Project Description

Gerald Lüttgen, Bamberg & Walter Vogler, Augsburg

1 State of the Art and Preliminary Work

Software engineers increasingly employ *heterogeneous* notations for specifying complex software systems, which mix declarative-logic and operational specification styles. This is because requirements are frequently stated in restricted forms of natural language or simple spreadsheets (*declarative*) and combined with design elements in the form of component interfaces (*declarative, also operational*) or state machines (*operational*). The popular Unified Modeling Language (UML) supports such heterogeneous notations, e.g., with its class diagrams, its state machine variant and its Object Constraint Language (OCL) [67]. A recent academic focus wrt. heterogeneity is on modelling formalisms for *reactive systems* software, including hierarchical state machines and protocol interface theories. For example, *Contractual State Machines* [33] uses logic constraints to express contracts between a reactive system and its environment and between the system's components, and employs patterns that allow designers to systematically refactor and refine components. Another example are *Modal Interfaces* [61] where a system component may be required to implement several interfaces simultaneously, i.e., to satisfy their conjunction.

One concrete application example where such heterogenous modelling formalisms are useful is the development of mode logics in aircraft control systems. Mode logics monitor an aircraft's sensors, e.g., speed and altitude, and select the control laws that in turn compute values for adjusting the aircraft's actuators, e.g., its elevators. Engineers typically develop such safety-critical systems stepwise [55]. In a first step, system requirements are stated in controlled English, the basic ones of which describe modes and their behaviour via declarative constraints. For example, a mode "isolate" may be specified as "the actuator is turned off indefinitely", and several concurrent modes may be governed by the invariant "exactly one actuator per elevator is active at any one time" [55]. Additionally, more advanced requirements for fault detection, isolation and recovery (FDIR) are specified. In a second step, the basic requirements are casted manually into a simple state-machine controller, e.g., using Mathwork's popular Simulink/Stateflow tool. This state machine is then successively extended in a third and fourth step by adding components/states and transitions that realize the FDIR requirements. Finally, in a fifth step, the system is tested to ensure that its requirements have been met. Hence, the engineers have followed a component-based method of stepwise refinement, successively trading declarative content (requirements) for operational content (design, in the form of states and transitions). Unfortunately, the lack of formal support for heterogeneous specification notations and associated component-based refinement means that this method is applied informally and not managed by tools. This is why verification activities are only conducted at the end, and not already along the way.

Indeed, the foundations of heterogeneous specification notations, which are the focus of this project, are relatively immature. Related work often avoids mixing operational and logic specification styles by translating one style into the other. For example, logic content may be translated into operational content, such as in Kurshan's work on ω*-automata* [39], which employs trace inclusion for refinement. Trace inclusion, however, is insensitive to deadlock and thus inadequate in the presence of concurrency. Dually, operational content may be stated as logic formulas, as is done by Lamport in [40] where implication is refinement. A similar approach is followed in Hoare and He's UTP [35], which has been used in the literature to give semantics to languages that consider both behavioural aspects and data aspects of systems, e.g., to Woodcock and Cavalcanti's Circus [59] which combines the process algebra CSP with the model-based language Z. However, UTP does not treat inconsistent operational content as logically false.

This project shall explore novel ways of deeply integrating (temporal) logic operators in concurrent state machine formalisms, and develop notions of refinement that are (i) deadlock sensitive, (ii) compositional, (iii) fully-abstract wrt. logic consistency, and (iv) compatible with logic satisfaction. These shall be applied to new variants of Modal Interfaces and Contractual State Machines, thus strengthening their foundations. The following describes preliminary work on our framework *Logic LTS*, and related work on interface theories (in particular *Modal Interfaces*), *Statecharts* based formalisms, and tool support (see Figure 1 for an overview).

Logic LTS. A seminal step towards a mixed operational and logic setting was taken by Olderog in [58], where process-algebraic constructs are combined with *trace formulas* and refinement is based on failure semantics. Here, trace formulas can serve as processes, but not vice versa. Thus, Olderog does not support the unrestricted mixing of specification styles. This shortcoming has been addressed by us in our setting of *Logic Labelled Transition Systems* (Logic LTS) [51, 52], which is based on prior – but mathematically less robust – work of the first applicant [14, 15].

Logic LTS combines operational and logic styles of specification within a unified framework. It includes operational, i.e., process-algebraic operators such as parallel composition and hiding, and the propositional-logic operators conjunction and disjunction. Logic LTS extends labelled transition systems by an *inconsistency* predicate on states, where an inconsistent state, or process, denotes empty behaviour that cannot be implemented. Inconsistencies may arise when conjunctively composing processes with different *ready sets*, i.e., initial action sets [51]. The refinement preorder we adapted for Logic LTS is a variant of *ready simulation* [9], which is compositional regarding all operational and logic operators of interest. A specification that cannot be refined further has deterministic behaviour: we call such specifications *concrete implementations* and, in the context of component-based design, these can be regarded as implemented components. Our ready simulation is sufficiently expressive regarding the specification of concrete implementations [26]. It is also fully abstract wrt. a reference preorder that preserves consistency when refining specifications [52], i.e., it is the coarsest compositional preorder wrt. parallel composition and conjunction when taking consistency into account. Full-abstraction results are a repeating theme in the applicants' research [16, 50, 62] and testify to the *optimality* of a mathematical setting. Such results have not been attempted in any work directly related to this research proposal.

We have extended Logic LTS in [53] by embedding a temporal logic for specifying safety properties. The chosen branching-time logic allows one to state properties regarding the enabledness of actions, using standard temporal operators such as *always* and *unless*, which are shown to be compositional for ready simulation. The embedding is conservative in that ready simulation, when restricted to pairs consisting of a process and a temporal formula, coincides with the logic's satisfaction relation. Moreover, ready simulation, when restricted to formulas, is entailment. The extended setting is unique in that it lends itself to *freely* mixing operational and temporal-logic styles of specification, with the fully-abstract ready simulation facilitating component-based design and analysis via compositional refinement checking. The setting's practical utility has been demonstrated in [53] by modelling and analysing a simple mode logic.

Notably, Fecher and Grabe [27] also employ ready simulation as implementation relation and define a specific satisfaction for temporal-logic formulas similar to our approach. In [27], whenever a process satisfies a formula, each concrete implementation of the process satisfies the formula; however, Fecher and Grabe do not allow the free mixing of operators.

Große-Rhode considers heterogeneous specifications where different viewpoints on a system are described in different formalisms [31]. These formalisms are then related via a category of transformation systems that plays the role of our Logic LTS. Refinement in terms of process-algebraic behavioural relations – a core notion in our approach – is not considered, and neither are declarative behavioural constructs such as temporal-logic operators.

Interface Theories. A key notion in component-based software development is the one of *interface*, which is an abstract description of a component's behaviour. Interfaces can be composed in parallel and refined in a compositional manner. Characteristically, an interface formulates as-

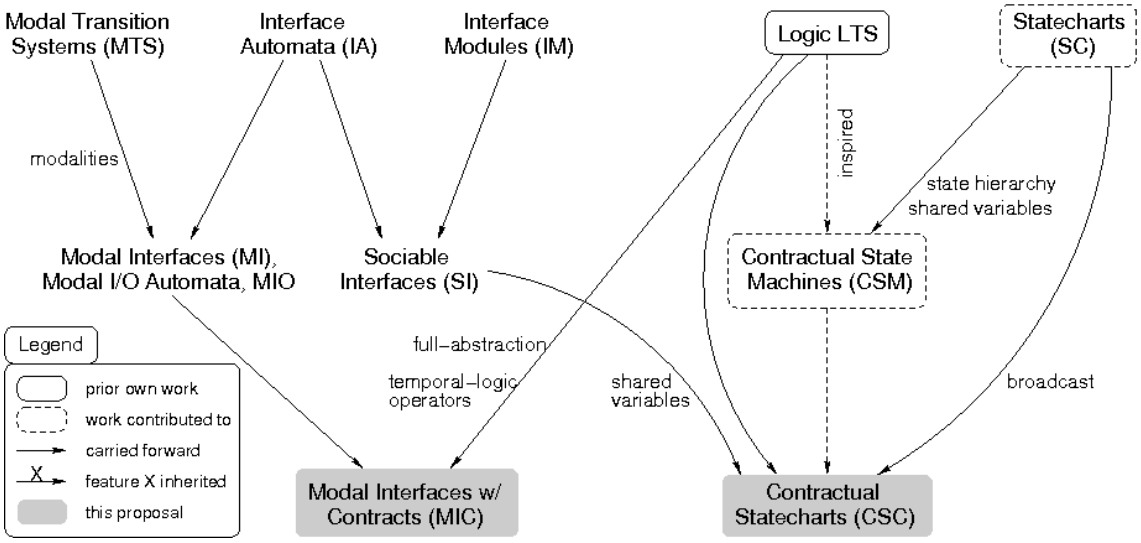

Figure 1: Overview of prior, related and proposed work.

sumptions on the inputs from its environment and guarantees regarding its outputs. Care has to be taken in case one interface in a composition violates the assumptions of another component in some reachable state. Usually, the treatment is *optimistic*, i.e., the composition is defined such that the assumptions on the environment ensure that entering such *incompatible* states can be avoided. Two interfaces are called incompatible if no such helpful environment exists. Naturally, compatibility of interfaces should be preserved by refinement. Interfaces may also be understood as behavioural types such that refinement is type-checking, as is done in Ptolemy II [46].

