JOHN WILEY & SONS, LTD., THE ATRIUM, SOUTHERN GATE, CHICHESTER P019 8SQ, UK

*** PROOF OF YOUR ARTICLE ATTACHED, PLEASE READ CAREFULLY ***

After receipt of your corrections your article will be published initially within the online version of the journal.

PLEASE NOTE THAT THE PROMPT RETURN OF YOUR PROOF CORRECTIONS WILL ENSURE THAT THERE ARE NO UNNECESSARY DELAYS IN THE PUBLICATION OF YOUR ARTICLE

☐ **READ PROOFS CAREFULLY**

ONCE PUBLISHED ONLINE OR IN PRINT IT IS NOT POSSIBLE TO MAKE ANY FURTHER CORRECTIONS TO YOUR ARTICLE

- This will be your only chance to correct your proof
- Please note that the volume and page numbers shown on the proofs are for position only

☐ **ANSWER ALL QUERIES ON PROOFS** (Queries are attached as the last page of your proof.)

- List all corrections and send back via e-mail to the production contact as detailed in the covering e-mail, or mark all corrections directly on the proofs and send the scanned copy via e-mail. Please do not send corrections by fax or post

☐ **CHECK FIGURES AND TABLES CAREFULLY**

- Check sizes, numbering, and orientation of figures
- All images in the PDF are downsampled (reduced to lower resolution and file size) to facilitate Internet delivery. These images will appear at higher resolution and sharpness in the printed article
- Review figure legends to ensure that they are complete
- Check all tables. Review layout, titles, and footnotes

☐ **COMPLETE COPYRIGHT TRANSFER AGREEMENT (CTA) if you have not already signed one**

- Please send a scanned signed copy with your proofs by e-mail. **Your article cannot be published unless we have received the signed CTA**

☐ **AUTHOR SERVICES**

- If you have registered this article in Wiley-Blackwell Author Services, the article's status will be updated shortly after you have returned your proof corrections (you will also receive an e-mail alert if you have opted to receive them). **You are entitled to free access to the PDF from Author Services when your article is published online.** This free access is considered your PDF offprint, and you will only have access from within Author Services; you will not be sent a PDF. You may also nominate up to 10 colleagues for free access. All accesses from Author Services count towards the usage of your article. For options to order print copies or additional electronic access, please see below.

☐ **OFFPRINTS**

- Free access to the final PDF offprint of your article will be available via Author Services only. Please therefore sign up for Author Services if you would like to access your article PDF offprint and enjoy the many other benefits the service offers.

Additional reprint and journal issue purchases

- Should you wish to purchase additional copies of your article, please click on the link and follow the instructions provided: http://offprint.cosprinters.com/cos/bw/
- Corresponding authors are invited to inform their co-authors of the reprint options available.
- Please note that regardless of the form in which they are acquired, reprints should not be resold, nor further disseminated in electronic or print form, nor deployed in part or in whole in any marketing, promotional or educational contexts without authorization from Wiley. Permissions requests should be directed to mailto: permissionsuk@wiley.com
- For information about 'Pay-Per-View and Article Select' click on the following link: http://www3.interscience.wiley.com/aboutus/ppv-articleselect.html

SPE 1006
pp: 1–10 (col.fig.: Nil)
PROD. TYPE: COM
ED: Arun
PAGN: Sabari – SCAN:

SOFTWARE – PRACTICE AND EXPERIENCE
Softw. Pract. Exper. (2010)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/spe.1006

Developing UPPAAL over 15 years

Gerd Behrmann[1], Alexandre David[2,*,†], Kim Guldstrand Larsen[2], Paul Pettersson[3] and Wang Yi[4]

[1]*NORDUnet A/S, Copenhagen, Denmark*
[2]*Department of Computer Science, Aalborg University, Denmark*
[3]*Mälardalen Research and Technology Centre, Mälardalen University, Sweden*
[4]*Department of Information Technology, Uppsala University, Sweden*

SUMMARY

UPPAAL is a tool suitable for model checking real-time systems described as networks of timed automata communicating by channel synchronizations and extended with integer variables. Its first version was released in 1995 and its development is still very active. It now features an advanced modeling language, a user-friendly graphical interface, and a performant model checker engine. In addition, several flavors of the tool have matured in recent years. In this paper, we present how we managed to maintain the tool during 15 years, its current architecture with its challenges, and we give the future directions of the tool. Copyright © 2010 John Wiley & Sons, Ltd.

Received 31 March 2010; Revised 31 May 2010; Accepted 14 July 2010

KEY WORDS: UPPAAL; real-time; model-checker; development

INTRODUCTION

UPPAAL is first of all a research tool born from the collaboration of Uppsala and Aalborg universities [1]. It takes its theoretical roots from Alur and Dill's pioneer work on timed automata [2]. Its performance originally comes from zones [3] as a representation for states and the efficient implementation of operators on its canonical data-structure known as difference-bound matrix (DBM) [4]. Since then the development has been fueled by scientific results on algorithms or new data structures [5–10], academic case-studies [11–15], industrial case-studies [16–20], and also teaching [21].

On the other hand, having such a tool helps to develop and test new theories and algorithms, which has given us synergy during the last decade between tool development and theoretical results.

Recently, the tool has blossomed into several domain specific versions, namely, CORA [6, 22] (cost-optimal reachability), TRON [23, 24] (online testing), COVER [25, 26] (offline test generation), TIGA [27] (timed game solver), PORT [28] (component based and partial order), PRO (extension with probabilities, in progress), and TIMES [29, 30] (scheduling and analysis). These extensions are made based on a common code base, re-using basic data structures to represent states, store them, and perform common operations such as delay, intersection, or computing successor states.

*Correspondence to: Alexandre David, Department of Computer Science, Aalborg University, Denmark.
†E-mail: adavid@cs.aau.dk

CORA is based on linearly priced timed automata [31]. The model extends timed automata with a special cost variable whose rate is specified for every state. The algorithm uses guiding to solve minimum cost reachability problems.

TRON is a testing tool suited for black-box conformance testing [32, 33] of timed systems. It is mainly targeted for embedded software commonly found in various controllers. Testing is done online in the sense that that tests are derived, executed, and checked while maintaining the connection to the system in real-time.

COVER is a tool for creating test suites from UPPAAL models with coverage specified by coverage observers a.k.a. observer automata.

TIGA is an extension for solving reachability and safety problems on timed game automata. Its algorithm [34] is a symbolic extension of the on-the-fly algorithm suggested by Shann *et al.* [35] for linear-time model-checking of finite-state systems. It is used for controller synthesis [36], it has application to testing [37], and it has been extended to synthesis under partial observability [38].

PORT is a version targeted at component-based modeling and verification. Its interface is developed as an Eclipse plug-in. The tool supports graphical modeling of internal component behavior as timed automata and hierarchical composition of components. It is able to exploit the structure of such systems and apply partial order reduction techniques successfully [39].

PRO is an extension of timed automata with probabilities [40, 41]. The model is extended with branching nodes that allow the user to specify weights for every outgoing edge. The engine can then compute probability bounds to reach specified states. It is work-in-progress.

