Les techniques formelles de spécification et les outils de vérification

Ana Cavalli

Télécom SudParis

Plan de la présentation

- Les protocoles de communication
- Techniques de spécification
- Vérification
- Test
- Problèmes de recherche
- Perspectives

Technique de spécification

- Qu'est-que c'est une spécification?
  - Décrit ce qui fait le système sans expliquer comment il le fait
- Technique de spécification
  - Informelle (basée sur le langage naturel)
  - Formelle
    - utilise une notation mathématique
    - associe une signification mathématique à cette notation

Techniques formelles de spécification

(Formal Description Techniques)

- Pourquoi des techniques formelles de spécification ?
  - Pour normaliser la communication
    - - entre systèmes informatiques hétérogènes
    - - entre systèmes de commutation téléphonique
  - ISO et ITU ont normalisé le concept de:
    - ARCHITECTURE OUVERTE OSI
  - Les normes (standards) font:
    - Description de l'architecture OSI (couches, services, protocoles) en langue naturelle
    - +
    - Diagrammes des transitions par des tables d'états
    - Les spécifications sont imprécises, ambiguës et pleines d 'erreurs:
      - Nécessité de définir des formalismes pour la description de ces systèmes

ISO et ITU normalisent trois TFS: LDS, Estelle et Lotos

- Ces techniques (appelés aussi langages de spécification):
  - spécifient les propriétés fonctionnelles d'un système vis à vis de son environnement
  - sont conçues pour décrire des systèmes distribués composés par des processus que s'exécutent en parallèle, se synchronisent et communiquent par messages
- Autres techniques: algèbre des processus (CCS), machines d'états finis, logique temporelle, réseaux de Petri

Aperçu sur les TFS normalisées

- **LDS (Langage de Description et de Spécification):**
  - défini et normalisée par l'ITU, fondée sur le modèle des machines d'états finis étendue, possède deux versions: LDS-PR (textuel) et LDS-GR (graphique)
- **Estelle**
  - normalisée par l'ISO, comme LDS il est basé sur le modèle de machines d'états finis étendues
- **Lotos**
  - normalisée par l'ISO, fondé sur les algèbres des processus (CCS, CSP), comme LDS il possède deux versions: graphique et textuelle

Avantages des TFS

- Définition stable et normalisée (consensus international)
- Evolution contrôlée
- Application à des systèmes réels
- Large communauté d 'utilisateurs
- Réduction du coût de développement des logiciels:
  - le coût de correction des erreurs de logiciels croit vite dans le cycle de vie des logiciels
  - intérêt à corriger ces erreurs dès la phase de conception

Avantages des TFS (suite)

- Absence d ambiguïté
- Précision
- Cohérence
  - pas de contradiction
- Abstraction
  - indépendance des implémentations
  - technique formelle de spécification ≠ langage de programmation

Validation des spécifications

- Cohérence des spécifications
- Analyse statique
  - vérification de la syntaxe
  - références inter-processus
  - sémantique statique
- Analyse dynamique
  - simulation exhaustive

Validation des spécifications (suite)

- Analyse dynamique exhaustive
  - analyse de la machine d'états finis (accessibilité entre les états)
  - analyse des réseaux de Petri
  - vérification en utilisant la Logique Temporelle (model checking)

Validation des spécifications (suite)

De la spécification à l'implémentation:

Description informelle du système

↓

Spécification formelle

↓

Spécification détaillé (détection et traitement d'erreurs, interfaces, etc.)

↓

Génération du code (C, C++, etc.)

Validation des spécifications (suite)

- Raffinement des spécifications pour:
  - génération du code («debugging testing »)
  - « testing » de l 'implémentation (« conformance testing »)

Techniques normalisées

	Modèle de calcul	Modèle de communication	Data types
• Estelle	machines d 'état finis étendues	par file d 'attente	PASCAL
• LDS	machines d 'état finis étendues	par file d 'attente	Abstract Data Types (ACT ONE)
• Lotos	CCS (algèbre des processus)	rendez-vous	Abstract Data Types (ACT ONE)

Qui s'intéresse aux TFS?

