BAMBERGER BEITRÄGE

ZUR WIRTSCHAFTSINFORMATIK UND ANGEWANDTEN INFORMATIK

ISSN 0937-3349

Nr. 99

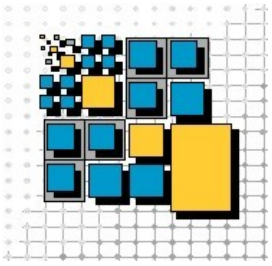
Static Analysis Rules of the BPEL Specification: Tagging, Formalization and Tests

Christian Preißinger, Simon Harrer

August 2014

FAKULTÄT WIRTSCHAFTSINFORMATIK UND ANGEWANDTE INFORMATIK

OTTO-FRIEDRICH-UNIVERSITÄT BAMBERG



Distributed Systems Group

Otto-Friedrich Universität Bamberg

An der Weberei 5, 96052 Bamberg, GERMANY

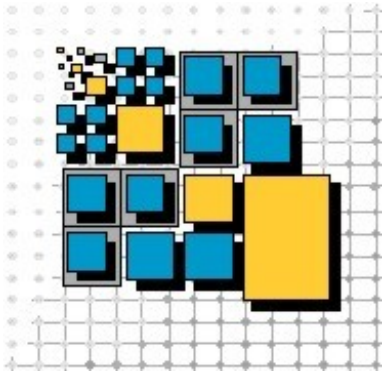
Prof. Dr. rer. nat. Guido Wirtz

Due to hardware developments, strong application needs and the overwhelming influence of the internet, distributed systems have become one of the most important topics for nowadays software industry. Owing to their ever increasing importance for everyday business, distributed systems have high requirements with respect to dependability, robustness and performance. Unfortunately, distribution adds its share to the problems of developing complex systems. Heterogeneity in both, hardware and software, frequent changes, concurrency, distribution of components and the need for inter-operability between systems complicate matters. Moreover, new technical aspects like resource management, load balancing and guaranteeing consistent operation in the presence of partial failures put an additional burden onto the developer. *Our long-term research goal is the development, implementation and evaluation of methods helpful for the realization of robust and easy-to-use software for complex systems in general while putting a focus on the problems and issues regarding distributed systems on all levels.* This includes design methods, visual languages and tools for distributed systems development as well as middleware, SOA and cloud computing issues. Our current research activities focus on different aspects centered around that theme:

- *Implementation of Business Processes and Business-to-Business-Integration (B2Bi)*: Starting from requirements for successful B2Bi development processes, languages and systems, we investigate the practicability and inter-operability of different approaches and platforms for the design and implementation of business processes.
- *Quality, esp. Robustness, Standard-conformance, Portability, Compatibility and Performance of Process-based Software and Service-oriented Systems*: In both, industry and academia, process languages have emerged, e.g. Windows Workflow (WF), Business Process Model and Notation (BPMN) and Web Services Business Process Execution Language (WS-BPEL). Although widely used in practice, current implementations of these languages and models are far from perfect. We work on metrics to compare such languages w.r.t. expressive power, conformance and portability as well as additional quality properties, such as installability, replaceability, adaptability and inter-operability. These metrics are developed and validated formally as well as evaluated practically. In the context of BPMN, we work on tools to check for and improve the standard compliance for human-centric process models on different layers of abstraction. Runtime environments for process languages with a focus on BPEL are investigated by means of a framework that eases the comparative test of different run-times and process engines.
- *Cloud Application Portability:* The hype surrounding the Cloud has lead to a variety of offerings that span the whole cloud stack. We examine important aspects of portability in cloud environments and enhance the portability of cloud applications by applying common standards between heterogeneous clouds. We make use of a holistic view of the cloud including important aspects like cloud specific restrictions, platform configurations, the deployment and life cycle of cloud applications.
- *Visual Programming- and Design-Languages*: The goal of this long-term effort is the utilization of visual metaphors and visualization techniques to make design- and programming languages more understandable and, hence, more easy-to-use. Currently, languages for designing and programming sensor networks are at the focus of this effort.

