Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 181 (2007) 81–96
www.elsevier.com/locate/entcs

Architectural Interoperability Checking in Visual Coordination Networks[1]

David Šafránek[2]

Department of Computer Science, Faculty of Informatics
Masaryk University Brno, Czech Republic

Abstract

In this paper, the approach [1] of architectural interoperability checking is revisited and utilized for interoperability checking of architectures with connectors and components treated as behaviorally and expressively different elements of architecture description. By that way, the framework of architectural interoperability checking for a diagrammatic language Visual Coordination Networks is established.

Keywords: Visual Coordination Networks, architecture interoperability checking, visual specification, concurrent systems, exogenous coordination model

1 Introduction

Hierarchical component-based system design can be underlied with precise operational semantics which allows to join traditional software and hardware design methods with formal methods known from the theory of process algebras (e.g., refinement, equivalence or model checking). Additionally, syntactical and semantical separation of modeling of coordination (interaction) aspects from modeling of behavioral (computation) aspects in architectural descriptions makes it possible to model a static communication infrastructure of a system independently of its behavioral parts. In such a setting, it can be ensured by architectural description that the system under specification is correct by design concerning the interoperability correctness of component cooperation.

In this paper we present a formal architectural interoperability checking methodology for a visual formalism Visual Coordination Networks (VCN). This language is based on our previous work [12].

[1] This work has been supported by the Grant Agency of Czech Republic grant No. 201/06/1338, the Academy of Sciences of CR grant No. 1ET408050503, and the research center Institute for Theoretical Computer Science (ITI), project No. 1M0021620808.

[2] Email: xsafran1@fi.muni.cz

doi:10.1016/j.entcs.2007.01.055

VCN employ an exogenous coordination model [3]. In such a model, coordination aspects are semantically separated from computational aspects. VCN can be viewed as static architecture diagrams specifying connections among components. To capture coordination aspects, connectors — so-called VCN buses — are introduced. By a particular VCN bus the behavior of some coordination model is specified.

1.1 Related Work

Graphical calculus of communicating systems [6] and Architectural Interaction Diagrams [9] represent the previous work on this topic. To our best knowledge, for none of those languages interoperability aspects were studied.

In the community of coordination languages, there is a large group of languages which have properties of architectural languages. The most recent language from this domain is Reo [4]. Reo supports control-driven exogenous coordination and is execution-oriented in contrast to design-oriented VCN.

In the process of hardware and software design, it has appeared useful to treat behavioral aspects separately from architectural aspects of a system under design. Such ideas of aspect separation have been tackled in the most of work concerning architectural description languages [7]. These principles are also employed in the recent version of the UML language.

1.1.1 Our Contribution

There is an exhaustive piece of work by Bernardo et al. on the topic of interoperability checking of architectural descriptions formalized in traditional process algebraic framework. In [5] it has been proved that for checking of an acyclic component topology it suffices to check interaction compatibility of all pairs of mutually connected components. The notion of such compatibility is based on weak bisimilarity of the two components in a pair. An abstraction of both components is considered comprising only actions of mutual interaction, while all other actions are hidden. In [1] this methodology of compatibility checking has been extended to arbitrary network topologies which can include cyclic relationships among components.

We show that a similar approach can be extended to the behavioral model of VCN. In contrast to [1,5], dependency graph of an architecture in VCN is bipartite, as VCN distinguishes connectors and components as two semantically different members of the architecture. In other words, such an extension has to to define the notion of architectural compatibility between any two adjacent nodes, which are given as a pair of a connector and a component. Moreover, VCN introduces a set-labeled transition operational semantics to capture behavioral model of connectors. This extension increases the expressiveness of the supported coordination model. In this paper, the notion of architectural interoperability is revisited and extended to capture needs of such a setting. Especially, there is no traditional notion of a parallel composition operator in VCN and therefore general congruence results employed in [1,5] cannot be directly employed here. The main result of this paper shows how the ideas of [1] are extended for the VCN setting.

2 Visual Coordination Networks

2.1 Overview

At the most abstract level of view, VCN introduces hierarchy of two separate layers — the *computation layer* and the *coordination layer*. The computation layer focuses on computational aspects of a system under design, while the coordination layer deals with interaction aspects. The principle of such a layered structure reflects the nature of component-based system design, and is inspired by the work on software architecture description (i.e., Wright [2]).

In our setting, the computation layer is treated as a low-level layer of system specification, upon which the coordination layer rests. Thus from the designers point of view, both the top-down and the bottom-up design methodologies can be applied in system design using VCN. On one hand, VCN allows the computation layer to be considered as a supplementary layer which can be added to the modeled system hierarchy later during the particular design process (top-down approach). On the other hand, one can specify the computation of components at first, while the coordination layer can be added later (bottom-up approach).

2.1.1 Coordination layer

The main idea of the VCN coordination layer, which revises the concept introduced in Wright [2], is to describe interaction aspects of a static component-based architecture. The notion of such a static structure concerns topology of concurrently running *components* permanently coordinated by specific *connectors*. This topology is determined by the point-to-point *links* which connect component interface ports to particular connectors. The respective VCN construct that represents such topologies of components and connectors is called a *network*. An example of a network is depicted in Figure 1.

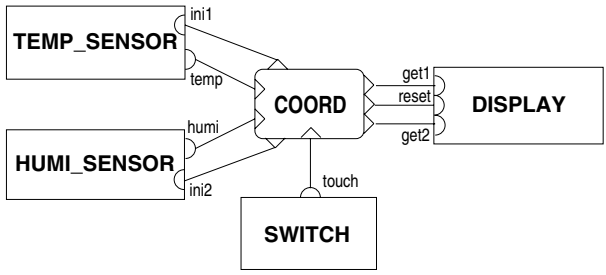

Fig. 1. A network of four components coordinated by one bus

Similarly to Wright, connectors are treated as first-class citizens (at the same specification level as components). In the VCN setting, connectors are called *buses*. Buses represent coordination mechanisms which control component interaction. The coordination model represented by buses is characterized by atomic actions called cooperations, which have the meaning of atomic multi-synchronization of a set of some components in a particular network.

2.1.2 Computation layer

Component computation is described by VCN *leaves*. VCN leaves are cornerstones of the computation layer. In our setting, leaves are assumed to be abstract computation models of system components. More particularly, a leaf is an atomic element in the VCN structure that can be specified in an arbitrary formalism for formal description of reactive computation. It is assumed that the kind of formalism which can be used for this purpose is compatible with the semantic model employed in VCN. In general, the potential set of such compatible formalisms includes any reactive computation description language which can be encoded into a labelled transition system.