An interface theory in the above sense was first developed by de Alfaro and Henzinger in an automaton setting with input and output actions, called *Interface Automata* (IA) [19]. They define refinement via an *alternating simulation*, i.e., interface M refines interface N whenever (i) M can simulate all inputs prescribed by N and (ii) N can simulate all outputs of M. In other words, M must accept at least as many inputs and provide at most as many outputs as N. Originally, de Alfaro and Henzinger introduced a slightly weaker refinement relation [21] that is close to ready simulation for Logic LTS. The first step for defining parallel composition on Interface Automata is to consider the automaton product; a state in this product is incompatible if one interface performs an output that is not expected by the other. Now, parallel composition is obtained by *pruning* this product, i.e., by removing all states from which an incompatible configuration can be reached via outputs of the composition. The intuition behind this pruning is that outputs cannot be controlled by the environment and, thus, the environment must refrain from certain inputs to avoid incompatibilities. Pruning for parallel composition is quite similar to backward propagation for conjunction in Logic LTS, where one deals with inconsistencies rather than incompatibilities.

Interestingly, the idea of preserving inputs and not adding outputs in a refinement step, and the idea of pruning, can already be found in Dill's language-based – and thus not deadlock-sensitive – setting for modelling asynchronous circuits [23]. A similar approach to the one of Interface Automata was developed in Petri net models of asynchronous circuits, in order to define when a collection of STGs – i.e., Petri nets labelled with signal edges – is a correct decomposition of a specification STG N [63]. STG-refinement requires that the collection simulates the inputs and bisimulates the outputs of N. Thus, one cannot simply relinquish all outputs in a refinement step as is the case for Interface Automata. Furthermore, STG-refinement ensures that the collection does not get into an incompatible state if the environment obeys the assumptions specified by N.

Interface theories have also been developed for web services [8], and in settings with shared-variable communication where one speaks of *Interface Modules* (IM) rather than Interface Automata [13, 24]. In *Sociable Interfaces* (SI) [18], for example, interfaces synchronise via actions

and change data values during output actions. Assumptions on the environment are formulated for input actions – essentially as pre- and post-conditions on data states –, and interfaces are deterministic for inputs. Variables can be written by several concurrent interfaces. To enhance compositional reasoning, SI enforces that an interface is informed about value changes of variables of interest. Moreover, Doyen et al. [24] study a conjunction operator for another interface theory involving shared-variable communication in order to reason about *combined* specifications, i.e., when a component implements several interfaces, for component reuse. However, parallel composition in [24] permits only limited communication, if at all, and no full-abstraction result is proved. As an aside, the conjoining of specifications has also been studied in TLA, a temporal logic of actions with shared-memory communication [1].

Modal Interfaces. The framework of *Modal Transition Systems* (MTS) [41] was introduced by Larsen et al. and refines plain LTS by distinguishing between may- and must-transitions. The employed refinement preorder, called *modal refinement*, ensures that a refinement implements all must-transitions of a specification and never adds any transition that is not allowed by a may-transition of the specification. This is formally defined via two-way simulations that are strong bisimulations [57] for those transition systems for which may- and must-transitions coincide. Modal refinement is also characterised via a temporal logic in the style of Hennessy and Milner in [41], and is related to ready simulation in Logic LTS in [53]. Larsen and Xinxin also consider a generalization of *quotienting*, namely solving equation systems of process-algebraic terms up to strong bisimilarity [45]. They construct a so-called *disjunctive* MTS whose concrete implementations are exactly the solutions of the equation system. Quotienting facilitates incremental, component-based design and analysis, and is thus of relevance to us.

Larsen also defined a conjunction operator on MTS which, however, results in systems that violate the MTS requirement that any must-transition is also a may-transition [41]; related decision problems such as the existence of a common implementation, have been studied in [3]. In the context of an MTS-based proof methodology, Larsen and others later restricted conjunction to *independent* specifications that avoid inconsistencies [44]. Their interest was in a proof methodology for an MTS-variant of the process algebra CCS [57], where parallel composition and conjunction can be mixed more freely than in [58]; in particular, conjunction is shown to distribute over parallel composition. Larsen et al. also employ a typical pattern of MTS within their proof methodology that corresponds to simple invariance formulas in the temporal logic CTL [25]; in contrast to Logic LTS, an algebraic theory of mixing operational and logic operators is not considered in [44].

The idea of may- and must-transitions may sensibly be adopted to Logic LTS for specifying ranges of ready sets more compactly, as is shown by us in [53]. Notably, it may also be used to address a major shortcoming of Interface Automata, namely that an interface can always be refined by a do-nothing interface that accepts all inputs and provides no outputs. For example, Nyman et al. do so in [42] by extending MTS by explicitly distinguishing input and output actions. For these *Modal I/O Automata*, they adopt the concepts of compatibility and pruning from Interface Automata, and then show that Interface Automata can be embedded in Modal I/O Automata so that alternating simulation and modal refinement coincide. However, no conjunction is defined for Modal I/O Automata; we have recently remedied this lacking in [54] for the modal refinement preorder of [42], though this preorder requires further scrutinizing. Caillaud et al. [61] consider modal extensions of interfaces, too, which they call *Modal Interfaces*. In contrast to Modal I/O Automata, they study deterministic automata only, where alternating simulation becomes a trace inclusion, and consider conjunction and quotienting, as well as parallel composition and compatibility with pruning. Again, no full-abstraction result is established, nor are standard logic laws proved.

Hennicker et al. also investigated Modal I/O Automata but, in contrast to [42] and the Interface Automata of [19], adopted a *pessimistic* view on compatibility and called the resulting setting *MIO* [5]. Here, 'pessimistic' means that, if the composition of two MIOs might reach an incompatible state, then they are deemed incompatible and thus cannot be composed. Hennicker et al. then studied variations of compatibility and alternating-simulation-based refinement [6], and extended their setting to include data [4]. As in Sociable Interfaces [18], transitions are accompanied

by pre- and post-conditions on data states.

Finally, it shall be mentioned that temporal logic has also been introduced to Modal Interfaces. Feuillade and Pinchinat [28] define a logic that is similar to ours for Logic LTS [53], and prove that this logic and MTS are equally expressive wrt. concrete implementations; this also involves translating formulas into MTSs. In contrast to our work, their setting is not mixed, does not consider nondeterministic systems and specifications, and does not include a refinement relation. Indeed, a unique feature of Logic LTS in comparison to related work is that its refinement relation subsumes the standard temporal-logic satisfaction relation, thereby naturally embedding temporal logic in a state-machine-based specification language.

Statecharts. Statecharts (SC) is a family of visual languages for specifying reactive and/or object-oriented systems, which extend finite-state machines by mechanisms for state hierarchy and for concurrency via event broadcast and shared-variable communication [34], thus facilitating the *compact* specification of complex systems. Statecharts and their semantics have been investigated extensively by the first applicant, e.g., in [22, 47]. Most well-known is the Statecharts dialect of the UML, called *UML state machines*. Statecharts have also been extended with various logics; for example, Galloway and Toyn [30] augment Statecharts to include assertions at states, which are then used for the formal validation of Statecharts specifications via theorem proving. In an article misleadingly entitled "Extending Statecharts with Temporal Logic" [65], Sowmya and Ramesh translate Statecharts into a temporal logic based on first-order predicate calculus, for which they also construct an axiomatic proof system. In contrast to [30], their goal was not only to reason about the behaviour of Statecharts but also about their structure. However, no mixing of Statecharts with (temporal) logic operators is considered in [30] and [65].

