Plate-forme pour l'étude et la conception de systèmes automatisés

C. André, M-A. Peraldi-Frati, D. Gaffé
Laboratoire Informatique Signaux et Systèmes (I3S)
Université de Nice Sophia Antipolis
CNRS UPRES-A 6070
BP121
06903 Sophia Antipolis
Email {andre, map, gaffe}@i3s.unice.fr

Résumé

Les enseignements d'informatique temps réel, s'appuient largement sur des concepts et techniques souvent spécifiques du domaine. Notre objectif est de faciliter l'assimilation par les étudiants, des méthodes et outils correspondants.
Pour cela, nous avons développé une plate-forme logicielle, qui regroupe les outils et simplifie leurs utilisations. Cette plate-forme permet un usage en groupe aussi bien que personnel. Elle privilégie les accès interactifs que ce soit au niveau des modèles (description et analyses), ou au niveau des réalisations (simulation réaliste de processus et de leur contrôleurs). L'article présente cette plate-forme comme le résultat d'une réflexion pédagogique et donc applicable à d'autres domaines. Son contenu technique est seulement esquissé puisqu'il n'est pertinent que pour les spécialistes du domaine.
Mots-Clefs : Environnement d'apprentissage interactif, Simulation et formation, Conception d'environnements

Abstract

A "real-time systems" course relies on concepts and techniques, often very specific to this field. Our goal is to facilitate the learning of associated methods and tools.
To address this problem, we have developed a software platform, which includes several tools and makes their use easier. The platform allows individual and large group uses. Interactivity has been favoured. This feature is present at the model level (description and analysis) and at the design level (realistic simulation and implementation testing), as well.
This paper reflects our teaching experience. Technical details specific of real-time systems are beyond the scope of the paper. We focus on concepts and techniques that can be extended to other domains.
Keywords : Interactive learning environment, Simulation and training, Environment design.

Introduction

Cette expérience concerne les enseignements d'informatique industrielle au sens large c'est-à-dire recouvrant à la fois les modèles pour l'expression des comportements et des traitements, leurs méthodes d'analyse, ainsi que la mise en œuvre par programmation logicielle ou matérielle. Généralement, les aspects modélisation, conception, programmation et expérimentation relèvent de modules d'enseignement différents. Ceci ne facilite pas, pour les étudiants, une perception globale de la problématique des applications temps réel.

La maîtrise de ces concepts et outils passe, en particulier, par une *expérimentation* aussi *réaliste* que possible de leur utilisation. Ceci suppose une prise en compte de l'environnement soit réel (maquette physique), soit simulé (modèle comportemental). Cette deuxième solution a l'avantage de réduire les coûts et les risques de dégradation.
Nous proposons un *enseignement intégré* depuis la spécification jusqu'à la réalisation en dégageant à chacun de ces niveaux les principes et des méthodes de mise en œuvre. Cet enseignement s'appuie sur des outils et méthodes ad hoc, souvent issus de recherches. Cet ensemble est regroupé au sein d'une *plate-forme logicielle* avec interface conviviale, qui privilégie les *accès interactifs*. Les étudiants peuvent ainsi se familiariser avec les modèles et tester leurs réalisations par simulation. Ces phases d'expérimentation peuvent se faire aussi bien en groupe qu'individuellement.
La plate-forme permet de choisir le niveau d'expertise en fonction du cursus et du niveau d'étude. Elle offre également une flexibilité au niveau des outils : l'enseignant peut choisir l'outil le mieux adapté à sa pédagogie parmi plusieurs analogues. Ceci est nécessaire pour nos enseignements qui sont dispensés dans divers cycles à des informaticiens et des automaticiens. En revanche, l'ossature de ces enseignements reste la même depuis les concepts (modèles et analyse) jusqu'aux réalisations.

Dans une première partie nous précisons le contexte de ces enseignements. Nous identifions des objectifs pédagogiques liés à notre domaine d'enseignement et nous présentons les réponses apportées par notre plate-forme. La troisième partie décrit la plate-forme elle-même. Son utilisation est ensuite illustrée. Nous résumons dans la conclusion les points qui nous paraissent les plus positifs et susceptibles d'être étendus à d'autres domaines.