TIMES TIMES is a tool-set for modeling, schedulability analysis, synthesis of (optimal) schedules and executable code. Its modeling language is timed automata extended with tasks. It models systems that can be described as a set of tasks that are triggered periodically or sporadically by time or external events. The release pattern is given by a timed automaton and the tool performs schedulability analysis on it. TIMES works by encoding the problem into timed automata and it uses the UPPAAL engine for the checks. It translates back the answer in terms of Gantt chart to visualize schedules. There are other tools that are using UPPAAL as a back-end verification engine, e.g. REX [42].

In this paper we focus on the 'core' tool UPPAAL and present our experience in developing and maintaining it for the last decade. In particular, we present the backbone architecture that has allowed us to expand the tool on different variants of timed automata. The following sections give an overview of the tool architecture, our experience in the process of building the tool, and the future development directions.

OVERVIEW OF THE TOOL ARCHITECTURE

Client–server architecture. The tool has two main components: a graphical user interface written in Java and a model-checker engine written in C++. The interface runs almost effortlessly on different platforms and we can exploit the rich functionalities available for programming interfaces inherent to the libraries that come with the Java programming language. The C++ language gives us both advanced object-oriented programming and performance. These two components form a basic client–server architecture with the graphical interface (client) communicating with the model checker (server) via a local pipe‡ or the network§. This separation of concerns makes UPPAAL easier to port and maintain on different platforms.

The graphical interface has three 'tabs' that correspond to the main tasks a user needs to do: to edit a model in the editor, to simulate it in the simulator, and submit verification queries to the model-checker. Additionally, the user may come back to the simulator to visualize a trace generated by the verification. Figure 1 gives a view of the simulator of the tool. The different variants of the tool have specialized interfaces and the figure shows the simulator used in TIGA. F1

‡A common inter-process communication mechanism.

§The verification can be done on a remote server, which is a rarely exploited feature.

DOI: 10.1002/spe

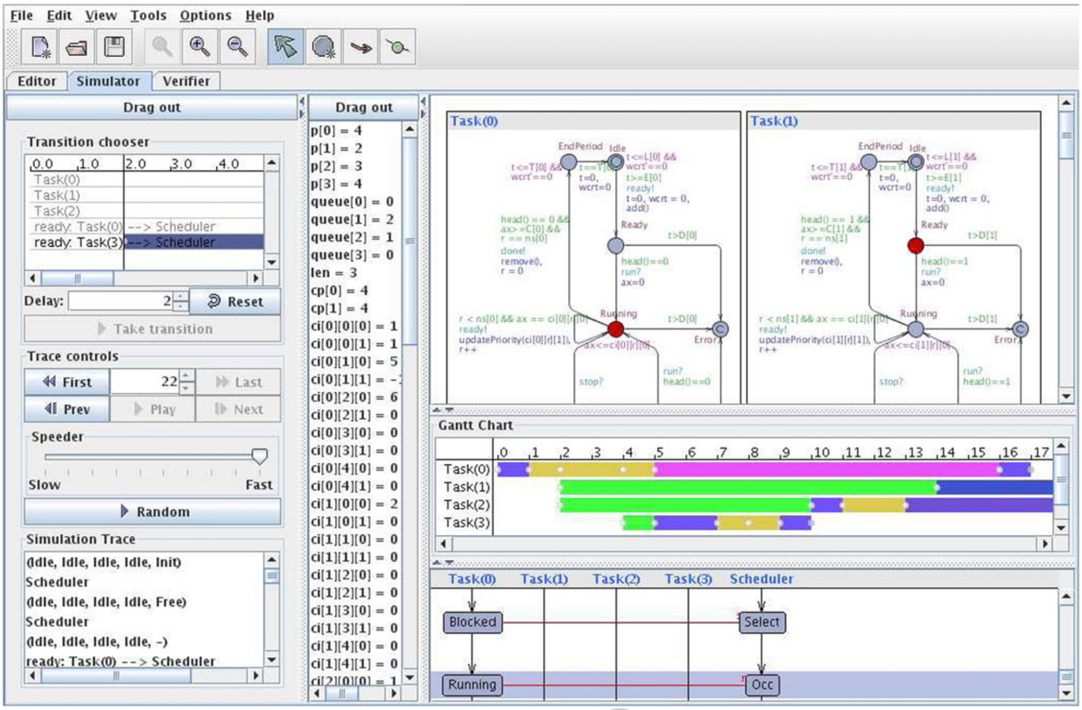

Figure 1. View of the 'concrete' simulator of TIGA.

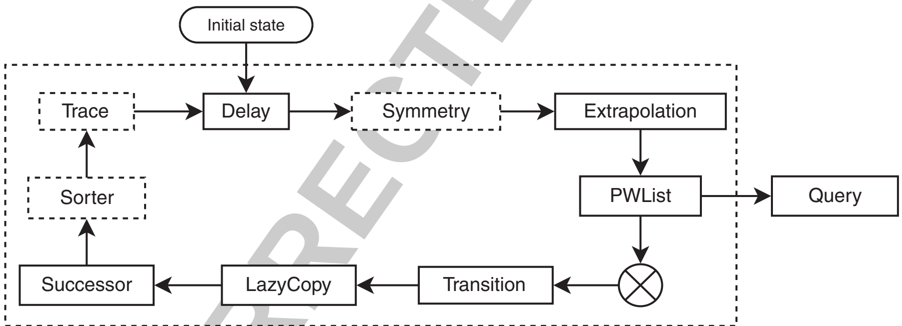

Figure 2. Pipeline architecture for the reachability filter.

On the left is the *command* part where the user can select transitions, go back in the trace, play randomly, or navigate through the current trace (history of states). The list in the middle shows the values of the variables and clocks. The timed automata are shown on the right and below them a Gantt chart and a message sequence chart. The simulator of UPPAAL lacks the Gantt chart and has other similar components, although instead of navigating with *concrete* clock valuations, the user sees *symbolic* states. The point here is to stress reuse of components across different tools. It is important to amortize development costs over time on different specializations of the tool without having to rewrite everything from scratch. This is obvious but the insidious consequence is that it is often difficult in practice to publish on these new additions. This is due to the lack of dedicated conferences where tool developments can be reported on.

Pipeline architecture. The model-checker itself (the *engine*) is designed around a pipeline architecture [5] where each block or *filter* processes states and sends them to the next stage as shown in Figure 2. The figure shows the configuration for the *reachability* filter. Other algorithms such as the *liveness* and the *leadsto* checkers have their own filters built on the same principle. F2

DOI: 10.1002/spe

In this example, the initial state is pushed to the reachability filter in its delay component to start the exploration. Then it runs its main loop that takes states from our (unified passed and) waiting list structure, explores them, and puts the successors in that structure. This structure (also called the *PWList*) implements one (colored) state set with states marked waiting and passed. It is unified in the sense that we have one structure instead of the traditional waiting and passed lists that need two lookups in hash tables per loop iteration of the reachability algorithm. Only states colored as waiting are explored and inclusion check between symbolic states is done against all states. The main chain for the exploration is *Transition* (which transitions can be taken)—*Successor* (execution of the transitions)—*Delay* (let time pass)—*Extrapolation* (apply an appropriate extrapolation to ensure finite exploration)—*PWList* (inclusion check and mark the state to be explored)—*Query* (evaluate the formula if the state was not included).

