmes et les règles d'inférence du système formel (principe de ré écriture)
- Des tables de vérité
- Logique décidable, complète et consistante

52

Raisonner... [Peirce]

- Langage de modélisation du système
- Langage de modélisation de propriétés
- ➔ Système de preuve

☞ Dédution

- Supposons un paquet plein de haricots blancs. La Loi "tous les haricots de ce paquet sont blancs » est valide. Si on prend à l'aveuglette une poignée de haricots dans le paquet (...) alors on pourra écrire avec certitude "les haricots dans cette main sont blancs".
- La déduction d'une Loi à partir d'un cas prédit avec une absolue certitude

53

Raisonner... [Peirce]

- Langage de modélisation du système
- Langage de modélisation de propriétés
- ➔ Système de preuve

☞ Induction

- On ne sait pas de quelle couleur sont les haricots dans la paquet. Si on en retire une poignée de haricots et qu'ils sont tous blancs (on peut recommencer l'opération un nombre x de fois pour s'en assurer) alors on peut écrire que tous les essais ayant donnés une poignée de haricots blancs, il est raisonnable de penser que tous les haricots de ce paquet sont blancs.
 - L'induction consiste à essayer de trouver des propriétés générales en partant de cas particuliers.
 - Elle peut alors être contredite par des contre-exemples.
hypothèse contraire
- Tous les chats que j'ai rencontré étaient gris donc tous les chats sont gris**

54

Raisonner... [Peirce]

- Langage de modélisation du système
- Langage de modélisation de propriétés
- Système de preuve

↳ Abduction

- Si on en retire à nouveau une poignée de haricots et effectivement, certains ne sont pas blancs (c'est le contre exemple attendu) alors on se trouve dans une situation inexplicable car contredisant les lois établies. on peut écrire que tous les essais ayant donnés une poignée de haricots blancs, il est raisonnable de penser que tous les haricots de ce paquet sont blancs.
- L'abduction consiste à essayer de formuler une hypothèse permettant d'exprimer la loi générale attendue toujours en partant de cas particuliers.
- On fait une hypothèse qui restera plausible jusqu'à sa contradiction : Si tous les haricots du paquet sont blancs, il est naturel que ceux qui ont été enlevés soient eux aussi blancs et cela suffit comme loi...

55

Technique de preuve

- Langage de modélisation du système
- Langage de modélisation de propriétés
- Système de preuve

↳ Techniques : Theorem Proving / Model Checking

↳ Deux approches théoriques essentielles pouvant s'appliquer

- $S \supset p$
 - S vérifie p : preuve de propriétés
- $S1 \subseteq S2$
 - $S1 \cong S2$: équivalence comportementale entre deux modèles
 - $S1 > S2$: raffinement d'un modèle par un autre

56

Outil de preuve : Theorem Prover

- ☞ **Prouver des théorèmes** qui modélisent des propriétés en utilisant les axiomes de base et des techniques de déduction propres à la logique
- ☞ **Fournit plutôt une assistance de vérification**
- ☞ **Non limité par l'explosion combinatoire**
- ☞ **Souvent limité à la preuve de propriétés essentiellement statiques**
- ☞ **Lesquels ?**
 - Z-Eves, STEP, MEC, COQ, HOL-Z, Object-Z, VDMTool, etc.

57

Outil de preuve : Model Checker

- ☞ **Littéralement ' contrôle du modèle '**
- ☞ **S'applique sur des automates d'états finis** : modèles de comportement
- ☞ **Principe** : explorer tous les états possibles c'est à dire exécuter l'automate (éventuellement de manière symbolique) jusqu'à obtention d'un contre exemple
 - preuve ' automatisée '
 - trace possible, contre-exemple
- ☞ **Permet de vérifier des propriétés temporelles**
- ☞ **L'apparition de logiques dédiées, de techniques symboliques et de mécanismes d'abstraction, de raffinement ou encore de projection permettront d'utiliser des MC sur des modèles plus complexes** (*Mu, Pi, Lambda, Algèbres, ...*)
- ☞ **Lesquels ?**
 - CADP, PVS, STEP, SMV, SPIN, Kronos, etc.

58

Quelles propriétés ?

☞ Remarques essentielles

- Elles font intervenir les moments d'évolution, voire le temps lui-même
- Elles doivent être prouvables sur des modèles comportementaux...

☞ Conclusion

- Il faut les décrire au moyen d'une logique permettant effectivement de représenter ces moments d'évolution, c'est à dire des états successifs du modèle de comportement, ici un automate...