Contexte d'enseignement

Le domaine d'étude est celui des systèmes à événements discrets plus particulièrement appliqué aux systèmes automatisés. Ces systèmes ont des spécificités liées au caractère réactif affirmé, à la richesse des interactions

avec l'environnement et à la diversité des contraintes imposées par cet environnement.
Il est certain que l'étude de ces systèmes est abordée différemment en fonction du cursus (Informatique, Électronique-Automatique) des étudiants mais aussi de manière plus ou moins détaillée en fonction de leur niveau (1er, 2ème ou 3ème cycle ; DUT, École d'ingénieurs, licence, maîtrise, DEA). De nos expériences d'enseignement dans ces divers cursus nous retenons les idées fortes suivantes :

- la nécessité d'une vision d'ensemble sur la façon d'aborder ce domaine d'applications quels que soient la sensibilité des étudiants et leur degré d'avancement ;
- le choix d'accorder un poids plus important aux aspects conception pour les filières DUT et Écoles d'ingénieurs, et aux modèles et analyse de propriétés pour les étudiants de Faculté des sciences ;
- le besoin d'adapter les enseignements pour combler les lacunes dues aux origines diverses des étudiants.

Ce dernier point est celui qui amène le plus de variété sur le contenu des enseignements. Le tableau suivant résume pour les filières Informatique et EEA les acquis (A), les compléments (C) et les découvertes (D).

	Informatique	EEA
Modèles physiques /Matériel	C	A
Algorithmique / programmation	A	C
Systèmes d'exploitation	A	C
Méthodes de Conception	C	C
Interface électronique	D	A
Interface Graphique	A	D
Programmation Réactive	D	D
Aspects Temps Réel	D	D

Tableau 1 : Connaissances en fonction des cursus
A : Acquis, C : Complément, D : Découverte

Nous enseignons les méthodes et outils qui permettent la mise en œuvre des systèmes évoqués précédemment [APR97].

Approche pédagogique

La première phase de ces enseignements consiste à bien faire comprendre la *problématique liée à ce domaine d'applications*. Ces applications sont souvent conçues comme des sous-systèmes coopérants. Nous insistons sur le parallélisme de leurs évolutions, l'importance des communications et des synchronisations entre ces sous-systèmes ainsi que sur les problèmes de partage de ressources.

Un deuxième aspect à dégager est l'existence de contraintes liées au fort couplage entre les contrôleurs et leur environnement. Ceci induit des contraintes sur les temps d'exécution, la sécurité des systèmes et des exigences de sûreté de fonctionnement, ensemble de contraintes que nous qualifions de « temps réel ».
La deuxième phase est *l'apprentissage des modèles*. Nous insistons surtout sur les modèles comportementaux qui permettent d'exprimer la dynamique des systèmes. Les applications très réactives que nous étudions se prêtent bien à une modélisation par événements discrets. Nous nous intéressons essentiellement aux modèles état/transition. Les machines à états finis sont les modèles de base, les réseaux de Petri et le Grafcet facilitent la représentation du parallélisme, les Statecharts et autres modèles synchrones introduisent les aspects hiérarchiques et les préemptions.

La troisième phase aborde le *problème de la conception* (préliminaire aussi bien que détaillée). Les enseignements s'appuient sur des études de cas permettant d'illustrer les méthodes dans le but d'acquérir peu à peu un savoir faire. Pour la conception temps réel, les méthodes présentées sont SA/RT et surtout des méthodes modernes orientées objets (ROOM, ROPES, Real-time UML).
En ce qui concerne la conception détaillée, notre effort porte sur des techniques systématiques de programmation conduisant à des mises en œuvre rapides et maintenables. Lorsqu'il s'agit de mise en œuvre logicielle, nous utilisons comme langages de programmation C, C++, Java ainsi que les langages synchrones pour les troisièmes cycles.

La dernière phase est celle d'*expérimentation et de validation*. Durant cette phase, les étudiants doivent évaluer de manière qualitative et quantitative leur solution.

À chacune de ces phases, nous avons fait en sorte que l'étudiant puisse expérimenter les enseignements par des outils qui facilitent leur assimilation.

