Aperçu rapide des méthodes formelles

Christian Attiogbé
Faculté des sciences – Université de Nantes
Christian.Attiogbe@univ-nantes.fr

Développement de logiciels

Nature des logiciels :

séquentiels, parallèles,

flot de données, transactionnels,

autonomes, centralisés,

répartis, réactifs, temps-réels

embarqués, protocoles, mobiles,

...

Cycles de vie

Ils ont été pendant longtemps le **support méthodologique** du développement du logiciel.

Les plus représentatifs de ces cycles de vie :

– Cycle en V,

– Cycle en cascade,

– Cycle de Balzer

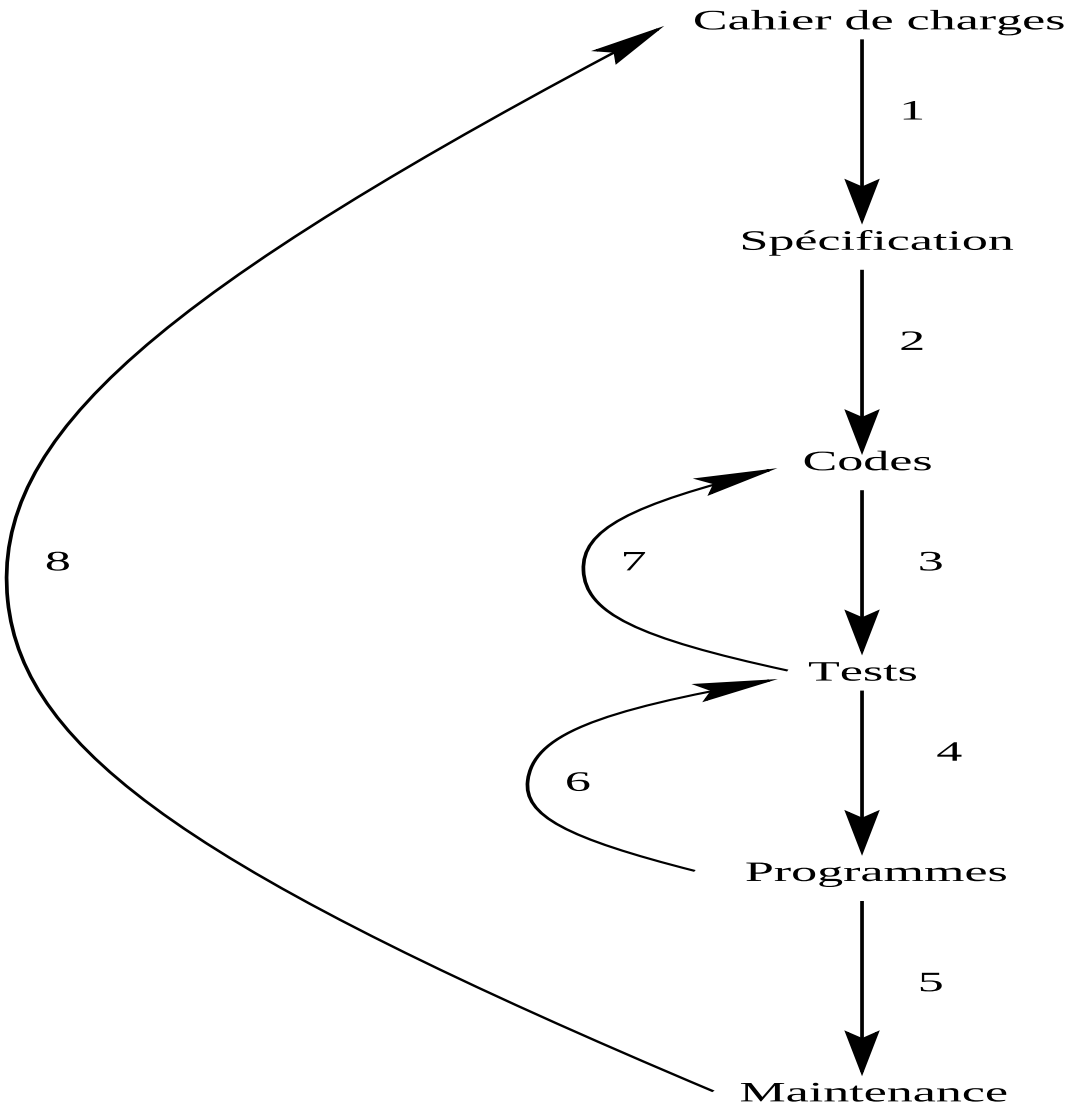

FIG. 1 – Cycle de vie en cascade (Boehm 1977)

Intérêts et limites des méthodes semi-formelles

– SADT, SA-RT, SSADM, ...
– JSD-JSP,
– Merise, Axial, ...
– OOA, OMT, ...
– UML

L'analyse du problème est faite. $\longrightarrow$ une contribution positive même si insuffisante.