Such a mixing was instead the focus of an industry-supported research project at the University of York, England, which was carried out between 2007 and 2010 [49], and of which the first author of this proposal was a co-investigator until his departure from York in Spring 2009. The strategic goal of that project was to improve the theoretical basis and tool support for the design languages and methodologies that are widely used for building avionics and aerospace systems. Existing languages, and in particular Statecharts, lacked in expressiveness and tool support for refinement-based designs as practiced by engineers. Accordingly, the project had intended to combine Statecharts with temporal logics, so as to add declarative constraints – or contracts (cf. [56]) – to states and transitions. However, this was never carried out; instead, our Logic LTS framework was enriched by state hierarchy and global variables, and called *Contractual State Machines* (CSM) [33]. This was then taken as the foundation for developing a set of *refinement patterns* that capture standard rules for translating between operational and logic styles of specification. In addition, a tool assisting in the application of refinement patterns was implemented, but no compositional checker for computing ready simulation on given CSMs was built.

Tools. The behavioural interface approach to component-based development is supported by a variety of academic tools. For example, the *Ptolemy II* tool [46] includes an editor and a simulator for manipulating and animating Interface Automata, respectively. It also offers algorithms for checking compatibility and refinement. The *TICC* tool [2] has similar functionality but targets Sociable Interfaces. It employs decision-diagram-based techniques for storing and manipulating complex state spaces and a game-based computation of alternating simulation. Again, this functionality is also available for MIO interfaces via the *MIO Workbench* [6], which is implemented within the Eclipse framework [29]. However, it currently supports only action-based MIOs and no data states. Last, but not least, York's toolset for Contractual State Machines [33] combines an editor and a simulator with an assistant for applying refinement patterns. It is implemented within Eclipse, too, and the pattern applier is realised using model transformation techniques.

1.1 Project-Related Publications

1.1.1 Articles Published by Outlets with Scientific Quality Assurance

[PSP1] R. Cleaveland and G. Lüttgen. *A semantic theory for heterogeneous system design.* In 20th Intl. Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2000), vol. 1974 of LNCS, pp. 312–324, 2000. Springer.

[PSP2] H. Fecher, D. de Frutos-Escrig, G. Lüttgen and H. Schmidt. *On the expressiveness of refinement settings.* In 3rd Intl. Conf. on Fundamentals of Software Engineering (FSEN 2009), vol. 5961 of LNCS, pp. 276–291, 2009. Springer.

[PSP3] G. Lüttgen and W. Vogler. *Conjunction on processes: Full-abstraction via ready-tree semantics.* Theoret. Comp. Sc., 373(1-2):19–40, 2007. An extended abstract appeared in 9th Intl. Conf. on Foundations of Software Science and Computation Structures (FOSSACS 2006), vol. 3921 of LNCS, pp. 261–276, 2006, Springer.

[PSP4] G. Lüttgen and W. Vogler. *Ready simulation for concurrency: It's logical!* Inform. and Comput., 208:845–867, 2010. An extended abstract appeared in 34th Intl. Coll. on Automata, Languages and Programming (ICALP 2007), vol. 4596 of LNCS, pp. 752–763, 2007, Springer.

[PSP5] G. Lüttgen and W. Vogler. *Safe reasoning with Logic LTS.* Theoret. Comp. Sc., 412(28): 3337–3357, 2011. An extended abstract appeared in 35th Conf. on Current Trends in Theory and Practice of Computer Science (SOFSEM 2009), vol. 5404 of LNCS, pp. 376–387, 2009, Springer.

[PSP6] G. Lüttgen, M. von der Beeck and R. Cleaveland. *Statecharts via process algebra.* In 10th Intl. Conf. on Concurrency Theory (CONCUR '99), vol. 1664 of LNCS, pp. 399–414, 1999. Springer.

[PSP7] A. Rensink and W. Vogler. *Fair testing.* Inform. and Comput., 205(2):125 –198, 2007.

[PSP8] M. Schäfer and W. Vogler. *Component refinement and CSC-solving for STG decomposition.* Theoret. Comp. Sc., 388(1-3):243–266, 2007.

1.1.2 Other Publications

[PSP9] W.-P. De Roever, G. Lüttgen and M. Mendler. *What is in a step: New perspectives on a classical question.* In Z. Manna and D. Peled, eds., Pnueli Festschrift, vol. 6200 of LNCS, pp. 370–399, 2010. Springer.

2 Objectives and Work Programme

2.1 Anticipated Total Duration of the Project

The project has not yet started. Its duration shall be 36 months, for which DFG funding is sought.

2.2 Objectives

Aim. The project's aim is to significantly enhance the theoretical foundations and practical utility of those heterogeneous specification formalisms for reactive systems that mix operational and logic styles of specification. Our particular interest is in formalisms that add logic elements to state machine notations, such as Caillaud et al.'s *Modal Interfaces* (MI) [61], which emphasise

logic conjunction for composing specifications, and Paige et al.'s *Contractual State Machines* (CSM) [33], which extend a simple hierarchical state machine notation by temporal-logic operators. This will lead to mathematically well-founded, yet practical theories of *Modal Interfaces with Contracts* (MIC) and *Contractual Statecharts* (CSC), respectively, as is highlighted in Figure 1.

Adding logic specification facilities to state-machine formalisms has distinct advantages, even if the expressiveness of the formalism is not increased. Firstly, logic formulas enable more compact and concise specifications. Secondly, it may simply be adequate to describe system properties, e.g., mutually exclusive behaviour, with logics instead of providing partial operational descriptions. This is particularly true in the context of the component-based development of reactive systems. While the rough architecture of a system is often known from the outset, typically only few of its components can initially be described operationally. Components that are identified later in the design process, as well as off-the-shelf components, may instead be specified by *contracts* [56] which express the components' assumptions on, and its guarantees to related components within the architecture and the system environment. Conceptually, contracts are nothing else than logic formulas that must be satisfied by any implementation. Therefore, a component-respecting, compositional approach to refining and analysing such specifications is necessary, whose foundations will be developed and applied in this proposed project.

Objectives. The concrete objectives of this project centre around the following themes:

1. *Study of consistency & compatibility notions (WP A1+2, WP BA1+2).*

   In heterogeneous specification formalisms of the kind mentioned above, key semantic notions when composing specifications conjunctively and in parallel are *consistency* [51] and *compatibility* [19], respectively. The conjunction operator of a formalism is closely related to the employed notion of refinement on specifications, which in turn is influenced by the choice of parallel composition [52]. Consequently, this project shall begin by studying consistency and compatibility notions implied by popular parallel composition operators.

2. *Development & evaluation of fully-abstract semantic theories (WP A3, WP BA3, WP 4).*

   The findings on consistency and compatibility shall provide the foundations for developing semantic theories that involve conjunction, parallel composition and refinement, and that are applicable to MI and CSM. The novelty here is that the refinement preorder of each theory shall be *fully-abstract* wrt. preserving deadlock, compatibility and the theory's operators, thus providing a strong mathematical foundation for component-based specification and development, which the MI and CSM formalisms are currently lacking. The theories shall be completed by studying algebraic laws of various operational and logic operators, as well as laws mixing those operators. The theories shall also be carefully compared and contrasted to related work.

3. *Application to MI: Extending MI by contracts → MIC (WP A5).*

   The theoretical results obtained will be the cornerstone for extending MI by temporal-logic operators, so as to be able to express safety properties [53]. One may view this as a language mechanism for phrasing logic contracts that govern operational behaviour, which will permit system designers employing interfaces to capture specifications more concisely. In addition, *MI with Contracts* (MIC) shall be equipped with a quotienting operator that enables incremental development and verification [61]. MIC shall be given a strong semantic theory along the lines outlined in Item 2 above, which will also put the existing theory of MI [61] on a stronger footing via a full-abstraction result.

4. *Application to CSM: Defining the formal semantics of CSM/CSC (WP BA5).*

   Our theoretical results shall be applied further to alter CSM to the originally intended Statecharts dialect [49], by replacing its process-algebraic parallel composition with the broadcasting parallel composition of Statecharts [34], and equipping it with a formal semantics.

This will fix the shortcomings of [33] in a mathematically elegant, fully-abstract way. It will also lead to a specification formalism, to which we refer as *Contractual StateCharts* (CSC), which has similarities to MIC except for the different concept of parallel composition.

