

Méthode B

Ce document présente les principes généraux de la **méthode B** et décrit l'**Atelier B**, l'outil associé à B développé par Clearsy. Enfin, il donne des références vers d'autres media d'information.

Présentation

B pour la conception de systèmes

B est une méthode de spécification formelle qui permet, grâce à un langage adéquat, d'exprimer très rigoureusement les propriétés exigées dans un cahier des charges. Nous pouvons ensuite **prouver de manière automatisée** que ces propriétés sont non ambiguës, cohérentes et non contradictoires. Puis nous **garantissons** par **preuve mathématique** que ces propriétés sont respectées au fur et à mesure des étapes de conception.

Cette méthode et la preuve qui lui est associée permettent :

- D'obtenir des spécifications techniques et des cahiers des charges système **clairs, structurés, cohérents et sans ambiguïté,**
- De développer des **logiciels garantis contractuellement sans défaut**

dans des domaines tels que le temps réel, les automatismes industriels, les protocoles de communication, les protocoles cryptographiques, l'informatique embarquée.

B pour le développement de logiciel prouvés

Les systèmes automatiques développés aujourd'hui contiennent une proportion logicielle de plus en plus significative. Cette évolution touche particulièrement les domaines où la sûreté de fonctionnement constitue une préoccupation majeure : le transport, l'énergie, l'espace, l'avionique ou les télécommunications.

Les parties logicielles d'un système sont particulièrement vulnérables aux erreurs de conception. En effet, la moindre défaillance d'un logiciel critique de sécurité peut entraîner des pertes humaines, la destruction du système ou une perte d'intégrité des données.

B est une méthode de développement de logiciels. Sa caractéristique fondamentale est que les logiciels développés en B sont **corrects par construction**.

La méthode B est une nouvelle approche adaptée à la maîtrise des développements de logiciels critiques de sécurité. Elle est constituée des éléments suivants :

- un langage modulaire qui couvre l'intégralité du cycle de développement ;
 - Les phases de spécification, de conception et d'implémentation sont ainsi réalisées dans un cadre homogène,
- un langage d'expression des propriétés et exigences de sûreté de fonctionnement ;
 - Fiabilité, Disponibilité, Maintenabilité sont exprimées dans un même cadre,
- la démonstration mathématique que le logiciel respecte ces propriétés,
- une rigueur de développement absolument nécessaire lorsque le logiciel est de taille et de complexité importante.

La méthode B a été appliquée avec succès sur la réalisation de logiciels critiques de sécurité pour de grands projets industriels dans le domaine du ferroviaire, grâce à un ensemble d'outils logiciels d'assistance : l'**Atelier B**, développé par Clearsy.

La méthode B

Synopsis

La méthode B a été inventée par Jean-Raymond ABRIAL, à partir des travaux de E. W. DIJKSTRA et de C. A. R. HOARE. Le langage B est une évolution du langage Z, adapté à une utilisation industrielle, et à l'ensemble du cycle de développement. Le langage B est fondé sur les concepts mathématiques de la théorie des ensembles.

The B Book (de J.R. ABRIAL, paru en 1996 chez Cambridge University Press) est l'ouvrage de référence sur B. Son sous-titre "*assigning programs to meanings*" reflète le principe fondamental de B : le logiciel est réalisé à partir de sa sémantique ; il est *correct par construction*.

Un développement B débute par l'écriture d'un **modèle** reprenant tous les aspects du besoin. Les principales données manipulées par le système sont décrites (ce sont les *variables d'état*), ainsi que les propriétés fondamentales de ces données. Des services assurent les transformations de ces données tout en préservant leurs propriétés. Le modèle B obtenu constitue une spécification de ce que devra réaliser le système (le quoi).

Le modèle B est ensuite transformé (*raffiné* dans le vocabulaire B), jusqu'à obtenir une implantation logicielle complète du système (le comment). En théorie, plusieurs implantations peuvent satisfaire une spécification ; concrètement, le choix d'une implantation repose sur des critères divers comme la rapidité de traitement, la précision des calculs ou la simplicité des démonstrations mathématiques.