Logique temporelle

59

Logique Temporelle Linéaire

☞ Extension de la logique propositionnelle

- Opérateurs booléens
- Introduit des opérateurs temporels (permettant de décrire l'évolution des états du modèle dans le temps):

O (suivant - next) unaire	« dans l'état suivant, ... »
□ (toujours - always) unaire ... »	« dans tous les états futurs,
◇ (inévitabilité - eventually) unaire	« dans un état futur, ... »
U (jusqu'à - until) binaire	« ... jusqu'à ... »

Si φ et ψ sont des formules de logique temporelle, alors:

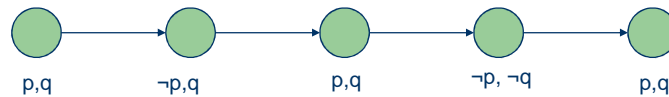
$(\varphi \wedge \psi)$ $(\varphi \vee \psi)$ $(\neg \varphi)$ $(\varphi \cup \psi)$
 $(O \varphi)$ $(\square \varphi)$ $(\diamond \varphi)$

sont des formules de logique temporelle.

60

Exemple

- ☞ Chaque état dans une séquence d'états modifie la valeur (vraie ou fausse) de chaque proposition ou formule



- ☞ Les opérateurs indiquent donc quels sont les états qui modifient la valeur d'une formule
- ☞ Dans le 4ème état on a:
 - ☐ $\neg p \vee q$
 - ☐ $O(p \vee q)$
- ☞ Mais pas:
 - ☐ $\Box(p \vee q)$

61

Logique Temporelle Linéaire : plus formellement...

- ☞ On considère un ensemble $E = \{e_1, \dots, e_n\}$
 - Exemple : e_i = un état du modèle en cours d'analyse
- ☞ Une propriété p des éléments de E est un sous ensemble de E
 - $e \in E$ a la propriété p quand $e \in p$
 - la propriété p =Vraie satisfait tous les éléments de E
 - la propriété p =Fausse ne satisfait aucun élément de E
- ☞ Notons $P = \{p_1, p_2, \dots, p_k\}$ un ensemble de propriétés de E

[G.Vidal-Naquet]

62

Logique Temporelle Linéaire : plus formellement...

Soit $\sigma = \sigma_0, \sigma_1, \dots, \sigma_{n-1}$, une suite finie d'éléments de E (exemple, une suite d'états du modèle)

On note:

$(\sigma, i) \models \varphi$ signifie "la suite σ satisfait la formule de logique temporelle φ au rang i "

$\sigma \models \varphi$, le fait que $(\sigma, 0) \models \varphi$

On a:

1. $(\sigma, i) \models p$ si et seulement si $i < |\sigma|$ et σ_i vérifie la propriété p
2. $(\sigma, i) \models (\varphi \vee \psi)$ si et seulement si $(\sigma, i) \models \varphi$ ou $(\sigma, i) \models \psi$
3. $(\sigma, i) \models (\varphi \wedge \psi)$ si et seulement si $(\sigma, i) \models \varphi$ et $(\sigma, i) \models \psi$

63

[G.Vidal-Naquet]

Logique Temporelle Linéaire : plus formellement...

4. $(\sigma, i) \models (\neg \varphi)$ si et seulement si $i < |\sigma|$ et $(\sigma_i) \not\models \varphi$ ((σ_i) ne satisfait pas φ)

5. $(\sigma, i) \models (O \varphi)$ si et seulement si $i < |\sigma|$ et $(\sigma, i + 1) \models \varphi$.

6. $(\sigma, i) \models (\Box \varphi)$ si et seulement si $\forall j, i \leq j < |\sigma|, (\sigma, j) \models \varphi$.

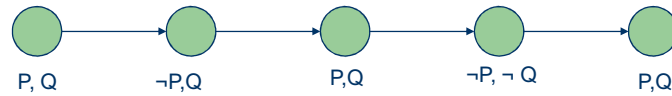
7. $(\sigma, i) \models (\Diamond \varphi)$ si et seulement si $\exists j, i \leq j < |\sigma|, (\sigma, j) \models \varphi$.