Le problème est "dégrossi".

Intérêts et limites des ... (suite)

Mais il est **impossible de raisonner/analyser** formellement sur le système en vue.

Il peut y avoir des ambiguités, ...

sources de BUG !

Spécification formelle

⇒ Expression dans un langage formel du quoi d'un système à développer.

– Résultat de la phase d'analyse

– Plusieurs formes possibles selon la nature du système

⟶ langages ou formalismes de spécification formelle :

Logique, Z, Langages de spécification algébriques, algèbres de processus, etc

Démarche de spécification

Les aspects *données* ou bien les aspects *opérations* ?

Deux grandes écoles de spécification formelle et de méthodes formelles :

- **Les données d'un système permettent de décrire les états du système**

- **les opérations du système permettent de décrire son fonctionnement ou son comportement par des axiomes**

↝ *paradigme des données* et *paradigme des opérations*.

Il convient de distinguer :

– les opérations exprimant le comportement d'un système

– des opérations caractérisant les données du système.

Les premières opèrent sur les données du système,

les dernières permettent de construire et exploiter les données du système.

Il y a un troisième paradigme qui semble être transversal.

C'est ***le paradigme des processus*** (les algèbres de processus).

Dans ce paradigme des processus les systèmes sont décrits par les équations exprimant leur comportement ou leur états.

Les principales algèbres de processus à la base des autres formalismes sont :

– CSP (Hoare)
– CCS (Milner)
– ACP (Bergstra)

Spécifications orientées états

(model-based approach ou state-based approach)

On privilégie les données du système, on s'intéresse à l'état du système.

– **Principe** Une spécification permet de construire, à l'aide des concepts de base fournis, un modèle du système (logiciel) à développer. Ce modèle doit avoir les propriétés du système. On peut ensuite raisonner sur le fonctionnement du système en utilisant le modèle.

Spécifications orientées états (suite)

- **Concepts de base** On utilise essentiellement les mathématiques discrètes (Théorie des ensembles), et la Logique.

- **Quelques exemples de formalismes**
  Z, VDM, B pour les systèmes séquentiels,
  Réseaux de Pétri, CCS, CSP, Unity, ... pour les systèmes concurrents et distribués. (CCS et CSP relèvent du paradigme des processus)

Spécifications orientées propriétés ou axiomatiques

(axiomatic approach, algebraic approach, operation oriented)

– **Principe** La spécification permet de construire une axiomatisation qui fournit les propriétés (comportementales) du système à développer en utilisant les concepts de base cités ci-après.

– **Concepts de base**

Logique,

Algèbre,

Domaines définis inductivement.

– **Exemples de formalisme**

les langages de spécification algébriques : LPG, ASL, ACT ONE, CLEAR, OBJ, PLUSS,...

les algèbres de processus : LOTOS

Spécifications hybrides (hybrid approach)

- **Principe** La spécification permet de compléter une axiomatisation par un modèle de données ou bien de compléter un modèle de données par une axiomatisation.

- **Concepts de base**
  Concepts de base des approches orientées état et propriétés.

Spécifications hybrides (suite)

- **Exemples de formalisme**
  - RAISE (Combine VDM et CSP)
  - VDM-SL et Réseaux de Petri,
  - Extended LOTOS(combine ACT ONE, CCS et CSP),
  - ZCCS (combine Z et CCS),
  - ...

Méthodes de développement formel

Pour résumer l'essentiel des idées nous reprenons le slogan :

prouver c'est programmer et

programmer c'est prouver !

La deuxième façon de résumer est de considérer l'interprétation suivante de l'isomorphisme de Curry-Howard.

Preuve	$\frac{Axiomes}{Theoremes}$

est équivalent à

Développement	$\frac{Specifications}{Programmes}$

Quelques définitions

Méthode formelle =

– Langage formel (syntaxe précise et sémantique précise) et

– Système de preuve ou de raisonnement formel.