2.1.3 Network hierarchy

The computation layer makes the bottom most level of the VCN hierarchy. It is determined by the set of all leaves which are used in the particular system design. As it has been mentioned above, leaves directly represent computation of components and are interconnected by buses to induce network topologies.

In a natural sense, such a network topology of leaves and buses can be abstractly viewed as a black box with complex behavior hidden inside (defined by computation of leaves coordinated by buses). More specifically, computation is represented not only by leaves, but also by entire network topologies. This idea leads us to consider the notion of component to be more abstract than the notion of leaf. In particular, either a leaf or a network can be sensed as a component in the VCN style of thinking. The possibility of taking a network as a component allows the coordination layer to have more levels of hierarchy.

3 VCN Structure and its Behavioral Model

In this section, we present a simplified version of VCN language. The considered simplification is focused on VCN networks in order to develop the framework for architectural interoperability checking.

3.1 Structural Terms

Similarly as in the case of Statecharts-like formalisms [8], the semantics of VCN is defined by textual terms. In VCN we distinguish between *structural terms* that formalize a system architecture, and *behavioral terms* that formalize a behavioral model (component and network computation). Key elements of VCN are *ports* which form *component interfaces*. Each port is identified by a label which is unambiguous in the scope of a particular level of hierarchy. A port can be sensed as a place on which events observed during the particular component computation occur. We distinguish two kinds of ports – *input ports*, on which the relevant events of the environment are received, and *output ports*, by which the relevant events are emitted to the environment.

Definition 3.1 Fix $\mathcal{L}$ a countable set of *labels* and assume $\tau \notin \mathcal{L}$. Define *ports*

as members of the set $\mathcal{P} \stackrel{\text{df}}{=} \mathcal{L} \times \{in, out\}$. Further define projections of $\mathcal{P}$ – the set of *output ports* $\mathcal{W} \stackrel{\text{df}}{=} \{p \in \mathcal{P} \parallel p = \langle l, out\rangle, l \in \mathcal{L}\}$ and the set of *input ports* $\mathcal{R} \stackrel{\text{df}}{=} \{p \in \mathcal{P} \parallel p = \langle l, in\rangle, l \in \mathcal{L}\}$.

To ensure that each port label is unique in the particular network scope, we annotate each port with an index denoting the component to which it belongs. In particular, we assume that each component in a network is identified by a unique natural number.

Definition 3.2 Define set of *annotated ports* $\mathcal{P}^\sharp \stackrel{\text{df}}{=} \{\langle p, i\rangle \parallel p \in \mathcal{P}, i \in \mathcal{N}\}$. Further define respective projections as sets of *annotated output* and *annotated input ports*, $\mathcal{W}^\sharp \stackrel{\text{df}}{=} \{\langle w, i\rangle \parallel w \in \mathcal{W}, i \in \mathcal{N}\}$, and $\mathcal{R}^\sharp \stackrel{\text{df}}{=} \{\langle r, i\rangle \parallel r \in \mathcal{R}, i \in \mathcal{N}\}$.

Remark 3.3 For some $i \in \mathcal{N}$ we denote the set of ports annotated by i as $\mathcal{P}^{\sharp i} \stackrel{\text{df}}{=} \{\langle p, i\rangle \parallel p \in \mathcal{P}\}$. Moreover, members of the set $\mathcal{P}$ are usually represented by symbols $p, p_1, p_2, \ldots$, members of $\mathcal{W}$ by $w, w_1, w_2, \ldots$, and, finally, members of $\mathcal{R}$ by $r, r_1, r_2, \ldots$ Note the important fact that $\mathcal{W} \cap \mathcal{R} = \emptyset$. By the notation $p^{\sharp i}$ for some $i \in \mathcal{N}$ we mean $\langle p, i\rangle \in \mathcal{P}^\sharp$. Thus an input port $r \in \mathcal{R}$ annotated by $i \in \mathcal{N}$, is denoted $r^{\sharp i}$. Annotated output ports are denoted in the same way. Whenever the annotation number i is not important in the particular context for an annotated port $p^{\sharp i}$, we omit the upper index '$\sharp i$' and write simply p.

For a specific set of (unannotated) ports $P \subseteq \mathcal{P}$, the set containing these ports all annotated by $i \in \mathcal{N}$ is denoted $P^{\sharp i}$. For the set of ports of all annotation indices the notation $P^\sharp$, $P^\sharp \stackrel{\text{df}}{=} \bigcup_{i \in \mathcal{N}} P^{\sharp i}$, is used.

Now we introduce a fundamental notion of the concept of buses – the notion of cooperations. A cooperation can be sensed as a group of ports on which the respective computational events are atomically synchronized when the cooperation occurs.

Definition 3.4 Let $W^\sharp \subset \mathcal{W}^\sharp$ and $R^\sharp \subset \mathcal{R}^\sharp$ finite sets of ports. Define *cooperation* as the pair $\langle W^\sharp, R^\sharp\rangle$, denoted $\langle W^\sharp / R^\sharp\rangle$, satisfying:

(1) $\forall i, j \in \mathcal{N}, w_1, w_2 \in \mathcal{W}, r_1, r_2 \in \mathcal{R}.\, (w_1{}^{\sharp i}, w_2{}^{\sharp j} \in W^\sharp \Rightarrow i \neq j) \wedge (r_1{}^{\sharp i}, r_2{}^{\sharp j} \in R^\sharp \Rightarrow i \neq j)$

We say that a *port p is included in a cooperation* $c := \langle W^\sharp / R^\sharp\rangle$, and write $p \in c$, if either $p \in W^\sharp$ or $p \in R^\sharp$. The set of all cooperations is denoted Coops.

The condition (1) ensures the natural requirement that at most one input port or one output port of each component can be involved in a cooperation.

Definition 3.5 Define *bus* B as a finite set of cooperations $B \subset$ Coops such that $B \neq \emptyset$. For a particular bus B the respective set of cooperations is denoted $coop(B)$. Countable set of all buses is denoted Buses. Members of Buses are typically denoted $B, B_1, B_2, \ldots$

Input of the bus B is denoted $In(B)$ and defined as the set $In(B) \stackrel{\text{df}}{=} \bigcup\{W^\sharp \parallel \langle W^\sharp / R^\sharp\rangle \in B\}$. Similarly, *output of the bus* B is denoted $Out(B)$ and defined as the set $Out(B) \stackrel{\text{df}}{=} \bigcup\{R^\sharp \parallel \langle W^\sharp / R^\sharp\rangle \in B\}$.

