Zeitschrift: Comtec : Informations- und Telekommunikationstechnologie = information and telecommunication technology

Herausgeber: Swisscom

Band: 77 (1999)

Heft: 1

Artikel: TIMS : un laboratoire TMN de poche

Autor: Eberhardt, Rolf

DOI: https://doi.org/10.5169/seals-876989

Extrait des Programmes d'exploration de Corporate Technology (6):

TIMS, un laboratoire TMN de poche

Est-il possible d'abaisser les coûts généraux d'introduction d'un système TMN et de raccourcir le processus depuis les spécifications jusqu'aux tests et à la mise en service, en investissant plus dans la phase de spécification? C'est probable! Une partie de la réponse se trouve dans le simulateur TMN du modèle d'information – TIMS, développé par Corporate Technology et par l'institut Eurécom. Avec TIMS, le concepteur peut rapidement développer des agents et des managers TMN exécutables et examiner les conflits internes. Bien que TIMS ait été prévu au départ pour être utilisé dans le domaine des TMN, il permet de concevoir n'importe quelles simulations d'objet, raison pour laquelle il ne contient pas seulement des composants Q3 mais aussi des composants pour la spécification de l'interface de CORBA. Mais TIMS offre aussi une protection des investissements car les résultats du développement du prototype peuvent être utilisés comme base pour l'acquisition ou la réalisation de séries de tests.

Le Programme d'exploration* EP97-6 «Operational Processes & Customer Care» analyse l'introduction de nouvelles technologies sur les processus «Service Order & Delivery» et «Service Assurance» du point de vue du développement des services et de l'efficience des coûts.

* Les Programmes d'exploration sont réalisés par Corporate Technology sur mandat de la Direction du groupe et font l'objet de comptes-rendus réguliers.
Ce sont des activités à moyen ou long terme, de 2 à 7 ans selon les domaines.

Il y a quatre ans, Swisscom a décidé d'introduire la technologie de réseau SDH, livrée par deux constructeurs. Sans parler de la meilleure qualité de la technologie de transmission, la gestion intégrée du réseau devait apporter des avantages significatifs dans l'exploitation de ce réseau. Nous avions émis des réflexions similaires au sujet du réseau ISDN. Dans les deux cas, c'est le concept TMN [1] – Telecommunication Manager Network – qui était prévu.

ROLF EBERHARDT, BERNE

Les interfaces entre les composants système des différents constructeurs sont la clé du succès du TMN. On ne peut connecter les composants entre eux sans trop de difficulté qu'une fois le mode de fonctionnement de ces interfaces clairement défini. Si au contraire leur définition n'est pas suffisante, le constructeur doit interpréter les prescriptions et court le risque de développer un composant incompatible. La plupart du temps, l'exploitant du réseau ne détecte cette erreur que lors du test d'intégration, ce qui peut entraîner des retards et des dépassements de coûts considérables. Il ne faut donc pas s'étonner que la gestion d'un réseau à base de TMN puisse être un réel cauchemar pour l'exploitant du réseau.

Prototyping rapide et TMN

Un laboratoire d'essai où l'on puisse simuler l'utilisation et examiner les problèmes d'intégration à peu de frais peut y remédier. Aussi bien Internet Engineering Task Force que différentes industries (p.ex. ATM Forum ou OMG) partent du principe qu'un standard ne peut être publié qu'après avoir subi de minutieux essais en laboratoire. Il existe bien des Toolkits TMN mais ils ne sont pas appropriés pour concevoir rapidement et à peu de frais des systèmes TMN. Suite à cette prise de conscience, Corporate Technology, de concert avec l'Institut Eurécom, a développé un laboratoire virtuel TMN, un simulateur de modèles d'information TMN, le TIMS. TIMS est un environnement de prototyping pour agents et managers TMN. GDMO et GRM décrivent aussi bien la structure des objets que les relations entre eux, alors que leur comportement est programmé dans un langage appelé Scheme, basé sur LISP. Grâce à des bibliothèques spéciales, le développeur CMIP ou CORBA peut construire des interactions avec d'autres composants TMN, basées sur IIOP. L'environnement de développement ainsi que le pilotage du simulateur sont entièrement intégrés dans l'éditeur de programmation emacs. Grâce à ce dernier et à l'interpréteur Scheme intégré, on peut obtenir des cycles de développement courts, comparables à un Microsoft Visual Basic. A part le Runtime Debugger, un outil de visualisation permet de représenter graphiquement la Management Information Base (MIB), la banque de données des Agents TMN, en temps réel. L'utilisateur voit où et comment les objets et leurs liaisons évoluent les uns par rapport aux autres et il peut appeler le contenu de ces objets MIB par un simple clic. Une simulation d'un Agent ou d'un Manager qui fonctionne est appelée TIMS-box. Soit la TIMS-box est pilotée par des fichiers de scénarios qui décrivent des processus définis, soit elle communique avec des systèmes externes via des interfaces (Figure 1).

