

5.1 Applications synchrones : langages et programmation

5.1.1 Définitions

Ces langages ont été développés au début des années 80 pour programmer les systèmes 'réactifs'.

On appelle un système **réactif** tout système de contrôle automatique qui doit réagir continuellement à son environnement. Leur vitesse de réaction est imposée par l'environnement qui ne peut pas attendre (ne pas confondre avec les systèmes interactifs : Le rythme de l'interaction est déterminé par le système et non par l'environnement).

Les systèmes réactifs sont souvent caractérisés par leur: *Criticité, parallélisme et déterminisme*.

Approche classique :

Les approches classiques et les langages classiques ne peuvent plus représenter correctement ses systèmes car ils ne permettent pas le parallélisme ni le respect du déterminisme strict (car asynchrones).

Abstraction synchrone :

Les langages synchrones sont basés sur le principe de la simultanéité. Tous les processus parallèles évoluent en même temps (*abstraction logique temporelle*). Cette approche est celle du langage Esterel. Tous les processus mis en parallèle partagent une même échelle de temps globale. C'est le modèle de base de l'électronique, de l'automatique et plus largement des systèmes embarqués.

Une autre approche consiste à considérer le programme synchrone comme un système dynamique, spécifié comme un système d'équations dynamiques. Le rôle du compilateur dynamique consiste alors à résoudre ce système d'équations. C'est le cas des langages Lustre et Signal.

5.1.2 Avantages de l'approche synchrone

Le principal avantage est celui de l'abstraction logique temporelle. La sémantique temporelle est simplifiée.

Comme le Pascal qui est un langage séquentiel de haut niveau typé et structuré, les langages synchrones sont des langages parallèles de haut niveau, ils sont temporellement typés et structurés.

La programmation avec Pascal réduit les bugs fonctionnels et la programmation avec un langage synchrone réduit les bugs temporels.

Dans les langages synchrones, le système évolue pas à pas et entre deux pas successifs, rien ne se passe. Ceci simplifie la vérification, le test et la validation.

Un autre avantage des langages synchrones est qu'ils permettent de générer automatiquement du code embarqué avec des performances qui peuvent être mesurées facilement.

5.1.3 Exemples de langage synchrone

➔ **ESTEREL** Historiquement le langage ESTEREL, développé au Centre de Mathématiques Appliquées (CMA) de l'École des Mines de Paris, à Sophia-Antipolis (France). Ce langage est dédié aux systèmes contrôlés par événements discrets.

Autres langages par ordre chronologique :

➔ **LUSTRE**: C'est un langage fonctionnel déclaratif de flux de données. Il est inspiré de **Lucid**. Les outils **SCADE** qui ont été développés initialement par Verilog et Aerospatiale sont basés sur LUSTRE. SCADE est commercialisé par Esterel Technologies (<http://www.esterel-technologies.com>).

➔ **SIGNAL** : Il est plus général que LUSTRE car il est plus relationnel que fonctionnel. **Polychrony** est le compilateur domaine public alors que **Sildex** est l'outil commercial développé par TNI-Valiosys.

➔ **ARGOS** : Il est la version synchrone du formalisme **Statecharts**. **SyncCharts** et **Mode Automata** sont tous les deux inspirés de Argos.

➔ **POLIS** : C'est un outil graphique pour implémenter des *Codesign Finite State Machines* CFSM. Les CFSM sont un ensemble de FSMs synchrones qui communiquent entre elles d'une manière asynchrone. C'est connu comme des GALS (*Globally Asynchronous Locally Synchronous*). L'outil **Cierto VCC** développé par **Cadence** est basé sur Polis.

➔ **SL** : C'est une variante de ESTEREL et qui a résolu le problème de causalité des signaux. Les hypothèses sur la présence du signal ou non ne sont pas autorisées. La réaction au signal est retardée jusqu'au prochain instant. **SL** fut le point de départ pour plusieurs autres langages synchrones **Sugar Cubes**, **Junior**...

Notes :

Esterel, **Argos**, et **SL** sont plus adaptés aux systèmes discrets contrôlés par événement. **Lustre**, **Signal** et **Polis** sont plus prêts des formalismes de spécification utilisés par les ingénieurs de contrôle automatique et commande numérique (Blocs diagramme, automate, équations différentielles, réseaux de data-flow, etc.).

Impact sur l'industrie des systèmes temps réel, critiques et embarqués

Les langages synchrones connaissent un grand succès dans l'industrie du temps réel, de l'embarqué et des systèmes critiques pour des raisons qu'on verra plus en avant dans ce cours.

Par exemple, Lustre est utilisé pour développer le logiciel de contrôle pour les stations nucléaires ainsi que pour les avions d'Airbus.

Esterel est utilisé pour développer des puces DSP pour l'industrie de la téléphonie mobile et pour le logiciel de contrôle de vol pour le chasseur Rafale de Dassault.

Signal est utilisé pour développer des logiciels de commande de réacteur d'avions.

Les arguments avancés par ces compagnies en faveur de l'utilisation de ces langages synchrones est qu'ils permettent un **développement** de logiciel critique **meilleur** et **rapide** grâce à leur rigueur mathématique et sémantique.

Un autre avantage de ces langages que les **ingénieurs apprécient** le plus est qu'ils permettent une expression des exigences proche de la culture de l'ingénieur de contrôle-commande. Ils permettent de faire de la **conception à base de modèles**. Leur succès le prouve.