Definition 3.6 Define the *link relation* L as the relation $L \subset \mathcal{P}^\sharp \times \text{Buses}$ satisfying for each $B \in \text{Buses}$ all of the following conditions:

(2) $$\forall i, j \in \mathcal{N}, p_1{}^{\sharp_i}, p_2{}^{\sharp_j} \in \mathcal{P}^\sharp, \langle p_1{}^{\sharp_i}, B\rangle \in L.\ \langle p_2{}^{\sharp_j}, B\rangle \in L \Rightarrow p_1 \neq p_2 \vee i \neq j$$

(3)
$$\forall p \in \mathcal{P}^\sharp.\ \langle p, B\rangle \in L \Rightarrow \exists c \in B.\ p \in c$$
$$\forall w \in \mathcal{W}^\sharp.\ w \in In(B) \Rightarrow \langle w, B\rangle \in L$$
$$\forall r \in \mathcal{R}^\sharp.\ r \in Out(B) \Rightarrow \langle r, B\rangle \in L$$

Denote the set of all links of the bus B as $links(B, L)$ and define $links(B, L) \stackrel{\text{df}}{=} \{l \parallel \exists p \in \mathcal{P}^\sharp.\ l = \langle p, B\rangle \in L\}$.

The condition (2) in the definition above ensures that at most one link can be defined for a particular component interface port. The set of conditions (3) guarantees consistency of an embedding of a particular bus into a particular link relation. More precisely, all the ports linked to a particular bus must be reflected by some cooperation and for each port of any cooperation of a particular bus there must exist a link connecting the bus with the respective port.

Remark 3.7 For a particular link $l \in L$ of some link relation L we denote its port as $port(l) \stackrel{\text{df}}{=} p$, where $l \equiv \langle p, B\rangle$.

Definition 3.8 Define the set of *structural terms*, denoted $\mathbf{T}_{st}$, as the least set satisfying:

(i) $A \in \mathbf{T}_{st}$ is a *leaf*

(ii) $N \in \mathbf{T}_{st}$, if N is a *network* defined as a tuple $N \stackrel{\text{df}}{=} \langle \bar{C}, \bar{B}, L\rangle$ where
- $\bar{C} = \langle C_1, \ldots, C_n\rangle$ for some $n > 0$ is a tuple of component terms, for each $i \in \{1, ..., n\}$, $C_i = \langle S_i, I_i, G_i\rangle$ is a *component* satisfying:

(a) $I_i \subset \mathcal{P}$ finite set of ports is an *interface*, denoted $I(C_i)$

(b) $S_i \in \mathbf{T}_{st}$

(c) G_i is a *gate* which maps the component body S_i to the interface I_i

- $\bar{B} = \langle B_1, \ldots, B_m\rangle$ for some $m \geq 0$ is a tuple of buses
- L a link relation satisfying

$$L \subseteq \bigcup_{i \in \{1,\ldots,n\}} I_i{}^{\sharp_i} \times \{B_i \parallel i \in \{1, ..., m\}\}$$

Additionally, we say that a port p is a *free port* in a network N if it has no link attached.

We do not define the notion of gate here, as we focus only on inner-level (horizontal) aspects of the architecture. However, the gate is an important element of VCN concerning the inter-level (vertical) aspects [12]. For the purpose of this paper, we treat a gate as an injective mapping $gate_G$ which maps a port of a component body to a port of the respective interface.

3.2 Behavioral Model

3.2.1 Leaves

Each leaf is characterized by the set of events which can occur during its computation. The behavioral model of a leaf is determined by the traditional transition system $S = \langle \mathtt{S}, T, \mathtt{s}_0 \rangle, \mathtt{s}_0 \in \mathtt{S}$ where the transition relation has the form $T \stackrel{\text{df}}{=} \mathtt{S} \times \mathcal{P} \times \mathtt{S}$. Behavioral model of the leaf S, denoted $\Phi(S)$, is determined by the initial state $\mathtt{s}_0$, $\Phi(S) = \mathtt{s}_0$. The τ-event represents a silent event (unobservable by the environment).

3.2.2 Buses

A particular behavioral model of a bus represents a coordination mechanism. An example of bus behavior is depicted in Figure 2. The model reflects the coordination

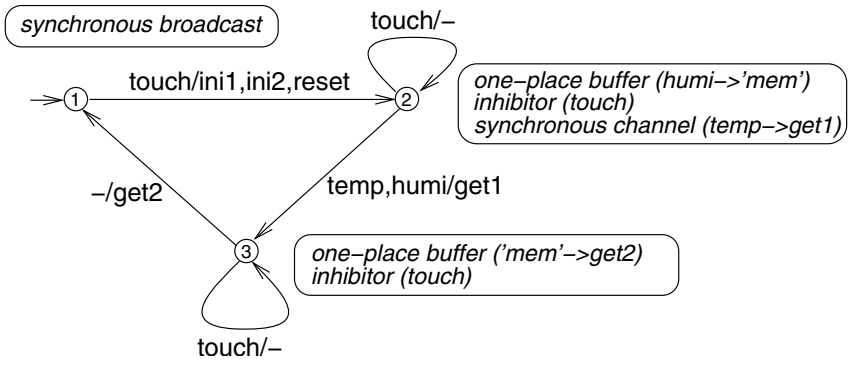

Fig. 2. Behavioral model of a *COORD* bus (a cooperation machine)

in the network from Figure 1, in particular, it deals with components $Temp\_sensor$, $Humi\_sensor$, which represent weather conditions sensors, the $Display$ component representing the LCD panel reporting the current theater information, and the $Switch$ component which initiates entire system computation. The coordination model of these components is determined by the bus $COORD$. The meaning of this bus is coordination behavior which combines atomic broadcast, synchronous channel, inhibitor, and a one-place buffer. The overall principle of the coordination model determined by the bus $COORD$ in this particular example can be summarized in the following phases:

(i) In the initial phase, pushing of the *touch* trigger causes broadcasting of initiating signal through the $ini1, ini2$, and $reset$ ports to the corresponding components. This broadcasting is performed atomically in an indivisible instant of time to ensure immediate and uninterruptable reaction of the system to the initiating signal.