8. $(\sigma, i) \models (\varphi U \psi)$ si et seulement si $\exists j, i \leq j < |\sigma|, (\sigma, j) \models \psi, \exists k, i \leq k < j, (\sigma, k) \models \varphi$

64

[G.Vidal-Naquet]

Exemple



☞ **Qu'en est-il des formules suivantes ?**

- ⇒ $\Box(p \Rightarrow O(q))$
- ⇒ $\Box(p \Rightarrow O(\neg qUr))$ (on suppose que r existe...)
- ⇒ $\Box\Diamond p$
- ⇒ pUq

65

Logique Temporelle Linéaire : Règles de déduction

- ☞ **TAU (tautologie)** Pour toute formule d'état valide p
 $\models p$
- ☞ **GEN (généralisation)**: Pour toute formule d'état p
 $\models p$

 $\models \Box p$
- ☞ **SPEC (Spécialisation)**: Pour toute formule d'état p
 $\models \Box p$

 $\models p$
- ☞ **MP (Modus Ponens)** :
 $(p1 \wedge \dots \wedge pn) \rightarrow q, p1, \dots, pn$

q
- ☞ **PAR (Particularisation)** : Pour toute formule d'état p
 $\Box p$

p

66

Propriétés : reprenons...

- ↵ **Décrivez les propriétés suivantes** (*donnez en une forme...*)
 - **Sûreté** : 'Quelque chose de mauvais ne doit pas se produire'
 - **Vivacité** : 'Quelque chose de bon doit fatalement arriver'
 - **Sécurité** : 'Quelque chose doit toujours arriver' ou 'Son contraire ne doit jamais arriver'
 - **Atteignabilité** : 'Une situation sera toujours atteinte tôt ou tard'

67

Algorithme de model checking (informel...)

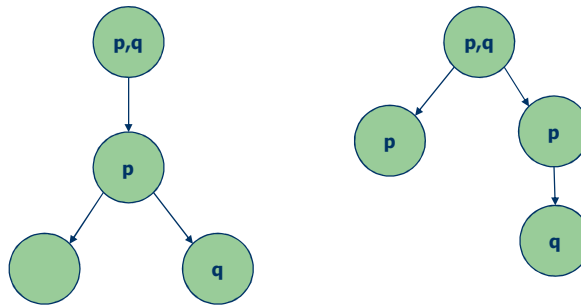
- ↵ **But** : vérifier que le modèle satisfait une formule ψ
 - Est-ce que pour toutes les exécutions σ de l'automate A, on a toujours $(A, \sigma) \models \psi$?
- ↵ **Comment ?**
 - Chercher un contre exemple qui montre qu'il existe une exécution σ de l'automate A qui ne satisfait pas ψ
- ↵ **Méthode**
 - Construire un automate B dans lequel on décrit les exécutions de A qui ne satisfont pas ψ
 - Synchroniser A et B pour obtenir un nouvel automate C
 - Si C est vide alors A satisfait ψ

68

Logique Temporelle Arborescente

☞ Certaines propriétés ne peuvent pas se décrire avec la Logique Temporelle Linéaire

☞ Exemple :



$O p \rightarrow q ?$

69

Logique Temporelle Arborescente

☞ Extension de la Logique Linéaire Temporelle par l'ajout d'opérateurs de chemin notés :

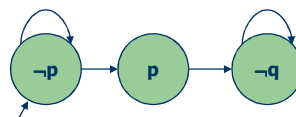
- E pour un chemin donné
- A pour tous les chemins

☞ Une formule de LTL + un opérateur de chemin devient alors une formule d'état

☞ Une formule d'état peut ensuite permettre d'écrire des formules de chemin

☞ Exemple

- $AO(EOq \wedge EO\neg q)$
- $A\Box p$
- $E\Box p$
- ... à vous...



70

Logique Temporelle Arborescente

Une formule d'état est :

$p \in P$, l'ensemble des propriétés de l'automate
 $\neg\phi$, $\phi \vee \psi$ et $\phi \wedge \psi$ où ϕ et ψ sont elles-mêmes des formules d'état
 $A\phi$ et $E\phi$ où ϕ est une formule de chemin

Sémantique

$(A, s) \models p$	ssi	$p \in P \vee$
$(A, s) \models \neg\phi$	ssi	$(A, s) \not\models \phi$
$(A, s) \models \phi \wedge \psi$	ssi	$(A, s) \models \phi$ et $(A, s) \models \psi$
$(A, s) \models \phi \vee \psi$	ssi	$(A, s) \models \phi$ ou $(A, s) \models \psi$
$(A, s) \models E\phi$ si	ssi	$\exists \sigma$ à partir de s / $(A, \sigma) \models \phi$
$(A, s) \models A\phi$ si	ssi	$\forall \sigma$ à partir de s / on a $(A, \sigma) \models \phi$