In fact we inserted a *LazyCopy* filter to reduce copies of states between the transition and successor filters. This filter really copies states only when necessary, e.g. computing one successor only does not require a copy and two successors require one copy only. It acts as a one-place buffer. When priorities are used in the model, this filter is swapped by another filter that is going to buffer transitions and sort them by priority, without changing the rest of the pipeline. Some filters are optional, such as *Sorter* that can sort transitions, *Trace* that is used to store traces or *Symmetry* that is projecting the states to a representative of its equivalence class (orbit) when symmetry is used in the model. In addition, different kinds of extrapolations can be used depending on the model, which results in different kinds of instances for the *Extrapolation* filter. We note that it is simpler to have the logic (in terms of if statements) to instantiate the right type of a component once and use the generic design to connect the components and use them transparently. The reader understands that the combination of these features gives rise to a lot of configurations. The point here is to keep orthogonal features separated.

The overall pipeline architecture allows us to reason about the algorithm in terms of blocks that we can change if we need another semantics. Implementing another checker, e.g. a timed game solver, is relatively easy and consists in adding components that will do the backward propagation, changing the first filter to either explore forward or backward, adding a post-processing filter to detect what is winning or losing in the game after *Extrapolation*, and changing the graph representation. The new pipeline still has the same structure and follows the same design. To change the semantics of the game, e.g. to implement simulation checking [43], we change *Transition* that implements the transition relation and *Delay* to allow turn-based delays.

There are two important points that this architecture illustrates: object-oriented programming and reuse of components. The filters are in practice abstract classes hence these components are managed at a high level. Second, we can reuse these filters for different pipeline configurations, i.e. for different checkers. We note that the architecture is also fit for functional languages.

Additional components. In addition to these components, UPPAAL contains a virtual machine to execute the compiled byte-code of our C-like input language supporting user defined functions and types. This allows the user to write complex and compact models while still limiting the state-space explosion—complexity can be concentrated in functions to avoid using intermediate states.

We currently distribute some open source components, such as the parser and the DBM library. The DBM library has a Ruby binding, which allows for quick prototyping. The parser understands the XML format we use in UPPAAL, which allows other researchers to use the same format. The DBM library handles DBMs and federations (unions of DBMs) used to represent symbolic states. The DBM library supports a wide range of operations including subtractions and merging of DBMs.

TOOL BUILDING PROCESS

Tools are not prototypes. It is relatively easy to produce prototypes as proofs of concept of some theory or algorithm to strengthen a paper but it is notoriously more difficult to develop a tool that is going to survive the test of time. Unfortunately, prototypes are more common in practice. Building and maintaining tools take a lot of time and is generally given less academic credit compared

DOI: 10.1002/spe

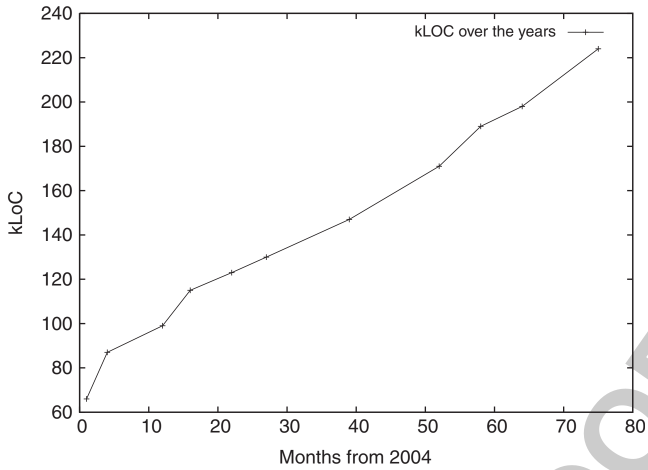

Figure 3. Evolution of the code base.

to more theoretical work, which explains the limited number of maintained tools. In the domain of formal methods, tools are crucial and they also serve as a dissemination means for theoretical results. Tools do have a positive impact through the case-studies that they allow other users to do, often in collaboration, which is important to amortize the development cost (in terms of time) and earn publications (otherwise we perish)[¶].

Who develops? The first question in developing tools is who is going to do it? Most of the time it is done by masters or PhD students, which makes sense economically since professors cannot afford writing C++ code. However, when temporary developers who have their own agenda (own thesis to write) work on the tools, there is the obvious issue that someone needs to take over, otherwise the tool will disappear. In addition, temporary developers do not have a long-term vision and are interested (rightfully so) in their own thesis. Over the years, changing teams without a common interest or focus means that the code will degrade if there is no control. What happened with us was that there were some PhD students who stayed in the team for a long time, long enough to lay down a solid architecture and durable design. As a first rule of survival, *one should have a solid design* and encourage people to stick to it even if they do not like it. At some point in time old design decisions will not make sense any more but that is a different issue.

Code size. When the code grows (see Figure 3) it is increasingly difficult for new people to use the code hence it becomes important to have some permanent staff to take care of it and revise it so that it can offer a limited and more useful interface. This is a considerable effort that is essential for the survival of the tool. In the past we had a few such revisions: the *pipeline* architecture, the *virtual machine*, and handling of *federations* in the model-checker. The size and complexity of the code has now become a barrier for new internal people but it is also a serious problem for external collaboration. We need a new revision to update the interfaces of the different components and add more abstraction to the code. For long-term development, it is important to have some permanent staff to take care of such revisions and keep a long-term vision. However, it is a trade-off between the academic and development work.

Code aging. Curiously code ages. This is due to the fact that developers forget old code and new methods or libraries appear over the years, which makes the code become older or deprecated. In addition, progress in compilers also means that special efforts in the past to make some algorithms efficient are now obsolete, e.g. we can commonly address iteratively elements in matrices by expressions like $dbm[i*dim+j]$ with confidence that the compiler will *not* use relatively expensive multiplications for element accesses. To counter code aging, documentation

[¶] The well-known motto *publish or perish* emphasized by a system holding that name is a testament of the tool development dilemma in academia.

is vital. Our experience has been to 'document' the code using `doxygen` formatted comments. There is no real documentation apart from these comments, although some efforts have been made to describe the overall design decisions and the communication protocol. We have crash courses to inform new programmers, which is a limited solution. As for the comments, they are extensive and they keep the memory of former developers. It is a weakness in the development process to lack stand-alone *white papers* that give technical details of the code but this has not been our priority.

Life cycles. The tool has gone over different life cycles over the years. A life cycle can be defined by major changes in architecture that are needed to accommodate new developments. It happens when old designs become too obsolete for new additions that were not foreseen in the past. The first cycle was with the original `ATG` graph editor‖ and an early custom simulator. The second cycle introduced an integrated graphical editor, the client–server architecture still in use today, and an improved engine. The third cycle is the current one with a modular pipeline architecture. This pipeline architecture is probably the determinant factor for keeping additions of new features without breaking the tool. In terms of features it is interesting to note that early development efforts were focused on performance improvements and then later on interface and language features. The later developments of the tool introduced new algorithms to handle different problems rather than improvements in the current algorithms.

During a cycle, the development is incremental, following the current design and making changes until the amount of desired new features and algorithms conflicts too much with the design. At this point there is a major effort to redesign (or re-factor) the code. The current architecture has lived up to its expectations for the approximately 8 years, during which we could re-use existing components and create new ones that we could literally plug together. However, the plethora of new variants of the tool hides the current internal issues with the architecture and now is the time for a major update.