Durant les phases 2 et 4 ils utilisent des outils interactifs. L'apprentissage des modèles (phase 2) se fait au travers d'outils qui permettent leur édition, leur transformation en d'autres modèles et surtout leur analyse par simulation et par preuve. L'objectif est triple : *compréhension* du modèle et de ses propriétés, *utilisation* du modèle sur des exemples et *évaluation* de la solution proposée. Les outils que nous mettons à disposition sont détaillés dans le paragraphe de description de la plate-forme.
Les expérimentations nécessaires à la phase 4 sont réalisées soit sur des maquettes physiques, soit en simulation. Les *simulateurs* ont l'avantage de représenter de manière réaliste l'environnement à contrôler, qui peut être inaccessible du fait de son coût, du danger à l'expérimenter, de son indisponibilité. Dans la plate-forme, nous mettons à disposition des outils

permettant de générer ces simulateurs incluant le processus et le contrôleur développé par l'étudiant. Ces simulateurs sont portables ce qui permet une utilisation personnelle, hors salle de travaux pratiques.
L'assimilation des techniques de conception est facilitée par le développement itératif de prototypes : répétition des phases 2, 3, 4 jusqu'à l'obtention d'un prototype satisfaisant.
En résumé, la plate-forme que nous détaillons dans les paragraphes suivants est conçue comme *un support pédagogique d'illustration de l'enseignement et d'incitation à la création.*

Description de la plate-forme

Contenu

La figure 1 représente les différents éléments de la plate-forme. Sur cette plate-forme apparaissent deux aspects de l'application : la partie contrôle (partie supérieure de la figure) et la partie opérative (partie basse) L'étudiant peut intervenir aux deux niveaux soit pour concevoir la solution (mise en œuvre du contrôleur), soit pour faire un apprentissage du processus à contrôler (utilisation du simulateur).
La partie à développer par l'étudiant est appelée le *contrôleur*. La partie opérative peut être soit le *processus* physique soit un *simulateur* de ce processus. Le simulateur possède une *interface graphique* écrite en TCL-TK qui représente avec plus ou moins de réalisme le processus physique. Cette interface est couplée à un *modèle comportemental* du processus.
Le fait de disposer d'un simulateur a deux avantages :

- Le premier est de pouvoir se familiariser avec le processus à contrôler indépendamment du contrôleur à réaliser.
- Le deuxième avantage est que ce simulateur permettra ensuite de tester la partie contrôle de manière beaucoup plus réaliste.

Le couplage contrôleur/simulateur conduit à une expérimentation plus réaliste qui évite de tester des scénarios d'événements qui ne se produisent jamais.
Le simulateur ainsi que le processus réel interfèrent avec la partie contrôle par l'intermédiaire d'une *interface* commune qui assure la coopération entre les deux parties (Figure 1).
Du coté de la partie contrôle, l'utilisateur choisit un modèle pour la mise en œuvre du contrôleur. Le contrôle est généralement exprimé avec des modèles de haut niveau (Automate, Grafcet, Statechart ...) ou bien directement implémenté par des langages impératifs classiques. Pour chacun de ces modèles, on dispose de compilateurs permettant la *génération de codes* intermédiaires communs aux différents outils. Il est souhaitable pour la mise au point des applications, de disposer, pour chaque format d'entrée, d'un « mécanisme de remontée dans les sources ». Cette technique permet de répercuter des résultats d'analyse ou de simulation dans le même formalisme que celui choisi pour décrire la solution.
Une autre approche est la *validation formelle* de ce même contrôleur, ce qui conduit à l'utilisation *d'outils de preuve*. Ces outils sont issus de l'environnement de développement des langages synchrones [BG92], [BOU98] développé au CMA de l'Ecole des Mines, d'autres ont été développés dans le cadre de travaux de recherche de notre équipe (SPORTS du laboratoire I3S) : syncCharts [ADB97], S-Grafcet [GA96], d'autres enfin sont classiques : Autograph [ATG], SIS [SS92].