Du prototype à l'implémentation, en passant par les spécifications.

Est-ce que ces arguments justifient le développement de sa propre plateforme? L'introduction du système se subdivise en trois phases, spécification, implémentation et mise en service. Comme chaque phase est généralement suivie par des experts différents, il y a une perte du savoir-faire spécifique à l'application lors du passage d'une phase à l'autre. Si l'on pouvait réutiliser facilement dans la phase suivante le savoir élaboré lors de la rédaction d'une spécification, cette dernière aurait infiniment plus de valeur (Figure 2).

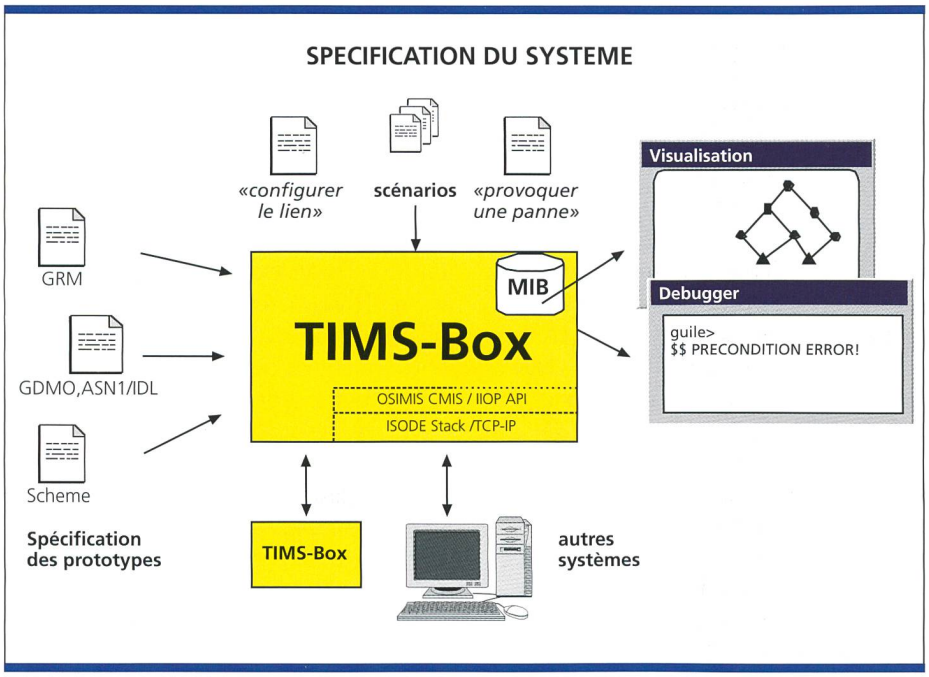


Fig. 1. Une fois la spécification des composants TMN réussie, il est possible de visualiser le comportement du système à l'aide de scénarios. Un debugger et des outils de visualisation permettent d'avoir un œil sur ce qui se passe au sein du système. TIMS peut communiquer avec d'autres systèmes via des interfaces.

TIMS devrait permettre de surmonter un grand nombre de ces obstacles. Par exemple, une spécification est automatiquement générée à partir du modèle de l'objet TIMS et du comportement de l'objet. Elle peut ensuite être utilisée comme source d'approvisionnement. De plus, son contenu est conforme à la nouvelle méthode basée sur ODP, selon la norme de l'UIT-T, ce qui améliore encore son acceptation.
Les spécifications des logiciels sont toujours criblées d'erreurs, souvent en raison d'inconsistances. Comment les détecter? En plus du code normal, le concepteur programme des règles supplémentaires qui doivent être respectées par le système en tout temps. Ces conditions cadre (pre- et post-conditions et invariants) sont constamment contrôlées en cours d'exécution. Si une spécification passe le test de validation, on peut affirmer qu'il n'y subsiste aucun conflit interne lié aux scénarios utilisés pour le test.
Équipé d'interfaces Q3 et CORBA, un prototype TIMS peut être introduit dans des architectures TMN comme configuration de référence et pour l'émulation. On peut citer comme exemple la copie de la gestion d'un réseau SDH européen.
Le factory acceptance test (FAT) et le test d'intégration de systèmes sont des défis particulièrement importants. A partir de la spécification, le spécialiste des tests développe des scénarios qui vérifient l'exactitude du produit. Les premières expériences ont montré que des suites de tests complexes peuvent être générées automatiquement à partir du modèle de l'objet et de son comportement ainsi que des scénarios. Ceci réduit considérablement le travail de préparation, ce qui permet au spécialiste des tests de se concentrer sur des tests complexes.