Utiliser B dans le développement d'un système c'est :

- lever toute ambiguïté dès l'interprétation du besoin,
- construire une spécification cohérente et conforme au besoin (le modèle),
- élaborer par étapes successives le système logiciel qui réalise la spécification.

La cohérence du modèle, puis la conformité du programme final par rapport à ce modèle sont garanties par des preuves mathématiques.

Machines Abstraites

La machine abstraite est l'élément de base d'un modèle B. C'est un concept très proche de notions bien connues en programmation, comme les modules, les classes ou les types abstraits de données.

Une machine abstraite contient des données ayant des propriétés *encapsulées*. Des services permettent d'accéder aux données et de les manipuler.

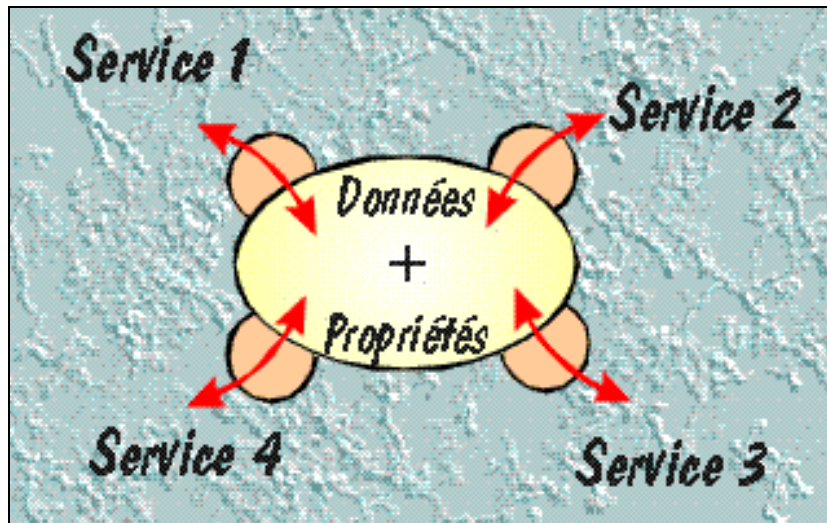


Figure 1 : la machine abstraite

Les données d'une machine abstraite (appelées *variables* dans la terminologie B) sont typées par des expressions mathématiques d'ensembles, de relations, de fonctions, de suites, etc. Les propriétés que les variables doivent vérifier continuellement, sont définies au moyen de *prédicats* et constituent l'*invariant* de la machine abstraite.

La spécification des services (appelés *opérations* dans la terminologie B) se fait dans un langage appelé *langage des substitutions généralisées*. Les substitutions généralisées sont des transformateurs de prédicats.

Par exemple, la substitution $x : \in E$ (« *x devient un élément de E* ») transforme le prédicat $(x \neq y)$ en $\forall x.(x \in E \Rightarrow x \neq y)$ (« *quelle que soit la valeur de x dans E, x est distinct de y* »).

Une opération d'une machine abstraite contient une précondition : il s'agit d'un prédicat qui exprime les conditions indispensables pour pouvoir invoquer l'opération.

Raffinements et Implantations

Le mécanisme de raffinement consiste à reformuler, par étapes successives, les variables et les opérations de la machine abstraite, de manière à aboutir finalement à un module qui constitue un programme informatique. Les étapes intermédiaires de reformulation se nomment les raffinements et le dernier niveau de raffinement s'appelle l'implantation.