Attention : σ est un chemin

71

Logique Temporelle Arborescente

Une formule de chemin est :

une formule d'état
 $\neg\phi$, $\phi \vee \psi$, $\phi \wedge \psi$, $O\phi$ et $\phi U \psi$ où ϕ et ψ sont des formules de chemin

Sémantique

$(A, \sigma) \models \phi$	ssi	$s = \text{premier état de } \sigma \text{ et } (A, s) \models \phi$
$(A, \sigma) \models \neg\phi$	ssi	$(A, \sigma) \not\models \phi$
$(A, \sigma) \models \phi \wedge \psi$	ssi	$(A, \sigma) \models \phi$ et $(A, \sigma) \models \psi$
$(A, \sigma) \models \phi \vee \psi$	ssi	$(A, \sigma) \models \phi$ ou $(A, \sigma) \models \psi$
$(A, \sigma) \models O\phi$ si	ssi	$(A, \sigma_1) \models \phi$
$(A, \sigma) \models \phi U \psi$ si	ssi	$\exists i. i \geq 0$ / $(A, \sigma_i) \models \psi$ et $\forall j. 0 \leq j < i$ / $(A, \sigma_j) \models \phi$

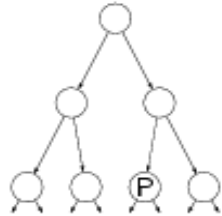
Par convention, on écrira :

X au lieu de O, F au lieu de \diamond , A au lieu de \square

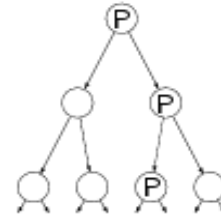
72

Logique Temporelle Arborescente : Exemples

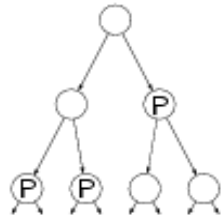
$E\Diamond P$:



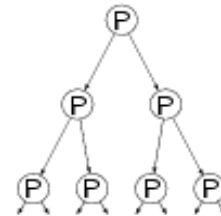
$E\Box P = E\neg\Diamond\neg P$:



$A\Diamond P = \neg E\neg\Diamond P$:



$A\Box P = \neg E\Diamond\neg P$:



73

Exercice

☞ Traduisez les formules suivantes en 'bon' Français sachant que :

- début, prêt, requête, accuséRéception, disponible et recommence sont des formules

☞ A vous...

- $EF(\text{début} \wedge \neg \text{prêt})$
- $AG(\text{requête} \rightarrow AF(\text{accuséRéception}))$
- $AG(AF(\text{disponible}))$
- $AG(EF(\text{recommence}))$

74

Un model checker : SMV



V.Chapurlat
LGI2P- Laboratoire de Génie Informatique et
d'Ingénierie de Production - France

75

SMV (Symbolic Model Verification)



- ↳ **Plusieurs versions (Cadence SMV)**
- ↳ **Utilise les BDD pour représenter le graphe d'états du modèle et les états qui satisfont certaines propriétés**
- ↳ **Principes de base**
 - Un modèle = un module SMV acceptant des paramètres en entrée
 - Un langage textuel unique permet de décrire le comportement d'un système et les propriétés à prouver
 - Décomposition comportementale possible

76

Exemple introductif

Systeme

Un circuit combinatoire de comparaison de signaux (arbitre deux bits)

Comportement

Deux signaux booléens ack1 et ack2 sont respectivement égaux à req1 et à req2. non req1 ou req1 et req2 sont des signaux d'entrée booléen du système

Propriétés

- On ne doit jamais avoir en même temps $ack1 = ack2 = 1$
- Toute modification de ack1 et de ack2 doit correspondre à une modification de req1 et/ou de req2
- Il faut toujours de chaque modification du signal req1 induise une modification du signal ack1
- Il faut toujours de chaque modification du signal req2 induise une modification du signal ack2