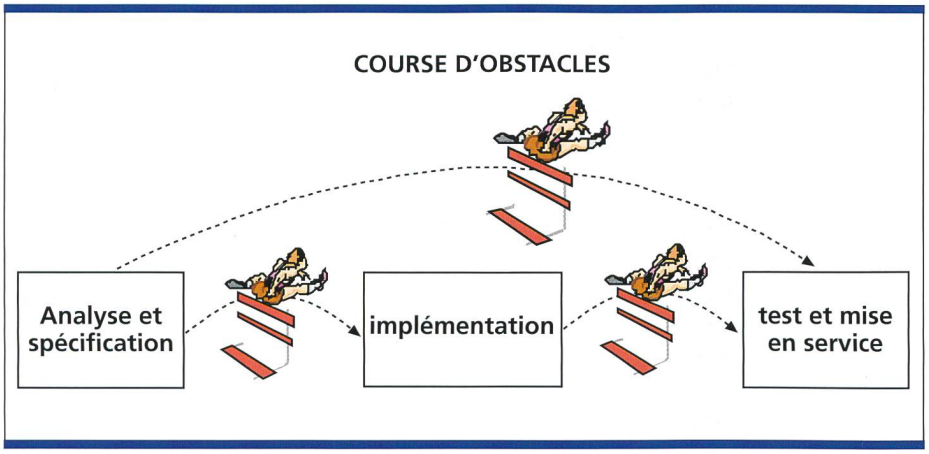


Fig. 2. Une amélioration des spécifications réduit la course d'obstacles d'une phase à l'autre.

Concevoir des TIMS boxes

Pour obtenir une simulation qui fonctionne, le concepteur doit définir deux composants: le modèle de l'objet et son comportement.
Les diagrammes d'entity-relationship servent de base pour la modélisation de modèles d'objets basés sur des données. Alors que les relations sont mieux connues dans la modélisation, dans l'implémentation elles sont souvent remplacées par des composants primitifs comme les pointeurs. Le code perd ainsi en stabilité, car sans relations, les décisions de conception concernées, comme le nombre ou le genre des objets, ne peuvent plus être contrôlées en temps réel. TIMS s'appuie sur les relations pour naviguer entre les objets. Les relations sont décrites dans le GRM [2] qui s'inspire largement du langage de définition des objets GDMO [3]. GRM propose aussi des opérateurs pour la gestion des relations et des rôles. L'opération ESTABLISH, par exemple, établit une instance de relation. Des objets peuvent ainsi être reliés à une relation existante par une opération BIND.
Le comportement (ou «behaviour») de l'objet est un véritable casse-tête pour le développement de logiciels. Les descriptions des interfaces d'application (API) décrivent bien la structure de l'interface et de l'information, mais pas comment cette dernière réagit. La plupart du temps, les développeurs se contentent d'une description en prose, par exemple: Si l'alimentation d'un équipement tombe en panne, son statut (operational state) et celui des services qui en dépendent (termination points), doivent commuter de ‹enabled› à ‹disabled›. Même le plus prudent des concepteurs ne peut pas

exclure la possibilité d'effets secondaires (side effects) dus à d'autres règles de comportement. Il pourrait par exemple y avoir à un autre endroit: En cas de panne de l'alimentation, l'équipement commute automatiquement sur l'alimentation de secours afin de provoquer un conflit. Jusqu'à présent, le comportement du système était à chaque fois assigné à un objet (le comportement de l'objet). Mais si plusieurs objets sont impliqués dans un comportement, la question se pose de savoir qui doit guider le comportement des autres objets.
Cette réponse dépend de l'implémentation du constructeur. L'approche choisie par TIMS, appelée «relationship behaviour formalisation» (RBF) [4], affecte le comportement de la relation qui lie les objets impliqués. Le problème d'affectation mentionné plus haut disparaît donc, et l'accès à une instance objet se simplifie parce qu'il devient indépendant de l'implémentation low level (p.ex. pointeur). Un nouveau comportement peut être introduit pas à pas dans le système, ce qui va profondément à l'encontre de l'esprit du Prototyping.
Un comportement est toujours provoqué par un événement. Un message, par exemple, peut servir de déclencheur. Ce message veut modifier quelque chose dans le système, par exemple modifier une valeur d'attribut, effacer un objet ou créer une relation. Pour s'assurer qu'aucun effet indésirable ne se produise, on définit un état initial et un état final (pre- et postcondition) pour chaque comportement. Pre et post sont incontournables pour la validation de la spécification. Ces conditions cadres sont également importantes en cas d'utilisation comme base d'approvisionnement, car elles décrivent le comportement du système sans spécifier l'algorithme (les constructeurs ne seraient assurément pas d'accord).