More information about our work can be found at `www.uni-bamberg.de/en/pi/`. If you have any questions or suggestions regarding this report or our work, don't hesitate to contact us.

Bamberg, June 2014 Guido Wirtz



Static Analysis Rules of the BPEL Specification: Tagging, Formalization and Tests

Christian Preißinger, Simon Harrer

https://github.com/uniba-dsg/betsy/tree/soca2014

Abstract In 2007, OASIS finalized their Business Process Execution Language 2.0 (BPEL) specification which defines an XML-based language for orchestrations of Web Services. As the validation of BPEL processes against the official BPEL XML schema leaves room for a plethora of static errors, the specification contains 94 static analysis rules to cover all static errors. According to the specification, any violations of these rules are to be checked by a standard conformant engine at deployment time. When a violation is not detected in BPEL processes during deployment, such errors are only detectable at runtime, making them expensive to find and fix. Due to the large amount of rules, we have created a tag system to categorize them, allowing easier reasoning about these rules. Next, we formalized the static rules and derived test cases based on these formalizations with the aim to evaluate the degree of support for static analysis of BPEL engines. Hence, this work is the foundation of the static analysis capabilities of BPEL engines.

Keywords SOA, BPEL, static analysis, tests, formalization, tagging

Contact:

christian.preissinger@morsepost.de, simon.harrer@uni-bamberg.de

Contents

List of Tables

Abbreviations

BPEL Web Service Business Process Execution Language

WSDL Web Service Description Language

1 Introduction

This work is the foundation of [1]. It analyzes the static analysis rules of the BPEL [3] specification, which can be found in [3, Appendix B]. The rules are numbered from 1 to 95, but because rule 49 is missing, the specification has 94 rules in total.

This technical report is structured as follows. First, due to the large amount of rules, we have created a tag system in section 2 to categorize them. This enables to reason within these rule groups as it is done in [1]. Next, we have created a formalization of these rules in section 3 including a reasoning on what different combinations of BPEL features these rules cover as well as how many combinations relate to rule violations and possible test cases. Last, in section 4, we provide links on where to find the test cases that have been created based on the formalizations and the resulting possible combinations of the BPEL features.

2 Tag System

Because of the large number of rules, we have tagged them according to two groups, namely, their *violation check* (How are the targets checked?) and their *target elements* (What BPEL features are restricted further?). These tags allow us to gain additional insight as they group the rules in a comprehensive and easy to interpret way. The tags and the rules that they refer to are given in table 1. A rule is tagged at least once per group, and can be tagged multiple times within each tag group.

The *violation check* tags describe the type of the check. The three tags with the highest number of rules, namely, *node requirement*, *choice* and *uniqueness* compensate the lax schema definition. The rules tagged with *node requirement* requires specific elements or attribute values, where the BPEL schema provides a greater choice. If a rule demands the usage of attributes and elements in specific combinations, it is tagged with *choice*. In this case, *choice* can refer to an *inclusive or* or an *exclusive or*. A more strict schema could have made these rules obsolete in the first place, e.g., by using the native schema choice mechanism instead. Rules with the *uniqueness* tag check the uniqueness of attributes or elements, e.g., the name attributes of `<variable>` definitions have to be unique per `<scope>`. The native schema mechanisms for uniqueness could have rendered these rules redundant. There are 14 rules tagged with *consistent redundancy* that deal with nodes which carry redundant information that can be derived from the context, e.g., from other attributes, elements or the location of an activity. The tag *location* is assigned to rules that restrict possible parents or ancestors of activities. Such rules are necessary due to undesired inheritance defects of the BPEL schema types, e.g., `<rethrow>` is a common activity even though it is solely applicable in the context of `<faultHandlers>`. The rules with the tag *execution instructions* instruct the BPEL engine how to execute the process, e.g., how to lookup a variable at runtime. Conformant static analysis can detect errors that occur when the execution would be performed in the correct order, but invalid execution at runtime remains unchecked by static analysis, of course. Hence, these rules can only be checked up to a certain point with static analysis. Rules are tagged with *definition resolution* if the rules require that a BPEL activity references other BPEL, WSDL or XSD elements by a qualified name (QName) or other references. Control cycles are forbidden in BPEL and the rules that describe their detection are tagged with *control cycle detection*.