Objectif

Modéliser ce comportement sous forme d'automate et prouver ces propriétés

77

Exemple introductif

```
module main(req1,req2,ack1,ack2) Paramètres E/S
{
  input req1,req2 : boolean;
  output ack1,ack2 : boolean; Déclarations

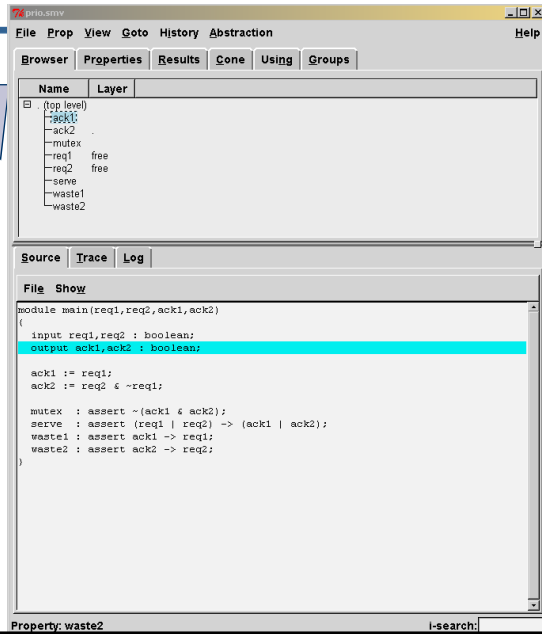
  ack1 := req1;
  ack2 := req2 & ~req1; Description du comportement

  mutex : assert ~(ack1 & ack2);
  serve : assert (req1 | req2) -> (ack1 | ack2); Propriétés à prouver
  waste1 : assert ack1 -> req1;
  waste2 : assert ack2 -> req2;
} Fin du module
```

78

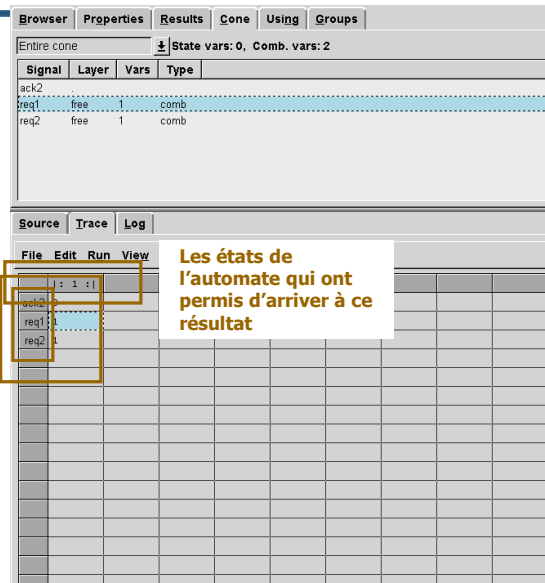
SMV : l'interface

- 1) Créer/modifier le texte source décrivant l'automate et les propriétés avec un éditeur externe (Notepad, etc.)
- 2) Sauvegarder avec l'extension '.smv'
- 3) Charger ce fichier avec smv (menu File...)



79

Nous obtenons alors...



Un contre exemple qui permet de déduire que cette dernière propriété est fausse...

Ces variables représentent le cône ('cone' en anglais) de la propriété : celles qui sont nécessaires à son analyse

Les états de l'automate qui ont permis d'arriver à ce résultat

80

Un model checker : UPPAAL



V.Chapurlat
LGI2P- Laboratoire de Génie Informatique et
d'Ingénierie de Production - France

81

UPPAAL



↳ **Uppsala University + Aalborg University = UPPAAL**

↳ <http://www.uppaal.com/>

↳ **Première version en 1995**

↳ **Principes**

- Langage de modélisation du comportement du système : automates temporisés et synchronisés ('networks of timed automata')
- Langage de modélisation de propriétés : Fragments de TCTL (Timed Computational Tree Logic) ('query language')
- Système de preuve

82

Automate temporisé (rappel)

- ↵ A timed automaton (TA) is a tuple (L, l_0, C, A, E, I) , where:
- L is a set of locations
 - $l_0 \in L$ is the initial location
 - C is the set of clocks
 - A is a set of actions, co-actions and the internal T-action
 - $E \subseteq L \times A \times B(C) \times 2C \times L$ is a set of edges between locations with an action, a guard and a set of clocks to be reset
 - $I : L \rightarrow B(C)$ assigns invariants to locations.