Distributed development. We use a centralized version management system (`CVS` and later subversion), which allows distributed teams to work on the same code. This is common for distributed projects. A given checkout of the repository contains all variants of the tool but each of them is located in its own separated module. Developers are responsible for few modules (their own) and modify other modules only occasionally. The key here is to have *responsibility* for the different parts for maintenance. In addition, we have the simple rules *committed code must compile* and *any distributed code must pass the regression test*. As breaking these rules produces heated reactions, they tend to be observed. The goal here is to keep discipline.

Testing. For a tool in the field of formal methods we would expect to apply formal verification techniques to it to ensure its correctness. Let us say research is not there yet. The code base has currently 200+ KLoC in C++ which implement algorithms that are themselves notoriously difficult to prove. There are tools we have used to assist us, such as `gcov`, `purify`, and `valgrind`. However, what we routinely do is to test. We use regression testing on a battery of known examples and results. When a bug is discovered, we insert that the new example in the test suite and make sure new versions pass the new tests. This is an automatic process handled by a script.

Bug management. Another well-known tool we are using is the bug management system *bugzilla*. Bugs are not only program errors but also requested features. They are sorted by priorities that developers can set. Errors usually come with examples to reproduce them. They are added to the regression tests when the errors are corrected. Sometimes a change in the code triggers a new error that was not present in the past. We use binary search on the revision number (in our subversion repository) to find which revision introduced the changes that triggered that error. This is a simple and very effective technique.

Cross-platform. An integral part of the development process is to take care of cross-platform development. Early on we decided to stick with one compiler, `gcc/g++`. We can use the same code and change a few headers only and compile for Windows, Linux, and MacOS. By doing so we can also take advantage of some useful gnu extensions. We dropped support for SunOS due to

‖This is an editor tool used by UPPAAL from 1995 to 1999.

DOI: 10.1002/spe

the absence of users and also machines installed with SunOS. All three supported platforms are actively used with an increase for MacOS in the recent years. What we do to manage this is to keep third-party libraries at a minimum. Currently, we need `libxml2` compiled for all platforms and we use `boost` headers only. The rest is generated code by tools such as `bison` and `flex`. Compilation is done under Linux for Linux and Windows, MacOS binaries are compiled on a Mac. We foresee problems in the future when supporting multi-threads since we will have to use additional libraries such as `Win32-pthread` to support POSIX threads (to begin with, the library needs to be patched for Win64).

Community. Finally, to survive, a tool needs its community. We have a discussion forum[**] that our user community uses to ask or answer questions and maintain an active discussion on the tool. In fact, this helps us tremendously because we cannot handle all newcomers to the tool individually and we are grateful to users who help each other. The community also provides us with new problems and case-studies, which in turn instill progress in algorithms and theory.

CHALLENGES

The first challenge is to manage the complexity and size of the project. Implementing advanced algorithms is tricky, specially when it is in a formal tool which is used for verification. As shown in Figure 3 the code (in kilo lines of code of C/C++) has been steadily growing. This growth comes from new variants and algorithms that are added to the repository. The count includes all code (used or not) for all variants of UPPAAL for the model-checker engine only. The graphical interface adds 40+ kLoC in Java.

The second challenge is to keep improving the performance and features of the tool despite the growing algorithm complexity. Table I shows the evolution of the performance of the tool. Experiments have been performed on a Pentium D 2.80 GHz with 1 GB RAM. We use `memtime` T1 that measures time and polls memory (not reliable below 0.1 s). Entries marked '—' denote verifications that were stopped after 2 h or 900 M. The models are available on *www.uppaal.org* under *Examples/benchmarks*. Apart from the performance improvements, the recent versions support user-defined functions and symmetry reduction. These features are not used in the experiments but they would further improve the performance.

The third challenge is to cope with new extensions of the tool to explore different theoretical paths. The current architecture has been pushed to implement the different known flavors of UPPAAL but also to extend every checker. Recent extensions to UPPAAL include priorities and stop-watches. TIGA was recently extended with a simulation checker. It is being extended with a new timed interface checker. Although the overall pipeline architecture accommodates these extensions, we have reached the limit of some 'implementation details'. These are: (1) there can only be one global system, (2) long-wished features, such as clock constraints on receiving edges of broadcast synchronizations, are now needed, (3) the engine is designed for 32-bit architectures, (4) there is no multi-core support, (5) there is only one kind of symbolic state, and the list goes on. These are obstacles for doing compositional model-checking where we would need to handle several systems and combine results. In addition, it is difficult to adapt the engine to different kinds of systems without changing core structures such as the states. Currently, when compiling CORA, one C macro is changed to swap to a different type of DBM supporting costs. This works because we made sure that the commonly needed interface was exactly the same. This is a very limited solution.

Another challenge is to use modern technology to its full potential. Updating to 64-bit is mainly technical. Taking real advantage of 64-bit is challenging. Modern compilers have the ability to *vectorize* code[††] but this is still limited to simple algorithms and not to the critical $O(n^3)$ algorithms that we have. Going for multi-core support (multi-threaded UPPAAL) is more difficult. There have

[**]http://tech.groups.yahoo.com/group/uppaal/.

[††]This in essence allows the use of SIMD instructions (single instruction, multiple data) on streams of data.

DOI: 10.1002/spe

Table I. Evolution of performance in terms of time (s) and memory consumption (MB).

Version	CSMA5	CSMA7	CSMA12	Fischer5	Fischer7	Fischer12	HDDI7	HDDI12
3.0.39	*8.4* s	—	—	*4.2* s	—	—	*36.3* s	—
	7.2 MB	—	—	10.6 MB	—	—	20.1 MB	—
3.2.12	*0.3* s	*417* s	—	*1.6* s	—	—	*7.2* s	—
	3.8 MB	145 MB	—	6.8 MB	—	—	11.9 MB	—
3.3.25	*0.2* s	*198* s	—	*1.1* s	—	—	*3.2* s	—
	3.4 MB	113 MB	—	6 MB	—	—	8.4 MB	—
3.4.6	*<0.1* s	*40.7* s	—	*0.3* s	*4706* s	—	*0.1* s	*5.3* s
	3.1 MB	34.5 MB	—	4.9 MB	267 MB	—	1.6 MB	14.1 MB
4.0.11	*<0.1* s	*0.2* s	*33.8* s	*<0.1* s	*0.4* s	*418* s	*<0.1* s	*0.4* s
	1.6 MB	38 MB	115 MB	1.6 MB	38.1 MB	300 MB	1.6 MB	38 MB
4.1.2	*<0.1* s	*0.2* s	*41.9* s	*<0.1* s	*0.3* s	*341* s	*0.05* s	*0.2* s
	1.6 MB	21.6 MB	99 MB	21 MB	21.6 MB	248 MB	1.6 MB	22.9 MB

been experiments in the past in this direction and we know that the current architecture could be adapted by having one thread per pipeline copy. This fits memory locality but we also know that it did not work so well because blocking data-structures (access protected by mutex) were major bottlenecks. It is crucial to have non-blocking structures such as [44] if we want to use multi-cores efficiently, although this is a temporary solution that will last at most 10 years‡‡. In addition, we want to make the components extendable more easily in particular to allow more people to work on UPPAAL without having to know what most of the code is doing. The bottom line is that there are research opportunities but not all issues are research related.