Utilisation du TIMS

Dans plusieurs études de cas, on a testé l'utilisation du TIMS pour la modélisation de modèles d'élément de réseau (SDH protection switch, V5.1) et de modèles de réseau (réseau de raccordement, Metran Xcoop). L'initiation au domaine spécialisé, la définition claire du problème TMN et l'interprétation correcte des règles de comportement ont créé la plupart des difficultés. Les pre- et postconditions se sont avérées très utiles, car les développeurs n'ont pas cessé d'ignorer des hypothèses faites auparavant.

Information d'arrière plan

Le concept TMN décrit des composants pour la commande et la surveillance de réseaux de télécommunication; il a été développé au sein de l'UIT-T. Les composants sont reliés par l'intermédiaire d'interfaces normalisées Q3 ou X. Une description normalisée d'interface précise quelle information doit être mise à disposition, ce qu'elle signifie et comment y accéder. Jusqu'à maintenant, on a utilisé pour cela le protocole CMIP sur une pile OSI, depuis peu également CORBA et EDI sur piles TCP/IP.
Il y a deux types de composants dans TMN, les agents et les managers. Les managers pilotent un système et offrent des interfaces à l'utilisateur. Les agents réceptionnent les ordres de gestion du Manager et les convertissent en ordres pour élément de réseau, où ils reçoivent des messages des éléments du réseau et les transmettent de manière appropriée au Manager.
La MIB est la banque de données des agents; elle garde en mémoire les informations pour élément de réseau; on peut y accéder via l'interface Q3.
Scheme est un langage semblable à LISP. Il se caractérise par une structure très compacte, ce qui facilite son apprentissage. Sa caractéristique dominante est son style d'écriture: alors que des langages connus comme C++ utilisent une notation infixe (p.ex. 2+3), Scheme utilise une notation préfixe (+2 3), ce qui nécessite une certaine accoutumance.

Abréviations

API	Application Programming Interface interface de programmation d'application
CMIP	Common Management Information Protocol protocole courant de gestion de l'information
CORBA	Common Object Request Broker Architecture architecture courante d'agent de requête objet
FAT	Factory Acceptance Test test d'acceptation d'usine
GDMO	Guidelines to the Definition of Managed Objects directives pour la définition d'objets gérés
GRM	General Relationship Model modèle général de relation
IIOP	Internet Interoperable Protocol protocole d'interaction Internet
UIT-T	Union Internationale des Télécommunications – Telecom
MIB	Management Information Base base d'informations de gestion
ODP	Open Distributed Processing processus distribué ouvert
OMG	Object Management Group groupe de gestion d'objet
RBF	Relationship Behavior Formalisation formalisation du comportement de relation
SDH	Synchronous Digital Hierarchy hiérarchie numérique synchrone
TIMS	TMN-Information Model Simulator simulateur de modèle d'information TMN
TMN	Telecommunications Management Network réseau de gestion des télécommunications
TTCN	Tabular Test Conformance Notation notation de conformité de test tabulaire
UML	Unified Modelling Language langage de modélisation unifié

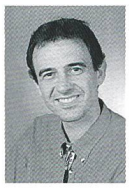
TIMS est tout à fait approprié pour développer pas à pas le comportement de modèles d'objet qui ont été préalablement décrits à l'aide d'une méthode d'ingénierie logicielle (p.ex. UML). Les relations, les conditions cadre et la possibilité de valider le comportement présentent un avantage considérable par rapport aux frameworks classiques orientés objet (p.ex. Smalltalk) pour transmettre des prescriptions claires aux constructeurs.
Contrairement à d'autres outils professionnels comme HP Openview, TIMS s'apprend beaucoup plus rapidement (< 1 mois) et il permet de réaliser rapidement des modèles (2–4 semaines pour une interface Q3 complète). En plus de la spécification très détaillée des interfaces, TIMS permet également d'élaborer des processus de gestion et des scénarios complets, comme on en rencontre parfois entre Manager et Agent. La connexion de plusieurs systèmes TIMS permet de construire des architectures TMN complètes. TIMS est particulièrement utile là où l'utilisateur a besoin de spécifications imperméables (p.ex. sur des systèmes TMN hétérogènes ou pour des normes) et lorsqu'il est nécessaire de connaître dans les détails le fonctionnement de l'interface (p.ex. pour des tests d'intégration ultérieurs). Par contre TIMS est moins approprié comme plate-forme de développement, pour des raisons de performance.

