

Views of the Future

Coalgebraic Methods in Computer Science and Mathematics.....	2
Scientific Computing and Numerical Software	3
Networks	5
Component Based Software Engineering	7
Information Systems of the Future.....	8

Coalgebraic Methods in Computer Science and Mathematics

Picture yourself sitting behind a computer: it has an internal state, given by, e.g., the contents of its memory cells. This state is not directly observable, but the screen provides certain observable information by means of specific operations (e.g., listing the names of all files created so far). A computer is also a dynamic system: its internal state can be modified (for instance, by creating a new file).

This complex view of systems, consisting of both observations and internal dynamics, is captured by coalgebras. They were only introduced in the early 1980s as formal duals of algebras, to deal with infinite data structures, but they have been around in disguise for much longer. In fact, many coalgebraic structures have been in use for a long time in various situations in both mathematics and computer science, although usually not identified as such. Only recently have they been recognized to form the underlying structure of various kinds of dynamic systems, automata, transition systems, infinite data types, object-oriented systems, formal power series, and even various classes of differential equations.

Coalgebra thus offers a wide and rather unique unifying perspective on various parts of mathematics and computer science, comprising parts of formal logic, analysis, combinatorics, control theory, and discrete mathematics on the one hand, and algorithmics, complexity theory, object- and component-based software engineering, and automated verification, on the other. As such, the study of coalgebra leads to a better understanding of seemingly unrelated disciplines, opening unexpected possibilities for crossfertilisation between techniques stemming from different worlds.

By now, the subject has gained worldwide recognition and interest among computer scientists, logicians, and mathematicians alike. CWI has been one of the first to put this exciting and promising field of fundamental research on its agenda, and is playing a leading role in its further development.

The highly interdisciplinary character of the study of coalgebra is illustrated by the various ways in which CWI cooperates both internally and externally on coalgebra and its applications. Among others, this involves the use of coalgebra for the supervisory control of discrete event systems, as well as applications of coalgebraic techniques for reasoning about programming logics and (secure) communication protocols.

Scientific Computing and Numerical Software

Scientific computing has become a major branch in many scientific, engineering and technological disciplines. The main output of scientific computing is numerical software, which is central to our industrialized, technological society. Numerical software lies at the heart of computer simulation and computer aided design technology.

For example, starting from differential equations based on basic laws such as Ohm's law and Kirchhoff's law, numerical software is used to design larger and faster electronic circuits for consumer goods like mobile phones, compact-disc players and personal computers. Starting from differential equations from structural mechanics, numerical software is used to design new infrastructure such as bridges, harbours, tunnels, dikes and waterways. Numerical software is used in the financial market to analyse future options, in the chemical industry to optimize plants and refineries, in the shipbuilding industry to design faster sailing ships. It is used around the world for weather forecasting, climate modelling and for analysing the threat of global air pollution. In more recent years scientific computing has entered research in biology and medicine. There is a common expectation that scientific computing and numerical software will play an important role to provide the understanding necessary for solving great challenges in medicine, like the treatment of cancer and heart diseases.

The development of user-oriented numerical software has begun end of the 1960s, at the time when the first time-sharing mainframe computers were installed in academic and research laboratories. Since then, research in leading-edge mathematical modelling, numerical algorithms and software, have brought us faster and faster algorithms. Research in computer hardware has brought us more and more processor speed and central memory. With this performance at the desktop, mathematicians, computational scientists, engineers and software developers have more computer power on hand than offered by the huge mainframe supercomputers of the Cray line of just a few years ago.

With respect to applications the trend is towards really novel, large-scale, highly complex problems. An example of such a novel application is a model for the whole human cardiovascular system. Another example, from cell biology, is the Silicon Cell, which is to become a computer model that 'computes life at the cellular level': a project of such computational complexity that a consortium of researchers in the Amsterdam area from cell biology, mathematics and computer science is preparing a decade-long research program on this subject. To keep up with demands from these challenging life-science applications, a substantial amount of novel mathematics and computer science must be developed.