FUTURE

We have shown in this paper the main challenges that we faced in building UPPAAL over the years along with our own solutions. The conclusion is to get the synergy theory—implementation—case-studies that in turn provides the *publications*. There is no bullet-proof solution and we consider ourselves to have been lucky to have started at the right time and got such a good response from the community to get this synergy.

UPPAAL has already spawned one company, UP4ALL§§, that sells a version of the tool for commercial uses. Another market we intend to target is testing. Research tools really have a future if they can be applied and used outside academia, as witnessed by Lustre/SCADE. However, their future as a free academic tool is uncertain as discussed in this paper in relation with the dilemmas. The situation is that tool paper tracks exist and show the interest in academic tool development but they are often on the side of main conferences and they usually accept short papers with short talks. This could be improved to stimulate tool development in the community.

To continue the development on the academic path, we are exploring different domains as the different flavors of UPPAAL show. This also means that a new life cycle with another architectural revision is now needed to cope with more extensions of UPPAAL. This will enable us to let other researchers experiment with the internals of UPPAAL while still maintaining our core engine.

ACKNOWLEDGEMENTS

It is important to remember that UPPAAL is the result of the cumulative efforts of many collaborators. Among them we would like to thank early pioneers Johan Bengtsson and Fredrik Larsen, former contributor of the graphical interface Carsten Weise, and active contributor and maintainer Marius Mikučionis (UPPAAL and TRON). We also thank contributors of different extensions of UPPAAL, among them

‡‡Shared memory architectures do not scale and message passing-based architectures will take over.

§§To contact UP4ALL email *sales@uppaal.com*.

Didier Lime (TIGA), John Håkansson (PORT), Anders Hessel (COVER), Leonid Mokrushin (TIMES), Jakob Illum (CORA), Arild Haugstad (PRO). Last but not least we thank our supporting user community.

REFERENCES

1. Yi W, Pettersson P, Daniels M. Automatic verification of real-time communicating systems by constraint-solving. *Proceedings of FORTE'94*, Hogrefe D, Leue S (eds.), North-Holland, 1994; 223–238. Q1
2. Alur R, Dill DL. Automata for modeling real-time systems. *Proceedings of ICALP* (*Lecture Notes in Computer Science*, vol. 443). Springer: Berlin, 1990; 322–335.
3. Larsen KG, Pettersson P, Yi W. Model-checking for real-time systems. *Proceedings of Fundamentals of Computation Theory* (*Lecture Notes in Computer Science*, vol. 965). Springer: Berlin, August 1995; 62–88.
4. Bengtsson J, Yi W. Timed Automata: Semantics, Algorithms and Tools. *Lectures on Concurrency and Petri Nets* (*Lecture Notes in Computer Science*, vol. 3098). Springer: Berlin, 2003; 87–124.
5. Behrmann G, David A, Larsen KG, Yi W. Unification and sharing in timed automata verification. *SPIN Workshop 03* (*Lecture Notes in Computer Science*, vol. 2648). Springer: Berlin, 225–229.
6. Behrmann G, Fehnker A, Hune T, Larsen KG, Pettersson P, Romijn J. Efficient guiding towards cost-optimality in UPPAAL. *Proceedings of the 7th International Conference on TACAS* (*Lecture Notes in Computer Science*, vol. 2301). Margaria T, Yi W (eds.). Springer: Berlin, 2001; 174–188.
7. Behrmann G, Larsen KG, Pearson J, Weise C, Yi W. Efficient timed reachability analysis using clock difference diagrams. *Proceedings of the 12th International Conference on CAV* (*Lecture Notes in Computer Science*, vol. 1633). Springer: Berlin, 1999. Q2
8. David A, Håkansson J, Larsen KG, Pettersson P. Model checking timed automata with priorities using DBM subtraction. *Proceedings of the 4th International Conference on FORMATS* (*Lecture Notes in Computer Science*, vol. 4202). Springer: Berlin, 2006; 128–142.
9. Larsson F, Larsen KG, Pettersson P, Yi W. Efficient verification of real-time systems: Compact data structures and state-space reduction. *Proceedings of the 18th IEEE RTSS*. IEEE Computer Society Press: Silver Spring, MD, 1997; 14–24.
10. Larsson F, Pettersson P, Yi W. On memory-block traversal problems in model checking timed systems. *Proceedings of the 6th Conference on TACAS* (*Lecture Notes in Computer Science)*, vol. 1785, Graf S, Schwartzbach M (eds.). Springer: Berlin, 2000; 127–141.
11. Lönn H, Pettersson P. Formal verification of a TDMA protocol startup mechanism. *Proceedings of the Pacific Rim International Symposium on Fault-tolerant Systems*, December 1997; 235–242. Q3
12. D'Argenio P, Katoen J-P, Ruys T, Tretmans J. The bounded retransmission protocol must be on time! *Proceedings of the Third Workshop on Tools and Algorithms for the Construction and Analysis of Systems* (*Lecture Notes in Computer Science*, vol. 1217), Brinksma E (ed.)., Springer: Enschede, The Netherlands, 1997; 416–431.
13. Gebremichael B, Vaandrager F, Zhang M. Analysis of the zeroconf protocol using uppaal. *Proceedings of the 6th ACM and IEEE International Conference on Embedded Software*. ACM: New York, 2006; 242–251.
14. Heidarian F, Schmaltz J, Vaandrager F. Analysis of a clock synchronization protocol for wireless sensor networks. *Proceedings of FM 2009*: *Formal Methods* (*Lecture Notes in Computer Science*, vol. 5850), Cavalcanti A, Dams D (eds.). Springer: Berlin, 2009; 516–531.
15. David A, Möller MO, Yi W. Formal verification of UML statecharts with real-time extensions. *FASE*, *The 5th International Conference 2002* (*Lecture Notes in Computer Science*, vol. 2306). Kutsche R-D, Weber H (eds.). Springer: Berlin, 2002; 218–232.
16. Lindahl M, Pettersson P, Yi W. Formal design and analysis of a gear-box controller. *Proceedings of the 4th Workshop on TACAS* (*Lecture Notes in Computer Science*, vol. 1384). Springer: Berlin, March 1998, 281–297.
17. David A, Yi W. Modelling and analysis of a commercial field bus protocol. *Proceedings of Euromicro—RTS'00*. IEEE Computer Society: Silver Spring, MD, 2000; 165–172.
18. Bengtsson J, Griffioen WD, Kristoffersen KJ, Larsen KG, Larsson F, Pettersson P, Yi W. Automated verification of an audio-control protocol using UPPAAL. *Journal of Logic and Algebraic Programming* 2002; **52–53**:163–181.
19. AlAttili I, Houben F, Igna G, Michels S, Zhu F, Vaandrager F. Adaptive scheduling of data paths using uppaal tiga. *Proceedings First Workshop on Quantitative Formal Methods*: *Theory and Applications* (*QFM'09*), S. A., *et al.* (eds.), vol. 13, Electronic Proceedings in Theoretical Computer Science, 2009; 1–12. Q4
20. Arnaud Y, Boimond J-L, Cury JE, Loiseau JJ, Martinez C. Using uppaal for the secure and optimal control of agv fleets. *The 7th Workshop on Advanced Control and Diagnosis ACD 2009*, 2009. Available at: http://hal.archives-ouvertes.fr/hal-00463480. Q5
21. Hamberg R, Vaandrager F. Using model checkers in an introductory course on operating systems. *Operating Systems Review* 2008; **42**(6):101–111.
22. Behrmann G, Fehnker A, Hune T, Larsen KG, Pettersson P, Romijn J, Vaandrager FW. Minimum-cost reachability for priced timed automata. *HSCC* 2001; 147–161.
23. Larsen K, Mikučionis M, Nielsen B. Online testing of real-time systems using UPPAAL. *FATES'04* (*Lecture Notes in Computer Science*). Springer: Berlin, Linz, Austria, September 2004.
24. Larsen KG, Mikucionis M, Nielsen B. Testing real-time embedded software using uppaal-tron: An industrial case study. *The 5th ACM International Conference on Embedded Software*. ACM Press: New York, NY, U.S.A., 2005; 299–306.