5.2 Modélisation

5.2.1 Définitions du model

- Un modèle est une **représentation simplifiée** d'une **partie de la réalité** avec un but spécifique
 - C'est une simplification parce que le monde réel est trop complexe
 - Le mécanisme d'abstraction (simplification) permet de contrôler la complexité intrinsèque du monde réel
 - Permettre de visualiser le système
 - Permettre de mieux comprendre le système
 - Permettre de spécifier la structure et le comportement du système
 - Permettre l'analyse, la simulation et la vérification du système
 - Documenter les décisions importantes

5.2.2 Modèle du système logiciel

Le modèle logiciel a certaines spécificités liées à l'ingénierie du logiciel.

- Le modèle est une représentation de quelque chose qui **n'existe pas encore** !
- Le but de la modélisation est de **faciliter la conception** et la **construction** du système !

5.2.3 Avantages de la modélisation

En plus des avantages cités ci-haut, la modélisation apporte certainement des outils intéressants à l'ingénieur de développement logiciel. Entre autres :

- Facilite la révision et l'évolution du système
- Réduit le risque d'erreurs, permet de détecter les erreurs tôt dans le cycle de vie
- Diminue les coûts de développement
- Réduit le temps à marché (*time-to-market*)
- Réduit le coût de développement
- Contrôle la complexité par le mécanisme d'abstraction
- Permet d'essayer facilement des solutions alternatives

5.2.4 La modélisation

Certaines conditions s'appliquent aux modèles, un bon modèle devrait :

- utiliser une notation standardisée
- être compréhensible pour les clients et utilisateurs

- permettre aux ingénieurs logiciel de bien saisir le système
- procurer une vue abstraite du système
- être visuelle

5.2.5 Modélisation visuelle

Souvent on utilise la modélisation visuelle pour les raisons suivantes :

On doit enregistrer nos pensées et communiquer en utilisant des langages visuels et schématiques (par exemple., UML) parce que on estime qu'au moins 50% de notre cerveau est impliqué dans le processus visuel. Les langages visuels sont naturels et faciles pour notre cerveau.

5.2.6 Vue multiple

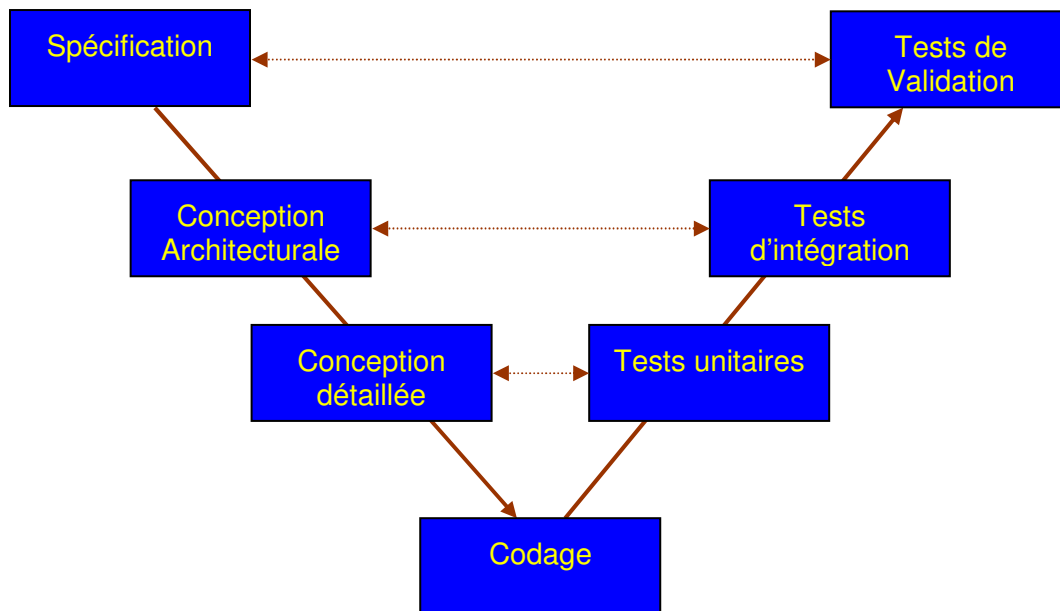
Un modèle peut représenter différents aspects du système logiciel

- Modèle de qualité (sécurité, fiabilité, etc.)
- Modèle de l'utilisation
- Modèle de test et de vérification
- Modèle métier (*business logic*)
- Modèle logiciel