Développement formel =

transformation systématique des spécifications en programmes en utilisant des lois prédéfinies.

Plusieurs approches dans les méthodes formelles :

- l'**approche opérationnelle** : le développement est basé sur la description d'un système par un modèle qui a les propriétés du système.

- l'**approche axiomatique** : le développement est basé sur l'axiomatisation qui décrit le comportement du système

- l'**approche hybride**

Nouveau cycle de vie

Avec les méthodes formelles les cycles de vie classique ne marchent plus.

Il faut les adapter.

Le cycle de vie de BALZER représente la nouvelle famille de cycles de vie adaptés au développement formel (ou rigoureux !).

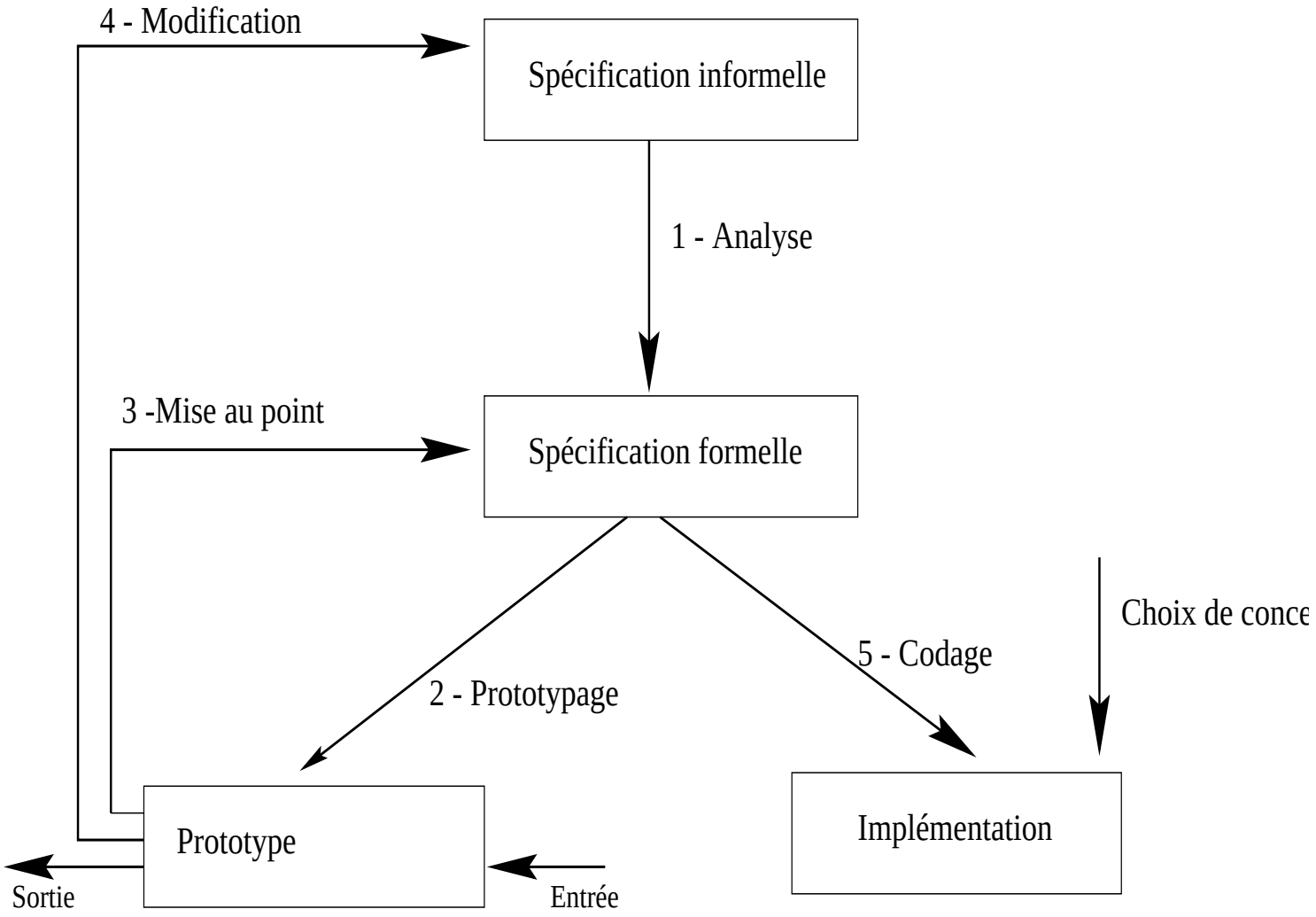

FIG. 2 – Cycle de vie de Balzer

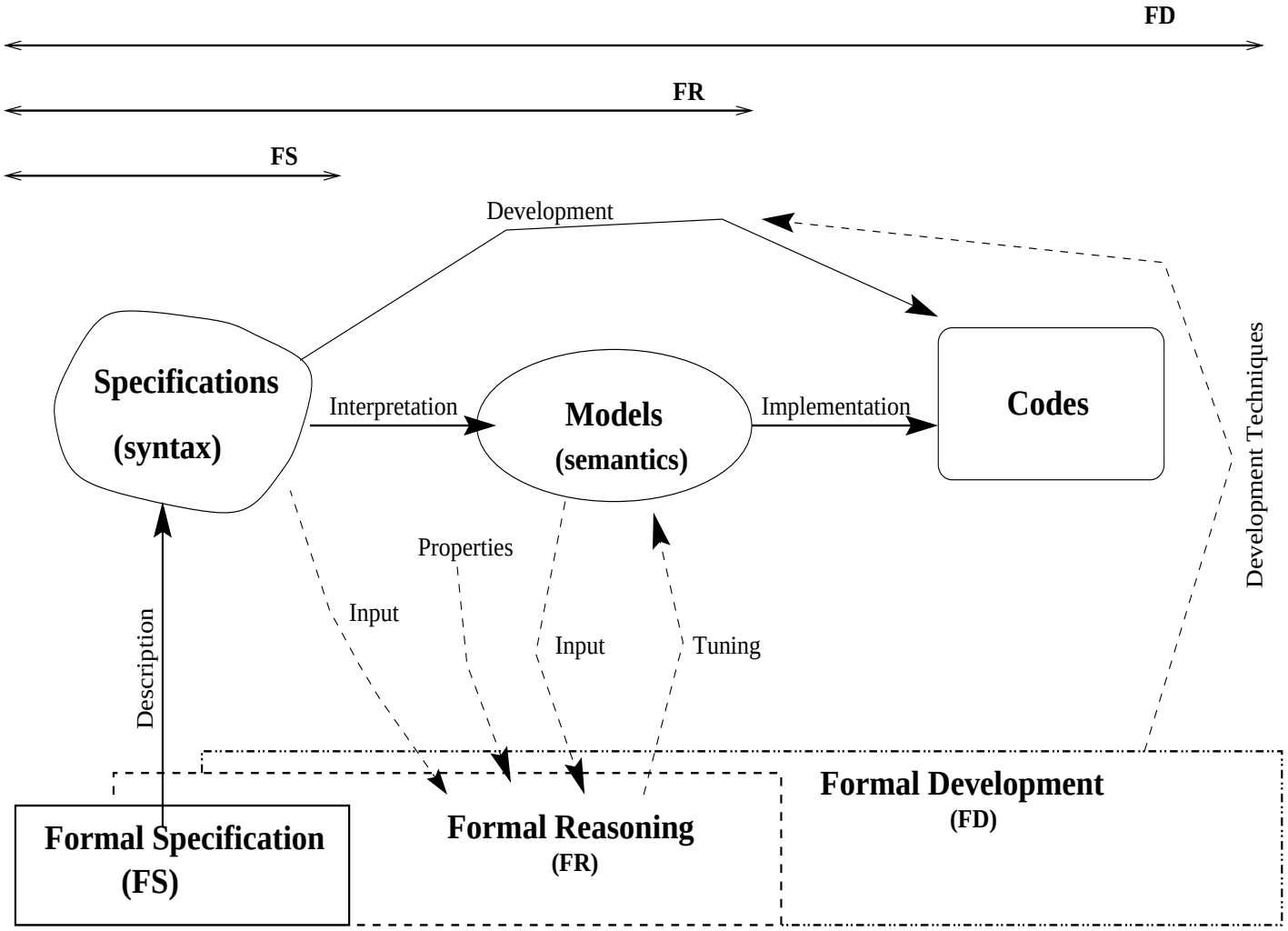

FIG. 3 – Présentation synthétique du développement formel