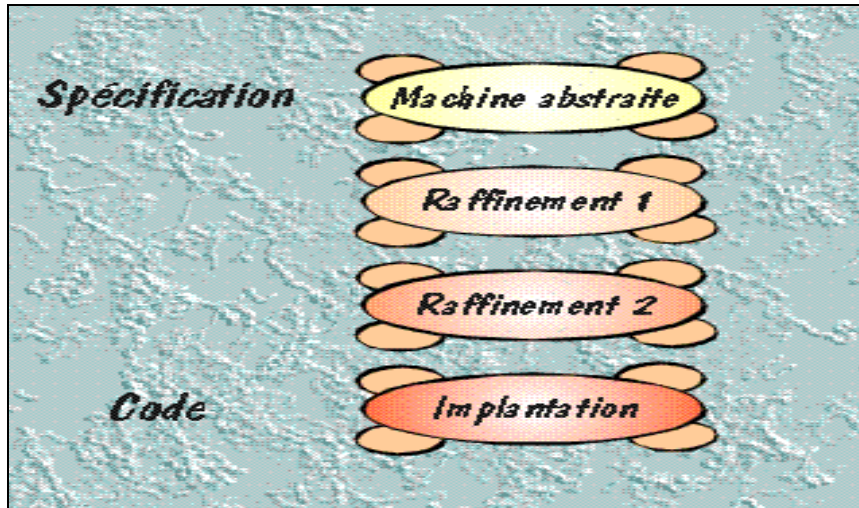


Figure 2 : mécanisme de raffinement

A chaque niveau de raffinement, des obligations de preuve représentent ce qu'il faut prouver pour garantir que le comportement d'une nouvelle opération ne contredit pas l'opération qu'elle raffine. Ainsi, après la dernière itération, et par transitivité, le code d'une implantation sera effectivement conforme aux spécifications de la machine abstraite correspondante.

La pratique du raffinement consiste à :

- transformer progressivement les structures de données abstraites (ensembles, relations, fonctions, suites) en structures données concrètes (variables scalaires, tableaux),
- lever peu à peu le niveau d'indéterminisme des substitutions,
- remplacer les substitutions abstraites (parallélisme, choix) par des substitutions concrètes (séquencement, conditions, boucles).

Projets

Un projet B est la réalisation en B d'un composant logiciel participant à un système complexe. Pour réaliser un projet B complet, on utilise le mécanisme de composition de machines abstraites et de décomposition des problèmes. Dès que le niveau de complexité d'une machine abstraite devient trop élevé, on décompose cette machine en plusieurs parties plus simples. Ainsi l'implantation d'une machine abstraite complexe s'appuie sur d'autres machines abstraites plus simples, elles-mêmes raffinables. De cette façon, un projet se construit peu à peu selon une architecture en couches.