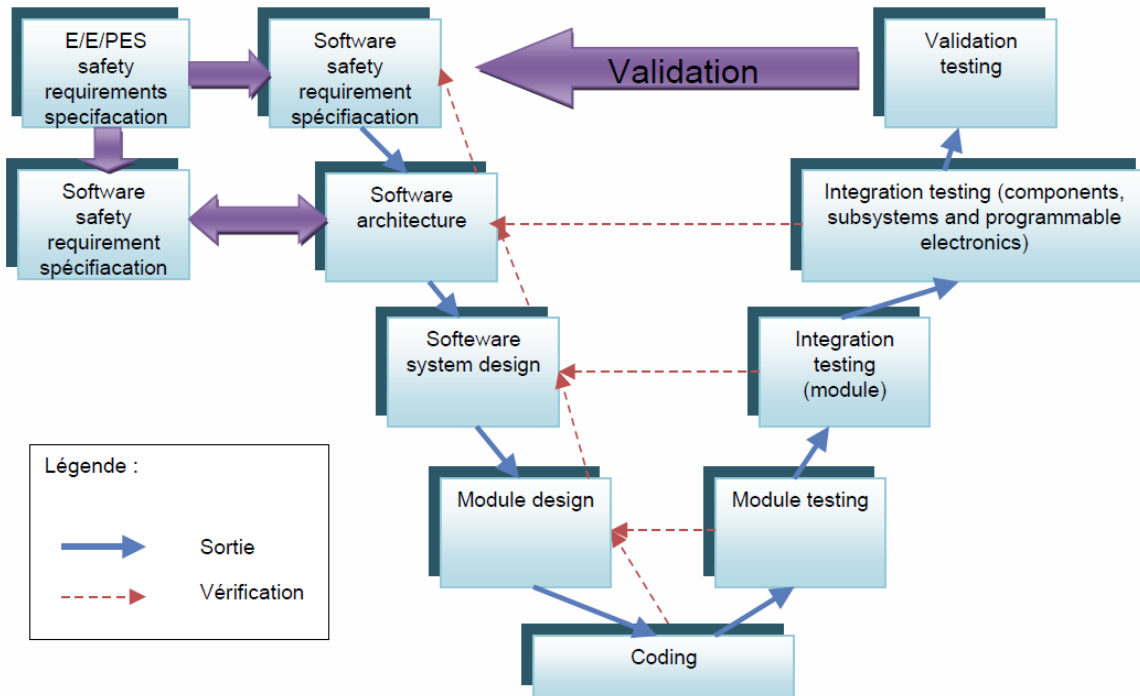
5.3 Conception à base de modèle MBD (*Model Based Development*)

5.3.1 Définitions

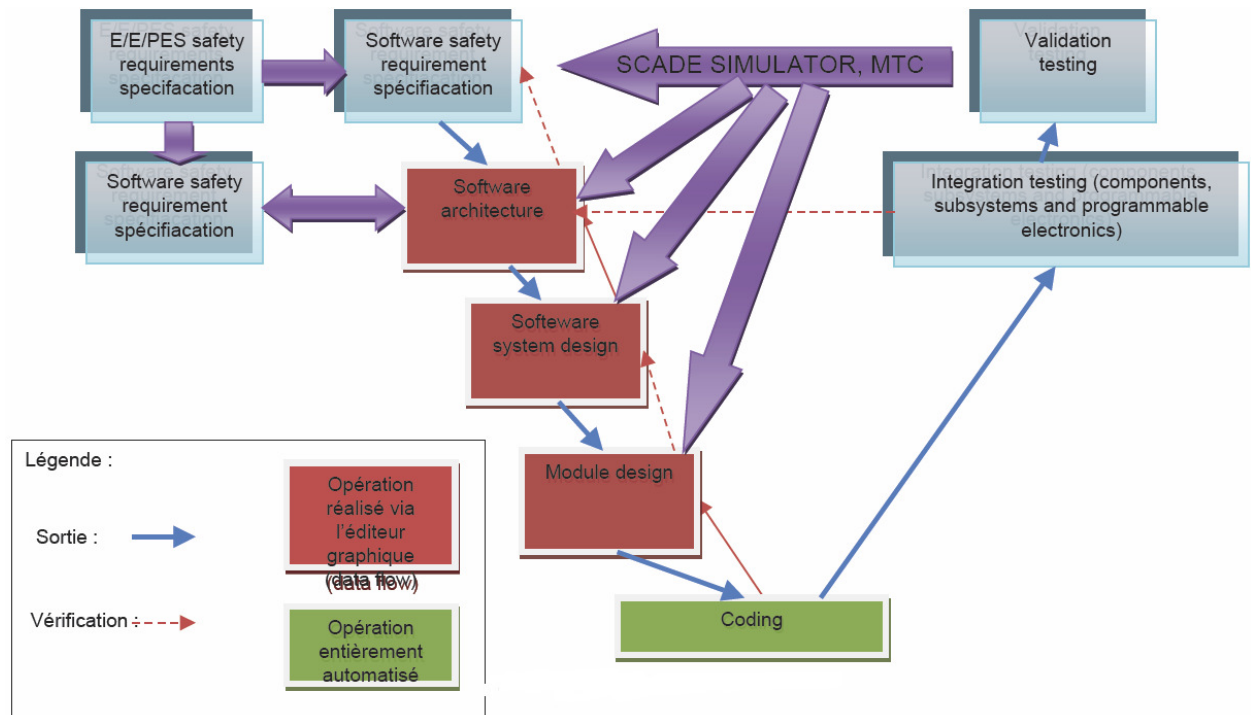
Dans le cycle de vie du logiciel classique on a, par exemple, le cycle en 'V'



Le cycle de développement classique suit le schéma suivant, si on introduit les critères de sécurité, on aura :