5. *Tool support for MIC & case studies (WP A6–8).*

   Interface theories will only be adopted in the practice of reactive systems development if they are accompanied by tools. Hence, prototypic tool support for MIC shall be provided in form of a toolset built within the Eclipse framework [29] and consisting of four components. A *visual editor* and a *simulator* will allow designers to draw MIC interface specifications and to animate their behaviour, respectively. At the centre of our attention, however, shall be the development and implementation of *compatibility* and *refinement checkers* for determining whether two specifications are compatible and, respectively, whether one specification formally refines another. The MIC framework shall be evaluated by means of case studies as described below.

   Regarding the automation of our refinement methodology, we will rely on and adapt existing, state-of-the-art automated verification technologies. The computational challenges of refinement checking for interface theories are currently being investigated by Dr. Benoit Caillaud at IRISA Rennes, France, who is a collaborator to our proposed research project. Therefore, investigations into this topic are not part of our proposal, but our research will certainly be informed by Dr. Caillaud's work.

6. *Tool support for CSC & case studies (WP BA6–8).*

   This objective is analogous to the previous one, except that it focuses on CSC rather than MIC. CSM is already supported by an editor and simulator implemented within Eclipse [29], and with a facility for manually refining system specifications via pre-defined templates using model transformation technologies [37]. This existing toolset shall be adapted to CSC and then be supplemented with an automated CSC refinement checker. The CSC framework and the refinement checker shall be evaluated via case studies as described below.

Outcome. The project's outcome will be mathematically robust, yet practical theories of refinement-driven reactive systems development. Thanks to the integration of logic contracts, state-based formalisms for specifying reactive systems will become more flexible and permit more concise specifications. The basis for this will be the concurrency-theoretic advancements in heterogeneous system specifications mixing operational and logic operators, which are at the heart of this project. The potential impact will be demonstrated via realistic case studies exercising our prototypic toolsets, which will lay the basis for a future technology transfer.

2.3 Work Programme incl. Proposed Research Methods

The proposed programme of work involves 15 work packages. The work packages related to MIC (WP A1–A8) will chiefly be conducted at Augsburg, while those related to CSC (WP BA1–BA8) will be led by Bamberg. Obviously, there are many parallels between our proposed work on MIC and CSC, which will require frequent interactions between the research teams at Augsburg and Bamberg. The scheduling of all work packages and the involvement of our international collaborators is depicted in Figure 2.

WP A1, BA1: Orientation. The project shall start off by giving the hired Research Assistants the opportunity to familiarise themselves with the relevant literature in the field (cf. Section 1), including the applicants' prior work on Logic LTS [51, 52, 53]. WP A1 will be conducted at Augsburg and focus on interface theories and foremost on Modal Interfaces (MI) [61], while WP BA1 will be carried out at Bamberg and concentrate on Statecharts-like formalisms and in particular

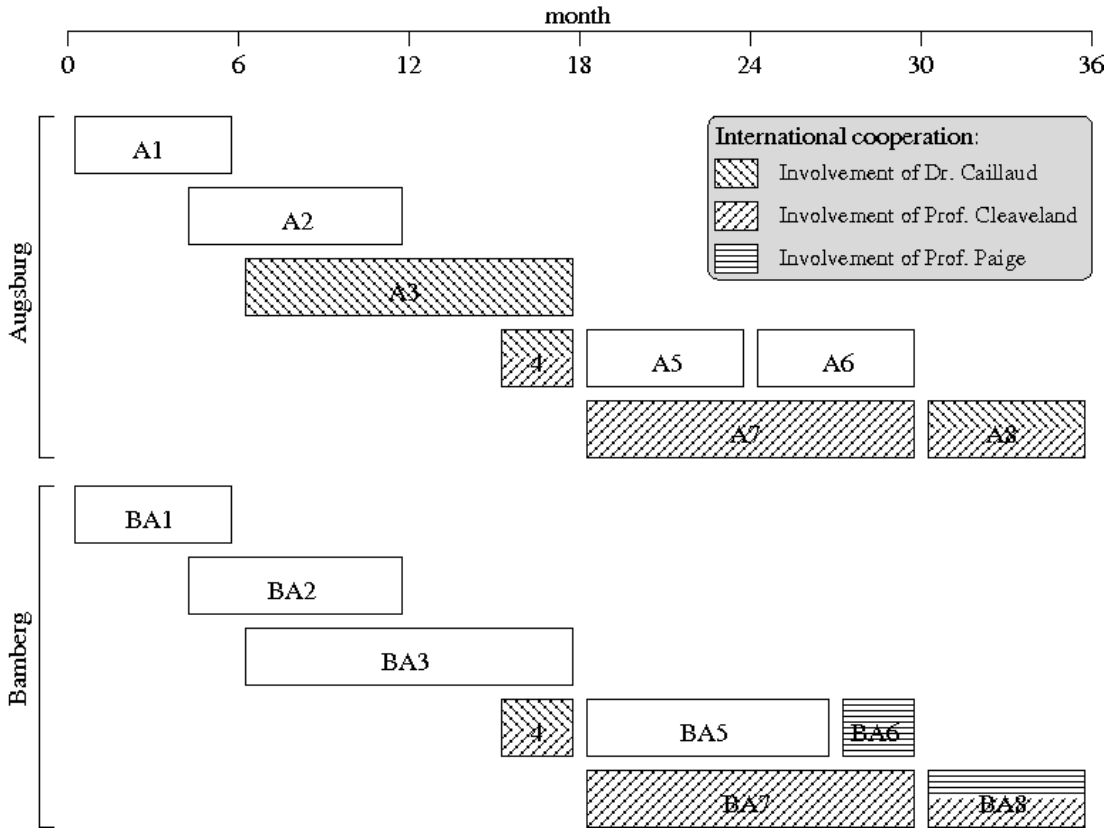

Figure 2: Proposed scheduling of work packages.

Contractual State Machines (CSM) [33]. Of interest to us is, e.g., shared-memory communication which is a common feature of interface theories and Statecharts; especially, we wish to see how the treatments of data in the MIO approach [5] and in Sociable Interfaces [18] compare.

WP A2, BA2: Studying compatibility & consistency notions. These work packages shall study compatibility and consistency notions implied by popular parallel composition operators, ranging from operators that rely on communication via synchronising actions [11, 57], to ones that distinguish between input and output actions [19, 60] or have Mealy-style input/output actions [34], to those that communicate via shared variables [13], to operators that mix some of these concepts [18]. Our goal is to revisit our work on CSP-style communication in Logic LTS [52] and to define conjunction in settings with shared-memory communication (WP A2) and broadcast communication (WP BA2), which is relevant to Statecharts dialects, as well as in settings involving parallel composition in the style of Interface Automata (WP A2).

Essential to our approach is the definition of conjunction on processes, which will be based on a notion of immediate inconsistency combined with backward propagation. The idea of immediate inconsistency is simply that two processes are inconsistent if they do not have a common implementation. Although the notion of implementation is not yet defined at this stage, this observation can give information on how the immediate activities of the two processes must be related, since an implementation must not introduce new deadlocks or incompatibilities in any environment.

WP A2 and BA2 shall study requirements for avoiding new deadlocks and incompatibilities in a shared-memory setting and, respectively, in a setting with broadcast communication. These scenarios have in common that communication is non-blocking for the sender and the writer, respectively, in contrast to the communication scheme investigated by us in [52]. In the broadcast scenario, there are typically no incompatibilities since unexpected messages are simply ignored,

and deadlock means that no further messages are sent. In the shared-memory scenario, there may be read-write and write-write conflicts and one may also add assumptions on values written [4], and deadlock arises when no further action is enabled. Both scenarios can be extended by transition modalities [41], which may influence the notions of inconsistency and incompatibility.

WP A3, BA3: Developing fully-abstract semantic theories. Here, we shall develop full-fledged semantic theories around our conjunction and parallel composition operators studied in WP A2 and BA2. Similar to our prior work on Logic LTS, we will characterise refinement preorders presumably based on variants of ready simulation, and prove these preorders *compositional* for our operators and *fully-abstract* relative to a naive refinement preorder that considers immediate inconsistency and deadlock only. This will be done for broadcasting systems (WP BA3) and shared-memory systems (WP A3) separately, and these results will be combined to a treatment of CSM (WP BA3), thus providing the groundwork for WP BA5. Furthermore, a refinement preorder will also be characterised for Modal Interfaces (WP A3), as a starting point for WP A5, for which we will consult with Dr. Caillaud who is an expert on interface theories.