The tags in the *target elements* group indicate which BPEL features are restricted further. The majority of the rules refer to activities related to the message exchanges and their required WSDL and XSD definitions, whereas only a minority restricts the structured activities.

Table 1: Static Analysis Rules: Grouped by Tag

tag	**rules**	**Σ**
violation check		
node requirements	1, 3, 4, 9, 13, 15, 17, 24, 26–31, 33, 35, 36, 38–43, 45, 47, 50, 53, 54, 57, 60, 62, 73–76, 78, 80, 91, 94	39
choice	16, 17, 19, 20, 25, 32, 34, 47, 51, 52, 55, 59, 63, 80, 81, 83, 85, 90	18
uniqueness	2, 14, 18, 22, 23, 44, 64, 66–69, 76, 86, 92, 93	15
consistent redundancy	5, 11, 12, 34–37, 46, 48, 57, 58, 79, 86, 87	14
location	6, 7, 8, 56, 60, 61, 65, 70, 71, 77, 79	11
definition resolution	10, 21, 42, 43, 60, 65, 73, 86, 94, 95	10
execution instructions	84, 88, 89, 95	4
control cycle detection	72, 82	2
target activities		
WSDL definitions	1, 2, 5, 10–14, 19–22, 45–48, 50, 53, 54, 58, 60, 84, 87, 88	24
message activities	5, 10, 21, 46–48, 50–55, 58, 59, 61, 63, 78, 84, 85, 87, 89, 90	22
expressions	26–31, 33, 38–43, 73–75, 94	17
process and scope	3, 18, 23, 44, 61, 77–80, 82, 83, 88, 91–93	15
message assignment activities	47, 48, 50–55, 58, 59, 63, 85, 87, 90	14
FCT handler activities	3, 6–8, 10, 70, 71, 77–81, 93	13
flow activities	64–72, 82	10
partner link activities	5, 10, 16–18, 35–37, 84	9
variable activities	10, 23–25, 34, 48, 58, 86, 90	9
XSD definitions	10, 21, 32, 34–37	7
assignment activities	10– 14, 45	6
correlation activities	10, 21, 44–46, 88	6
event handler activities	83, 86, 88, 89, 95	5
loop activities	62, 70, 76, 83	4
start activities	15, 56, 57, 62	4
engine specific activities	4, 9	2

In this work, we focus on a subset of 71 rules, leaving 23 rules out of scope as shown in table 2. In table 3, a subset of the tagging information of table 1 is shown by focusing only on the covered rules.

Table 2: Static Analysis Rules: Covered and Out of Scope

group	**rules**	**Σ**
Covered	1, 2, 3, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22, 23, 24, 25, 32, 34, 35, 36, 37, 44, 45, 46, 47, 48, 50, 51, 52, 53, 54, 55, 57, 58, 59, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 76, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 95	71
Out of Scope	4, 9, 21, 26, 27, 28, 29, 30, 31, 33, 38, 39, 40, 41, 42, 43, 56, 60, 73, 74, 75, 77, 94	23