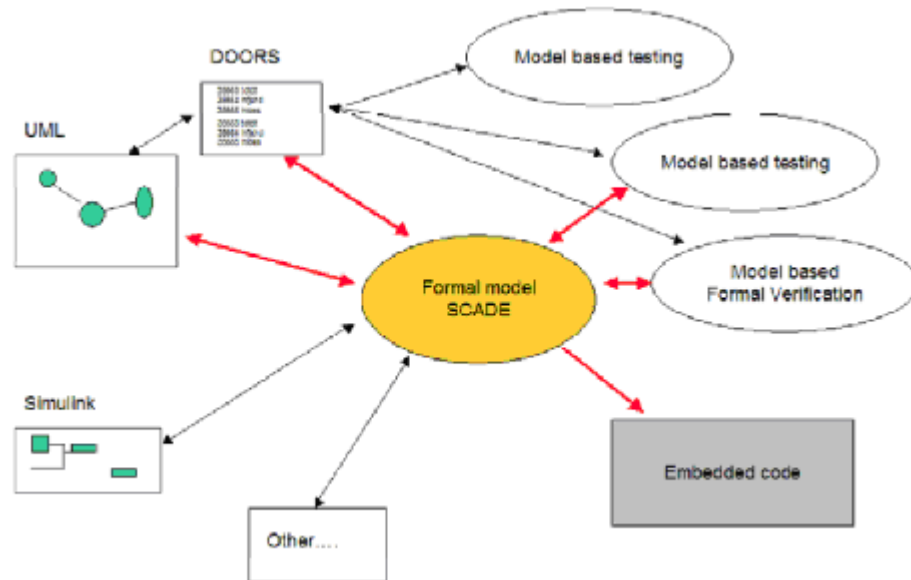


Dans un cycle de développement MBD (*Model based Development*) avec SCADE, on aura plutôt le schéma suivant :



5.3.2 Approche MBD

a. Définition des besoins



b. Spécifications du logiciel

- des diagrammes de flux de données
- des diagrammes d'états-transitions
- Un *mapping* des spécifications fonctionnelles vers une architecture cible spécifique

c. Génération de code

- Le code est **portable** (compilateur, cible et OS indépendants)
- Le code est **structuré** par fonction et par bloc.
- Le code est **lisible** et **traçable** au modèle via l'utilisation de noms et d'annotations
- La **mémoire** allouée est toujours **statique**
- Il n'y a **pas de pointeur**
- Il n'y a **pas de récursivité**

d. Vérification et validation du développement

- série de tests définie à partir du model
- couverture et traçabilité des exigences garanties
- rapport détaillé de la couverture obtenue

5.3.3 Avantage d'une approche MBD

- Une seule référence pour les spécifications
- Le modèle n'est pas ambigu
- Formalisme connu des ingénieurs
- Vérifiable mathématiquement
- Le code est facilement réutilisable
- Pas de codage manuel
- Séparation claire entre le code fonctionnel et le code spécifique à la plate-forme
- Possibilité de créer des banques de modèles et de générer du C MISRA suivant l'OS ou la cible.
- Détection des erreurs tôt dans le cycle de développement
- Application facile à maintenir
- Correction du model et re-génération du code
- Vérification inutile en aval du model

5.4 Introduction à SCADE Suite™

(Inspiré des Publications et présentations d'Esterel Technologies)

Le principe de SCADE est : ***Conception et code sont corrects par construction***

5.4.1 Le défi du développement de systèmes critiques

SCADE Suite™ est un environnement de développement intégré utilisé pour concevoir, simuler, vérifier et développer le code embarqué des systèmes temps-réel critiques, tels que les équipements aéronautiques, les systèmes " *drive-by-wire* " automobiles ou les systèmes de signalisation ferroviaires. Pour ces systèmes critiques et pour leur code embarqué, la qualité n'est pas facultative. C'est une exigence, comme l'indiquent les directives de la norme **RTCA DO178B** de l'aviation civile.

Dans le même temps, l'énorme pression du marché oblige à réduire les coûts et les délais de développement. SCADE Suite™ a été conçue en collaboration avec des sociétés industrielles rodées au développement de tels systèmes (avionneurs, concepteurs de systèmes de transport et de centrales nucléaires) et satisfait les besoins des projets de développement des logiciels embarqués critiques.

La solution SCADE Suite™ offre un bénéfice à ses utilisateurs, auxquels elle assure formellement et automatiquement que :

Ce qui est spécifié est :

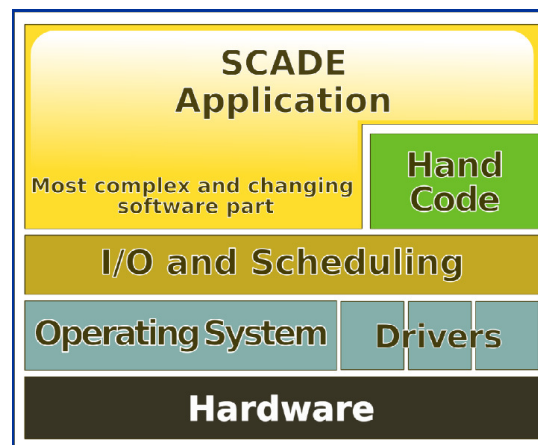
- Ce qui est compris par tous,
- Ce qui est simulé,
- Ce qui est vérifié,
- Ce qui est embarqué.

5.4.2 Caractéristiques techniques

- Simulink™ Gateway permet de récupérer le contrôleur prototypé dans Simulink™ en une représentation formelle discrète afin de le sécuriser et de générer le code embarquable et qualifiable.
- SCADE Suite™ utilise des schémas fonctionnels graphiques formels pour des spécifications précises et un dialogue facilité.
- Une fois décrites, les spécifications peuvent être exécutées. Le simulateur utilise le code généré par SCADE Suite™.
- En mode graphique interactif, le simulateur est un outil de *debug* très puissant. La simulation peut également être lancée en batch sur des scénarios prédéfinis.
- Alors que la simulation teste des scénarios explicites, les outils de preuve intégrés dans SCADE Suite™ permettent de détecter les erreurs de spécification dès la phase de conception en validant le respect ou non par le modèle, quel que soit le scénario d'entrées, des contraintes de sécurité ou de sûreté.
- Le code embarqué C ou Ada est généré automatiquement à partir du modèle SCADE Suite™. Esterel Technologies fournit en particulier un générateur de code C qualifiable pour la norme [RTCA DO178B](#) niveau [A](#).