Then, we will check the desirables for each of these settings. This will firstly include a sanity check whether our conjunction operators are indeed conjunctions in the relevant settings, i.e., whether a specification refines a conjunction of specifications if and only if it refines each specification. Secondly, we will prove compositionality for disjunction and for operational connectives like choice and action scoping. Thirdly, standard Boolean *laws* involving disjunction and conjunction will be verified. We will also consider laws that mix Boolean and operational operators, and the question whether our preorders are *thorough* [43], i.e., whether one specification refines another if and only if each of its concrete implementations refines the other specification. If one of these checks should fail, this would indicate a shortcoming in the definition of conjunction. Thus, WP A2/BA2 and WP A3/BA3 are interleaved in our schedule of work (cf. Figure 2).

WP 4: Comparing our theories to related work. This work package shall compare our conjunction operators and refinement preorders to related work, and reflect on the algebraic laws obtained. The related work of importance to us comprises (i) Interface Automata [19], (ii) Interface Modules [13, 20] and Sociable Interfaces [18], (iii) Modal I/O Automata [42], Modal Interfaces [61] and MIO [5], and (iv) Logic LTS [52]. It will be interesting to see whether our approach justifies the preorders adopted in the literature for Interface Automata and Modal I/O Automata; note that the early papers on Interface Automata [19, 21] present two subtly different preorders. Also, correctness notions in digital circuits [23, 63, 66] are of relevance here. Another aspect we will study in this work package is the expressiveness of our theories, which will be measured via thoroughness as in [26]. Our experienced collaborators Prof. Cleaveland and Dr. Caillaud will assist us in the scientifically robust assessment of the novel theories developed by us in WP A3 and BA3. This is particularly important, given the quickly moving field of interface theories.

WP A5: Extending MI by contracts (MIC). This work package shall extend Modal I/O Automata, or Modal Interfaces (MI), via a facility for specifying temporal-logic contracts. This will lead to a truly heterogeneous and more practical interface theory that supports concise interface specifications, and which we call *Modal Interfaces with Contracts* (MIC). The results of WP A2, A3 and BA3 will provide the context for working out an elegant semantic theory for MIC. Specifically, we will (i) define a temporal logic for safety properties and an intuitive satisfaction relation between processes and formulas (contracts), (ii) translate contracts into MIC's operational fragment, and (iii) show that contract satisfaction coincides with MIC refinement. If time permits, we will also investigate the modelling of interesting further properties outside our contract language, such as contracts involving simple liveness conditions [64].

Since interface technologies are essential for incremental, component-based software development, we will study the quotienting problem for MIC parallel composition, which provides the theoretical foundation for this development approach. In contrast to the seminal work on quotienting of Larsen and Xinxin [45], we will consider inequations based, e.g., on ready simulation

and involving a different kind of parallel composition. As in [45], it might turn out to be necessary to extend MIC in order to express such quotients. In addition to quotienting for parallel composition, it may also be interesting to look at quotients for conjunction. Usually, these would simply be implications, so we will have to investigate how to add a general implication operator to MIC.

WP BA5: Defining a formal semantics of CSM/CSC. The *Contractual State Machine* language (CSM) [33] is a visual variant of our Logic LTS [53], with added state hierarchy and shared-memory communication. The aim of this work package is to transform CSM to a proper Statecharts dialect, as was originally intended by the CSM research project carried out at the University of York [49]. This involves changing the CSP-style parallel composition operator of Logic LTS to an event broadcast operator [60]. In addition, the York project treated (finite) data simply by providing a ground semantics, i.e., by encoding data values into states, which makes practical verification difficult due to state explosion. Here, we will replace this treatment by explicitly adding to transitions pre- and post-conditions in the form of logic formulas over atomic data propositions, such as value comparisons. This is inspired by Hennicker et al.'s handling of data for MIO [5] and also by Sociable Interfaces, which directly reflects the idea of contracts [56] and naturally supports compositional and symbolic reasoning. We thus call our Statecharts dialect *Contractual StateCharts* (CSC).

Our work of WP A3 and BA3 will serve as the semantic backbone for giving an operational semantics and a full semantic theory to CSC. It will need to be extended by a hierarchy operator as well as by operators for dealing with boundary-crossing transitions. Our main result of WP A3 and BA3, which establishes the compatibility of the logic satisfaction relation with the fully-abstract refinement preorder, then provides a formal basis for the compositional verification of CSC, i.e., for incremental verification along state boundaries and state hierarchies.

WP A6: Implementing an editor & a simulator for MIC. A first step towards tool support for MIC shall be (i) an editor for drawing MIC specifications and (ii) a simulator for executing such specifications according to the MIC semantics defined in WP A5. Since such tool components are standard, we wish to save development effort by not starting from scratch, but rather by adapting the editor and simulator of Hennicker et al.'s MIO Workbench [6], which is implemented within the Eclipse framework [29] and available to us in source code. We will use these components for validating the MIC semantics and when conducting WP A8.

WP BA6: Adapting the existing CSM toolset to CSC. Paige et al.'s toolset for CSM [33] consists of an editor, a simulator and a refinement pattern applier, is also implemented within Eclipse, and made available to us in source code by our collaborator Prof. Paige. This work package shall adapt the CSM editor and simulator to our CSC language, which primarily means replacing the CSP-style parallel operator with the broadcast parallel operator of Statecharts, as well as substituting the underlying semantics machinery with the one developed in WP BA5. In addition, the CSC way of handling data is different from CSM, as it will involve pre- and post-conditions on data states. Analogously to WP A6, the editor and simulator will be used for validating the CSC semantics and when conducting WP BA8.

WP A7, BA7: Developing refinement checkers for MIC & CSC. Key to our MIC and CSC toolsets will be algorithms for automatically checking whether one given specification refines a second specification, according to the variants of ready simulation introduced in WP A3 and BA3. We shall develop and implement such refinement checkers for MIC and CSC within Eclipse.

This can be done via partition-refinement techniques as proposed by Bloom and Paige in [10]. These techniques may be combined with advanced data structures such as the decision diagrams employed by the TICC tool [2], and also with decision procedures if data states occur in a specification [38], as is the case for our CSC language. Alternatively, ready simulation on two specifications may be computed by translating one specification into its characteristic temporal-

logic formula, in the sense of Ingolfsdottir and Steffen [36], thereby reducing refinement checking to model checking this characteristic formula against the other specification [68].

Our collaborator Prof. Cleaveland has co-developed several preorder checkers – conceptually and in practice [17] – and will assist us in deciding which implementation strategy will work best for MIC and CSC. We will also implement compatibility checking for MIC, which – like refinement checking – requires an analysis of the state spaces of the underlying specifications. In this context, we will also compare our MIC toolset to other interface-theory tools such as TICC [2], Ptolemy II [46] or the MIO Workbench [6], wrt. (i) usability and conciseness and (ii) the performance of MIC's compatibility and refinement checkers.

As an aside, we stress that we do not aim here at advancing automated verification algorithms, but rather adapt them to our refinement-checking needs. In addition, our proposal – while being connected to model checking since our refinement relations include temporal-logic satisfaction as a specialisation – does not aim at compositional model checking. This is because the latter requires an automatic decomposition of a system implementation and a temporal-logic formula, while our focus is on the composition of operational and declarative-logic specification styles.

WP A8, BA8: Evaluating the MIC & CSC frameworks via case studies. Our approaches to heterogeneous specification formalisms shall be evaluated by means of case studies, which will employ our MIC and CSC toolsets. We will focus on two case studies which, although below industrial size, will make a first connection of our foundational research results with practical aspects. This will help us transferring our technology to software engineers who can then build tool-supported and industrial-strength specification and design methodologies on top of it.

The first case study is in the avionics domain and concerns the stepwise design of a mode logic (WP A8). This will allow us to validate as to how far the informal approach to designing mode logics currently employed in practice – as described on the first page of this proposal – can be formalized using our theory and supported by our prototypic tools. Questions of interest are the following: (a) Are our declarative-logic extensions to automata notation sufficient and natural enough for concisely expressing a mode logic's requirements? If not, what modelling constructs are missing? (b) Are the successive informal refinement steps conducted by engineers when transitioning from declarative requirements to operational designs supported by our formal refinement preorders? If not, what aspects are too strong in, or not covered by, our preorders, and how would an improved preorder look like? (c) Is the current automated verification technology adapted by our tools mature enough to deal with such realistic examples? If not, which aspects require improvement? Mode logics are a good example for investigating these questions since their basic structures are simple, since they are focussed on control and require only simple data types, and since realistic examples of mode logics are available in the public domain [55]. Indeed, the idea for part of this grant proposal dates back to when the first investigator was working on the formal modelling and analysis of mode logics for NASA [48]. If time permits, we will conduct a larger avionics case study donated to our collaborator Prof. Paige and the University of York by BAE Systems. This case study is not in the public domain and involves an *Integrated Flight and Propulsion Control System* that has already been studied at York and is specified using Stateflow state machines and English-language constraints.