DOI: 10.1002/spe

25. Hessel A, Pettersson P. A test case generation algorithm for real-time systems. *Proceedings of the Fourth ICQS*, Ehrich H-D, Schewe K-D (eds.). IEEE Computer Society: Silver Spring, MD, 2004; 268–273.
26. Hessel A, Pettersson P. Cover—A test-case generation tool for timed systems. *Testing of Software and Communicating Systems*: *Work-in-Progress and Position Papers*, *Tool Demonstrations*, *and Tutorial Abstracts of TestCom/FATES*, Petrenko A, Veanes M, Tretmans J, Grieskamp W (eds.)., 2007; 31–34. **Q6**
27. Behrmann G, Cougnard A, David A, Fleury E, Larsen KG, Lime D. UPPAAL-TIGA: Time for playing games! *CAV'07* (*Lecture Notes in Computer Science*, vol. 4590). Springer: Berlin, 2007; 121–125.
28. Håkansson J, Carlson J, Monot A, Pettersson P, Slutej D. Component-based design and analysis of embedded systems with UPPAAL PORT. *ATVA* (*Lecture Notes in Computer Science*, vol. 5311), Cha SD, Choi J-Y, Kim M, Lee I, Viswanathan M (eds.). Springer: Berlin, 2008; 252–257.
29. Fersman E, Pettersson P, Yi W. Timed automata with asynchronous processes: Schedulability and decidability. *Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems* (*Lecture Notes in Computer Science*, vol. 2280), Katoen J-P, Stevens P (eds.). Springer: Berlin, 2002; 67–82.
30. Amnell T, Fersman E, Mokrushin L, Pettersson P, Yi W. Times: A tool for modelling and implementation of embedded systems. *Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems* (*Lecture Notes in Computer Science*), vol. 2280), Katoen J-P, Stevens P (eds.). Springer: Berlin, 2002; 460–464.
31. Behrmann G, Larsen KG, Rasmussen JI. Priced timed automata: Algorithms and applications. *FMCO* 2004; 162–182.
32. Tretmans J. A formal approach to conformance testing. *PhD Thesis*, University of Twente, 1992.
33. Krichen M, Tripakis S. Black-box conformance testing for real-time systems. *Model Checking Software* (*Lecture Notes in Computer Science*, vol. 2989). Springer: Berlin, 2004; 109–126.
34. Cassez F, David A, Fleury E, Larsen KG, Lime D. Efficient on-the-fly algorithms for the analysis of timed games. *CONCUR'05* (*Lecture Notes in Computer Science*, vol. 3653). Springer: Berlin, 2005, 66–80.
35. Liu X, Smolka S. Simple linear-time algorithm for minimal fixed points. *Proceedings of 26th Conference on Automata, Languages and Programming* (*ICALP'98*) (*Lecture Notes in Computer Science*, vol. 1443). Springer: Berlin, 1998; 53–66.
36. Jessen JJ, Rasmussen JI, Larsen KG, David A. Guided controller synthesis for climate controller using UPPAAL-TIGA. *Proceedings of the 19th International Conference on Formal Modeling and Analysis of Timed Systems* (*Lecture Notes in Computer Science*, vol. 4763). Springer: Berlin, 2007; 227–240.
37. David A, Larsen KG, Li S, Nielsen B. Cooperative testing of uncontrollable real-time systems. *The 4th Workshop of Model-based Testing* (*MBT'08*), 2008.
38. Cassez F, David A, Larsen KG, Lime D, Raskin J-F. Timed control with observation based and stuttering invariant strategies. *Proceedings of the 5th International Symposium on Automated Technology for Verification and Analysis* (*Lecture Notes in Computer Science*, vol. 4762). Springer: Berlin, 2007; 192–206.
39. Håkansson J, Pettersson P. Partial order reduction for verification of real-time components. *Proceedings of the 5th International Conference on FORMATS* (*Lecture Notes in Computer Science*). Springer: Berlin, 2007.
40. Beauquier D. On probabilistic timed automata. *Theoretical Computer Science* 2003; **292**:65–84.
41. Kwiatkowska M, Norman G, Sproston J, Wang F. Symbolic model checking for probabilistic timed automata. *Formal Techniques, Modelling and Analysis of Timed and Fault-tolerant Systems* (*Lecture Notes in Computer Science*, vol. 3253. Springer: Berlin, 2004; 293–308.
42. Ericsson A, Berndtsson M, Pettersson P, Pettersson L. Verification of an industrial rule-based manufacturing system using rex. *The 1st International Workshop on Complex Event Processing for Future Internet*, September 2008.
43. Bulychev P, Chatain T, David A, Larsen KG. Efficient on-the-fly algorithm for checking alternating timed simulation, FORMATS'09 (*Lecture Notes in Computer Science*, vol. 5813). Springer: Berlin, 2009; 73–87.
44. Shann C-H, Huang T-L, Chen C. A practical nonblocking queue algorithm using compare-and-swap. *The 7th International Conference on Parallel and Distributed Systems* 2000; 470–475.

DOI: 10.1002/spe

Author Queries Form

John Wiley

JOURNAL TITLE: **SPE** **2/8/2010**

ARTICLE NO: 1006

Queries and / or remarks

Query No.	Details required	Author's response
Q1	Please provide publisher name for Ref. [1].	
Q2	Please provide page range for Refs. [7,23,39].	
Q3	Please provide place of proceeding for Refs. [11,42].	
Q4	Please provide editor names and publisher details for Ref. [19].	
Q5	Please provide place of proceeding and access date for Ref. [20].	
Q6	Please provide publisher name and location for Ref. [26].	

COPYRIGHT TRANSFER AGREEMENT

Date: ____________________ Contributor name: ____________________

Contributor address: ____________________

Manuscript number (Editorial office only): ____________________

Re: Manuscript entitled ____________________

____________________ (the "Contribution")

for publication in ____________________ (the "Journal")

published by ____________________ ("Wiley-Blackwell").