5.4.3 Quelques informations utiles sur SCADE

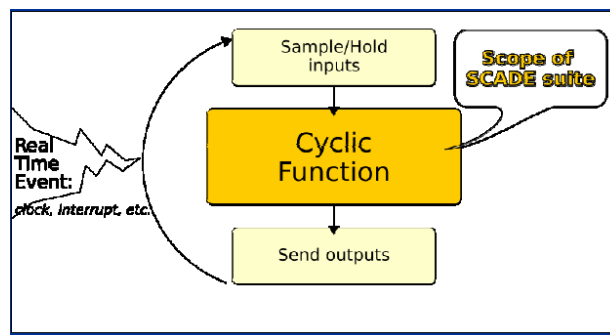
➔ *Place du Code SCADE dans une application typique (d'après Esterel Technologies) :*



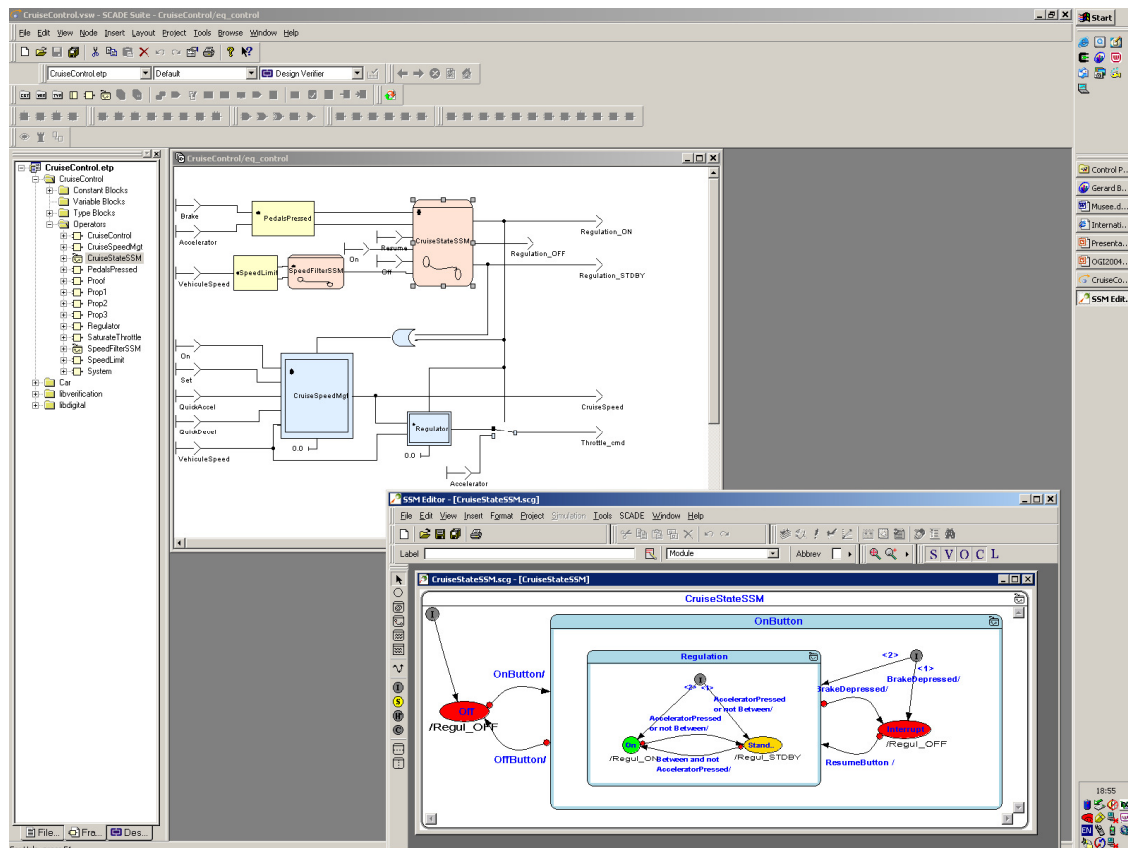
➔ *Programmatuion*

- Le langage SCADE (Lustre) est défini formellement avec des objectifs de sécurité :
 - Le langage est simple et stable. Il interdit des constructions dangereuses (par exemple : des boucle infinies, allocation dynamique de la mémoire, etc.)
 - L'interprétation du modèle ne dépend pas du lecteur ou de l'environnement.
 - 20 années de recherche active en collaboration avec le monde de l'aéronautique et le nucléaire.
- Le langage est :
 - Complètement modulaire.
 - Très typé.

- Opérateurs temporels 'explicites'.
 - Valeurs initiales des données 'explicitement' déclarées.
 - Complet et causal.
 - 'Data flow' et 'Control flow' sécuritaires.
 - L'expression de la concurrence est simple et sécuritaire.
- Pour chaque cycle, le programme calcule son vecteur de sortie à partir d'un vecteur d'entrée stable.
 - Il n'y a pas d'interaction entre le programme et le monde extérieur durant le cycle de traitement qui bénéficie de propriétés théoriques et pratiques très solides.
 - Comportement totalement déterministe, temps de traitement fini.
 - Beaucoup plus facile à vérifier que les programmes asynchrones.