The second case study will exercise our CSC toolset and re-consider the *Harbour Docking* case study investigated by Lishan Harbird – a PhD graduate of Prof. Paige – for CSM in her recent PhD thesis [32] (WP BA8). This case study involves a system that manages ships waiting to dock at a harbour, tracks the entering of ships into the port to the quays, and prevents the double allocation of ships to quays. The design of this system is based on heterogeneous refinement patterns catalogued in the CSM toolset, which provide a systematic basis for gradually introducing detail as design decisions are made. The design process starts with a contract, then applies refinement and refactoring patterns until a completed state machine design is achieved. We will redo this case study using our CSC toolset which will, firstly, help us in validating the CSC semantics and the underlying semantic design decisions using the CSC simulator. Secondly, our refinement

checker will testify to the correctness of the refinement-pattern instantiations employed in the original case study [32]. Thirdly and most importantly, we will compare and contrast the 'small-step' refinement via refinement patterns as in the CSM toolset [33] to the 'large-step' refinement supported by our CSC refinement checker. The latter will allow us to address the question whether the small-step refinement implied by Harbird's patterns is indeed a practical alternative to the large refinement conducted by engineers today when designing control systems.

Lastly and if time permits, we will additionally re-consider Broy and Lamport's *Remote Procedure Call* case study that has been introduced in [12] and considered by Larsen et al. for MTS with conjunction in [44], within our heterogeneous interface formalism. It should be noted that our novel approach to heterogeneous specification will allow us to deal with application scenarios that have rarely been studied before; therefore, re-considering a known case study will have the flavour of conducting a new one.

2.4 – 2.6 Not Applicable

2.4 Data handling; 2.5 Other information; 2.6 Descriptions of proposed investigations involving experiments on humans, human materials or animals.

2.7 Information on Scientific and Financial Involvement of International Cooperation Partners

Not applicable; see Section 5.4.1 for information on international cooperation partners with whom we will be working together informally. These partners neither have applied nor will apply for funding with the DFG or a partner organization in the context of this project proposal.

3 Bibliography

[1] M. Abadi and L. Lamport. Conjoining specifications. *ACM TOPLAS*, 1(3):507–534, 1995.

[2] B.T. Adler, L. de Alfaro, L.D. da Silva, M. Faella, A. Legay, V. Raman, and P. Roy. TICC: A tool for interface compatibility and composition. In *CAV 2006*, vol. 4144 of *LNCS*, pp. 59–62. Springer, 2006.

[3] A. Antonik, M. Huth, K.G. Larsen, U. Nyman, and A. Wasowski. Complexity of decision problems for mixed and modal transition systems. In *FoSSaCS 2008*, vol. 4962 of *LNCS*, pp. 112–126. Springer, 2008.

[4] S. Bauer, R. Hennicker, and M. Bidoit. A modal interface theory with data constraints. In *SBMF 2010*, vol. 6527 of *LNCS*, pp. 80–95. Springer, 2010.

[5] S. Bauer, R. Hennicker, and M. Wirsing. Interface theories for concurrency and data. *Theoret. Comp. Sc.*, 412(28):3101–3121, 2011.

[6] S. Bauer, P. Mayer, A. Schroeder, and R. Hennicker. On weak modal compatibility, refinement, and the MIO Workbench. In *TACAS 2010*, vol. 6015 of *LNCS*, pp. 175–189. Springer, 2010.

[7] A. Benveniste, B. Caillaud, L.P. Carloni, P. Caspi, and A.L. Sangiovanni-Vincentelli. Composing heterogeneous reactive systems. *ACM Trans. Embedded Comput. Syst.*, 7(4), 2008.

[8] D. Beyer, A. Chakrabarti, and T.A. Henzinger. Web service interfaces. In *WWW 2005*, pp. 148–159. ACM, 2005.

[9] B. Bloom. *Ready Simulation, Bisimulation, and the Semantics of CCS-like Languages*. PhD thesis, MIT, 1990.

[10] B. Bloom and R. Paige. Transformational design and implementation of a new efficient solution to the ready simulation problem. *Sc. Comp. Progr.*, 24(3):189–220, 1995.

[11] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of communicating sequential processes. *J. ACM*, 31(3):560–599, 1984.

[12] M. Broy, S. Merz, and K. Spies, eds. *Formal Systems Specification: The RPC-Memory Specification Case Study*, vol. 1169 of *LNCS*. Springer, 1996.

[13] A. Chakrabarti, L. de Alfaro, T.A. Henzinger, and F.Y.C. Mang. Synchronous and bidirectional component interfaces. In *CAV 2002*, vol. 2404 of *LNCS*, pp. 414–427. Springer, 2002.

[14] R. Cleaveland and G. Lüttgen. A semantic theory for heterogeneous system design. In *FSTTCS 2000*, vol. 1974 of *LNCS*, pp. 312–324. Springer, 2000.

[15] R. Cleaveland and G. Lüttgen. A logical process calculus. In *EXPRESS 2002*, vol. 68,2 of *ENTCS*. Elsevier, 2002.

[16] R. Cleaveland, G. Lüttgen, and V. Natarajan. Priority in process algebra. In *Handbook of Process Algebra*, ch. 12, pp. 711–765. Elsevier, 2001.

[17] R. Cleaveland and O. Sokolsky. Equivalence and preorder checking for finite-state systems. In *Handbook of Process Algebra*, ch. 6, pp. 391–424. Elsevier, 2001.

[18] L. de Alfaro, L.D. da Silva, M. Faella, A. Legay, P. Roy, and M. Sorea. Sociable interfaces. In *FroCoS 2005*, vol. 3717 of *LNAI*, pp. 81–105. Springer, 2005.

[19] L. de Alfaro and T.A. Henzinger. Interface automata. In *FSE 2001*, pp. 109–120. ACM, 2001.

[20] L. de Alfaro and T.A. Henzinger. Interface theories for component-based design. In *EMSOFT 2001*, vol. 2211 of *LNCS*, pp. 148–165. Springer, 2001.

[21] L. de Alfaro and T.A. Henzinger. Interface-based design. In *Engineering Theories of Software-Intensive Systems*, vol. 195 of *NATO Science Series*. Springer, 2005.

[22] W.-P. de Roever, G. Lüttgen, and M. Mendler. What is in a step: New perspectives on a classical question. In *Pnueli Festschrift*, vol. 6200 of *LNCS*, pp. 370–399. Springer, 2010.

[23] D.L. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits*. MIT Press, 1989. ACM Distinguished Dissertation.

[24] L. Doyen, T.A. Henzinger, B. Jobstmann, and T. Petrov. Interface theories with component reuse. In *EMSOFT 2008*, pp. 79–88. ACM, 2008.

[25] E.A. Emerson. Temporal and modal logic. In *Handbook of Theoret. Comp. Sc.*, vol. B, pp. 995–1072. North-Holland, 1990.

[26] H. Fecher, D. de Frutos-Escrig, G. Lüttgen, and H. Schmidt. On the expressiveness of refinement settings. In *FSEN 2009*, vol. 5961 of *LNCS*, pp. 276–291. Springer, 2009.

[27] H. Fecher and I. Grabe. Finite abstract models for deterministic transition systems. In *FSEN 2007*, vol. 4767 of *LNCS*, pp. 1–16. Springer, 2007.

[28] G. Feuillade and S. Pinchinat. Modal specifications for the control theory of discrete event systems. *J. Discrete Event Dyn. Syst.*, 17:211–232, 2007.

[29] The Eclipse Foundation. Eclipse Modeling Framework. www.eclipse.org/modeling/emf/.

[30] A. Galloway and I. Toyn. Proving properties of Stateflow models using ISO Standard Z and CADiZ. In *ZB-2005*, vol. 3455 of *LNCS*, pp. 104–123. Springer, 2005.

[31] M. Große-Rhode. *Semantic Integration of Heterogeneous Software Specifications*. Monographs in Theoret. Comp. Sc. Springer, 2003.