- Les industriels
  - influence croissante des systèmes OSI parmi les industriels
    - Alcatel, Motorola, Matra-Nortel, France Télécom, Bouygues, SFR, CISCO, SIEMENS, LUCENT
- Les centres de recherche universitaires
  - intérêt croissant pour les systèmes distribués
  - définition des langages et de modèles pour ces systèmes

Les outils en France au niveau prototype et industriel

- Estelle Development Tool Set
  - INT
- ObjectGEODE
  - Verilog-Telelogic
- Tau
  - Telelogic
- SPIN
  - Bell Labs

Nature des outils

- Editeurs syntaxiques et graphiques
- Analyseurs de la sémantique statique
- Vérificateurs
  - pour trouver des propriétés de « bon fonctionnement »: non blocage, équité, etc
- Simulateurs
  - simulation du comportement du système. L'utilisateur peut déclencher des séquences d 'actions, inspecter l 'état courant du système, les valeurs de variables, le contenu des files d 'attente

Nature des outils (suite)

- Générateurs de code
  - à partir de la spécification en langage TFS obtenir un programme exécutable
- Générateur de tests
  - a partir d'une spécification en langage TFS produire automatiquement des séquences de tests qui permettent de vérifier qu'une implémentation existante est conforme à la spécification

Les protocoles de communication

- Le monde OSI
  - Modèle de référence OSI : modèle d'architecture pour l'interconnexion de systèmes hétérogènes
    - ISO *7498* et ITU-T *X.200*
  - Nombreuses normes de protocoles pour chacune des couches du modèle
    - Normes ISO et ITU
    - Exemple pour la couche transport :
      - Protocole de transport : ISO *8072* et ITU-T *X.224*
      - Service de transport : ISO *8073* et ITU-T *X.214*

Les protocoles de communication (suite)

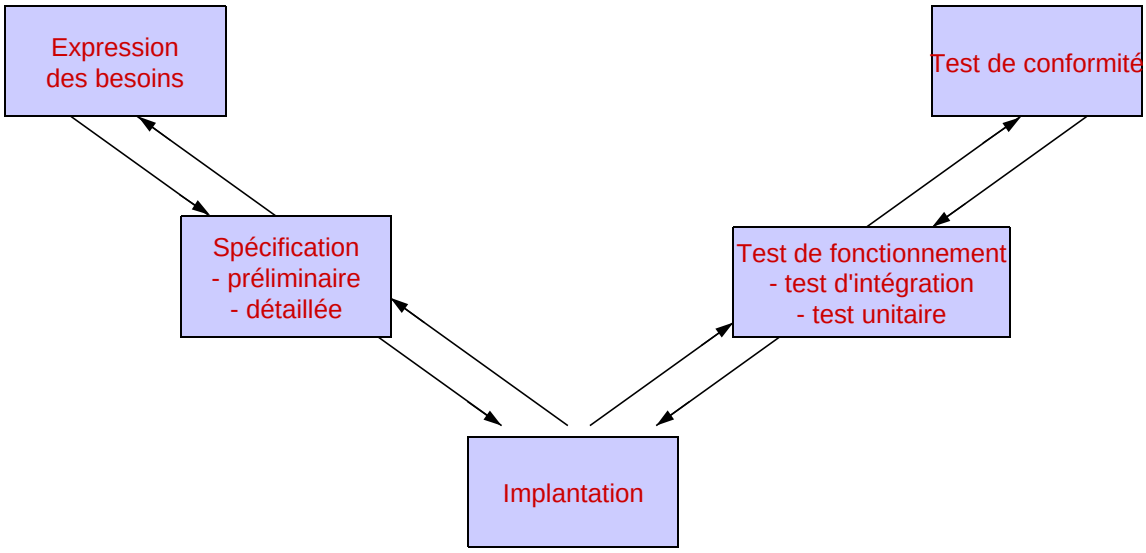
- Le monde TCP/IP
  - Un ensemble de protocoles : Internet Protocol Suite
    - RFC (Request For Comments) de l'IETF (Internet Engineering Task Force)
    - Exemple pour TCP : RFC *793*
- Les autres mondes…
  - Réseaux locaux (IEEE *802.**)
  - Systèmes d'exploitation réseau (Netware, OS/2…)
  - Architectures constructeurs (SNA, DNA, DSA…)
  - Réseaux de télécommunications (normes ITU, ETSI…)
  - etc.