Table 3: Covered Static Analysis Rules Grouped by Tag

tag	**rules**	**Σ**
violation check		
node requirements	1, 3, 13, 15, 17, 24, 35, 36, 45, 47, 50, 53, 54, 57, 62, 76, 78, 80, 91	19
choice	16, 17, 19, 20, 25, 32, 34, 47, 51, 52, 55, 59, 63, 80, 81, 83, 85, 90	18
uniqueness	2, 14, 18, 22, 23, 44, 64, 66–69, 76, 86, 92, 93	15
consistent redundancy	5, 11, 12, 34–37, 46, 48, 57, 58, 79, 86, 87	14
location	6–8, 61, 65, 70, 71, 79	8
definition resolution	10, 65, 86, 95	4
execution instructions	84, 88, 89, 95	4
control cycle detection	72, 82	2
target activities		
WSDL definitions	1, 2, 5, 10–14, 19, 20, 22, 45–48, 50, 53, 54, 58, 84, 87, 88	22
message activities	5, 10, 46–48, 50–55, 58, 59, 61, 63, 78, 84, 85, 87, 89, 90	21
process and scope	3, 18, 23, 44, 61, 78–80, 82, 83, 88, 91–93	14
message assignment activities	47, 48, 50–55, 58, 59, 63, 85, 87, 90	14
FCT handler activities	3, 6–8, 10, 70, 71, 78–81, 93	12
flow activities	64–72, 82	10
partner link activities	5, 10, 16–18, 35–37, 84	9
variable activities	10, 23–25, 34, 48, 58, 86, 90	9
XSD definitions	10,–14, 45	6
assignment activities	10, 32, 34–37	6
correlation activities	10, 44–46, 88	5
event handler activities	83, 86, 88, 89, 95	5
loop activities	62, 70, 76, 83	4
start activities	15, 57, 62	3

3 Formalization

The formalization of a static analysis rule lists all BPEL elements and attributes in categories that can be instantiated to get all possible violations and valid combinations concerning a rule. The categories are, for example, attribute values or affected message activities. Processes with one of the combinations are divided in the groups of valid and invalid processes. The different invalid processes, required to test a rule entirely, are identified by the formalization as well as the corresponding valid process.

The purpose of the formalization is the structured creation of good tests. No other formalization met our requirements for that task, because we see the importance in modeling both, correct and erroneous processes explicitly. Thus, we had to create our own formalization, starting from the model of Kopp et al. [2]. Many of the rule models could be negated to see how the invalid processes are. In table 4, a comparison of the rules is shown. In total, we cover five more rules than Kopp et al. [2]. Four rules (19, 25, 75 and 80) are mentioned in [2], but are not defined any further. Rule 53 is defined but excluded directly after its definition, and rule 21 is not mentioned at all. The other rules are explicitly in- or excluded.

Table 4: Static Analysis Rule Coverage: Our Approach vs. the Model in [2]

type	**rules**	**Σ**
both included	1–3, 5–8, 15–18, 20, 22–24, 32, 34–37, 44–48, 50–52, 54, 55, 57–59, 61–72, 76, 78, 79, 81–83, 85–87, 90–93, 95	59
both excluded	4, 9, 21, 26–31, 33, 39–41, 43, 75, 94	16
only our approach	10–14, 19, 25, 53, 80, 84, 88, 89	12
only Kopp et al. [2]	38, 42, 56, 60, 73, 74, 77	7

For the rules that we and Kopp et al. cover, we have created a more detailed comparison shown in table 5 which uncovers that ten rules are formalized wrongly or not sufficiently enough to create test cases from in [2]. Their issues are given in table 6.

Table 5: Static Analysis Rule Coverage: Our Approach vs. the Model in [2]: Detailed Comparison of the Static Analysis Rules Covered by Both Approaches

	rules	**Σ**
same expressive power	1–3, 5–8, 15–18, 22–24, 34–37, 44–46, 48, 50–52, 54, 55, 57–59, 62–64, 66–69, 71, 72, 76, 79, 81, 82, 85–87, 91–93	49
model of Kopp et al. [2] is insufficient	20, 32, 47, 61, 65, 70, 83, 95	8
model of Kopp et al. [2] is wrong	78, 90	2