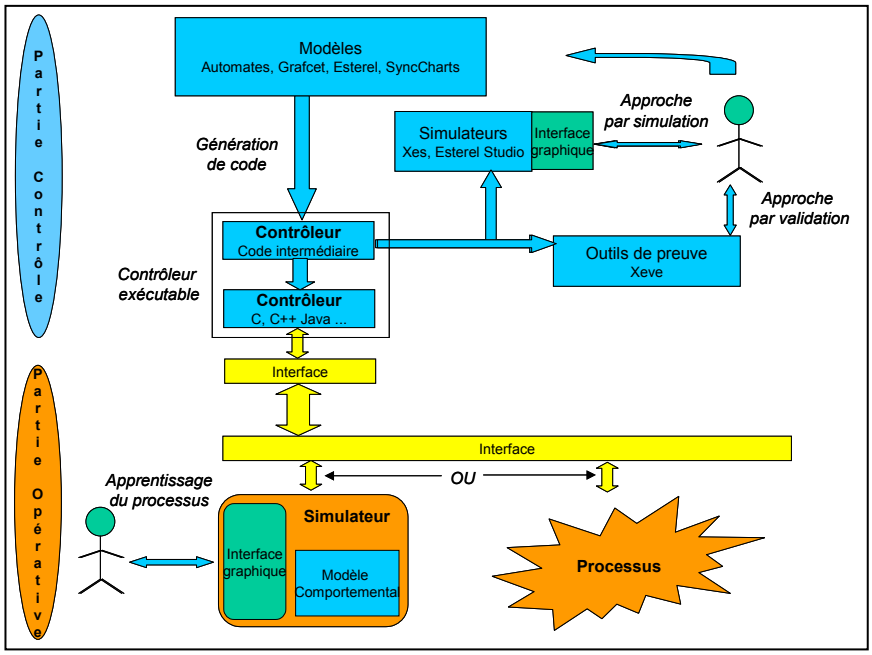

Figure 1 : Constituants de la plate-forme de simulation

Réalisation

La plate-forme a été développée suivant des règles de génie logiciel afin d'assurer :

- la *transparence* des échanges d'informations entre outils (formats intermédiaires, protocoles) ;
- la *portabilité,* pour son fonctionnement sur du matériel hétérogène (multi plate-forme d'exécution : windows, linux, solaris) ;
- l'*extensibilité* afin de pouvoir ajouter des fonctionnalités supplémentaires sans avoir à remettre à plat la construction de l'ensemble du simulateur ;
- un *degré d'expertise* adaptable en fonction du public visé.

Ces qualités participent à la convivialité de la plate-forme. Il est certain que sa conception a demandé un investissement important en temps. Cet effort permet maintenant l'intégration d'applications nouvelles de manière plus aisée.

Exemples d'expérimentations

L'idée de cette plate-forme nous est venue d'un cas d'école soumis à la communauté scientifique : l'atelier de production FZI (ForschungsZentrum der Informatik, Karlsruhe). Il avait été choisi comme exemple pour l'évaluation des méthodes formelles de développement des systèmes réactifs [LL95]. Le but était de mettre à disposition de la communauté scientifique un exemple de processus possédant les caractéristiques et contraintes des systèmes temps-réel. C'est une chaîne de production de pièces à usiner et à stocker. Un cahier des charges de l'application ainsi qu'une interface graphique représentant le processus sont fournis. Cette interface graphique, couplée à un modèle comportemental de l'atelier, permet de simuler les différents dispositifs de l'atelier et voir leurs évolutions.

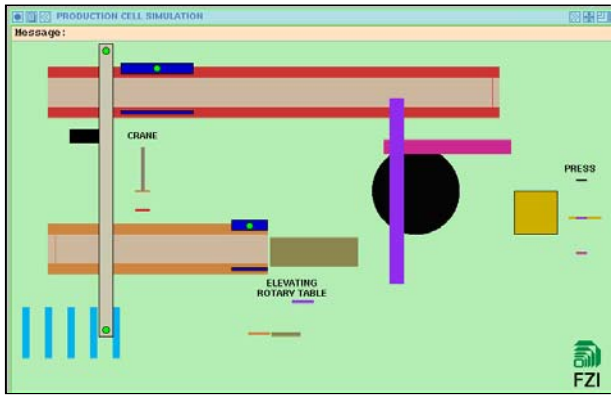

Figure 2 Simulateur TCL-TK d'un atelier d'usinage de pièces.

Cette maquette était prévue initialement pour un apprentissage du processus et fournissait un protocole rustique de communication permettant de brancher les contrôleurs à tester.
Nous avons repris ce principe en enrichissant le protocole pour offrir la possibilité d'évolutions concurrentes entre le contrôleur et le processus nous rapprochant ainsi un peu plus de la réalité. Cette technique de dialogue a été ensuite étendue aux outils associés aux modèles.
Le simulateur FZI a été le premier adapté et intégré dans notre plate-forme. Nous avons ensuite développé, à des fins d'enseignement, d'autres simulateurs de processus (Digicode, Lave linge, Four micro-onde, Ascenseur ...).
Il est même possible grâce au couplage entre simulateurs et à la remontée dans le source du contrôleur de visualiser simultanément l'état de la maquette logicielle et celui du contrôleur.
La figure 3 représente la maquette du lave linge agrémentée de quelques commentaires. Quant à la figure 4, c'est un instantané de l'état du contrôleur exprimé, dans ce cas là, en Grafcet.