Les protocoles de communication (suite)

- Qu'est-ce qu'un protocole ?
  - Un ensemble de règles et de formats permettant la communication entre des entités d'un système distribué
    - Règles de communication
    - Formats des messages échangés

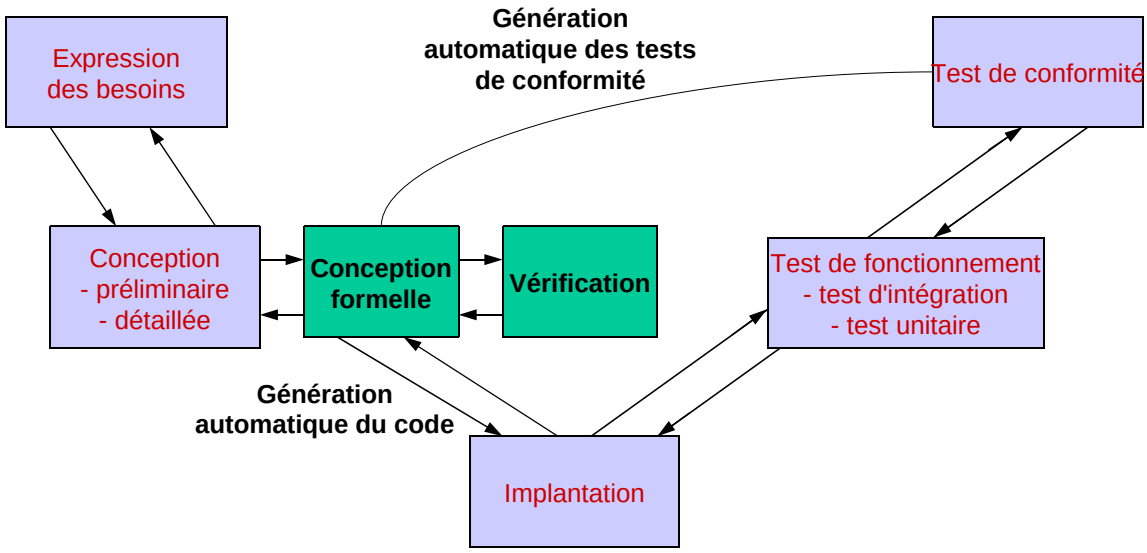
Le cycle de développement des protocoles

- Le modèle en V



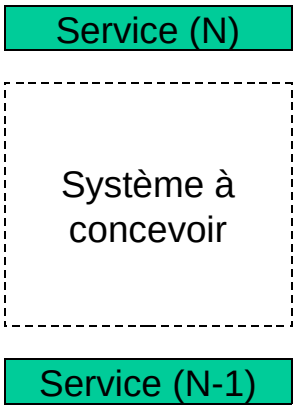
Le cycle de développement des protocoles (suite)

- Le modèle en V avec conception formelle



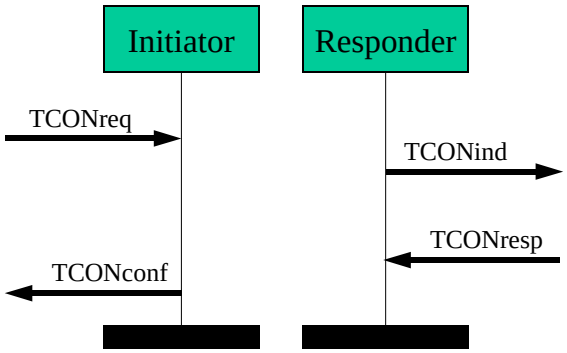
Expression des besoins

- Établissement d'un cahier des charges
  - Besoins de l'utilisateur en termes de services disponibles et de services désirés
  - Spécifier uniquement le comportement externe du logiciel
  - Spécifier les réponses
    - aux événements attendus
    - aux événements indésirables (robustesse)

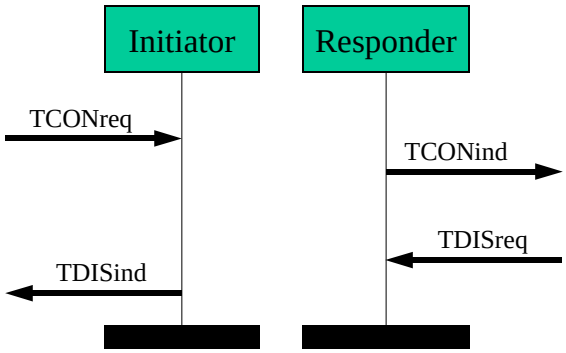


Expression des besoins (suite)

- Description du service sous forme de MSC
  - MSC (Message Sequence Chart) : norme ITU-T *Z.120*



Connexion acceptée



Connexion refusée

Vérification et test

- Qu'est-ce que la verification?
  - Vérifier que la spécification du système est correcte et ne contient pas d'erreur
- Qu'est-ce que le test?
  - Evaluer que l'implantation du système possède les propriétés attendues et ne contient pas des erreurs (aussi évaluer l'interoperabilité avec d'autres implantations ou d'autres composants)