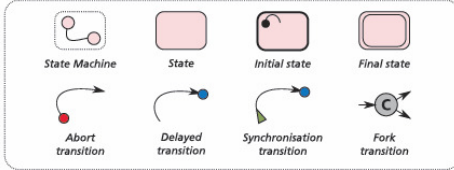


➔ Vue de l'environnement SCADE



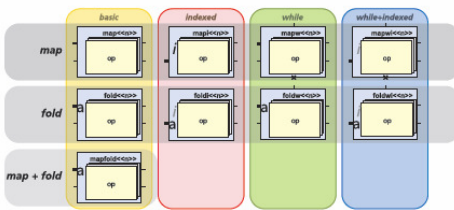
➔SCADE : Carte de référence

State Machines

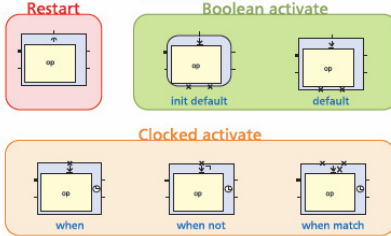


Higher Order

Iterators



Time modifiers



To contact us

For Technical Support: support@esterel-technologies.com
 For Commercial Inquiries: sales@esterel-technologies.com
 For more information: info@esterel-technologies.com
 For current and product details: www.esterel-technologies.com

Ctrl+L
 Ctrl+O
 Ctrl+P
 Ctrl+S
 Alt+V
 Alt+Z
 F1
 Shift+F1+click

F2
 Alt+Enter
 Ctrl+A
 Ctrl+V
 Ctrl+C
 Ctrl+X
 Ctrl+Z
 Ctrl+Y
 Ctrl+F
 Ctrl+G

Ctrl+Shift+P
 Ctrl+Shift+C
 Ctrl+Shift+T
 Ctrl+Shift+E
 Ctrl+Shift+N
 Ctrl+Shift+I
 Ctrl+Shift+O
 Ctrl+Shift+H
 Ctrl+Shift+L
 Ctrl+Shift+B
 Ctrl+Shift+S
 Ctrl+Shift+M
 Ctrl+Shift+R
 Alt+Shift+Up
 Alt+Shift+Down
 Ctrl+operator
 Ctrl+struct type
 Shift+struct type
 Ctrl+click on flow
 Ctrl+pin
 Shift+(de)selection

Ctrl+arrow
 Arrow
 Shift+arrow
 Ctrl+Spacebar
 Shift+ area
 Shift+resize
 Ctrl+resize

F4
 Shift+F4
 F8
 Ctrl+F6
 Alt+Up/Down arrow
 Alt+Left/Right arrow

Ctrl(+Shift)+Tab
 Esc
 (Shift)Tab
 Enter
 Spacebar

File management

Save all
 Open
 Print
 Save
 View Workspace
 View Output
 Online help
 Contextual help

Editing

Edit mode
 Edit Properties
 Select all
 Copy
 Paste
 Cut
 Undo
 Redo
 Find
 Goto line

Design

Insert Package
 Insert Constant
 Insert Type
 Insert Element
 Insert Operator
 Insert Input
 Insert Output
 Insert Hidden Input
 Insert Local Variable
 Insert Probe
 Insert Signal
 Connect by name
 Connect by rank
 Move Up
 Move Down
 Boolean Activate
 Compose
 Decompose
 Add Point
 Multiple flows
 Incremental (de)selection

Graphical layout

Align on arrow side
 Nudge
 Move
 Optimize flow
 Lasso selection
 Constrained resize
 Center resize

Navigation

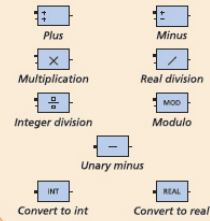
Next Item
 Previous Item
 Switch focus
 Goto Back
 Navigate Up/Down
 Navigate Previous/Next

Dialog box navigation

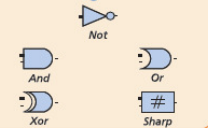
Tab display
 Close dialog
 Option focus
 Validate
 Checkbox selection

SCADE Predefined operators

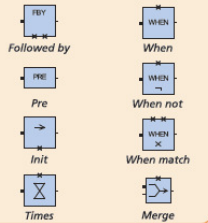
Arithmetics



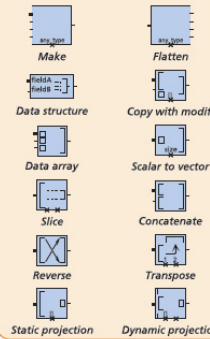
Logical



Time



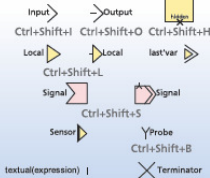
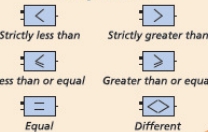
Structured