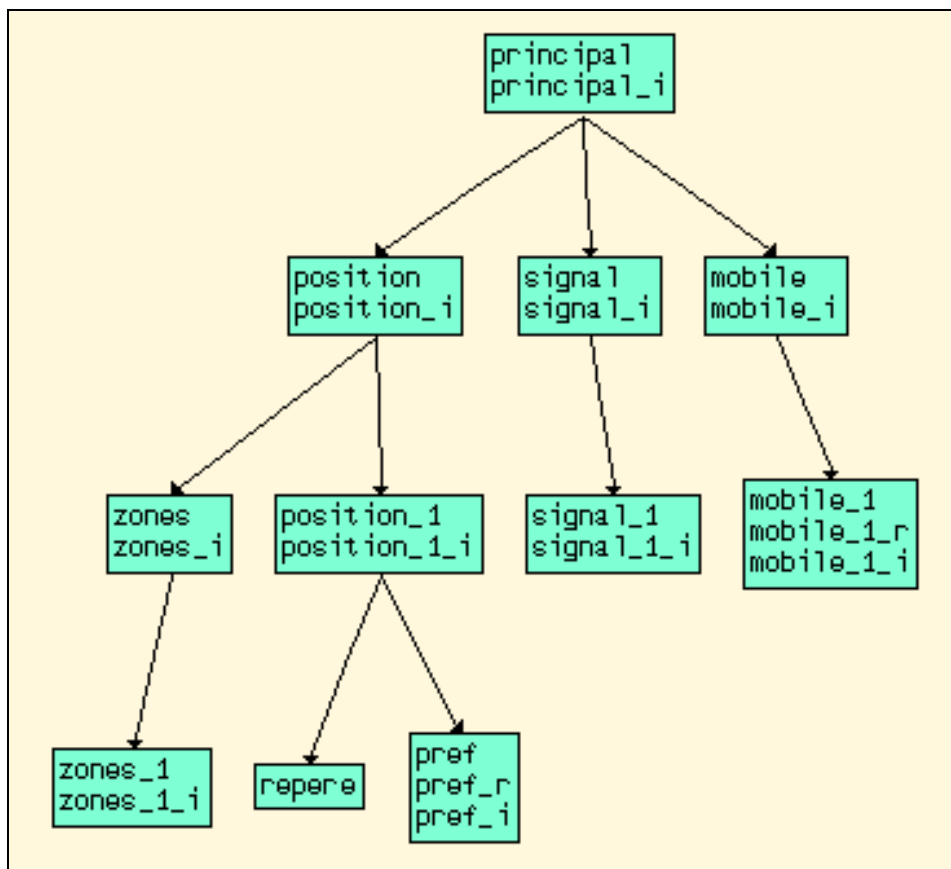


Figure 3 : un projet B

Les machines abstraites de plus bas niveau dans ce type d'architecture peuvent provenir de **librairies** (librairie standard livrée avec l'Atelier B ou librairie issue d'une réutilisation).

Le langage B

Le langage B se compose des parties suivantes :

- le langage des prédicats
- le langage des expressions
- le langage des substitutions généralisées
- le langage des composants (machines abstraites, raffinements, implantations)

Les prédicats

Les prédicats sont des formules qui permettent d'exprimer des propriétés sur des données. Les prédicats utilisés en B font partie de la logique du premier ordre.

Les prédicats sont utilisés directement dans le langage B pour exprimer des propriétés sur les données d'un composant. Par exemple, ils expriment les propriétés invariantes des variables, les préconditions sous lesquelles une opération peut être appelée (ces conditions portent particulièrement sur les paramètres d'entrée de l'opération).

Les prédicats sont également utilisés pour exprimer les obligations de preuve. Le but d'une obligation de preuve est un prédicat, il peut alors être prouvé ou réfuté. Les hypothèses sous lesquelles une obligation de preuve doit être établie sont également des prédicats.

Les prédicats du langage B sont :

1. les propositions simples (conjonction, négation, disjonction, implication, équivalence),
2. les prédicats quantifiés (universellement, existentiellement),
3. les relations entre expressions : égalité, appartenance, inclusion, comparaison arithmétique.

Les expressions

Une expression est une formule qui désigne une donnée. Toute donnée possède un type qui est l'ensemble de toutes ses valeurs possibles.

Les catégories d'expressions sont :

1. les expressions de base (désignation d'une donnée),
2. les expressions booléennes,
3. les expressions arithmétiques,
4. les couples,
5. les ensembles (ensemble vide, ensemble des entiers relatifs, des booléens...),
6. les constructions d'ensembles (ensemble des sous-ensembles, union, intersection...),
7. les relations (identité, inverse, projection, composition, itération, domaine...),
8. les fonctions (injections, surjections, bijections...),
9. les constructions de fonctions (fonctions constantes, lambda expressions...),
10. les suites (suites, permutations, concaténation, insertion, restriction...).

Les substitutions

Afin de garantir que l'appel d'une opération préserve les données et leurs propriétés, il est nécessaire de formuler la transformation effectuée par cette opération. La notion de substitution généralisée est utilisée pour construire les prédicats à prouver.

Les substitutions généralisées décrivent le comportement des opérations, au niveau des machines abstraites, des raffinements et des implantations. Les substitutions de spécifications peuvent être non-déterministes et non-exécutables, alors que les substitutions d'implantations correspondent à des instructions d'un langage de programmation impératif classique (de type Pascal, Ada, C, Modula 2, etc...).