Dear Contributor(s):
Thank you for submitting your Contribution for publication. In order to expedite the editing and publishing process and enable Wiley-Blackwell to disseminate your Contribution to the fullest extent, we need to have this Copyright Transfer Agreement signed and returned as directed in the Journal's instructions for authors as soon as possible. If the Contribution is not accepted for publication, or if the Contribution is subsequently rejected, this Agreement shall be null and void. **Publication cannot proceed without a signed copy of this Agreement.**

A. COPYRIGHT

1. The Contributor assigns to Wiley-Blackwell, during the full term of copyright and any extensions or renewals, all copyright in and to the Contribution, and all rights therein, including but not limited to the right to publish, republish, transmit, sell, distribute and otherwise use the Contribution in whole or in part in electronic and print editions of the Journal and in derivative works throughout the world, in all languages and in all media of expression now known or later developed, and to license or permit others to do so.

2. Reproduction, posting, transmission or other distribution or use of the final Contribution in whole or in part in any medium by the Contributor as permitted by this Agreement requires a citation to the Journal and an appropriate credit to Wiley-Blackwell as Publisher, and/or the Society if applicable, suitable in form and content as follows: (Title of Article, Author, Journal Title and Volume/Issue, Copyright © [year], copyright owner as specified in the Journal). Links to the final article on Wiley-Blackwell's website are encouraged where appropriate.

B. RETAINED RIGHTS

Notwithstanding the above, the Contributor or, if applicable, the Contributor's Employer, retains all proprietary rights other than copyright, such as patent rights, in any process, procedure or article of manufacture described in the Contribution.

C. PERMITTED USES BY CONTRIBUTOR

1. Submitted Version. Wiley-Blackwell licenses back the following rights to the Contributor in the version of the Contribution as originally submitted for publication:

a. After publication of the final article, the right to self-archive on the Contributor's personal website or in the Contributor's institution's/employer's institutional repository or archive. This right extends to both intranets and the Internet. The Contributor may not update the submission version or replace it with the published Contribution. The version posted must contain a legend as follows: This is the pre-peer reviewed version of the following article: FULL CITE, which has been published in final form at [Link to final article].

b. The right to transmit, print and share copies with colleagues.

2. Accepted Version. Re-use of the accepted and peer-reviewed (but not final) version of the Contribution shall be by separate agreement with Wiley-Blackwell. Wiley-Blackwell has agreements with certain funding agencies governing reuse of this version. The details of those relationships, and other offerings allowing open web use, are set forth at the following website: http://www.wiley.com/go/funderstatement. NIH grantees should check the box at the bottom of this document.

3. Final Published Version. Wiley-Blackwell hereby licenses back to the Contributor the following rights with respect to the final published version of the Contribution:

a. Copies for colleagues. The personal right of the Contributor only to send or transmit individual copies of the final published version in any format to colleagues upon their specific request provided no fee is charged, and further-provided that there is no systematic distribution of the Contribution, e.g. posting on a listserve, website or automated delivery.

b. Re-use in other publications. The right to re-use the final Contribution or parts thereof for any publication authored or edited by the Contributor (excluding journal articles) where such re-used material constitutes less than half of the total material in such publication. In such case, any modifications should be accurately noted.

c. Teaching duties. The right to include the Contribution in teaching or training duties at the Contributor's institution/place of employment including in course packs, e-reserves, presentation at professional conferences, in-house training, or distance learning. The Contribution may not be used in seminars outside of normal teaching obligations (e.g. commercial seminars). Electronic posting of the final published version in connection with teaching/training at the Contributor's institution/place of employment is permitted subject to the implementation of reasonable access control mechanisms, such as user name and password. Posting the final published version on the open Internet is not permitted.

d. Oral presentations. The right to make oral presentations based on the Contribution.

4. Article Abstracts, Figures, Tables, Data Sets, Artwork and Selected Text (up to 250 words).

a. Contributors may re-use unmodified abstracts for any non-commercial purpose. For on-line uses of the abstracts, Wiley-Blackwell encourages but does not require linking back to the final published versions.

b. Contributors may re-use figures, tables, data sets, artwork, and selected text up to 250 words from their Contributions, provided the following conditions are met:

(i) Full and accurate credit must be given to the Contribution.
(ii) Modifications to the figures, tables and data must be noted. Otherwise, no changes may be made.
(iii) The reuse may not be made for direct commercial purposes, or for financial consideration to the Contributor.
(iv) Nothing herein shall permit dual publication in violation of journal ethical practices.

D. CONTRIBUTIONS OWNED BY EMPLOYER

1. If the Contribution was written by the Contributor in the course of the Contributor's employment (as a "work-made-for-hire" in the course of employment), the Contribution is owned by the company/employer which must sign this Agreement (in addition to the Contributor's signature) in the space provided below. In such case, the company/employer hereby assigns to Wiley-Blackwell, during the full term of copyright, all copyright in and to the Contribution for the full term of copyright throughout the world as specified in paragraph A above.

2. In addition to the rights specified as retained in paragraph B above and the rights granted back to the Contributor pursuant to paragraph C above, Wiley-Blackwell hereby grants back, without charge, to such company/employer, its subsidiaries and divisions, the right to make copies of and distribute the final published Contribution internally in print format or electronically on the Company's internal network. Copies so used may not be resold or distributed externally. However the company/employer may include information and text from the Contribution as part of an information package included with software or other products offered for sale or license or included in patent applications. Posting of the final published Contribution by the institution on a public access website may only be done with Wiley-Blackwell's written permission, and payment of any applicable fee(s). Also, upon payment of Wiley-Blackwell's reprint fee, the institution may distribute print copies of the published Contribution externally.

E. GOVERNMENT CONTRACTS

In the case of a Contribution prepared under U.S. Government contract or grant, the U.S. Government may reproduce, without charge, all or portions of the Contribution and may authorize others to do so, for official U.S. Government purposes only, if the U.S. Government contract or grant so requires. (U.S. Government, U.K. Government, and other government employees: see notes at end)

F. COPYRIGHT NOTICE

The Contributor and the company/employer agree that any and all copies of the final published version of the Contribution or any part thereof distributed or posted by them in print or electronic format as permitted herein will include the notice of copyright as stipulated in the Journal and a full citation to the Journal as published by Wiley-Blackwell.

G. CONTRIBUTOR'S REPRESENTATIONS

The Contributor represents that the Contribution is the Contributor's original work, all individuals identified as Contributors actually contributed to the Contribution, and all individuals who contributed are included. If the Contribution was prepared jointly, the Contributor agrees to inform the co-Contributors of the terms of this Agreement and to obtain their signature to this Agreement or their written permission to sign on their behalf. The Contribution is submitted only to this Journal and has not been published before. (If excerpts from copyrighted works owned by third parties are included, the Contributor will obtain written permission from the copyright owners for all uses as set forth in Wiley-Blackwell's permissions form or in the Journal's Instructions for Contributors, and show credit to the sources in the Contribution.) The Contributor also warrants that the Contribution contains no libelous or unlawful statements, does not infringe upon the rights (including without limitation the copyright, patent or trademark rights) or the privacy of others, or contain material or instructions that might cause harm or injury.