With respect to software design, the future trend is towards making available more diagnostic tools for testing and evaluation, validation and verification. Numerical software packages have reached a certain level of maturity, but as a rule they are still mainly workable for experts with a broad numerical experience. This delays their use and distribution into new areas. Therefore, improvements are necessary to relieve users from technical intricacies. Encouraging new tools for this will come, among others, from software engineering, computational steering and visualization. These tools will further enhance the widespread use of numerical software.

There is no doubt that for industrialized countries like the Netherlands, the dependence on scientific computing of technological and scientific development is considerable and will further increase for many

years to come. This dependence calls for a strong and long lasting research effort. Because scientific computing can be seen as part of applied and numerical mathematics, of the computational sciences and computer science in general, developments in all these areas are relevant. In particular the complete spectrum from mathematical modelling, algorithm design and analysis to software implementation will be extremely important.

Networks

Traditionally, mathematical and computer science techniques have been applied to 'technical' networks, like telecommunication and railway networks. They are centrally organized, and all users have the same objective, or at least, act follow agreed rules, thus allowing the use of classical methods of operations research, such as linear programming and queueing theory. This has led to a good understanding of basic issues in network analysis like capacity, reliability, strength, congestion, and performance of networks and of steering and controlling operations like flows and circulations in networks.

Also distributed networks, which are not centrally organized but in which the actors yet act following commonly agreed rules, like computer and traffic networks, have been studied to some extent. The actors (computer users, car drivers, etc.) utilize a common network, each with their own objectives, but obeying accepted rules or protocols (generally). Here the methods of systems and control theory apply.

In both cases, a number of methods have been developed as a result of problems in actual networks.

The study of communication networks is highly relevant to new forms of communication, which often use the same network but with different types of messages: telephone, data, images, video, fax, etc., each with their own characteristics and priorities.

An important new aspect of traffic, communication, and railway networks is that several parties that use the same network (like internet or mobile phone providers, railway companies, car drivers) compete for its capacity, and that capacity assignment and priorities are controlled by new mechanisms like auctioning (train-line frequencies), pooling (mobile phone antennas), or market effects. This requires the development of new techniques in mathematical economics, game theory, and control theory. Also, software agents offer important new possibilities in capacity assignment and market effects. Apart from the design of the system, the investigation of emergent behaviour by simulation is relevant here. Thus, techniques from computer science are required, especially in systems of software agents, computational economics, and simulation of network systems by social learning techniques.

Then there are the transportation and logistics networks, with wide unexplored areas of problems and applications of network techniques, where different types of actors participate (transportation firms, factories, distribution centers, retail trade etc.), each type consisting of different, often competing actors.

In several other sectors of society too, networking is technologically underdeveloped: the health sector, agriculture, education, etc. A prominent initial issue is often that the network still has to be created: it contains psychological, cultural, and social components, in order to make the actors realize that the network, including its conventions and protocols, is in their interest. This can be approached by (new) techniques like evolutionary simulation, to address the social, economical, and psychological roles of actors in society, especially to address the social interactions and market properties in a society.

These issues merit attention, if only because of their potentially profound impact on society. CWI offers a highly integrated research environment, where experts in all disciplines can be found to actively work together on operations research, queueing theory, performance analysis, stochastics, network theory (including multicommodity flows in networks), optimization (linear, integer, and constraint programming),

algorithmics, systems and control, agent-based computational economics (ACE) and social learning (evolutionary simulation), computational intelligence (evolutionary, neural; for a.o. classification, optimization, negotiation, auctions, market mechanisms), electronic markets and e-business and applied algorithmics.

Component Based Software Engineering

The complexity of software systems sometimes lingers on the border of human comprehension and often hinders their timely development, effective deployment, proper maintenance, and incremental evolution. On the other hand, because complex software systems involve heavy investments of human and financial resources, they are likely to be critical to the missions of the institutions where they are deployed for very long periods of time. The longevity and significance of such software as mission-critical systems makes their change through maintenance and evolution their inescapable fate, and this intensifies the problem of coping with their complexity throughout their long lives.