Techniques de Vérification

- Basée sur:
  - specifications formelles
  - techniques d'analyse statique
  - model checking
- Pour vérifier:
  - propriétés globaux
  - propriétés de composants
- Pour détecter:
  - design errors
  - deadlocks, livelocks

Vérification d'un protocole

- Vérification de la spécification formelle
  - Propriétés liées au protocole
  - Propriétés liées au service
- Propriétés liées au protocole
  - Interblocage (Deadlock)
  - Bouclage (Livelock)
  - Réception non spécifiée
  - Code mort (état inaccessible, transition infranchissable…)
  - etc.
- Propriétés liées au service
  - Le service rendu par le protocole est bien le service attendu
  - Exemples :
    - Une indication ne peut être émise par la station destinataire qu'après la réception d'une requête par la station émettrice
    - Les messages sont reçus par la station destinataire dans le même ordre que l'ordre d'émission par la station émettrice
    - etc.

Vérification

Main techniques:

- Model checking
  - Sifakis, Clarke, Holzmann
  - vérification des propriétés sur un modèle donné du logiciel à vérifier (graphe d'accessibilité)
  - reverse engineering
- Analisis statique vérification par analyse du programme
  - Cousot (les deux)

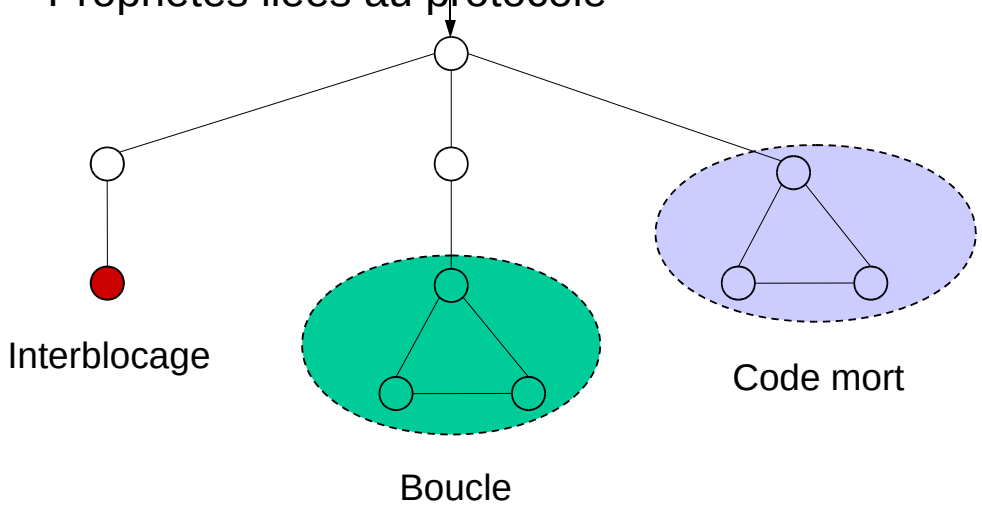
Vérification (model checking)

- Vérification exhaustive
  - Construction du graphe global représentant le comportement du protocole, à partir de la conception formelle
  - Vérification de propriétés sur le graphe global
  - Problème : explosion combinatoire du graphe global

État global du système
=
État de chaque automate
+
Valeur de chaque variable de chaque automate
+
Contenu de chaque file d'attente

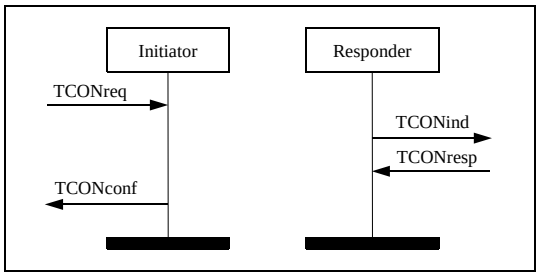
Vérification (model checking)

- Vérification de propriétés sur le graphe global
  - Propriétés liées au protocole



Vérification (suite)

- Propriétés liées au service
  - Propriétés exprimées sous forme de MSC



  - Propriétés exprimées en logique temporelle

$$\text{TCONreq} \Rightarrow \Box\,(\text{TCONind} \vee \text{TDISind})$$

"Après TCONreq, pour tous les comportements possibles, on aura TCONind ou TDISind."

  - Propriétés exprimées sous forme d'automates observateurs