83

Sémantique opérationnelle d'un TA

Definition 2 (Semantics of TA). Let (L, l_0, C, A, E, I) be a timed automaton. The semantics is defined as a labelled transition system $\langle S, s_0, \rightarrow \rangle$, where $S \subseteq L \times \mathbb{R}^C$ is the set of states, $s_0 = (l_0, u_0)$ is the initial state, and $\rightarrow \subseteq S \times \{\mathbb{R}_{\geq 0} \cup A\} \times S$ is the transition relation such that:

- $(l, u) \xrightarrow{d} (l, u + d)$ if $\forall d' : 0 \leq d' \leq d \implies u + d' \in I(l)$, and
- $(l, u) \xrightarrow{a} (l', u')$ if there exists $e = (l, a, g, r, l') \in E$ s.t. $u \in g$, $u' = [r \mapsto 0]u$, and $u' \in I(l')$,

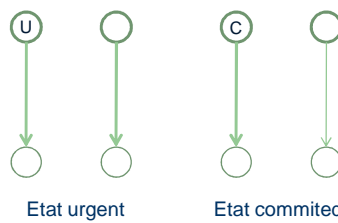
where for $d \in \mathbb{R}_{\geq 0}$, $u + d$ maps each clock x in C to the value $u(x) + d$, and $[r \mapsto 0]u$ denotes the clock valuation which maps each clock in r to 0 and agrees with u over $C \setminus r$. \square

84

Automate temporisé : synthèse

Etat

- Nom
- Invariant (horloge)
- Peut être initial, urgent ou committed

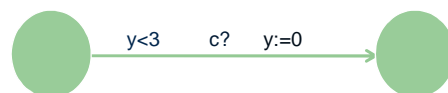


85

Automate temporisé : synthèse

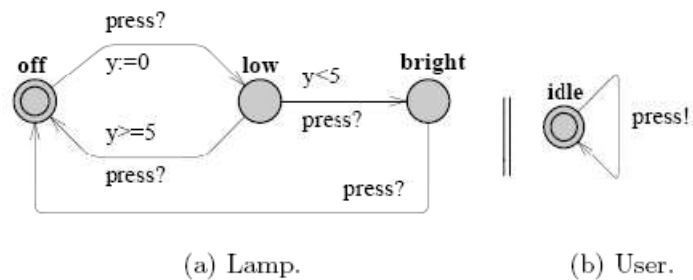
Transitions

- Garde (horloge)
- Synchronisation des canaux
 - $c!$: émetteur
 - $c?$: récepteur
- Remise à zéro (update)



86

Sémantique opérationnelle : exemple



As an example of the semantics, the lamp in Fig. 1 may have the following states (we skip the user): $(\text{Lamp.off}, y = 0) \rightarrow (\text{Lamp.off}, y = 3) \rightarrow (\text{Lamp.low}, y = 0) \rightarrow (\text{Lamp.low}, y = 0.5) \rightarrow (\text{Lamp.bright}, y = 0.5) \rightarrow (\text{Lamp.bright}, y = 1000) \dots$

87

Méthode

☞ **Un process est une instance d'un Automate temporisé**

☞ **Il faut donc :**

- Déclarer des variables globales (Interface Editor - Déclarations)
- Décrire le comportement du système = réseau d'automates temporisés (Interface Editor – Templates)
- Instancier les templates pour créer les processus (Interface Editor – System déclarations)
- Tester le modèle de comportement et/ou simplement visualiser ce comportement (Interface Simulator)
- Décrire les propriétés (queries) : fragment de TCTL (Interface Verifier)

88

Décrire le comportement du système : déclarations

☞ **Globale (tous les processus) ou locale (un seul template) : mêmes principes**

☞ **BNF**

Declarations ::= (VariableDecl | TypeDecl | Function | ChanPriority)*

VariableDecl ::= Type VariableID (',' VariableID)* ';'

VariableID ::= ID ArrayDecl* ['=' Initialiser]

Initialiser ::= Expression | '{' Initialiser (',' Initialiser)* '}'

TypeDecls ::= 'typedef' Type ID ArrayDecl* (',' ID ArrayDecl)* ';'