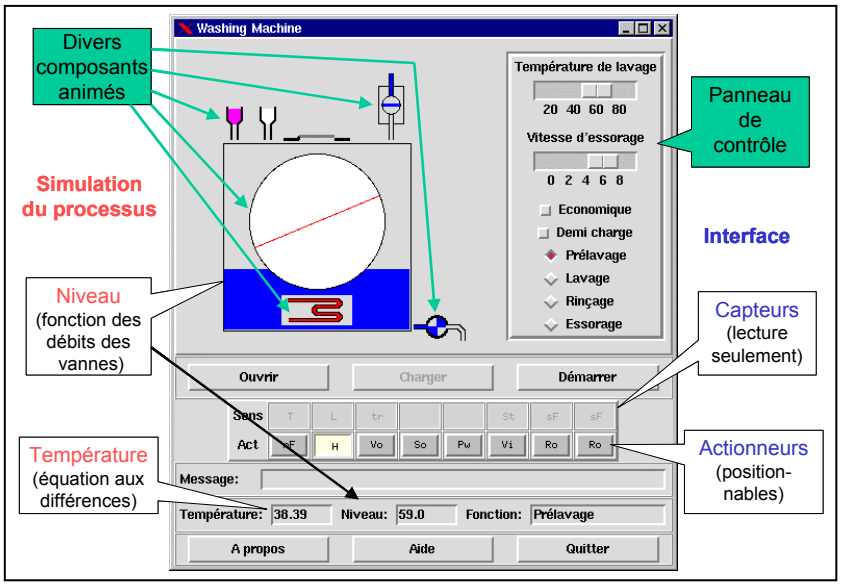

Figure 3 : Maquette logicielle de la machine à laver.

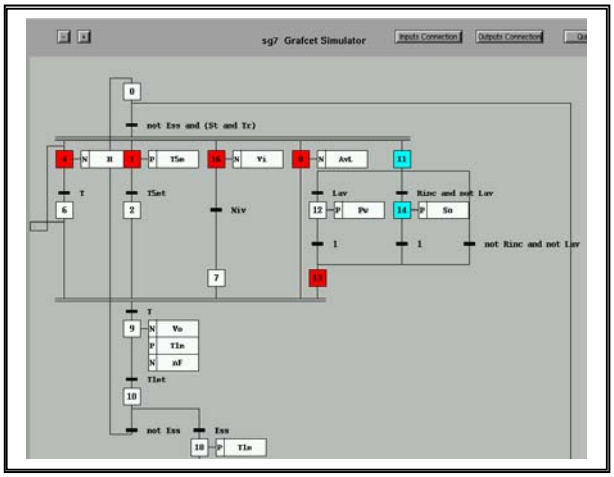

Figure 4 : Contrôleur Grafcet du lave linge.

Retours d'expérience

Nous allons distinguer d'une part le public concerné (Informatique, Automatique-Electronique) et d'autre part l'activité (modélisation ou simulation).

Un exemple à caractère industriel comme l'atelier FZI a suscité plus d'intérêt de la part des étudiants en Automatique par le caractère industriel, quoique simplifié, de l'application à laquelle ils n'avaient pas accès physiquement. Pour les informaticiens, les problèmes de coordination entre les différents équipements ont été très difficiles à surmonter. Au delà de ces difficultés, le fait de pouvoir activer individuellement les différents actionneurs du processus et enchaîner des réactions est une étape très positive qui s'est révélée indispensable à la compréhension du processus. C'est le premier pas vers l'élaboration du contrôle. De plus, nous avons fourni une solution opérationnelle exécutable en pas à pas qui a permis, à de nombreux étudiants, de mieux comprendre les nécessités de synchronisation et de gestion de ressources.

Sur les maquettes plus simples telles que la machine à laver, le processus lui-même n'a pas posé trop de problèmes. En revanche, cette classe d'applications a mis en évidence la difficulté d'appréhender le rôle du temps physique et sa prise en compte dans les programmes temps réel. Cette difficulté est plus éprouvée par les étudiants informaticiens qui ont tendance à considérer les mécanismes de gestion du temps comme étant de trop bas niveau.