A sensible approach to reducing this complexity to manageable levels is the application of a technique that has already been successfully deployed in other, older production environments: building systems out of components. Component-Based Software Engineering (CBSE) advocates this sensible approach. Unlike other programming abstractions, such as objects, software components are expected to accommodate certain non-functional requirements, such as commercial viability, and must be built 'defensively' with certain minimum assumptions about their deployment environments. They are expected to be deployed and used only in binary form in a language-independent (and sometimes even platform-independent) manner.

Component-based software architectures rationally postulate two types of code: the code that comprises individual software components, i.e., component code; and the code that composes a set of individual components into a coherent system, i.e., the glue code. Variants of the so-called scripting languages have been proposed and used with some success as glue-code languages. Regardless of what (programming, scripting, etc.) language is used to write the glue code, the success of the component-based methodology requires the semantics of the glue code to be independent of the internal semantics of the components it composes into a system. This semantic independence requires an interface layer where the externally observable semantics of a component can be precisely described independently of its irrelevant internal semantics.

The existing technologies that offer component-based software engineering, are generally restricted to syntactic interfaces for components (e.g., CORBA and COM) and furthermore, some (e.g., Jini) are language specific. The goal of our ongoing experimental work is to forge out a practically useful language called *Stream* (rheo means stream). *Stream* is a language for composition of mobile components using channels as their primitive connectors. Simpler connectors can be combined using the compositional operators in *Stream* to build more complex ones. The semantics of connectors in *Stream* are independent of the semantics of the components they connect. This enhances modularity of software systems, allows separate, independent verification of connectors and components, and facilitates reuse of connectors as well as components.

The work done on CBSE at CWI builds upon and extends our extensive earlier work on Coordination models and languages. It emphasizes the semantic independence of the components and the glue code, and focuses on compositional models and languages for developing the glue code. We have already developed a formal model for component-based systems, together with a formal-logic-based component interface description language that conveys the observable semantics of components. We have presented a formal system for deriving the semantics of a composite system out of the semantics of its constituent components, and the conditions under which this derivation system is sound and complete.

Information Systems of the Future

A challenge for governmental decision makers is to chart the contours of the market and socio-economic structures 6-15 years ahead. Given the current short product life cycles in ICT, such time-frames are far beyond the business horizon, because they reflect a generation of technology and services. A discussion framed in the context of current software and middle-ware problems therefore leads towards a rationalization of past decisions, rather than a technological outlook or challenge to enter new markets. A possible way out of this dilemma is to identify the boundary conditions and challenges software technology may encounter in a rapidly evolving field. Once identified, we may be able to pro-actively act in terms of concrete stimulus of small and large projects in light of their potential of opening new frontiers of IT.

As a small case study, we take three steps into the future. Through these futuristic examples, we can chart the contours of possible activities and how they relate to the near future software development activities. Likewise, they encapsulate the contours of possible product/market combinations.

The ingredients to realize the visions are readily available. The hardware can be produced at low cost, power can be obtained from batteries or solar cells, the processing capabilities are readily available. Java has become a platform-independent framework to describe algorithms, Jini technology provides a basis to distribute and publicize services, agent technology is trying to identify and re-use effective learning technology.

However, the competition in the market is such that no one company is able to research and develop the complete infrastructure needed to bring the visions about. Rather, a cooperation between companies and public-sector research centres is necessary if they are ever to be realized. This would take the form of a nationally recognized expertise centre with strong ties to selected industries and research groups (satellites). In terms of our visions, a company like Philips might take the steering and monitoring role, while several public (EU) labs develop (partial) solutions. The outcome is not necessarily a product, but a software/technology base that has intimate experience in the key software problems: scalability, autonomy, self-repair and -adaptation.

Research at CWI is strongly focused on such interaction with the private sector, both as a participant in contract research, and as a project partner in many public-private EU projects.