[32] L. Harbird. *Model-Based Refinement of Contractual State Machines*. PhD thesis, U. York, 2012.

[33] L. Harbird, A. Galloway, and R.F. Paige. Towards a model-based refinement process for Contractual State Machines. In *IEEE Symp. on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, pp. 108–115. IEEE, 2010.

[34] D. Harel. Statecharts: A visual formalism for complex systems. *Sc. Comp. Progr.*, 8:231–274, 1987.

[35] C.A.R. Hoare and J. He. *Unifying Theories of Programming*. Prentice Hall, 1998.

[36] A. Ingolfsdottir and B. Steffen. Characteristic formulae. *Information and Control*, 110(1):149–163, 1994.

[37] D.S. Kolovos, R.F. Paige, F. Polack, and L.M. Rose. Update transformations in the small with the Epsilon wizard language. *J. Object Technology*, 6(9):53–69, 2007.

[38] D. Kroening and O. Strichman. *Decision Procedures: An Algorithmic Point of View*. Springer, 2008.

[39] R.P. Kurshan. *Computer-Aided Verification of Coordinating Processes: The Automata-Theoretic Approach*. Princeton Univ. Press, 1994.

[40] L. Lamport. The temporal logic of actions. *ACM TOPLAS*, 16(3):872–923, 1994.

[41] K.G. Larsen. Modal specifications. In *Automatic Verification Methods for Finite State Systems*, vol. 407 of *LNCS*, pp. 232–246. Springer, 1989.

[42] K.G. Larsen, U. Nyman, and A. Wasowski. Modal I/O automata for interface and product line theories. In *ESOP 2007*, vol. 4421 of *LNCS*, pp. 64–79. Springer, 2007.

[43] K.G. Larsen, U. Nyman, and A. Wasowski. On modal refinement and consistency. In *CONCUR 2007*, vol. 4703 of *LNCS*, pp. 105–119. Springer, 2007.

[44] K.G. Larsen, B. Steffen, and C. Weise. A constraint oriented proof methodology based on modal transition systems. In *TACAS '95*, vol. 1019 of *LNCS*, pp. 17–40. Springer, 1995.

[45] K.G. Larsen and L. Xinxin. Compositionality through an operational semantics of contexts. *J. Logic Computation*, 1(6):761–795, 1991.

[46] E.A. Lee and Y. Xiong. A behavioral type system and its application in Ptolemy II. *Formal Aspects of Computing*, 16(3):210–237, 2004.

[47] G. Lüttgen, M. von der Beeck, and R. Cleaveland. Statecharts via process algebra. In *CONCUR '99*, vol. 1664 of *LNCS*, pp. 399–414. Springer, 1999.

[48] G. Lüttgen and V. Carreño. Analyzing mode confusion via model checking. In *(SPIN '99)*, vol. 1680 of *LNCS*, pp. 120–135. Springer, 1999.

[49] G. Lüttgen and R.F. Paige. Refinement patterns for Contractual Statecharts, 2007. EPSRC grant poposal, no. EP/E034853/1.

[50] G. Lüttgen and W. Vogler. Bisimulation on speed: A unified approach. *Theoret. Comp. Sc.*, 360:209–227, 2006. An extended abstract appeared in FOSSACS 2005, vol. 3441 of LNCS, pp. 79–94, Springer, 2005.

[51] G. Lüttgen and W. Vogler. Conjunction on processes: Full-abstraction via ready-tree semantics. *Theoret. Comp. Sc.*, 373(1-2):19–40, 2007. An extended abstract appeared in FOSSACS 2006, vol. 3921 of LNCS, pp. 261–276, Springer, 2006.

[52] G. Lüttgen and W. Vogler. Ready simulation for concurrency: It's logical! *Inform. and Comput.*, 208:845–867, 2010. An extended abstract appeared in ICALP 2007, vol. 4596 of LNCS, pp. 752–763, Springer, 2007.

[53] G. Lüttgen and W. Vogler. Safe reasoning with Logic LTS. *Theoret. Comp. Sc.*, 412(28):3337–3357, 2011. An extended abstract appeared in SOFSEM 2009, vol. 5404 of LNCS, pp. 376–387, Springer, 2009.

[54] G. Lüttgen and W. Vogler. Modal Interface Automata. In *7th Intl. IFIP Conf. on Theoret. Comp. Sc. (TCS 2012)*, LNCS. Springer, 2012. In press.

[55] Mathworks, Inc. Control design: Designing an aircraft elevator control system. URL http://www.mathworks.de/control-systems/demos.html?file=/products/demos/stateflow/fdir/FDIR_overview.html. Last accessed on 26th July 2012.

[56] B. Meyer. Applying design by contract. *IEEE Computer*, 25(10):40–51, 1992.

[57] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[58] E.-R. Olderog. *Nets, Terms and Formulas*. Cambridge Tracts in Theoret. Comp. Sc. 23. Cambridge Univ. Press, 1991.

[59] M. Oliveira, A. Cavalcanti, and J. Woodcock. A UTP semantics for Circus. *Formal Aspects of Computing*, 21:3–32, 2009.

[60] K.V.S. Prasad. A calculus of broadcasting systems. *Sc. Comp. Progr.*, 25(2-3):285–327, 1995.

[61] J.-B. Raclet, E. Badouel, A. Benveniste, B. Caillaud, A. Legay, and R. Passerone. A modal interface theory for component-based design. *Fundamenta Informaticae*, 107:1–32, 2011.

[62] A. Rensink and W. Vogler. Fair testing. *Inform. and Comput.*, 205(2):125–198, 2007.

[63] M. Schäfer and W. Vogler. Component refinement and CSC-solving for STG decomposition. *Theoret. Comp. Sc.*, 388(1-3):243–266, 2007.

[64] A. Siirtola, A. Puhakka, and G. Lüttgen. Introducing fairness into compositional verification via unidirectional counters. In *Application of Concurrency to System Design*, pp. 32–41. IEEE, 2012.

[65] A. Sowmya and S. Ramesh. Extending Statecharts with temporal logic. *IEEE TSE*, 24(3):216–231, 1998.

[66] W. Vogler and R. Wollowski. Decomposition in asynchronous circuit design. In *Concurrency and Hardware Design*, vol. 2549 of *LNCS*, pp. 152–190. Springer, 2002.

[67] J.B. Warmer and A.G. Kleppe. *The Object Constraint Language: Precise Modeling with UML*. Addison Wesley, 1999.

[68] D. Zhang and R. Cleaveland. Fast generic model-checking for data-based systems. In *FORTE 2005*, vol. 3731 of *LNCS*, pp. 83–97. Springer, 2005.

4 Requested Modules/Funds

Funds are requested for the full application period of 3 years. All funds other than for staff are to be split 50:50 between Augsburg and Bamberg.

4.1 Basic Module

4.1.1 Funding for Staff

At each site, Augsburg and Bamberg:

- *1 Research Assistant, RA* (PhD student) for 36 months full-time (Doktorandin/Doktorand und Vergleichbare, 100% der regelmäßigen Arbeitszeit)

  This RA will be the main researcher contributing to the project from Augsburg's and Bamberg's end, respectively. She or he will work on all work packages scheduled at Augsburg (WP A1–A8) and, respectively, Bamberg (WP BA1–BA8). That this post is full-time does not only match the volume of work proposed, but is also necessary to attract suitable candidates, given the excellent job prospects in and the high starting salaries offered by the IT industry. In Augsburg, Dipl.-Inf. Ference Bujtor shall be employed as RA; he currently holds a fixed-term teaching post at the University and is being introduced to the field of interface theories by Prof. Vogler. In Bamberg, the post will be advertised nationally and internationally if no suitable candidate among the graduates of Prof. Lüttgen can be recruited.

- *1 Student Assistant, SA* (Studentische Hilfskraft) for 44 hours per month for 21 months, at the standard rate.

  - *At Augsburg:* 12.16 Euro per hour, for a total of 11,235.84 Euro.
  - *At Bamberg:* 11.52 Euro per hour, for a total of 10,644.48 Euro.

  After a familiarisation period with the project of three months, the SA will assist the RA with tool development (mainly with the programming tasks of WP A6–7 and WP BA6–7, respectively) and evaluation (with the conduct of case studies in WP A8 and WP BA8, respectively). A student at Augsburg who has taken one or more of Prof. Vogler's modules on automata theory, process algebra and distributed algorithms, would be most suitable for the SA position at Augsburg. The SA at Bamberg will likely be recruited among the students who take Prof. Lüttgen's modules on software engineering and parallel programming.