A partir des substitutions généralisées, les obligations de preuve sont systématiquement engendrées. Par exemple, l'obligation de preuve correspondant à la conservation de l'invariant lors de l'appel d'une opération se construit en prenant

CLEARSY Société par Actions Simplifiée au Capital de 266 880 € - RCS Aix-en-Provence 433 901 402 - Code NAF 721 Z

Parc de la Duranne – 320, avenue archimède – Les Pleïades III – Bât A - 13857 AIX EN PROVENCE CEDEX 3

Tél : 04 42 37 12 70 – Fax : 04 42 37 12 71

comme hypothèse l'invariant et comme but à prouver la substitution de l'opération appliquée à l'invariant.

Les principales substitutions du langage B sont :

1. la substitution identité, la substitution bloc,
2. la substitution devient égal, la substitution devient tel que, la substitution devient élément de,
3. la substitution précondition, pour exprimer les préconditions d'appel d'une opération,
4. la substitution choix borné, la substitution SELECT,
5. les substitutions ANY et LET qui introduisent des données vérifiant certaines propriétés,
6. la substitution VAR qui introduit des variables locales,
7. les substitutions conditionnelles IF et CASE,
8. la substitution appel d'opération,
9. la substitution simultanée,
10. la substitution boucle tant que,
11. la substitution ASSERT qui introduit des propriétés pouvant faciliter la preuve.

Les composants

Dans la terminologie B, le terme *composant* est utilisé de façon générique pour représenter une machine abstraite, un raffinement ou une implantation. Les composants sont décrits par des clauses. Les principales clauses sont :

MACHINE	déclaration du nom de la machine abstraite et de la liste éventuelle de ses paramètres
REFINEMENT	déclaration du nom d'un raffinement
IMPLEMENTATION	déclaration du nom d'une implantation
REFINES	référence au composant raffiné
IMPORTS	lorsqu'une implantation importe une machine abstraite, elle peut librement utiliser ses opérations (mais pas ses données)
SEES	lorsqu'un composant voit une machine abstraite, il fait référence à cette machine et peut consulter ses données et utiliser les opérations qui ne modifient pas ses données,
INCLUDES	lorsqu'un composant inclut une machine abstraite il intègre toutes ses données
DEFINITIONS	déclaration des définitions textuelles qui seront expansées dans le composant avant toute autre analyse
CONSTRAINTS	déclaration des propriétés des paramètres formels de la machine abstraite
SETS	déclaration des ensembles abstraits et des ensembles énumérés

CONCRETE_CONSTANTS	déclaration de constantes implémentables, conservées jusqu'à l'implantation
ABSTRACT_CONSTANTS	déclaration des constantes abstraites non implémentables
PROPERTIES	déclaration des propriétés des constantes
CONCRETE_VARIABLES	déclaration des variables concrètes implémentables, conservées jusqu'à l'implantation
ABSTRACT_VARIABLES	déclaration des variables abstraites non implémentables, qui seront raffinées
INVARIANT	déclaration des propriétés des variables
INITIALISATION	initialisation des variables
OPERATIONS	déclaration des opérations sous la forme d'un en-tête et d'un corps

Exemples

Plusieurs études de cas d'application de B sont disponibles. Elles permettent d'illustrer l'apport de la méthode B sur des exemples concrets de développement de systèmes. Les cas présentés sont de taille significative et sont souvent utilisés dans la littérature pour comparer différentes méthodes de développement.

- Le contrôle d'accès à des bâtiments protégés¹,
- Le contrôle-commande d'une chaudière nucléaire²,
- Le distributeur de billets²,
- Le portrait robot¹.

¹ CD-ROM Etudes de cas : Disponible sur demande.

² Disponibles sur Internet ou fournis avec l'Atelier B.

L'Atelier B



L'Atelier B est un produit développé par Clearsy en coopération avec Jean-Raymond ABRIAL et Alstom, et avec le financement et la collaboration de la RATP, la SNCF et l'INRETS. Par leur utilisation intensive, ALSTOM et Matra Transport International ont contribué à faire progresser notablement l'ensemble des performances de l'Atelier B.

