R A P P O R T

RAPPORT DES/184 f

CLAIRE, UN OUTIL DE SIMULATION EVENEMENTIELLE POUR LE TEST DES LOGICIELS

RAGUIDEAU J., SCHOEN D.,
HENRY J.Y., BOULC'H J.

cea INSTITUT DE PROTECTION ET DE SURETE NUCLEAIRE
DEPARTEMENT D'EVALUATION DE SURETE

RAPPORT DES/184 f

CLAIRE, UN OUTIL DE SIMULATION EVENEMENTIELLE POUR LE TEST DES LOGICIELS

RAGUIDEAU J.[*], SCHOEN D.[*],
HENRY J.Y.[**], BOULC'H J.[**]

Specialists' Meeting on
"Advanced Control and Instrumentation Systems
in Nuclear Power Plants :
Design, Verification and Validation"
(IAEA, IWG-NPPCI, IWG-ATWR)
ESPOO (FINLAND), 20-23 juin 1994

[*] LETI, DEIN
[**] DES/SAMS

Juin 1994

CLAIRE
Un outil de simulation événementielle pour le test des logiciels

Jacques RAGUIDEAU
Dominique SCHOEN

Commissariat à l'Energie Atomique
LETI (CEA -Technologies avancées)
DEIN - CE/S F91191 Gif sur Yvette Cedex

Jean-Yves HENRY
Jacques BOULC'H

Institut de Protection et de Sûreté Nucléaire
Département d'évaluation de la sûreté
CEA CENFAR
B.P. 6 - 92265 Fontenay-aux-roses

RESUME : CLAIRE fournit un environnement purement logiciel permettant de valider des applications temps-réel distribuées, soit au niveau des spécifications, soit au niveau du code.
L'outil offre les facilités graphiques de modélisation de l'application et de son environnement; il effectue de façon très efficace la simulation du modèle entré et assure le contrôle de l'évolution des données dynamiquement ou en temps différé.

1 - INTRODUCTION

L'outil CLAIRE a été initialement réalisé pour répondre aux besoins de l'Institut de Protection et de Sûreté Nucléaire (IPSN) dont le rôle est d'effectuer les analyses de sûreté qui permettent d'évaluer les dispositions prises par les exploitants d'installations nucléaires.
Dans ce cadre, l'IPSN est amené à réaliser des études de dossiers pour ces installations. Certains systèmes ont un degré de criticité tel qu'il est apparu nécessaire de disposer d'outils de simulation devant permettre l'exécution de tests spécifiques sur les logiciels qu'ils intègrent.

Le besoin était de disposer d'un outil permettant de décrire l'environnement de logiciels afin d'évaluer leur comportement en réponse à des entrées normales ou à des fautes. L'outil devait permettre la simulation d'applications temps-réel distribuées sans altération de leur comportement temporel.
La multiplicité et la complexité des environnements ainsi que la souplesse souhaitée rendant difficile voire impossible toute simulation matérielle, le choix d'une simulation purement logicielle s'est imposé.

Ce besoin de modélisation et de simulation relève en fait d'un souci commun à tous les réalisateurs de logiciels complexes: comment dépasser le stade des tests unitaires, où les débogueurs peuvent suffire, pour valider une application temps-réel dans son ensemble?

Cette interrogation devient tout à fait cruciale pour les logiciels de "sûreté" dont les défaillances peuvent entraîner des situations catastrophiques.
Les phases de spécification de ces logiciels particulièrement critiques bénéficient déjà de formalismes permettant d'effectuer des vérifications sur ces spécifications: les réseaux de Pétri, les langages synchrones apportent chacuns des réponses dans le domaine qui leur est propre: vérifications temporelles, vérifications de propriété de sûreté.
Outre le fait que ces formalismes ne sont pas toujours applicables à la taille et à la complexité des applications, ils ne permettent pas une validation sur le code réellement implanté dans ces installations critiques.
CLAIRE permet à la fois d'effectuer des vérifications de propriétés de sûreté à partir d'une modélisation de spécifications mais également de valider le code final en simulant le comportement de ses interfaces matérielles et logicielles. Il permet d'observer et de commander dynamiquement le déroulement de la simulation, mais également d'enregistrer les évolutions des entités simulées pour les analyser après coup.