4.1.2 Direct Project Costs

4.1.2.1 Equipment up to 10,000 Euro, Software and Consumables. Specialised equipment for carrying out the proposed project is not required. All staff on the project will be equipped by the Universities of Augsburg and Bamberg with

- *State-of-the-art PCs and/or laptops*. All software necessary for carrying out the computing-intensive work packages WP A6–8 and WP BA6–8 is available free of charge.
- *Headsets and webcams*. These will enable the necessary frequent communication (via Skype or similar services) between the project sites Augsburg and Bamberg, and with the project partners at Maryland, Rennes and York.

No funds for consumables are requested.

4.1.2.2 Travel Expenses. Funds are requested to cover the cost of travel required for carrying out the project:

- *Conferences/workshops*

  Each site is expected to actively participate in two international conferences/workshops during each project year, with one travel being within Europe and one overseas. Examples of targeted meetings are, in alphabetical order, the international conferences on *Application of Concurrency to System Design* (ACSD), *Computer Aided Verification* (CAV), *Concurrency Theory* (CONCUR), *Fundamental Approaches to Software Engineering* (FASE), *Foundations of Software Science and Computation Structures* (FOSSACS), *Foundations of Software Engineering* (FSE), *Automata, Languages and Programming* (ICALP) and *Tools and Algorithms for the Construction and Analysis of Systems* (TACAS), plus some of their affiliated, high-quality workshops.

  The cost of a European trip is estimated at 1,400 Euro (including registration fees, accommodation and travel expenses), and the one for an overseas trip at 2,000 Euro. Therefore, 20,400 Euro is requested in total for all conference/workshop travel.

- *Augsburg–Bamberg cooperation*

  The cooperation between Augsburg and Bamberg will require of the RAs mutual, bi-monthly short visits of three days each during the first 18 months of the project. The visiting intervals can then be relaxed to once per calendar quarter for the remaining 18 months. In addition, each applicant will travel to the partner site once per year for three days.

  This totals 21 three-day trips, estimated at 300 Euro per trip. Thus, 6,300 Euro is requested to support the Augsburg–Bamberg cooperation throughout the application period.

- *International cooperation*

  The three international project partners are located in Rennes, France (Dr. Caillaud); Maryland, USA (Prof. Cleaveland); and York, England (Prof. Paige). For the project duration, one one-week visit to each of the partners by one member of the project team is planned.

  The travel expenses for the European trips will be approx. 1,000 Euro per trip, and those for the Transatlantic trips will be approx. 1,600 Euro per trip. This totals 3,600 Euro for all travel related to the project's international cooperations.

- *Summer schools*

  Each of the two RAs working on the project will apply to a distinguished international summer school, such as the Marktoberdorf Summer School, related to the topics Formal Specification, Concurrency Theory and Automated Verification that are relevant to this proposal.

  The projected costs are 1,800 Euro per RA for travel, accommodation and participation fees, thus totalling 3,600 Euro.

In summary, the requested funds for travel are 33,900 Euro overall.

4.1.2.3 Visiting Researchers. In addition to the funds for visiting the international project partners as explained under bullet point "*International cooperation*" above, we request funds for one one-week visit of each project partner to Augsburg or Bamberg. Analogously to above, these costs total 3,600 Euro.

4.1.2.4 & 4.1.2.5 Not Applicable. 4.1.2.4 Expenses for laboratory animals; 4.1.2.5 Other costs.

4.1.2.6 Project-Related Publication Expenses. 250 Euro p.a. are requested for each site, Augsburg and Bamberg, totalling 1,500 Euro over the application period.

This fund will enable us to publish in some of the increasing number of good-quality open-access journals and conference/workshop proceedings that impose moderate charges, such as those

appearing in the *Leibniz International Proceedings in Informatics* (LIPIcs) and *Electronic Notes in Theoretical Computer Science* (ENTCS) series. However, it is envisaged that we will mainly publish in journals and proceedings that do not charge authors.

4.1.3 Instrumentation

Funds for equipment exceeding 10,000 Euro or major instrumentation exceeding 50,000 Euro are not requested.

4.2 – 4.7 Not Applicable

4.2 Module temporary position for funding; 4.3 Module replacement funding; 4.4 Module temporary clinician substitute; 4.5 Module Mercator fellows; 4.6 Module workshop funding; 4.7 Module public relations funding.

5 Project Requirements

5.1 Employment Status Information

(a) Lüttgen, Gerald, Universitätsprofessor (W3), University of Bamberg (lifelong civil servant)
(b) Vogler, Walter, Universitätsprofessor (C3), University of Augsburg (lifelong civil servant)

5.2 First-Time Proposal Data

Not applicable

5.3 Composition of the Project Group

Both applicants will be the only people working on the proposed project, who will *not* be paid by the DFG. However, if it should turn out at the beginning of the last year of funding that the project would benefit from additional human resources in order to provide better tool support (see WP A6–8 and WP BA6–8), then the first applicant will make extra funds of the University of Bamberg available to pay for one additional Student Assistant (Studentische Hilfskraft) for up to 44 hours per month for a maximum of 12 months, at the standard Bamberg rate of 11.52 Euro per hour, for a total of up to 6,082.56 Euro.

5.4 Cooperation with Other Researchers

5.4.1 Researchers Cooperating on this Project

The project team at Augsburg and Bamberg will collaborate with three internationally leading researchers and their teams in the context of the proposed research project:

Dr. Benoît Caillaud, IRISA Rennes, France. Dr. Caillaud is heading IRISA's research team on realising algorithmic methods of reactive and distributed systems from partial and heterogeneous specifications [7]. He is an expert in contracts and interface theories [61], and will advice us when

developing our algebraic theory of modal interfaces extended with temporal-logic contracts (MIC) and when contrasting this theory to related work (WP A3, 4, 8).

Prof. Rance Cleaveland, University of Maryland, USA. Prof. Cleaveland is also Executive and Scientific Director of the Fraunhofer Center for Experimental Software Engineering, Maryland, USA. He has previously collaborated with the first applicant on a heterogeneous specification formalism combining process algebra and temporal logic [14, 15], and is therefore particularly suited for discussing the theories developed by us and for comparing them to related work (WP4). His main contribution to this project, however, will be in consulting us when developing and evaluating refinement checkers for MIC and CSC (WP A7, A8, BA7, BA8). Prof. Cleaveland is a leading expert in behavioural preorder checking for finite-state systems [17] and in automatically checking data-based systems [68]. As Chairman and co-founder of Reactive Systems, Inc., he has much experience in providing industrial-strength automated verification support and conducting realistic case studies.

Prof. Richard Paige, University of York, England. Prof. Paige is an expert in formal aspects of software engineering and in model-driven development. He has recently co-developed the Contractual State Machines (CSM) specification formalism [33] as the Principal Investigator (PI) of a UK research project that was co-authored by the first applicant, Prof. Lüttgen, and of which Prof. Lüttgen was a co-PI until his departure from York to join the University of Bamberg in April 2009. In contrast to what is proposed here, the CSM toolset provided by the UK project is manual rather than automated, and focuses on applying patterns of refinement via model transformations [37]. Prof. Paige will help us with adapting York's CSM toolset to simulate models wrt. our fully-abstract CSC semantics (WP BA6). In addition, he will assist us with the conduct of case studies employing CSC (WP BA8); in particular, re-considering some of the examples and case studies used in the York project will likely provide interesting insights regarding the practical utility of our automated refinement checking technique for CSC when compared to York's manual refinement pattern applier for CSM.

5.4.2 Past Collaborators

Within the past three years, the applicants have worked with the following national and international researchers on joint projects: Prof. Flavio Corradini, University of Camerino, Italy; Dr. Maria Rita di Berardini, University of Camerino, Italy; Dr. Andrew Galloway, University of York, England; Victor Khomenko, University of Newcastle, England; Dr. Jan Tobias Mühlberg, University of Leuven, Belgium; Prof. Richard Paige, University of York, England; Dr. Ralf Wollowski, Hasso-Plattner-Institut, Potsdam, Germany.

5.5 Scientific Equipment

All equipment necessary for carrying out the proposed project, i.e., PCs/laptops with standard software, headsets and webcams, will be provided by the Universities of Augsburg and Bamberg.

5.6 Project-Relevant Interests in Commercial Enterprises

Not applicable

6 Additional Information

Not applicable