Présentation

L'**Atelier B** est un outil qui permet une utilisation opérationnelle de la méthode B. Il offre, au sein d'un environnement cohérent, de nombreuses fonctionnalités permettant de gérer des projets en langage B. Ces fonctionnalités se regroupent en quatre catégories :

- les tâches automatisables lors du développement d'un projet : les vérifications syntaxiques des composants, la génération automatique des obligations de preuve, la traduction automatique des implantations B vers les langages C ou Ada,
- une aide à la preuve, pour démontrer les obligations de preuve, grâce à des outils de preuve adaptés,
- une aide au développement : gestion automatique des dépendances entre composants B, bibliothèques réutilisables, génération de documentation et génération automatique de métriques,
- des outils de confort pour l'utilisateur : représentation graphique de projets, navigateur hypertexte pour se diriger dans un projet, affichage de l'état et des statistiques d'un projet, génération automatique du dictionnaire des termes d'un projet, archivage de projets.

L'Atelier B s'utilise soit par l'intermédiaire d'une Interface Homme Machine au format Motif, soit en utilisant directement des commandes (mode de commandes). L'Atelier B est multi-utilisateurs.

Les principales fonctionnalités de l'Atelier B

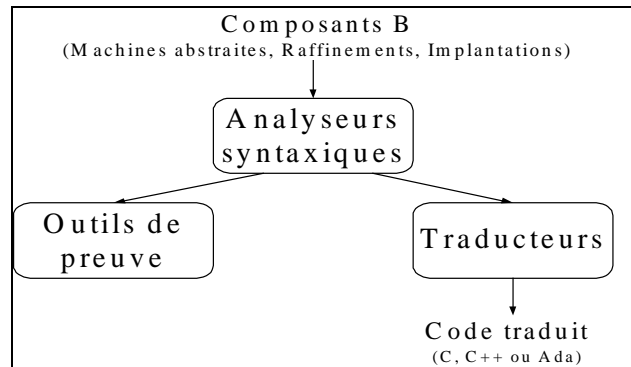


Figure 4 : atelier B

Les analyseurs syntaxiques

Ils permettent d'effectuer l'ensemble des vérifications syntaxiques sur les fichiers en langage B :

- le Type Checker effectue d'abord la vérification grammaticale d'un composant B, et ensuite un certain nombre de vérifications contextuelles dont le contrôle de type et le contrôle de portée des identificateurs. Le passage d'un composant au travers du Type Checker est un requis avant sa preuve et avant d'autres vérifications présentées ci-dessous,
- le B0 Checker effectue des vérifications spécifiques au langage B0 utilisées dans les implantations (une sous-partie du langage B) pour assurer que celles-ci peuvent être traduites. Le passage d'une implantation au travers du B0 Checker est un requis avant la traduction du composant,
- le vérificateur de projets effectue des vérifications sur l'ensemble des composants d'un projet pour contrôler son architecture (les liens entre les composants). La vérification d'un projet est un requis avant la traduction finale du projet.

Les outils de preuve

Les outils de preuve assurent les fonctionnalités suivantes :

- la génération automatique des obligations de preuve à partir des composants en langage B
- **un composant B est correct lorsque ses obligations de preuve sont démontrées**
- la preuve en mode automatique : sans intervention de l'utilisateur, la plupart des obligations de preuve sont démontrées
- la preuve en mode interactif utilisée lorsque le mode automatique a échoué : l'utilisateur guide le prouveur dans sa démonstration d'une obligation de preuve à l'aide de commandes interactives (ajout de lemmes, preuve par cas...),
- le prouveur de prédicats : démonstration des règles ajoutées par l'utilisateur
- la visualisation des obligations de preuve, leur origine et leur état (triviales, prouvées, non prouvées). Les obligations de preuve peuvent également être imprimées aux formats : Word, FrameMaker, Interleaf ou LATEX
- la gestion d'une base de règles validée comprenant plus de 2.200 règles
- la visualisation graphique de l'arbre de preuve d'une obligation de preuve

Les traducteurs automatiques

Les implantations constituent l'étape de codage d'un développement en langage B. Elles sont écrites à l'aide d'un sous-ensemble du langage B, similaire à un langage de programmation impératif. Afin de faciliter la génération de code sur n'importe quel système cible, les implantations sont traduites automatiquement en langage de programmation classique. Les programmes obtenus peuvent alors être compilés et assemblés sur la machine cible pour produire le logiciel exécutable.