2 - PRINCIPES DE L'OUTIL

Les principes sur lequels CLAIRE est basé sont la modélisation par flots de données et la simulation événementielle de cette modélisation. Le choix "événementiel" est lié à la validation de code: la production d'un événement constitue la réaction associée aux passages du code par des états attendus: valeur du PC, écriture à une adresse donnée... Ce choix s'applique sans difficulté aux validations de spécifications.

2.1 - Modélisation

Un modèle flot de données est aisément exprimable sous forme graphique. La modélisation adoptée dans CLAIRE consiste à décrire de façon hiérarchique descendante les différents éléments de l'application, selon un formalisme proche de celui de la méthode SA/RT.
Chaque entité est représentée par une boîte, chaque boîte peut se décomposer et se raffiner en boîtes de niveau inférieur. Les boîtes feuilles de la décomposition sont des procédures écrites en langage C dans lesquelles est décrit le rôle fonctionnel de la boîte.

Ces boîtes échangent des flots d'information symbolisés par des fils. Ces flots peuvent être définis comme continus ou discrets. Les modifications des flots discrets vont déclencher les procédures feuilles auxquelles ces flots aboutissent.

2.2 - Simulation événementielle

Toute modification d'un flot est associée à la création d'un événement. Les informations portées par un événement sont la variable concernée, la valeur à affecter à la variable, la date d'échéance c'est à dire l'instant auquel cette affectation devra se faire, et enfin la liste des fonctions à déclencher lors de cette affectation.
Les événements créés par les procédures sont placés dans un échéancier et ordonnés en fonction de leur date d'échéance. La simulation exploite cet échéancier, met à jour les données et active les fonctions associées aux flots déclenchants.

3 - ROLE DE L'OUTIL

CLAIRE permet d'effectuer des validations de spécifications et des validations de codes exécutable.

3.1 - Validation de spécifications

L'outil permet de réaliser des modélisations de systèmes à partir de leurs spécifications. Les éléments entrant dans la décomposition sont décrits suivant le degré de finesse nécessaire: ils peuvent être décrits comme des boîtes noires si on ne s'intéresse qu'aux interactions avec l'extérieur ou comme des boîtes blanches si on s'intéresse au fonctionnement interne.
Un élément "boîte noire" peut, lors d'une modélisation plus fine, être remplacé par sa modélisation "boîte blanche", voire à terme par son algorithme réel. Cette facilité permet de valider des applications de façon incrémentale.

Il est possible avec CLAIRE de faire des modélisations assimilables aux réseaux de Pétri et donc d'effectuer des validations sur le comportement temps réel de l'application modélisée. La transposition consiste à associer les fonctions aux transactions et les jetons aux événements: une fonction est déclenchée quand un jeton (un événement sur une donnée) arrive sur une place d'entrée de la transition qu'elle réalise. La fonction évalue ses conditions sur les jetons d'entrée et si elles sont satisfaites, la fonction absorbe les jetons, c'est à dire réinitialise ses conditions d'entrée, exécute son corps puis place des jetons dans ses places de sortie, c'est à dire crée d'autres événements qui sont les jetons d'entrée d'autres fonctions.
L'avantage de cette transposition réside dans la possibilité de décrire et d'éxécuter des applications très conséquentes, ce qui est difficile avec les réseaux de Pétri.

3.2 - Validation de logiciels

L'outil peut être utilisé pour tester des logiciels exécutables, distribués sur un ou plusieurs microprocesseurs. Il permet d'étudier leurs comportements dans les cas limites et d'évaluer leur résistance aux fautes.
Les boîtes décrivent alors l'environnement des logiciels à valider: interfaces matérielles ou logicielles, évolutions des paramètres de l'installation...
Des boîtes associées non plus à l'environnement mais à une stratégie de tests, peuvent être intégrées à la modélisation: observateurs chargés de détecter l'occurrence de tel ou tel événement ou oracles permettant la comparaison à un comportement de référence.

L'environnement est décrit selon la méthode proposée plus haut, le logiciel à tester donne lieu à une boîte particulière dans la description graphique et à une procédure particulière au sens de la simulation événementielle.
Les entrées/sorties de cette procédure correspondent aux échanges du microprocesseur avec son environnement. La procédure est réalisée par un simulateur de microprocesseur qui reçoit en paramètres le code à exécuter et une description de la correspondance entre les échanges et les événements internes.