(ii) After initiation, the coordination model is waiting for the information to be signaled on the producers ($Temp\_sensor$ and $Humi\_sensor$) output ports. In this phase, it is also capable of receiving the *touch* signal, to preserve blocking of the initiation switch.

When the *temp* and *humi* information appears on the respective ports, the coordination model acts like both a synchronous channel (relaying the *temp* information to the *get*1 port) and a one-place buffer (storing the *humi* information to the internal memory — the state (3)).

(iii) After filling of the buffer, the information stored in the buffer is transmitted to the *get*2 port. It ensures that the *get*1 and *get*2 ports are filled with the appropriate information in a sequence of a given order (i.e., the information of the current weather conditions can appear always in the predefined order on the display panel).

(iv) The coordination model returns to the initial phase (i).

The coordination behavior described above can be formally captured as a transition system labeled with cooperations. We call such a variant of transition system as *cooperation machine*. A cooperation machine is declared formally by the following definition:

Definition 3.9 Assume B is a bus. *Cooperation machine of* B, denoted $cm(B)$, is a tuple $cm(B) \stackrel{\text{df}}{=} \langle Q, T, q_0 \rangle$ where

- Q is a finite set of states,
- $q_0 \in Q$ is an initial state,
- $T \subseteq Q \times coop(B) \times Q$ is a finite transition relation.

Remark 3.10 For a given bus B, the set of all states of the cooperation machine $cm(B)$ is denoted $Q(B)$. Further, the initial state of the cooperation machine $cm(B)$ is denoted $\Phi_B(B)$. The fact that $\langle q, \langle W/R \rangle, q' \rangle \in T$ for some $q, q' \in Q(B)$ and $\langle W/R \rangle \in B$ is denoted $q \stackrel{W/R}{\rightarrow}_B q'$. Additionally, the set of all transitions in which can be evolved from a state $q \in Q(B)$ is denoted $en(q)$, $en(q) \stackrel{\text{df}}{=} \{\langle W/R \rangle \in B \,\|\, \exists q' \in Q(B).\, q \stackrel{W/R}{\rightarrow}_B q'\}$.

3.2.3 Components and Networks

Definition 3.11 For each component $C = \langle S, I, G \rangle$ and each network or leaf configuration $\mathtt{S}$ define the *component configuration* $\langle \mathtt{S}, I, G \rangle$. Behavioral model of a component C is denoted $\Phi_C(C)$ and determined by the initial state of S. Formally, $\Phi_C(\langle S, I, G \rangle) \stackrel{\text{df}}{=} \langle \Phi(S), I, G \rangle$.

Definition 3.12 Define the set of *network configurations* as the set

$$\{\langle\langle \mathtt{c_1}, .., \mathtt{c_n} \rangle, \langle q_1, .., q_m \rangle\rangle \,\|\, \begin{array}{l} \forall n, m \in \mathcal{N},\ \mathtt{c}_1, .., \mathtt{c}_n \ \ldots \text{component configurations}, \\ q_1 \in Q(B_1), .., q_m \in Q(B_m) \ \ldots \text{bus states} \ \} \end{array}$$

Behavioral model of the network $N = \langle\langle C_1, .., C_n \rangle, \langle B_1, .., B_m \rangle, L \rangle$ is denoted $\Phi(N)$ and defined $\Phi(N) \stackrel{\text{df}}{=} \langle\langle \Phi_C(C_1), .., \Phi_C(C_n) \rangle, \langle \Phi_B(B_1), .., \Phi_B(B_m) \rangle\rangle$.

Let $\mathtt{N} = \langle\langle \mathtt{c}_1, .., \mathtt{c}_n \rangle, \langle q_1, ..., q_m \rangle\rangle$ a network configuration. By the notation $\mathtt{N}[c_i := \mathtt{c}']$ we denote a network configuration which differs from $\mathtt{N}$ only in the ith component configuration, provided that the component configuration $\mathtt{c}_i$ is replaced with $\mathtt{c}'_i$.

Similarly, by the notation $\mathtt{N}[q_j := q']$ we denote a network configuration which differs from $\mathtt{N}$ only in the jth bus state, provided that the bus state q_j is replaced with q'.

For some $\theta \subseteq \{1, ..., n\}$ denote $\mathtt{N}[\bigwedge_{i\in\theta} c_i := \mathtt{c}'_i]$ the network configuration which differs from $\mathtt{N}$ in all component positions included in θ, provided that each component configuration $\mathtt{c}_i$ for some $i \in \theta$ is replaced with the configuration $\mathtt{c}'_i$. Moreover, the network configuration which differs from $\mathtt{N}$ in all component positions included in θ and additionally in the jth state of the bus B_j which has moved to state q' is denoted $\mathtt{N}[\bigwedge_{i\in\theta} c_i := \mathtt{c}'_i,\ q_j := q']$.

For an arbitrary configuration t of any kind denote $en(t)$ the set of all events which can be evolved from t, $en(t) \stackrel{\text{df}}{=} \{e \parallel \exists t'.\, t \xrightarrow{e} t'\}$ where '$\rightarrow$' represents the relevant transition relation. Note that also the internal τ-event can be included in $en(t)$.

Now we present structural operational semantics rules which derive the behavioral model of components and networks from behavioral models of elementary entities (buses and leaves). The first rule defines the operational semantics of the bottom most components. Let $C = \langle S, I, G\rangle$ a component, and S a leaf. The operational semantics of C is defined by the transition relation '$\rightarrow_C$' over the relevant component configurations which is given by the inference rule:

$$(1)\quad \frac{\mathtt{s} \xrightarrow{e}_s \mathtt{s}'}{\langle \mathtt{s}, I, G\rangle \xrightarrow{gate_G(e)}_C \langle \mathtt{s}', I, G\rangle} \quad \Big[\, e \in en(\mathtt{s}) \,\Big]$$

Let $C = \langle N, I, G\rangle$ a component, and $N = \langle\langle C_1, C_2, ..., C_n\rangle, \langle B_1, ..., B_m\rangle, L\rangle$ a network. The operational semantics of C is defined by the transition relation '$\rightarrow_C$' over the relevant component configurations which is given by the following rule:

$$(2)\quad \frac{\mathtt{N} \xrightarrow{e^{\sharp_i}}_N \mathtt{N}'}{\langle \mathtt{N}, I, G\rangle \xrightarrow{gate_G(e^{\sharp_i})}_C \langle \mathtt{N}', I, G\rangle} \quad \left[\begin{array}{c} i \in \{1, ..., n\} \\ e^{\sharp_i} \in en(\mathtt{N}) \end{array}\right]$$