Static Analysis Rules which are modeled wrong or insufficient in [2] The formalization of rule 78 by Kopp et al. [2, p. 58] is about unique names of peer scopes, but the standard [3, p. 203] deals with the location of the `@taget` of a `<compensateScope>`. Thus, the formalization of Kopp et al. is wrongly named for rule 78 and actually is a rule 92 duplicate, as rule 92 has its own correct formalization [2, p. 57]. Rule 90 is defined along with rules 47, 58, and 87 which deal with correct type usage in message exchange activities, but rule 90 makes either `@messageType`

Table 6: Issues with Model of [2]

rule	what?	why?
20	insufficient	parts are not considered in the model
32	insufficient	definition for `<from>` is missing
47	insufficient	regular messages are missing
61	insufficient	check only name uniqueness instead of name
65	insufficient	incomplete
70	insufficient	`<eventHandlers>` not covered
83	insufficient	required element undefined
95	insufficient	handles part of rule 85 instead
78	wrong	handle different elements and different behavior
90	wrong	focus on types instead of the occurrence of attributes

or `@element` a necessary attribute if `@variable` is used in an `<onEvent>`. Therefore, the rule formalization by Kopp et al. is unsuitable, so less tests are required as for the other rules defined along [2, p. 27]. A more detailed explanation why something is modeled wrong or insufficient can be found in the details of each rule.

Static Analysis Rules which are Out of Scope for our Approach The rules 21, 56, 60 and 77 are considered out of scope according to table 2, however, a formalization is given in this section nevertheless. As these rules are not part of the tables in this section, we state their differences briefly in the following. Rule 21 is missing from the model by Kopp et al., rule 56 is included but the formalization cannot be used to drive test cases from, rule 60 is included as well but because tests derive from negating the model would also violate rule 61 this definition is insufficient for our purposes, and rule 77 is also included but its negation is incomplete, hence, cannot be used to drive every test cases.

SA00001

"A WS-BPEL processor MUST reject a WS-BPEL that refers to solicit-response or notification operations portTypes."[3, p. 194]

This rule describes two possible error types that are excluded in the formal model by [2, p. 27]. These errors narrow the definition of the four possible message exchange patterns [4, see section 2.4] down to two, namely, one-way and request-response. In terms of implementation, these patterns differ in the order and existence of `<output>` and `<input>` messages for an `<operation>` in a WSDL definition. Hence, the permutation of both `<output>` and `<input>` in a WSDL `<operation>` definition creates four combinations. Each combination refers to a specific message exchange pattern. Consequently, two of these combinations are forbidden whereas the remaining two are valid. We solely change the locations of the Web Service Description Language (WSDL)s in the derived Web Service Business Process Execution Language (BPEL) processes and modified WSDLs.

SA00002

"A WS-BPEL processor MUST reject any WSDL portType definition that includes overloaded operation names."[3, p. 194]

This rule ensures that the `@name` of an `<operation>` is unique within its `<portType>` in a WSDL definition. The negation of the formalization of this rule in [2, p. 11] reveals that only a single test is required which contains a name duplicate.

SA00003

"If the value of exitOnStandardFault of a <scope> or <process> is set to 'yes', then a fault handler that explicitly targets the WS-BPEL standard faults MUST NOT be used in that scope."[3, p. 194]

Due to inheritance rules of `@exitOnStandardFault` in both `<process>` and `<scope>`, we can distinguish between two different inheritance situations: 1) `<process>` and `<scope>` explicitly defining the value, and 2) a `<scope>` and `<process>` use the implicit value, namely, a `<scope>` that inherits the value `"yes"` for the attribute or a `<process>` that uses the default value. The exceptions are derived from the specification which lists all available standard faults [3, pp. 193], except for the `joinCondition` as this standard fault is explicitly excluded in the rule definition. In [2, p. 56], the same list of standard faults is used, as in their definition they exclude the standard fault `joinCondition` as well. This results in 152 combinations, of which half are valid combinations whereas the other half corresponds to invalid ones, as `exitOnStandardFault="yes"` always holds for all erroneous tests.