Vue d'ensemble

TIMS est conçu en deux parties: TIMS validator et TIMS emulator. Le validator contient toutes les fonctionnalités pour le prototyping et la validation des modèles d'information. Le TIMS emulator complète le paquet avec les possibilités d'interfaces Q3 et CORBA. Des informations plus détaillées sur le TIMS sont disponibles à l'adresse http://www.vptt.ch/~tims.
Les premières expériences ont permis de tester la génération automatique de suites de tests TTCN. TTCN sert à décrire des «factory acceptance tests» (FAT). Cette application pourrait s'avérer très intéressante car le développement FAT, notamment de scénarios, prend beaucoup de temps. Le développement du TIMS dans cette direction est actuellement en test. 4, 11

Références

[1] R Sellin TMN – die Basis für das Telekom-Management der Zukunft, R. von Deckers Verlag Heidelberg 1995, ISBN 3-76854294-7.
[2] ITU-T X.725 Generic Relationship Model 1994.
[3] ITU-T X.722 Guidelines to the Definition of Managed Objects 1992.
[4] R. Eberhardt S. Mazziotta D. Sidou Design and Testing of Information Models in a Virtual Environment Proceedings of the Integrated Management Symposium 1997 (IM 97) 1997.



***Rolf Eberhardt**, ing. dipl. EPF en informatique, a rejoint Swisscom Corporate Technology après ses études. Il travaille dans le domaine de la gestion du réseau de raccordement et du service, plus spécialement sur les processus end to end, donc sur tout, depuis l'événement commercial jusqu'à l'interface Q3.*
En outre, il a représenté Swisscom dans l'ETSI ainsi que dans le «network management forum» (NMF) et il a dirigé plusieurs projets Unisource OTC. En tant que Programme Manager, il est responsable depuis 1997 du programme «Operational Processes & Customer Care».

Summary

TIMS – a pocket-size TMN laboratory

Is it possible to lower the overall introduction costs of a TMN system and to snorten the elaborate process from specification to the test by investing more into the specification phase? Presumably yes. A part of the answer can be found in the TMN informat on model sinriulator TIMS developed by Corporate Technology and Institut Surecom Irl TIMS the designer can develop executable TMN agents and managers n a short time and check them for internal conflicts. Although it was originally intended for use in the TMN area any object s mulation can be built. For this reason TIMS contains not only Q3 components but also compnents for COBRA interface specification But TIMS provides also investment protection because the results of the prototype development can be used as the basis for procuring and creating test suites.

Ein am Kopf tragbarer PC

Diese Weltneuheit hat IBM Japan entwickelt. Der «Kopf-PC» (IBM nennt ihn «wearable computer») ist gerade mal so gross wie ein Stereokopfhörer, wiegt weniger als 300 Gramm und misst 8×12×2,6 cm. Darin enthalten sind die Zentraleinheit, ein integriertes Display mit Kopfhörer, ein Controller für den «Trackpoint» mit einer «Klicktaste» sowie ein winziges Mikrophon. Der unter WINDOWS 98 laufende PC arbeitet mit einem 233-MHz-MMX-Prozessor, hat eine Festplatte mit 340 MByte Speicherkapazität, 64 MByte Arbeitsspeicher und nutzt den Universalbus USB. Das Display ist 1×1 cm gross und hängt 3 cm vor dem Auge. Es löst 320×240 Pixel auf, was etwa dem Eindruck eines gängigen 26-Zoll-Displays in 2 m Abstand entspricht. Der PC soll etwa so viel kosten wie ein Notebook und wendet sich vor allem an Mechaniker in der Autowerkstatt oder in der Luftfahrtindustrie, die ständig während der Arbeit in irgendwelchen Manuals blättern müssen, um ihre Aufgabe erledigen zu können.
Das Gerät bedarf im Detail noch der Verfeinerung, soll aber in etwa einem Jahr an den Markt gehen.

IBM Japan Ltd.
2-12, Roppongi 3-chome, Minato-ku
Tokyo 106; Japan
Tel. +81-3-3586-1111
Fax +81-3-5563-4904