Operational model of the network N is determined by the transition relation '$\rightarrow_N$' over network configurations. The following rule captures the situation when a component performs an action occurring on a free port.

$$(3)\quad \frac{\mathtt{c}_i \xrightarrow{e}_C \mathtt{c}'}{\mathtt{N} \xrightarrow{e^{\sharp_i}}_N \mathtt{N}[c_i := \mathtt{c}']} \quad \left[\begin{array}{c} \mathtt{N} = \langle\langle \mathtt{c}_1, .., \mathtt{c}_i, .., \mathtt{c}_n\rangle, \bar{q}\rangle \\ e \in en(\mathtt{c}_i) \\ e^{\sharp_i} \text{ is a free port} \end{array}\right]$$

Finally, we present the rule responsible for network coordination. At each network configuration a maximal cooperation is chosen to be performed from all the enabled cooperations. Before we state the synchronization rule itself, we define the notion of maximal enabled cooperation which realizes such a choice.

Definition 3.13 Let $\mathtt{N} = \langle\langle \mathtt{c}_1, ..., \mathtt{c}_n\rangle, \langle q_1, ..., q_m\rangle\rangle$ a network configuration. Further let $\langle W/R\rangle \in$ Coops a cooperation satisfying $\langle W/R\rangle \in en(q_j)$ for some $j \in \{1, ..., m\}$. We say that $\langle W/R\rangle$ is *enabled in configuration* $\mathtt{N}$ and write $enabled(\langle W/R\rangle, \mathtt{N})$ if and

only if there exists a set of ports $P \subset \mathcal{P}^\sharp$ satisfying the following conditions:

- If $e^{\sharp i} \in P$ for some $i \in \{1, ..., n\}$ then $e^{\sharp i} \in en(\mathtt{c}_i)$.
- $e \in P \Leftrightarrow e \in W$ or $e \in R$.

We say that $\langle W/R \rangle$ is *maximal enabled cooperation in* q_j *of* $\mathtt{N}$ and write $maxenabled(\langle W/R \rangle, \mathtt{N}, q_j)$ if and only if $enabled(\langle W/R \rangle, \mathtt{N})$ and $\forall \langle W'/R' \rangle \in en(q_j).\, (W' \subseteq W \wedge R' \subseteq R) \vee (W' \cap W = \emptyset \wedge R' \cap R = \emptyset)$.

The following rule is responsible for the network coordination:

$$(4)\quad \frac{q_j \overset{W/R}{\rightarrow}_{B_j} q'_j \quad \forall e^{\sharp i} \in \mathcal{M}_{B_j}.\, \mathtt{c}_i \overset{e}{\rightarrow}_C] \mathtt{c}'_i}{\mathtt{N} \overset{\tau}{\rightarrow}_N \mathtt{N}[\bigwedge_{i \in \theta} c_i := \mathtt{c}'_i, q_j := q'_j]} \qquad \left[\begin{array}{c} \mathtt{N} = \langle\langle \mathtt{c}_1, .., \mathtt{c}_n \rangle, \langle q_1, .., q_j, .., q_m \rangle\rangle \\ maxenabled(\langle W/R \rangle, \mathtt{N}, q_j) \\ \mathcal{M}_{B_j} \overset{\text{df}}{=} W \cup R \\ \theta \overset{\text{df}}{=} \{i \parallel e^{\sharp i} \in \mathcal{M}_{B_j}\} \end{array}\right]$$

4 Network Interoperability Correctness

Our formal solution for checking of interoperability correctness of VCN behavioral model revisits and extends the approach previously presented and proved by Bernardo et.al. in [1]. Inspired by that work, we establish the solution for checking of arbitrary interoperability-critical property which is preserved by weak bisimulation equivalence of VCN terms. As weak bisimulation equivalence preserves validity of all μ-calculus formulae with weak versions of diamond and box operators, the interoperability checking method includes an exhaustive set of interesting properties including deadlock freedom. The principle of interoperability checking relies on the idea of searching for a situation in a particular network hierarchy in which validity of a property being checked is violated by interaction of some components and buses which otherwise satisfy the considered property (if taken as stand-alone entities). With respect to the inductive definition of VCN diagrams the interoperability check traverses the hierarchy from leaves up to higher network levels.

Note that the approach of Bernardo et.al. deals with architectures composed of uniform components connected by links which can be of one-to-many character. Anyway, those links are static (stateless) connectors. There is no explicit notion of a connector like in Wright or in our approach. However, connectors can be there still modeled explicitly as components which are logically treated differently than common components. What is not possible there is modeling of connector dynamism concerning atomic many-to-many cooperations, as our approach allows by the behavioral model of buses. This is the reason why the results of [1] cannot be directly applied for developing the interoperability checking framework for the behavioral model of VCN. We follow the way of utilizing and extending these results to fit the character of behavioral model of buses.

Remark 4.1 For a component C_i and a bus B_j of some network $\langle \bar{C}, \bar{B}, L \rangle \in \mathbf{T}_{\text{st}}$ denote $Llinks(C_i, B_j)$ the set of all links between C_i and B_j:

$$Llinks(C_i, B_j) \overset{\text{df}}{=} \{l \in links(B_j, L) \parallel \exists p \in I_i.\, port(l) = p^{\sharp i}\}$$

Definition 4.2 Let $N = \langle\langle C_1, ..., C_n\rangle, \langle B_1, ..., B_m\rangle, L\rangle$ a network. Define the *dependency graph of* N, denoted $\mathcal{G}(N)$, as a bipartite graph $\mathcal{G}(N) \stackrel{\text{df}}{=} \langle\{C_1, ..., C_n\} \cup \{B_1, ..., B_m\}, E\rangle$ where E is defined in the following way:

$$E \stackrel{\text{df}}{=} \{\langle C_i, B_j\rangle \parallel Llinks(C_i, B_j) \neq \emptyset\}$$

In the following part, we will assume φ a formula of modal μ-calculus expressing some interoperability safety property. An example of such a property can be deadlock freedom expressed by the formula $\nu Z.\langle\langle -\rangle\rangle \mathtt{tt} \wedge [[-]]Z$. A significant feature of this formula is that it does not refer to any particular event and hence its validity is independent of the behavioral model alphabet.