2005 - An ambient climate control system

19:00 the garden sprinkler installation collects the temperature and humidity levels of the thermometer pods distributed in the garden by wireless access. It contacts the local weather forecast site and obtains the weather prediction. It generates a sprinkling scheme and sends it to the respective sprinkler installations, which realize the plan autonomously using up-to-date information.

20:00 the lawn sprinkler starts its 40-minute rain program, but is interrupted by the dog, who disconnects the water hose. The sprinkler detects this and sends a wireless message to the house to warn the gardener. He being out, the message is forwarded to his car, from where he instructs the sprinkler system to suspend the program until he returns home by 22:00.

This reply is also picked up by the home air conditioning system, which adjusts its plans to have lowered the temperature to 18 °C by 22:00.

22:15 the owner reconnects the water hose, and the sprinkler continues the plan taking into account the actual soil humidity reached before it was interrupted. The gardener installs some new thermometer pods of the latest model. Upon activation they gracefully integrate with the system, sharing historical data about the premises.

2010 - Ubiquitous video at your fingertips

There are 1000 TV channels available from various broadcasters in the Netherlands; moreover, all public video archives (Nederlands Audiovisueel Archief, Museums, libraries) are digitized and stored on large public servers. However, more channels lead to more confusion. There is no time to see everything and no time to find what one is really interested in.

Everyone has various mobile devices capable to receive and display any type of audiovisual data format for entertainment (see a movie in the train), professional use (video conference in the car) or educational purposes (training video, university lecture).

How will technology provide solutions? Every home has a digital vault for keeping data from TV and radio broadcasts, PCs, digital camera and Internet. The user does not need to be aware of the existence of the vault: the different devices interface seamlessly with it.

The vault is connected to an intelligent module which filters, indexes and organizes the data automatically based on metadata from service providers, devices, the user and the information contained in the data itself, using algorithms for image/video/ audio/speech/text analysis and natural language processing

Service providers will also provide general metadata, such as genre information, short textual summary, boundaries of informative video segments, such as news reports and topic units in a documentary and topics of these segments. More specific metadata comes from the user. This allows the extraction of (parts of) different programs interesting to the user. Here the development of content models describing the video segments of interest are required and the extraction of suitable effective features is necessary for detecting highlights.

2020 - PolyTabloids

6:30 Mark is awakened by his clockradio. As soon as he sets his feet on the floor, the sleeping room control center warns the coffee machine downstairs to start preparing a nice espresso. Still half asleep, Mark picks up the PolyTabloid, an A3-sheet of electrified material, from its printer.

With the coffee near at hand, he sits at the kitchen table. As he touches the Poly, pieces of text appear. Upon seeing a photo of a riot last night against a gasoline station in Chicago, he draws a circle on the image with his finger and calls up an enlargement. The Poly interacts with the WallDisplay, telling it to show the news-broadcast associated with the news item. In the basement, the newsflash is located by the Media Server among the hundreds of news casts collected last night. It is beamed up and the electronic copy of Van Gogh's Sunflowers is replaced by the inferno of a burning gas station.

Still unsure about the reason for this attack, he issues a search for related news items by pointing out the words on the Poly. His Media Search Engine retrieves a few more documents from his home archive, giving precedence to information already privately classified, or related to his work as logistics manager.

10:00 Mark enters the board room with his PolyTabloid filled with relevant information on the incident, including a 3D model for a quick-replacement station. The PolyTabloid with the dossier is taken by a secretary for authorized inclusion in the company's media archive. After inspection of the latest news bulletins it becomes clear that there is no relationship with the riot crowd and the well-guarded brand name of the oil company.

It doesn't take the board long to decide that a video message to its personnel is warranted and budget is freed to take this opportunity to create a new station. External communication quickly produces the necessary message using the fast local media based and those directly accessible from the agents, the message is packaged for multi-channel delivery, with the appropriate visual and audio packaging.

11:00 All field operators receive a message on their wristwatch. Its polymer top shows a short video message of the director. The office workers at the same time receive the high-volume video message on their laptops or PolyTabloids.