Choice



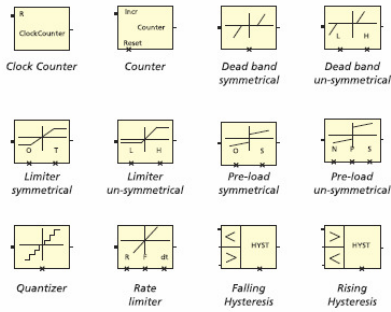
Comparison



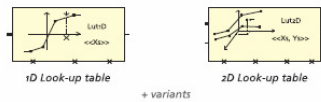
Modeling constructs



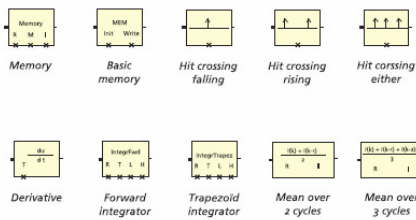
libpwnlinear pwnlinear package



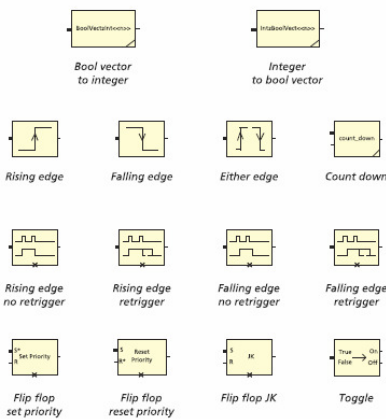
lut package



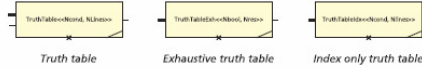
liblinear linear package



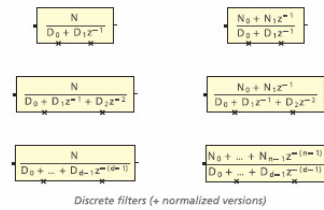
libdigital digital package



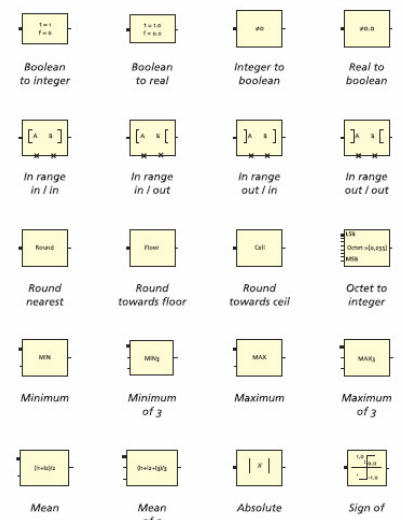
truthtables package



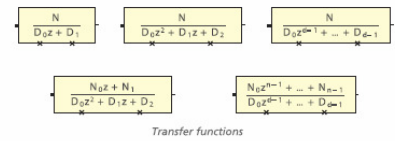
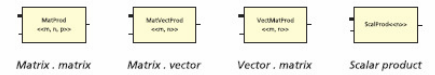
filters package