89

Décrire le comportement du système : déclarations

☞ **Constante**

Const name value

☞ **Tableaux**

Int [min,max] avec min et max sont des entiers

☞ **Canaux de synchronisation (emetteur c! / récepteur c?)**

chan name

☞ **Diffusion (broadcast channel)**

broadcast chan name

☞ **Urgent (attention : ne pas mettre de contraintes sur une horloge)**

– urgent

90

Les déclarations: BNF générale

```
Expression → ID | NAT
            | Expression '[' Expression ']'
            | '(' Expression ')'
            | Expression '++' | '++' Expression
            | Expression '--' | '--' Expression
            | Expression AssignOp Expression
            | UnaryOp Expression
            | Expression BinaryOp Expression
            | Expression '?' Expression ':' Expression
            | Expression '.' ID

UnaryOp     → '-' | '!' | 'not'
BinaryOp   → '<' | '<=' | '==' | '!=' | '>=' | '>'
            | '+' | '-' | '*' | '/' | '%' | '&'
            | '|' | '^' | '<<' | '>>' | '&&' | '||'
            | '<?' | '>?' | 'and' | 'or' | 'imply'
AssignOp   → ':=' | '+=' | '-=' | '*=' | '/=' | '%='
            | '|=' | '&=' | '^=' | '<<=' | '>>='
```

91

Instancier les templates pour créer les processus : system declaration

The screenshot shows the UPPAAL software interface with a project named 'UPPAAL' and a file named 'demo/2doors.xml'. The main editor displays the following code:

```
bool activated1, activated2;
urgent chan pushed1, pushed2;
urgent chan closed1, closed2;

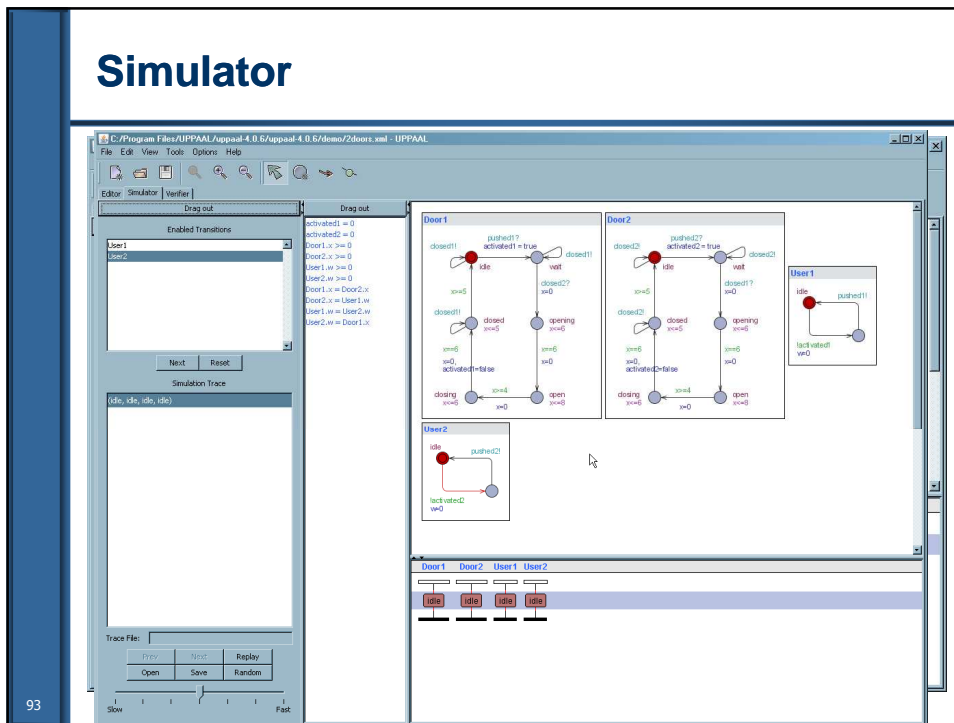
Door1 = Door(activated1, pushed1, closed1, closed2);
Door2 = Door(activated2, pushed2, closed2, closed1);
User1 = User(activated1, pushed1);
User2 = User(activated2, pushed2);

system Door1, Door2, User1, User2;
```

The interface includes a 'Project' tree on the left with 'System declarations' selected. On the right, there are three panels: 'Paramètres', 'Instances', and 'Le système à analyser'. The 'Instances' panel shows the instantiated components: Door1, Door2, User1, and User2. The 'Le système à analyser' panel shows the system declaration: system Door1, Door2, User1, User2;

92

Simulator



93

Langage de modélisation de propriétés

TCTL (fragment)