Actuellement, les traducteurs en langage Ada et en langage C sont disponibles.

La représentation graphique des projets

Elle permet de représenter graphiquement les composants d'un projet et leurs liens, en les positionnant automatiquement au sein du graphe. L'utilisateur peut choisir différentes options d'affichage, comme la nature des liens à visualiser, la visualisation du graphe de dépendance complet d'un projet ou du graphe de dépendance d'un composant.

Les bibliothèques réutilisables

Les composants B peuvent être regroupés dans des bibliothèques, afin d'être réutilisés au sein de plusieurs projets.

Une bibliothèque prédéfinie est fournie avec l'**Atelier B**. Elle comprend de nombreux composants pour faciliter, par exemple, la manipulation de tableaux, de suites ou bien pour assurer l'interface vers des services d'entrées/sorties.

Le générateur de documentation

Il construit des modèles de documents (au format SGML) pour pouvoir générer automatiquement la documentation d'un développement. La documentation peut contenir les informations suivantes :

- une page de garde, un sommaire, des titres, du texte,
- des composants B en notation mathématique,
- la représentation graphique d'un projet ou d'un sous projet,
- l'état et les statistiques d'un projet (taux de preuve...),
- le dictionnaire des termes utilisés dans le projet,
- les règles rajoutées par l'utilisateur pour la preuve, en notation mathématique.

Actuellement, la documentation est produite aux formats : Word, FrameMaker, Interleaf et LATEX.

Le navigateur hypertexte

Le navigateur produit des pages hypertextes au format HTML, afin :

- d'accéder aux informations du projet (représentation graphique, dictionnaire),
- d'accéder aux informations de chaque composant (liens, variables, opérations...),
- de modifier chaque composant.

La gestion de projets

L'**Atelier B** offre des services de gestion de projets de grande taille (comprenant par exemple 500 composants). En particulier :

- l'utilisation de l'Atelier B par plusieurs utilisateurs en réseau. Ces utilisateurs peuvent travailler en même temps sur le même projet,
- d'archiver et de restaurer un projet complet,
- d'architecturer un projet en plusieurs sous-projets ou bibliothèques. L'Atelier B permet ainsi des développements importants et modulaires, par plusieurs équipes de développeurs,
- de visualiser de manière synthétique l'état d'un projet, en fournissant pour chaque composant, son état (passé au Type Checker, traduit en C ou en Ada), le nombre d'obligations de preuve et le pourcentage prouvé,
- de générer automatiquement les dépendances entre composants d'un projet. Ainsi, pour effectuer une action (passage au Type Checker, au Générateur d'obligations de preuve...) sur une sélection de composants, l'Atelier B reporte la ou les actions nécessaires sur les composants dont ils dépendent.

Pourquoi choisir l'Atelier B ?

Le projet **Atelier B** est soutenu par plusieurs industriels et donneurs d'ordres. Ses outils ont été validés par des suites de tests intensifs. La base de règles utilisée par les outils de preuve a été totalement démontrée par un comité interindustriel des experts français de la méthode B.