CHECK ONE BOX:

☐ Contributor-owned work

ATTACH ADDITIONAL SIGNATURE PAGES AS NECESSARY

Contributor's signature ______________________ Date ____________

Type or print name and title ______________________

Co-contributor's signature ______________________ Date ____________

Type or print name and title ______________________

☐ Company/Institution-owned work (made-for-hire in the course of employment)

Company or Institution (Employer-for-Hire) ______________________ Date ____________

Authorized signature of Employer ______________________ Date ____________

☐ U.S. Government work

Note to U.S. Government Employees
A contribution prepared by a U.S. federal government employee as part of the employee's official duties, or which is an official U.S. Government publication, is called a "U.S. Government work," and is in the public domain in the United States. In such case, the employee may cross out Paragraph A.1 but must sign (in the Contributor's signature line) and return this Agreement. If the Contribution was not prepared as part of the employee's duties or is not an official U.S. Government publication, it is not a U.S. Government work.

☐ U.K. Government work (Crown Copyright)

Note to U.K. Government Employees
The rights in a Contribution prepared by an employee of a U.K. government department, agency or other Crown body as part of his/her official duties, or which is an official government publication, belong to the Crown. U.K. government authors should submit a signed declaration form together with this Agreement. The form can be obtained via http://www.opsi.gov.uk/advice/crown-copyright/copyright-guidance/publication-of-articles-written-by-ministers-and-civil-servants.htm

☐ Other Government work

Note to Non-U.S., Non-U.K. Government Employees
If your status as a government employee legally prevents you from signing this Agreement, please contact the editorial office.

☐ NIH Grantees

Note to NIH Grantees
Pursuant to NIH mandate, Wiley-Blackwell will post the accepted version of Contributions authored by NIH grant-holders to PubMed Central upon acceptance. This accepted version will be made publicly available 12 months after publication. For further information, see www.wiley.com/go/nihmandate.

WILEY AUTHOR DISCOUNT CARD

As a highly valued contributor to Wiley's publications, we would like to show our appreciation to you by offering a **unique 25% discount** off the published price of any of our books*.

To take advantage of this offer, all you need to do is apply for the **Wiley Author Discount Card** by completing the attached form and returning it to us at the following address:

The Database Group
John Wiley & Sons Ltd
The Atrium
Southern Gate
Chichester
West Sussex PO19 8SQ
UK

In the meantime, whenever you order books direct from us, simply quote promotional code **S001W** to take advantage of the 25% discount.

The newest and quickest way to order your books from us is via our new European website at:

http://www.wileyeurope.com

Key benefits to using the site and ordering online include:

- Real-time SECURE on-line ordering
- The most up-to-date search functionality to make browsing the catalogue easier
- Dedicated Author resource centre
- E-mail a friend
- Easy to use navigation
- Regular special offers
- Sign up for subject orientated e-mail alerts

So take advantage of this great offer, return your completed form today to receive your discount card.

Yours sincerely,

Verity Leaver
E-marketing and Database Manager

*TERMS AND CONDITIONS

This offer is exclusive to Wiley Authors, Editors, Contributors and Editorial Board Members in acquiring books (excluding encyclopaedias and major reference works) for their personal use. There must be no resale through any channel. The offer is subject to stock availability and cannot be applied retrospectively. This entitlement cannot be used in conjunction with any other special offer. Wiley reserves the right to amend the terms of the offer at any time.

REGISTRATION FORM
FOR 25% BOOK DISCOUNT CARD

To enjoy your special discount, tell us your areas of interest and you will receive relevant catalogues or leaflets from which to select your books. Please indicate your specific subject areas below.

Accounting [ ]
- Public [ ]
- Corporate [ ]

Chemistry [ ]
- Analytical [ ]
- Industrial/Safety [ ]
- Organic [ ]
- Inorganic [ ]
- Polymer [ ]
- Spectroscopy [ ]

Encyclopedia/Reference [ ]
- Business/Finance [ ]
- Life Sciences [ ]
- Medical Sciences [ ]
- Physical Sciences [ ]
- Technology [ ]

Earth & Environmental Science [ ]

Hospitality [ ]

Genetics [ ]
- Bioinformatics/Computational [ ]
  Biology [ ]
- Proteomics [ ]
- Genomics [ ]
- Gene Mapping [ ]
- Clinical Genetics [ ]

Medical Science [ ]
- Cardiovascular [ ]
- Diabetes [ ]
- Endocrinology [ ]
- Imaging [ ]
- Obstetrics/Gynaecology [ ]
- Oncology [ ]
- Pharmacology [ ]
- Psychiatry [ ]

Non-Profit [ ]

Architecture [ ]

Business/Management [ ]

Computer Science [ ]
- Database/Data Warehouse [ ]
- Internet Business [ ]
- Networking [ ]
- Programming/Software [ ]
  Development [ ]
- Object Technology [ ]

Engineering [ ]
- Civil [ ]
- Communications Technology [ ]
- Electronic [ ]
- Environmental [ ]
- Industrial [ ]
- Mechanical [ ]

Finance/Investing [ ]
- Economics [ ]
- Institutional [ ]
- Personal Finance [ ]

Life Science [ ]

Landscape Architecture [ ]

Mathematics/Statistics [ ]

Manufacturing [ ]

Material Science [ ]

Psychology [ ]
- Clinical [ ]
- Forensic [ ]
- Social & Personality [ ]
- Health & Sport [ ]
- Cognitive [ ]
- Organizational [ ]
- Developmental and Special Ed [ ]
- Child Welfare [ ]
- Self-Help [ ]

Physics/Physical Science [ ]

[ ] I confirm that I am a Wiley Author/Editor/Contributor/Editorial Board Member of the following publications:

SIGNATURE: ……

PLEASE COMPLETE THE FOLLOWING DETAILS IN BLOCK CAPITALS:

TITLE AND NAME: (e.g. Mr, Mrs, Dr) ………………………………………………………………………………

JOB TITLE: ……

DEPARTMENT: ……..

COMPANY/INSTITUTION: …………………………………………………………………………………………………

ADDRESS: ……….

………

………

………

TOWN/CITY: ………

COUNTY/STATE: …….

COUNTRY: ……

POSTCODE/ZIP CODE: ……………………………………………………………………………………………………

DAYTIME TEL: ……

FAX: ………

E-MAIL: ………

YOUR PERSONAL DATA
We, John Wiley & Sons Ltd, will use the information you have provided to fulfil your request. In addition, we would like to:

1. Use your information to keep you informed by post, e-mail or telephone of titles and offers of interest to you and available from us or other Wiley Group companies worldwide, and may supply your details to members of the Wiley Group for this purpose.

   [ ] Please tick the box if you do not wish to receive this information

2. Share your information with other carefully selected companies so that they may contact you by post, fax or e-mail with details of titles and offers that may be of interest to you.

   [ ] Please tick the box if you do not wish to receive this information.

If, at any time, you wish to stop receiving information, please contact the Database Group (databasegroup@wiley.co.uk) at John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, UK.

E-MAIL ALERTING SERVICE
We offer an information service on our product ranges via e-mail. If you do not wish to receive information and offers from John Wiley companies worldwide via e-mail, please tick the box [ ].

This offer is exclusive to Wiley Authors, Editors, Contributors and Editorial Board Members in acquiring books (excluding encyclopaedias and major reference works) for their personal use. There should be no resale through any channel. The offer is subject to stock availability and may not be applied retrospectively. This entitlement cannot be used in conjunction with any other special offer. Wiley reserves the right to vary the terms of the offer at any time.

Ref: S001W