L'outil peut même dans ce cas être utilisé comme un débogueur de code puisqu'il permet d'accéder aux différents registres et valeurs de la mémoire interne du microprocesseur.

4 - COMPOSANTES DE L'OUTIL

CLAIRE se compose des éléments suivants: un éditeur graphique, un générateur de code, un noyau de simulation, un analyseur de résultats.

L'*éditeur graphique* permet la manipulation des boîtes et fils de la modélisation et assure la vérification de la cohérence des informations entrées.

Le *générateur de code* analyse la description graphique entrée par l'opérateur. Un formalisme de dénomination des boîtes permet au générateur d'identifier les boîtes "modules" auxquelles seront associés les fichiers source C qui sont les unités de compilation.
Le générateur de code exploite les flots de données décrits dans le graphique pour créer automatiquement les parties déclaratives de chaque module. Il génère les déclarations de procédures et insère le code C associé entré pour chaque boîte terminale dans la description graphique.
Ces modules sont compilés. L'édition de liens qui crée le module exécutable de la simulation réunit les modules générés associés à l'application et les modules composant le noyau de simulation.

Le *noyau de simulation* exploite un échéancier contenant des événements datés. Ces événements sont produits par les procédures lors des modifications, immédiates ou différées, des valeurs des flots. Ils sont insérés dans l'échéancier à une place correspondant à la date à laquelle ils doivent être exécutés.
A la date prévue, le noyau de simulation affecte aux flots les valeurs portées par les événements, provoque l'exécution des procédures associées aux flots déclenchants et mémorise la nouvelle valeur du flot.

L'*analyseur de résultats* facilite l'exploitation des fichiers souvent très volumineux contenant l'évolution des flots au cours de la simulation.
Cette analyse statique des résultats complète l'analyse dynamique effectuée par les observateurs qui peuvent, pendant la simulation, détecter des situations programmées. L'analyseur de résultats permet, lui, de naviguer a posteriori dans les événements issus de la simulation pour les analyser plus finement,et éventuellement détecter des situations non voulues.
L'analyseur offre des facilités graphiques de tracé de courbes, de chronogrammes ou de tableaux de valeurs; des possibilités de zoom, de modifications d'échelle, de repérages entre courbes sont également offertes.

5 - PROGRAMMATION DU MODELE

La programmation consiste à décrire le fonctionnement des boîtes feuilles de la description graphique; seule la partie exécutive des procédures est à effectuer par le programmeur: la partie déclarative est prise en compte par le générateur de code.
Le langage C a été choisi comme langage de programmation pour les procédures modélisant l'environnement. C'est également le langage de développement du noyau de simulation ce qui favorise l'intégration du tout.

Le choix du langage C résulte bien sûr de sa large diffusion mais aussi de ses caractéristiques pour traiter le problème : compilation séparée, portée des variables, structuration des données, manipulation des fonctions qui permet d'associer une adresse de fonction à un événement et donc d'avoir une simulation performante.

De plus, en disposant du langage C pour développer ses procédures, l'utilisateur dispose de toute la puissance de ce langage et de toutes ses possibilités d'interface avec des produits existants: il peut faire appel à toutes les procédures de la "run time library", utiliser XWINDOW pour animer sa simulation...

Le développement du modèle qui constituera la simulation ne nécessite donc l'apprentissage d'aucun langage spécifique dédié au test: il est à la portée de tout développeur C.

6 - APPLICATIONS DE L'OUTIL

6.1 - Expérience d'évaluation des logiciels des systèmes classés 1E

6.1.1- Présentation de la démarche d'évaluation

Le processus d'autorisation de fonctionnement des centrales nucléaires comprend des étapes obligatoires qui donnent lieu notamment à un examen détaillé du contrôle-commande.

Cet examen prend en compte les aspects liés aux technologies (circuits intégrés, logiciels) qui ont été choisies par le fabricant pour les systèmes programmés qui assurent des fonctions classées de sûreté.