$E \langle \rangle p$: il existe un chemin le long duquel p est vrai un jour

$E [] p$: il existe un chemin le long duquel p est toujours vrai

$A \langle \rangle p$: le long de tous les chemins p est vrai un jour

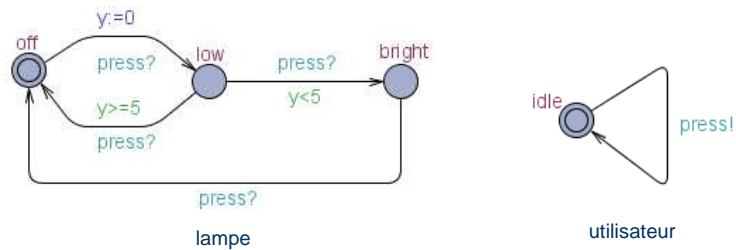
$A [] p$: le long de tous les chemins p est toujours vrai

$q \rightarrow p$: quand q est vrai alors p sera forcément vrai un jour ($A [] (q \Rightarrow A \langle \rangle p)$)

Avec $p, q = \text{Aut.état} \mid \text{clock} \sim \text{valeur} \mid p \text{ and } q \mid p \text{ or } q \mid \text{not } p \mid p \text{ imply } q \mid \text{deadlock}$

94

Exemple de prise en main



Propriétés :



$E \leftrightarrow \text{lampe.y} < 5 \text{ imply } \text{lampe.bright}$

$A[] \text{ utilisateur.idle}$

$E[] \text{ utilisateur.idle \& lampe.off}$

97

Etude comparative rapide et non exhaustive

 Uppaal	 SMV
Model-checker quantitatif à temps dense	Model-checker qualitatif
Automates temporisés	Automates non temporisés (Structures de Kripke, Langage LUSTRE)
Interface graphique + interface textuelle	Interface textuelle
Fragment de la logique TCTL	Logique CTL
Vérifie des propriétés temporisées	Vérifie des propriétés temporelles
Avantage : Horloges + simulateur	Avantage : Utilisation de ROBDD + un contre exemple

98

Conclusion



V.Chapurlat
LGI2P- Laboratoire de Génie Informatique et
d'Ingénierie de Production - France

99

Problématique : V&V (ou VVQC ou IVVQ ?)

- ☞ **La vérification et la validation représentent aujourd'hui un enjeu...**
 - la vérification : *construisons-nous correctement le modèle du système ? vs. construisons-nous correctement le système ?*
 - la validation : *construisons-nous le bon modèle du système ? vs. le système construit répond-t-il aux attentes réelles des parties prenantes ?*
- ☞ **Vérification & Validation (V&V) :** s'assurer que le système développé correspond aux attentes de l'utilisateur final ou, à défaut, le rassurer partiellement...
À lui de faire d'autres modèles venant compléter le premier (*interopérabilité des modèles ?*)

100

Problématique : V&V

☞ Vérification formelle pourrait se développer en ME pour plusieurs raisons:

- Le **temps de mise sur le marché** d'un produit ou d'un service : plus il est court, naturellement, mieux c'est, or les erreurs rallongent les délais de conception et de fabrication. *Cependant...*
- La **sûreté** : pour les systèmes critiques, le zéro-défaut est un objectif sans faille mais aussi pour les organisations par exemple impliquées dans des gestions de crises, des collaborations douteuses, etc.
- La **qualité** : la vérification formelle certifie une bonne réutilisation et une maîtrise des coûts de maintenance
- ...

101

V&V : limites des outils actuels



☞ Model checker (vérificateur de modèles)

- Principe: vérification des propriétés en parcourant exhaustivement un modèle comportemental fini du système
- Avantage: automatique + génération des contre-exemples
- Inconvénient: explosion combinatoire, modèles comportementaux

☞ Theorem prover (démonstrateurs de théorèmes)

- Principe: utilisation d'axiomes ainsi qu'un ensemble de règles d'inférences logiques pré programmées.
- Avantage: technique très rigoureuse, abstraction, réécriture, déduction.
- Inconvénient: complexe et formalisme difficilement accessible aux non spécialistes

102

Avant tout, mon avis...

↳ **Système formel incluant des mécanismes de représentation et de raisonnement** (déduction, abduction, induction) :

- Appartenant au langage de modélisation ou de la méthode de modélisation employée (Z, B, VDM, Algèbres - de processus (Millner, Canuto, ...) 'tropicales', ...-)
- Adaptés à ce langage : il s'agit de traduire le modèle obtenu dans un langage supportant des mécanismes de preuve (Graphes conceptuels vers logique du premier ordre, modèles de machine séquentielle vers langages automates, ...)

↳ **De manière plus intuitive...**

- Règles et contraintes de construction qui peuvent être vérifiés 'à la main' par le modeleur

↳ **Exemple : le projet CARIONER...**