Les points clés en faveur de l'Atelier B sont les suivants :

- L'**Atelier B** permet le développement de projets dans un contexte industriel : plusieurs utilisateurs peuvent travailler simultanément sur un même projet, les limites intrinsèques de l'Atelier B (nombre de composants, taille des composants...) sont suffisamment élevées pour une application en contexte industriel.
- La partie sécuritaire du SAET Météor représente 107.000 lignes de code en langage B (1075 composants), génère 29.000 obligations de preuve, et 87 000 lignes de code Ada traduites.
- Les outils de preuve sont performants.
- Le pourcentage de preuves démontrées automatiquement, sans ajout de règles utilisateur, est supérieur à 80% pour les projets présentés en études de cas.

Enseignement et recherche

La méthode B a été appliquée avec succès dans l'industrie, ce qui en fait une méthode opérationnelle. Elle est également diffusée dans le domaine universitaire pour l'enseignement et la recherche. Le tableau ci-dessous donne la liste des établissements possédant des licences de l'Atelier B :

Sites utilisateurs
CRIN Nancy
IRIN
Ecole des Mines de Nancy
ENSIMAG
Université Franche Comté
Université de Rennes
CNAM Paris
CNAM Nantes
CNRS LAAS
CNAM IEE
Télécom Paris
ENSEEIH
ENSEIRB
Université de SHEERBROOKE (Canada)
Abo Akademi University (Finlande)
Université de Nantes
ENSMA
Ecole Polytechnique Lausanne
CNAM Versailles

INT
Université de Montpellier
SCSSI Enseignement
ESIAL
Université de Freiburg (Allemagne)
Université de Breme (Allemagne)
ESSI
ENSEM
ENSTB
Université d'orléans
Université de Savoie
INRIA Lorraine
UFRN (Université Brésil)
KTH (Suède)
CNAM Evry/IIE
Université de Metz
ESIL
ENSSAT
Université d'Angers
Université d'Evry
Université de la Rochelle
IRIT
Université de Compiègne
Université de Bordeaux
Loria Nancy
IUP GSI Annecy

CLEARSY Société par Actions Simplifiée au Capital de 266 880 € - RCS Aix-en-Provence 433 901 402 - Code NAF 721 Z

Parc de la Duranne – 320, avenue archimède – Les Pleiades III – Bât A - 13857 AIX EN PROVENCE CEDEX 3

Tél : 04 42 37 12 70 – Fax : 04 42 37 12 71

Monash University (Australie)
Loria Nancy
Mac Master University (Canada)
University Of Southampton(UK)
Université de Poitiers
Université du Havre
Université d'Angers (ISTIA)
IUT Vélizy - Versailles
Université de Strasbourg
Université de Paderborn (Allemagne)
Université de Belfort
Université de Cali (Colombie)
Université d'Orsay - LRI
IUP NTTIE
Université de Rouen
ENSAE
Brigham Young University (UK)
ETH (Suisse)
LSR IMAG
PRISM
TEESIDE University (UK)
Université d'Evry
Fraunhofer Institute (Allemagne)
ENSMA
SUPELEC
University of New South Wales (Australie)

ENST
Université de Limoges
ESIREM
TESS Ltd (Japon)
University of Surrey (UK)
Université d'Indonésie
National Institute of Informatics (Japon)
Zhuhai College / Jilin University (Chine)
UBI COVIHLA (Portugal)
Ecole Centrale de Lille



Contacts

Clearsy

Clearsy offre un ensemble complet de services autour de la méthode B :

- la commercialisation de l'Atelier B
- des formations à la méthode B
- le conseil
- la réalisation de développements en B
- la conduite d'études à l'aide de la méthode B
- l'aide au développement (preuve, conseil sur l'architecture)

Contacts :

Clearsy

1330, rue Guilibert Gauthier de la Lauzière

Europarc de Pichaury - Batiment C1

13856 AIX EN PROVENCE CEDEX 3 - FRANCE

Tél. : (33) 04 42 37 12 70 - Fax : (33) 04 42 37 12 71

Email : contact@clearsy.com

Site Clearsy : <http://www.clearsy.com>

Site Atelier B : <http://www.atelierb.societe.com>

Site B académique : <http://www-lsr.imaq.fr/B/>

Site B4free: <http://www.b4free.com>