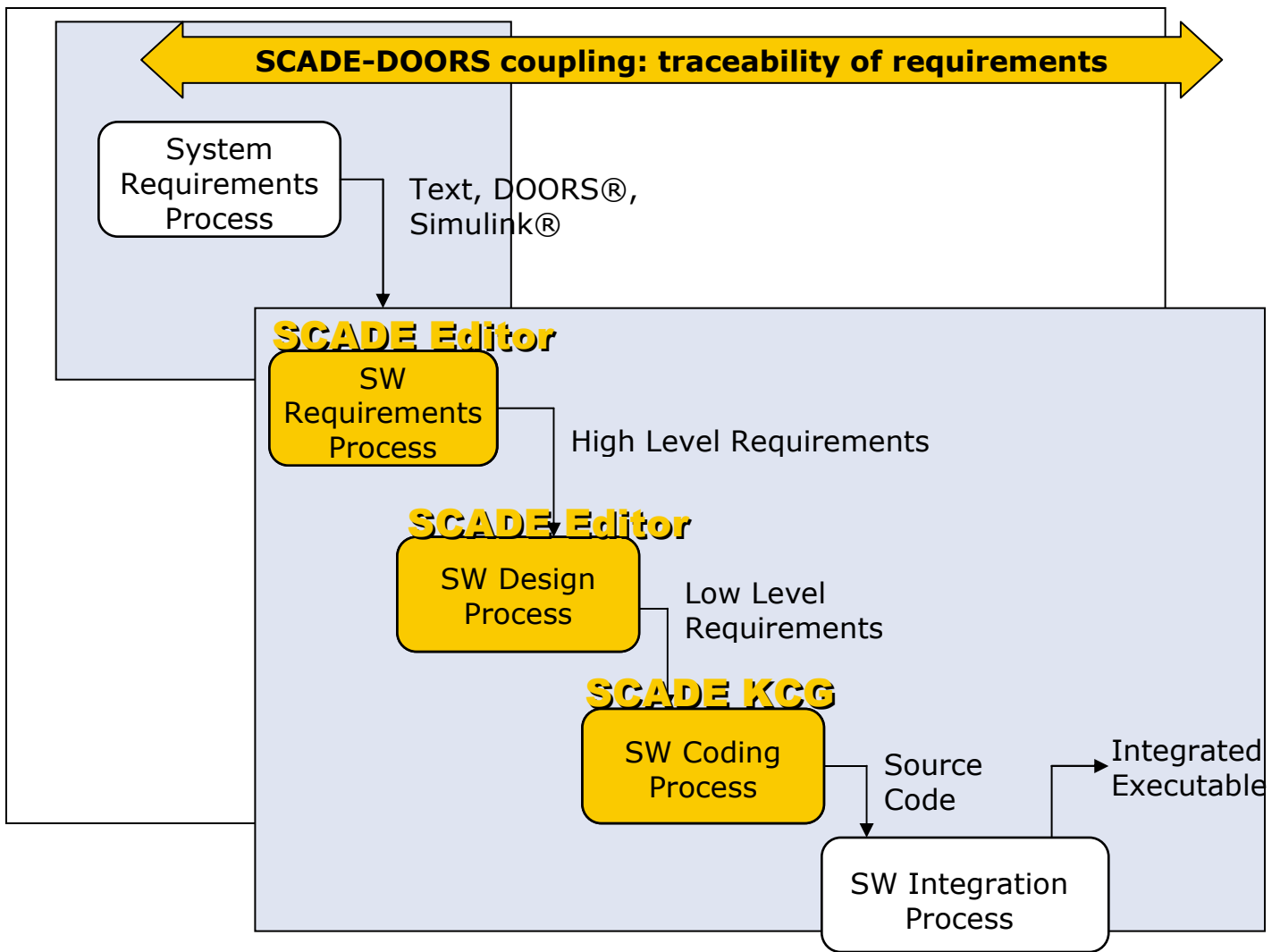
libmath math package



vect package



➔ *Flot formal SCADE:*



Typical Aerospace & Defense Applications

- Flight control systems
- Power management
- Reconfiguration management
- Autopilots
- Engine control systems
- Braking systems
- Cockpit display and alarm management
- Fuel management





















Estotel Technologies © 2005 — www.esterel-technologies.com

Typical Automotive Applications

- Airbags
- Braking Systems, ABS & ESP
- Steering
- Chassis & Suspension Systems
- Driver Assistance Systems
- Restraining systems
- Engine regulation
- X-By-Wire applications

Estotel Technologies © 2005 — www.esterel-technologies.com

Typical Rail & Heavy Industry Applications

- ▶ Interlocking systems control
- ▶ Signaling
- ▶ Ground stations
- ▶ Automatic Train Operations
- ▶ Train Control Systems
- ▶ Critical Graphics Displays
- ▶ Level Crossings
- ▶ Safe Platforms
- ▶ Heavy Duty Land systems (tanks, tractors, construction equipments...)



Amato - EUROSTAR



RATP - Paris Subway (Metro line)



John Deere - EU

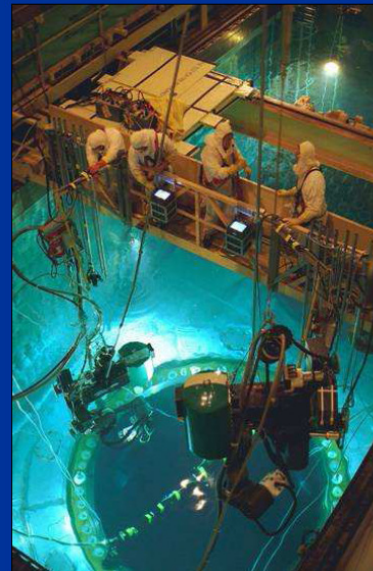


12

Esterel Technologies © 2005 — www.esterel-technologies.com

Typical Nuclear I&C Applications

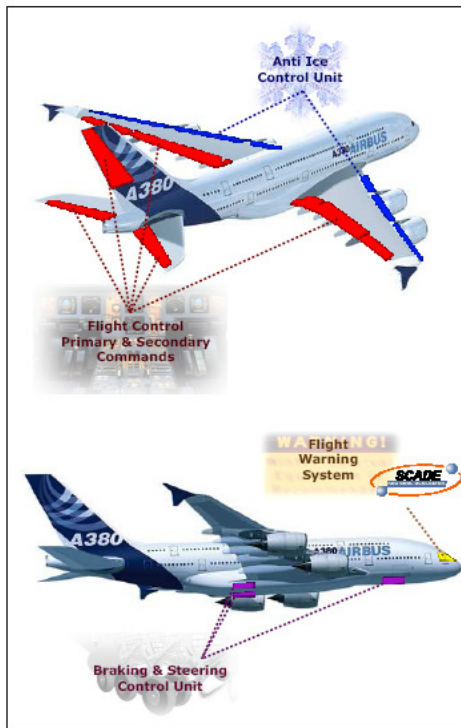
- ▶ Reactor Protection Systems (Safety Class 1E)
 - ▶ Reactor limitation system
 - ▶ Trip processing & Emergency shutdown
 - ▶ Safety actuation
- ▶ Neutron Instrumentation Systems
 - ▶ Power measurement system
 - ▶ Neutron detectors (Safety Class 1E)
 - ▶ Pressurizer heating controllers
 - ▶ Neutron instrumentation systems (Safety Class 1E)
 - ▶ Boron meters
- ▶ Other Safety Systems
 - ▶ Safety valve control system
 - ▶ Rod control systems (Non Safety Class 1E)
 - ▶ Diesel sequencing system (Safety Class 1E)
 - ▶ Rod position instrumentation systems



13

Esterel Technologies © 2005 — www.esterel-technologies.com

SCADE in the A380



- SCADE = Airbus corporate standard for all new airplanes developments

- Flight Control system
- Flight Warning system
- Electrical Load Management system
- Anti Icing system
- Braking and Steering system
- Cockpit Display system
- Part of ATSU (Board / Ground comms)
- FADEC (Engine Control)
- EIS2 : Specification GUI Cockpit:
 - PFD : Primary Flight Display
 - ND : Navigation Display
 - EWD : Engine Warning Display
 - SD : System Display