In the previous section cooperation machines have been introduced as specific transition systems with cooperations appearing in transition labels. Hence we at first utilize the traditional (weak) bisimulation equivalence in order to establish the notion of so-called cooperation-labeled weak bisimulation.

Remark 4.3 Let $\gamma \in \text{Coops}^*$ a sequence of cooperations. Denote $\hat{\gamma}$ the following sequences of cooperations:

- $\hat{\gamma} \stackrel{\text{df}}{=} \epsilon$, if $\gamma = \langle\emptyset/\emptyset\rangle$;
- $\hat{\gamma} \stackrel{\text{df}}{=} \hat{\gamma}'$, if $\gamma = \langle\emptyset/\emptyset\rangle^* \cdot \gamma' \cdot \langle\emptyset/\emptyset\rangle^*$ where $\gamma' \neq \epsilon$.

Denote $\Rightarrow^{\gamma}_{B}$ the sequence of succeeding transitions $\Rightarrow^{\gamma}_{B} \stackrel{\text{df}}{=} (\stackrel{\emptyset/\emptyset}{\rightarrow}_B)^* \stackrel{\gamma}{\rightarrow}_B (\stackrel{\emptyset/\emptyset}{\rightarrow}_B)^*$.

Definition 4.4 Let B_1 and B_2 buses. Further let $q \in Q(B_1), b \in Q(B_2)$ states of the respective cooperation machines. We say q and b are *(weakly) bisimilar* and write $q \approx^{cl} b$ if and only if for each $\gamma \in B_1 \cup B_2$ both of the following holds:

(i) If $q \stackrel{\gamma}{\rightarrow}_{B_1} q'$ then $\exists b' \in Q(B_2).\ b \stackrel{\hat{\gamma}}{\Rightarrow}_{B_2} b'$ and $q' \approx^{cl} b'$.

(ii) If $b \stackrel{\gamma}{\rightarrow}_{B_2} b'$ then $\exists q' \in Q(B_1).\ q \stackrel{\hat{\gamma}}{\Rightarrow}_{B_1} q'$ and $b' \approx^{cl} q'$.

The fundamental idea of interoperability checking is based on the notion of so-called compatibility which is based on pairwise comparison of behavior of components and buses in the network. More particularly, the behavior observed on links between a component and a bus is considered. In order to enable such comparison we define a *projection of bus behavior* to a particular subset of links.

Definition 4.5 Let $N = \langle\langle C_1, ..., C_n\rangle, \langle B_1, ..., B_m\rangle, L\rangle$ a network, and let B_j for some $j \in \{1, ..., m\}$ a bus, $cm(B_j) = \langle Q, T, q_0\rangle$ a cooperation machine, and $L' \subseteq L$ a link relation. Define *L'-projection of* B_j, denoted $\pi(B_j, L')$, as the bus B' with the semantics determined by the cooperation machine $cm(B') \stackrel{\text{df}}{=} \langle Q, T', q_0\rangle$ where T' is defined in the following way:

$$\langle q, \langle W'/R'\rangle, q'\rangle \in T' \stackrel{\text{df}}{\Leftrightarrow} \langle q, \langle W/R\rangle, q'\rangle \in T$$

where $W' \stackrel{\text{df}}{=} W \cap ports(B_j, L')$ and $R' \stackrel{\text{df}}{=} R \cap ports(B_j, L')$.
The bus B' is defined as the set of all cooperations appearing in labels of T'.

Note that the cooperation-labeled transition relation of cooperation machines can be comprehended formally as an extension of the classical transition relation which is used for determining the operational semantics of VCN terms. This classical transition relation can be lifted to the format of cooperation-labeled transition relation. More precisely, we can consider behavioral model of a VCN term in the form of so-called saturated cooperation machine (defined below). This way we achieve uniform framework for behavioral analysis of VCN terms. In such a setting, we represent the internal τ-event as an empty cooperation $\langle\emptyset/\emptyset\rangle$.

Definition 4.6 The *cooperation machine* $cm(B)$ is called *saturated* if each transition of $cm(B)$ has the form $\langle q, b, q'\rangle$ satisfying just one of the following possibilities:

- $b \equiv \langle\{w\}/\emptyset\rangle$ for some $w \in \mathcal{W}^\sharp$;
- $b \equiv \langle\emptyset/\{r\}\rangle$ for some $r \in \mathcal{R}^\sharp$;
- $b \equiv \langle\emptyset/\emptyset\rangle$.

To analyze interoperability of buses and particular components in a network we need to look into the network internal behavior. More specifically, cooperations cannot be hidden in such analysis. In particular, we need to observe cooperations occurring on the network link relation or on its specific subset. To capture this kind of observation of network behavior with respect to some link relation L, we define *L-observable model* of the network. This operational model is based on the cooperation-labeled transition relation. In the following definition the notion of L-observable model is given and consequently employed to formalize the notion of compatibility.

Definition 4.7 Let $N = \langle\langle C_1, .., C_n\rangle, \langle B_1, .., B_m\rangle, L\rangle \in \mathbf{T}_{\mathrm{st}}$ a network. Define *L-observable model of network N* by a transition relation $\rightarrow_L$ defined over network configurations by the following rules, which are modification of rules $(3, 4)$:

$$(3')\ \frac{\mathtt{c}_i \xrightarrow{e}_C \mathtt{c}'}{\mathtt{N} \xrightarrow{\gamma}_L \mathtt{N}[c_i := \mathtt{c}']} \quad \left[\begin{array}{c} \mathtt{N} = \langle\langle \mathtt{c}_1, .., \mathtt{c}_i, .., \mathtt{c}_n\rangle, \bar{q}\rangle \\ e^{\sharp_i} \text{ is a free port} \\ \gamma \stackrel{\mathrm{df}}{=} \begin{cases} \langle e^{\sharp_i}/\emptyset\rangle, \text{ if } e \in \mathcal{W}, \\ \langle\emptyset/e^{\sharp_i}\rangle, \text{ if } e \in \mathcal{R}. \end{cases} \end{array}\right]$$

$$(4')\ \frac{q_j \xrightarrow{W/R}_{B_j} q'_j \quad \forall e^{\sharp_i} \in \mathcal{M}_{B_j}.\, \mathtt{c}_i \xrightarrow{e}_C \mathtt{c}'_i}{\mathtt{N} \xrightarrow{W/R}_L \mathtt{N}[\textstyle\bigwedge_{i\in\theta} c_i := \mathtt{c}'_i, q_j := q'_j]} \quad \left[\begin{array}{c} \mathtt{N} = \langle\langle \mathtt{c}_1, .., \mathtt{c}_n\rangle, \langle q_1, .., q_j, .., q_m\rangle\rangle \\ maxenabled(\langle W/R\rangle, \mathtt{N}, q_j) \\ \mathcal{M}_{B_j} \stackrel{\mathrm{df}}{=} W \cup R \\ \theta \stackrel{\mathrm{df}}{=} \{i \parallel e^{\sharp_i} \in \mathcal{M}_{B_j}\} \end{array}\right]$$