L'appui technique (IPSN) de l'autorité de sûreté (DSIN) a pour charge de réaliser toutes les investigations qu'il juge nécessaires afin de s'assurer que les méthodes et techniques mises en oeuvre par le fabricant et l'exploitant garantissent, pour les logiciels des systèmes classés 1E, la sûreté attendue et permettent une testabilité et une maintenabilité suffisantes. Pour ce faire, il porte plus particulièrement son attention sur les aspects suivants:

- méthodes de développement des logiciels rationnelles et rigoureuses suivant un plan précis d'assurance de la qualité (documentation et code);
- règles strictes de programmation pour la production d'un logiciel testable et maintenable (code);
- tests mis en oeuvre pour assurer un taux de couverture suffisant aussi bien chez le fabricant que sur site (simulation).

L'évaluation des logiciels est réalisée en prenant en compte notamment les résultats des deux analyses suivantes:

- l'analyse des documents (conception du logiciel, procédures de qualité, maintenance),
- l'analyse dynamique à l'aide de l'atelier CLAIRE, réalisée sur le code binaire fourni par le fabricant de logiciel.

L'analyse dynamique a pour objectif de montrer le comportement du logiciel soumis:

- à des stimuli dont les valeurs sont prises parmi les conditions prévues par le domaine nominal de fonctionnement du système qui utilise le logiciel sous test (étude de consistance),
- à des stimuli dont les valeurs correspondent à des cas de dysfonctionnement du système qui utilise le logiciel sous test (étude de robustesse).

La démarche adoptée pour l'évaluation des logiciels du système de protection des Réacteurs à Eau sous Pression est présentée ci-après pour la partie qui concerne l'analyse dynamique.

6.1.2 - Etude de consistance

L'étude de consistance permet de vérifier les valeurs prises par les sorties du système (par exemple la commande d'arrêt d'urgence) lorsque les entrées prennent des valeurs choisies par l'analyste dans le domaine nominal de fonctionnement du système de protection.
L'atelier CLAIRE permet de réaliser la simulation du fonctionnement par le déroulement du programme binaire sans avoir besoin du matériel (carte de l'unité centrale, cartes périphériques...) qui est utilisé sur le site. Il permet :

- la constitution d'un environnement qui reproduit les échanges entre chaque microprocesseur et les circuits (horloge, circuit de communication, mémoires...) qui lui sont associés dans chaque unité du système de protection installé sur le site,
- l'exécution des programmes binaires des unités du système de protection par un simulateur de microprocesseurs, avec la production de fichiers spécifiques qui tracent toutes les interactions entre les microprocesseurs et leurs environnements, avec mention du temps d'exécution,
- la présentation, sous une forme synthétique (chronogramme, courbes...), des valeurs prises par les différentes variables surveillées, afin de permettre une analyse des résultats de simulation.

La reconstitution de l'environnement du programme binaire et du microprocesseur qui l'exécute est obtenue par le développement de logiciels spécifiques qui remplacent les matériels sollicités par ces programmes. Ce développement est fait en utilisant essentiellement une description graphique basée sur la méthode SA/RT.

L'exécution des programmes tient compte des valeurs des variables d'entrée données par les jeux d'essais conçus pour cette étude de consistance.

Dans un premier temps, une sélection des conditions de fonctionnement normales du système de protection sera exécutée pour s'assurer de l'adéquation de la modélisation obtenue par l'environnement développé pour cette étude. Dans un second temps, des exécutions seront réalisées pour vérifier le comportement des logiciels du système lorsque celui-ci est mis en situation de fonctionnement particulière (dégradation de la logique de vote 2/4 par exemple) prévue dans la spécification.

Le système simulé et ses jeux d'essais seront réutilisés pour vérifier la non régression du bon fonctionnement de chaque version de ces logiciels.

6.1.3 - Etude de robustesse

Cette étude a pour objectif principal de juger du comportement des logiciels de l'ensemble représentatif soumis à des jeux d'essais, définis auparavant, qui représentent des dysfonctionnements du système de protection ou des systèmes lui délivrant des informations. Les jeux d'essai sont focalisés sur les composants critiques ou sensibles détectés lors des étapes précédentes. Elle met en place une analyse qui présente un aspect complémentaire aux tests réalisés par le fabriquant.

Cette étude utilise les outils de simulation décrits pour l'étude de consistance, afin de constituer un environnement plus complet permettant notamment d'atteindre certaines variables internes des logiciels qui sont représentatives de dysfonctionnements recherchés.

Les résultats des simulations obtenus avec les différents jeux d'essais pour la robustesse doivent être analysés pour identifier l'état de chaque variable de sortie des logiciels impliqués dans ces simulations.