Concernant les modèles , l'existence des outils est un plus mais leur utilisation reste assez limitée. La raison essentielle est la difficulté, pour l'étudiant, de choisir le modèle adéquat et de conduire l'analyse des propriétés pertinentes. Ce dernier point peut difficilement s'acquérir par auto apprentissage avec notre plate-forme. Il nécessite des séances de travaux dirigés préalables.

Conclusion

Nous avons présenté dans cet article une expérience pédagogique menée à l'Université de Nice Sophia-Antipolis dans le cadre d'enseignements d'informatique industrielle. L'objectif est de faciliter l'acquisition par les étudiants d'informatique et d'EEA des concepts et méthodes pour la mise en œuvre d'applications embarquées et/ou temps réel. L'originalité de l'approche est d'associer les aspects visuels (simulation) aux modèles et outils sous jacents (spécification et analyse) au sein d'une même plate-forme.
Les principaux bénéficiaires de cette méthode sont les étudiants auxquels l'environnement offre un accès simplifié à un ensemble d'outils et méthodes. Cet ensemble facilite la compréhension globale de la chaîne de conception et son application à des cas concrets.
A côté de cet aspects « boîte à outils », l'utilisation de cet environnement cultive les qualités d'autonomie et d'initiative des étudiants (possibilité d'utilisation personnelle) tout en les sensibilisant aux avantages de la réutilisation d'existants. Les interfaces graphiques apportent un plus grand confort d'utilisation. Enfin l'aspect ludique de certaines maquettes n'est pas à négliger.
Du point de vue des enseignants, un avantage est de pouvoir proposer aux étudiants des projets significatifs avec des chances raisonnables de succès. Une deuxième retombée est de pouvoir multiplier le nombre et la diversité des maquettes à moindre coût, sans crainte pour l'intégrité du matériel et des personnes.
Ce choix de formation s'appuyant sur la simulation permet également de se placer dans des situations de pannes, voire même de catastrophes, sans pour autant en avoir les inconvénients.
En effet une vanne de lave linge qui se bloque (en simulation) conduit à un débordement (virtuel) qui ne nécessite pas le recours à des serpillières ...
Un dernier point à souligner est la possibilité, grâce au caractère extensible de la plate-forme, de pouvoir intégrer ou substituer de nouveaux modèles et générateurs. Cela devrait favoriser les échanges d'expériences entre établissements.
Le domaine des systèmes embarqués et temps réel se prête bien à ce type d'intégration. Notre souhait est que cette expérience puisse être adaptée à d'autres domaines.

Références

[ADB97] C. André, M. Bourdellès et S. Dissoubray. SyncCharts. Esterel : Un environnement graphique pour la spécification et la programmation d'applications réactives complexes. Actes Génie Logiciel 97, numéro 46, Paris, Décembre 1997

[APR97] C. André, M.-A. Peraldi et J.-P. Rigault. Introducing the Synchronous Approach into a Real-Time Course. Real Time Education Workshop 97 RTAS'97 Montréal (C), June 1997, published in Real Time Systems Education II, Mossé, Zalewski Ed. IEEE, pp 104-109.

[ATG] Projet INRIA- MEIJE A Quick Introduction to Auto/Graph.
www.inria.fr/meije/verification/quick-guide.html

[BG92] G. Berry, G. Gonthier. The Esterel. Synchronous Programming Language: Design, Semantics, Implementation Science of Computer Programming vol. 19, n°2, pp 87-152, 1992. www.esterel.org

[BOU98] A.Bouali. Xeve: an Esterel Verification Environment. Int'l Conference on Computer-Aided Verification (CAV'98), June/July 1998, Vancouver, BC Canada. Also available as a technical report INRIA RT-214, 1997.

[GA96] D. Gaffé et C. André. Grafcet et environnements synchrones**.** In Conf. Modélisation des Systèmes Réactifs, Afcet, pages 71--77, Brest (F), Mars1996.

[LL95] C. Lewerentz and T. Lindner. Formal Development of Reactive Systems. Volume 891 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, 1995.

[SS92] E. M. Sentovich, K. J. Singh and al. SIS : A System for Sequential Circuit Synthesis. Memorandum N° UCB/ERL M92/41 Berkeley CA. May 92 wwwcad.eecs.berkeley.edu/Respep/Research/sis/abstract.html