The *L-observation of network N*, denoted $\Phi_L(N)$, is determined as in the case of the common behavioral model by the initial network configuration $\Phi(N)$, but the transition relation $\rightarrow_L$ is taken instead of $\rightarrow_N$.

Definition 4.8 Let $N = \langle\langle C_1, .., C_n\rangle, \langle B_1, .., B_m\rangle, L\rangle$ a network, and let $C_i = \langle S, I, G\rangle$ and B_j (for $i \in \{1, .., n\}, j \in \{1, .., m\}$) adjacent nodes of $\mathcal{G}(N)$. Further let $L' \subseteq L$ defined as a link relation $L' \stackrel{\mathrm{df}}{=} \{l \in links(B_j, L) \parallel \exists p \in I.\, p^{\sharp_i} = port(l)\}$.

We say C_i is *compatible in* N *with* B_j, and write $C_i \bowtie^N B_j$, if and only if $\Phi_B(B') \approx^{cl} \Phi_{L'}(\langle\langle S, I', G'\rangle, \langle\pi(B_j, L')\rangle, L'\rangle)$ where

- $I' \stackrel{\text{df}}{=} I \cap P$ where $P \stackrel{\text{df}}{=} \{p \in I \parallel \exists l \in L', i \in \mathcal{N}.\, p^{\sharp_i} = port(l)\}$,
- G' is defined by G restricted to ports of I'.

An important subpart of an acyclic network topology is a so-called star topology. Intuitively, the star topology is a group of all components connected to the same bus in a particular acyclic network. Significance of this structure relies on the fact that for checking of interoperability it suffices to check compatibility of the central bus with every component included in the particular star topology. This result is included in the general theorem 4.11. In order to precise the notion of the star topology, we introduce formally the notion of subnetwork at first.

Definition 4.9 Let $N = \langle\langle C_1, ..., C_n\rangle, \langle B_1, ..., B_m\rangle, L\rangle \in \mathbf{T}_{\text{st}}$ a network. For some $n' \leq n$ and $m' \leq m$ define the *subnetwork* N' *of* N as a network $N' = \langle\langle C'_1, ..., C'_{n'}\rangle, \langle B'_1, ..., B'_{m'}\rangle, L'\rangle \in \mathbf{T}_{\text{st}}$ satisfying:

- $\{C'_1, ..., C'_{n'}\} \subseteq \{C_1, ..., C_n\}$ and $\{B'_1, ..., B'_{m'}\} \subseteq \{B_1, ..., B_m\}$
- $L' \stackrel{\text{df}}{=} \{l \in L \parallel l \in Llinks(C, B) \wedge C \in \{C'_1, ..., C'_{n'}\} \wedge B \in \{B'_1, ..., B'_{m'}\}\}$
- Each $B \in \{B'_1, ..., B'_{m'}\}$ is defined as $\pi(B_j, L')$ for some $j \in \{1, ..., m\}$.

Further define for some $j \in \{1, ..., m\}$ the *star topology of* B_j *in* N as a subnetwork N' of N having the form $N' \stackrel{\text{df}}{=} \langle\langle C'_1, ..., C'_{n'}\rangle, \langle B'\rangle, L'\rangle$ where each $C'_i \in \{C'_1, ..., C'_{n'}\}$ satisfies $Llinks(C'_i, B_j) \neq \emptyset$.

In checking of network interoperability, existence of cyclic component relationships requires a special care. An example of interoperability correctness violation in a cyclic topology is depicted in Figure 3. If we take the star topology

$$\langle\langle C'_1, C'_2\rangle, \langle\pi(B_1, L)\rangle, L\rangle,\ L \stackrel{\text{df}}{=} \{\langle in^{\sharp_1}, B_1\rangle, \langle out^{\sharp_1}, B_1\rangle, \langle in^{\sharp_2}, B_1\rangle, \langle out^{\sharp_2}, B_1\rangle\}$$

where the components C'_1, C'_2 are C_1, C_2 projected to links in L then such a subnetwork is deadlock free. Similarly, the same holds for the star topology of the bus B_2. However, if we consider the entire cyclic topology, a deadlock situation can arise, as it is shown by darkened states of the transition systems in the figure.

To capture the above stated problem, we present the main result of this paper declaring a set of conditions which are both necessary and sufficient to ensure an arbitrary network topology to satisfy some property φ. In the following theorem, by a *maximal cycle* we mean each subgraph of a network dependency graph which is not strictly contained in some larger cycle of the graph. Moreover, for each maximal cycle Ω of a network dependency graph we consider the respective subnetwork, so-called Ω-subnetwork, containing all the buses and components of the cycle and the link relation restricted to them. Additionally, we define the *border of a cycle* to denote the set of those components and buses each of which is identified by one of the following statements:

- A component of the cycle which has at least one free port.

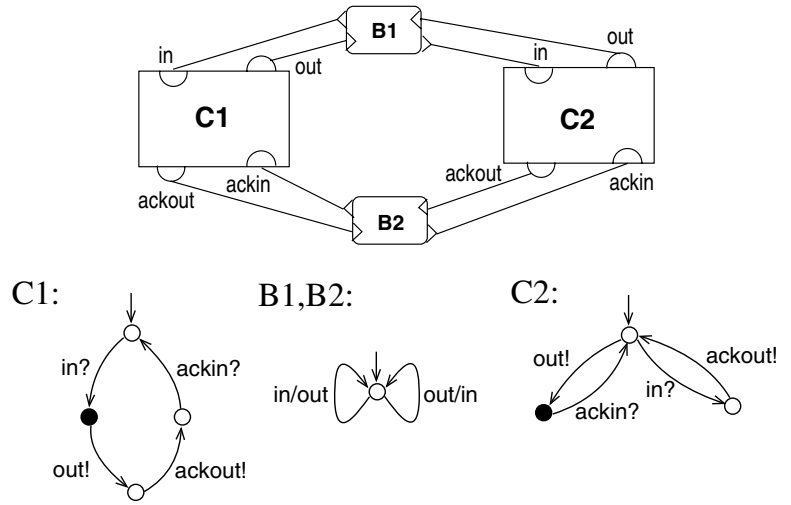

Fig. 3. Violation of network interoperability correctness by deadlock

- A component of the cycle linked to a bus outside the cycle.
- A bus in the cycle which is linked to a component outside the cycle.