L'analyse est poursuivie, au niveau des sorties du système, pour identifier les conséquences des dysfonctionnements introduits et d'en tirer les conclusions sur l'adéquation des comportements du système en regard des missions qu'il doit assurer.

6.2 - Validations de spécifications

Le projet ESCRIME en cours de développement au CEA consiste à évaluer les architectures de contrôle-commande des centrales nucléaires du futur: il s'agit de très gros systèmes répartis, dans lesquels les préoccupations de type temps-réel, anciennement confinées au plus bas niveau des automatismes, deviennent omniprésentes, en raison des croissances de l'automatisation du système, des contraintes d'optimisation du processus, et des exigences en matière de sûreté de fonctionnement.

L'application de contrôle-commande gère une hiérarchie d'objectifs:

- au plus haut niveau, l'objectif global s'exprime sous une forme 'externe', telle que le profil de puissance à fournir au réseau.
- cet objectif global est transformé dynamiquement en une suite d'objectifs courants (par exemple : amener la tranche dans l'état d'arrêt intermédiaire diphasique), en fonction de l'état du réacteur, de la disponibilité de ses sous-systèmes...
- l'objectif courant est finalement décliné en objectifs fonctionnels concernant les grandeurs physiques telles que la pression, la température, la réactivité... Ces objectifs sont atteints au moyen de la mise en oeuvre de sous-systèmes physiques et de boucles de régulation de bas niveau.

Les comptes rendus et messages d'erreurs remontant d'un niveau conduisent le niveau supérieur à revoir sa stratégie pour respecter ses objectifs, ou, à défaut, à faire remonter l'erreur. On en arrive ainsi à mettre en oeuvre des logiciels de prise de décision complexes.

L'évaluation d'un tel système nécessite en premier lieu un outil permettant d'évaluer son comportement dynamique, en jouant des scenarii prédéfinis ou interactifs, en observant graphiquement l'évolution de grandeurs physiques simulées.

Le système est décrit dans le formalisme CLAIRE sous forme de boîtes hiérarchisées, échangeant des flots de données et de contrôle. Les boîtes du dernier niveau de décomposition sont écrites en C, ou peuvent incorporer des codes complexes existants: dans la réalité, le contrôle-commande est bien sûr rebouclé par le processus physique; dans sa version simulée avec CLAIRE, il est rebouclé à un simulateur de process déjà existant qui est figuré par une des boîtes de la description hiérarchique.

La phase de simulation est interactive, certaines variables du système simulé étant représentées par des objets graphiques tels que potentiomètres, cadrans... Ces éléments peuvent, de plus, influencer le cours de la simulation en répercutant les actions graphiques de l'utilisateur vers les variables et contrôles du simulateur.

L'utilisateur peut choisir dynamiquement les variables représentées, afin de pouvoir traiter des applications de grande taille, comportant un grand nombre de variables.

Des observateurs peuvent être juxtaposés au système simulé, afin de détecter en ligne des situations telles que :

- blocage,
- indéterminismes,
- insuffisance de ressources CPU,
- non respect de contraintes temporelles,
- non respect de séquences d'événements.

Une mesure de couverture peut être effectuée.

Les simulations peuvent être rejouées afin de rechercher la cause d'une situation anormale, et afficher les résultats sous forme de courbes, extraction de configurations...

7 - EXTENSION

Les travaux en cours portent sur l'extension des possibilités d'interactions de l'opérateur au cours de la simulation : actuellement, le suivi dynamique du déroulement de la simulation se fait de façon textuelle. Des moyens graphiques de visualisation et de commande rendraient ce suivi plus aisé.
Il est prévu :

- d'offrir à l'opérateur de choisir le mode de représentation le plus approprié (potentiomètre,cadran,texte...) pour suivre l'évolution dynamique de ses variables,
- de modifier la valeur courante d'une variable avec ce même mode de représentation,
- d'enrichir les moyens actuels de contrôle de la simulation en prévoyant notamment des possibilités de pas à pas et de modulation de la vitesse d'exécution.

8 - DISPONIBILITE

CLAIRE est écrit en langage C et XWindow; il est disponible sur Vax/VMS et est en cours de portage sur SUN.

Les simulateurs de microprocesseurs actuellement disponibles sur VAX sont ceux des M6800, M68000, M68010, M68020, I8051, I8086.