Definition 4.10 Let $N = \langle\langle C_1, ..., C_n\rangle, \langle B_1, ..., B_m\rangle, L\rangle$ a network and Ω some (maximal) cycle of $\mathcal{G}(N)$. Define *border* of Ω, denoted $\beta(\Omega)$, as the set

$$\beta(\Omega) \stackrel{\text{df}}{=} \{C \in \Omega \parallel \exists p \in ports(I(C)), \forall B \in \Omega.\, p \notin ports(Llinks(B, C))\} \cup \{B \in \Omega \parallel \exists \gamma \in B, p^\sharp \in \gamma, \forall C \in \Omega.\, p \notin ports(Llinks(B, C))\}$$

Theorem 4.11 *Let $N = \langle\langle C_1, ..., C_n\rangle, \langle B_1, ..., B_m\rangle, L\rangle$ a network with $\mathcal{G}(N)$ which is a connected graph of arbitrary shape, and let φ a property. N satisfies φ if and only if the following conditions hold:*

(i) *Each bus $B_j \in \{B_1, ..., B_m\}$ satisfies φ.*

(ii) *Each component $C_i \in \{C_1, ..., C_n\}$ satisfies φ.*

(iii) *Each bus B which is not included in any cycle of $\mathcal{G}(N)$ satisfies $C \bowtie B$ for each C such that $Llinks(C, B) \neq \emptyset$.*

(iv) *Each bus $B \in \beta(\Omega)$ of each maximal cycle Ω of $\mathcal{G}(N)$ satisfies $C \bowtie B$ for each C such that $Llinks(C, B) \neq \emptyset$ and C is not included in Ω.*

(v) *For each maximal cycle Ω of $\mathcal{G}(N)$ the Ω-subnetwork of N satisfies φ.*

To realize interoperability checking for arbitrary network topology and that way constructively prove the above stated theorem, we apply the idea of step-by-step reduction of the entire potentially cyclic topology to a smaller acyclic topology which has the behavioral model observationally equivalent to the original topology. The intuition about such an observational behavior preserving reduction of cyclic topologies to acyclic topologies is the following. Each cycle in the entire network is replaced with a star topology which has the behavior model equivalent to the original cyclic topology. This star topology is defined in such a way that all the relevant links

leading from components and buses of the cycle to the buses and components outside the cycle are remapped to equivalent links leading from components and buses of the star topology. The important property that must be satisfied by such a replacement is mutual compatibility of those components and buses. Moreover, the star topology must itself satisfy compatibility of its components with the bus forming its center. Then by replacing all the cyclic subnetworks with such compatible star topologies the acyclicity of the entire network is achieved and interoperability checking then relies on checking mutual compatibility of buses and components. The formal proof of the theorem is based on the intuition described in this paragraph and is given precisely in the full version of this paper [11].

5 Conclusion and Future Work

In this paper we have presented simplified version of the language VCN for hierarchical specification of component-based concurrent systems. The key concept of the language are buses which represent coordination models used in system architectures. We have utilized the process algebraic approach of [5] and proved its extension which has been introduced due to the specific features of VCN which are not incorporated directly in traditional process algebraic approaches for architectural description.

We are currently implementing a graphical tool [10] which allows VCN diagrams to be simply created and modified. In our future work, we would like to discuss endogenous and exogenous extensions of the VCN models in the style of [1]. We also aim at extending architectural interoperability checking to VCN bus classes which allow generalized definitions of buses [12].

References

[1] Alessandro Aldini and Marco Bernardo. On the usability of process algebra: An architectural view. *Theor. Comput. Sci.*, 335(2-3):281–329, 2005.

[2] R. Allen and D. Garlan. A formal basis for architectural connection. *ACM Trans. Softw. Eng. Methodol.*, 6(3):213–249, 1997.

[3] F. Arbab. What do you mean, coordination? *Bulletin of the Dutch Association for Theoretical Computer Science (NVTI)*, 1998.

[4] Farhad Arbab. Reo: a channel-based coordination model for component composition. *Mathematical. Structures in Comp. Sci.*, 14(3):329–366, 2004.

[5] Marco Bernardo, Paolo Ciancarini, and Lorenzo Donatiello. Architecting families of software systems with process algebras. *ACM Trans. Softw. Eng. Methodol.*, 11(4):386–426, 2002.

[6] R. Cleaveland, X. Du, and S. A. Smolka. GCCS: A Graphical Coordination Language for System Specification. In *Proceedings of COORD'00*. LNCS, Springer Verlag, 2000.

[7] Paul C. Clements. A survey of architecture description languages. In *IWSSD '96: Proceedings of the 8th International Workshop on Software Specification and Design*, page 16, Washington, DC, USA, 1996. IEEE Computer Society.

[8] D. Harel. Statecharts: A Visual Formalism for Complex Systems. Technical report, The Weizmann Institute, 1987.

[9] A. Ray and R. Cleaveland. Architectural Interaction Diagrams: AIDs for System Modeling. In *Proc. of ICSE 2003*. IEEE, 2003.

[10] Z. Rehak. *VCNE project home page.* ParaDiSe Laboratory, Masaryk University Brno, 2006. `http://www.fi.muni.cz/~xrehak5/vcne/`.

[11] D. Šafránek. Architectural Interoperability Checking in Visual Coordination Networks. Technical Report FIMU-RS-2006, Faculty of Informatics, Masaryk University Brno, 2006.

[12] D. Šafránek and J. Šimša. VCD: A Visual Formalism for Specification of Heterogeneous Software Architectures. In *Theory and Practice of Computer Science: 31st Conference on Current Trends in Theory and Practice of Computer Science*, volume 3381 of *LNCS*. Springer, 2005.