Combinations for process The formalization shown below is complete as every possible state of a `<process>` as well as every possible standard fault for this case is listed. In total, this amounts to $19 * 3 = 57$ combinations. There are $19 * 1 = 19$ test cases, namely, one for each standard fault with the `<process>` having `exitOnStandardFault="yes"`.

`<catch>` with `faultName="bpel:VALUE"`
`VALUE` ∈ [ambiguousReceive, completionConditionFailure, conflictingReceive, conflictingRequest, correlationViolation, invalidBranchCondition, invalidExpressionValue, invalidVariables, mismatchedAssignmentFailure, missingReply, missingRequest, scopeInitializationFailure, selectionFailure, subLanguageExecutionFault, uninitializedPartnerRole, uninitializedVariable, unsupportedReference, xsltInvalidSource, xsltStylesheetNotFound]
×
[`exitOnStandardFault="yes"`, `exitOnStandardFault="no"`, default `exitOnStandardFault="no"`]

Formalization 1: SA00003 for `<process>`

Combinations for scope Below, the formalization for `<scope>`s is shown. To combat test explosion, we considered to include both explicit setting of `@exitOnStandardFault` and inheriting the value from an enclosing `<scope>` or `<process>`. But we did not include arbitrary large inheritance dependencies, e.g., `<scope>` in `<scope>` in `<scope>`, as this is not feasible. In total, this amounts to $19 * 3 * 4 * 3 = 684$ combinations. In this case, we have $19 * 1 * 4 * 3 = 19 * 12 = 228$ test cases when the `<scope>` has `exitOnStandardFault="yes"` set explicitly, and $19 * 3 * 1 * 1 = 19 * 3 = 57$ when the `<scope>` inherits `exitOnStandardFault="yes"` from the enclosing `<scope>` that has set `exitOnStandardFault="yes"` explicitly, and $19 * 1 * (1 + 1) = 19 * 2 = 38$ when it inherits the explicitly set `exitOnStandardFault="yes"` from the enclosing `<process>` via the enclosing `<scope>` or directly from the enclosing `<process>`. Hence, we have $228 + 57 + 38 = 323$ test cases in total for the `<scope>`.

`<catch>` with `faultName="bpel:VALUE"`
`VALUE` ∈ [ambiguousReceive, completionConditionFailure, conflictingReceive, conflictingRequest, correlationViolation, invalidBranchCondition, invalidExpressionValue, invalidVariables, mismatchedAssignmentFailure, missingReply, missingRequest, scopeInitializationFailure, selectionFailure, subLanguageExecutionFault, uninitializedPartnerRole, uninitializedVariable, unsupportedReference, xsltInvalidSource, xsltStylesheetNotFound]
×
[`exitOnStandardFault="yes"`, `exitOnStandardFault="no"`, inherited]
×
enclosing `<scope>` [`exitOnStandardFault="yes"`, `exitOnStandardFault="no"`, inherited, no enclosing scope]
×
enclosing `<process>` [`exitOnStandardFault="yes"`, `exitOnStandardFault="no"`, default]

Formalization 2: SA00003 for `<scope>`

SA00004

"If any referenced queryLanguage or expressionLanguage is unsupported by the WS-BPEL processor then the processor MUST reject the submitted WS-BPEL process definition."[3, p. 194]

This rule depends on engine implementations, therefore, it is postponed to future work.

SA00005

"If the portType attribute is included for readability, in a <receive>, <reply>, <invoke>, <onEvent> or <onMessage> element, the value of the portType attribute MUST match the portType value implied by the combination of the specified partnerLink and the role implicitly specified by the activity."[3, p. 194]

For each of the five different message activities, namely, `<invoke>`, `<receive>`, `<reply>`, `<onMessage>` and `<onEvent>`, there is one error if the `@portType` that has not the already

implied value. Additionally, [2, p. 26] distinguish between receiving and sending message activities because the implication is resolved via the communication role of the `<partnerLink>`, being `@partnerRole` and `@myRole`. However, this is not important for the tests, as we still require five tests, one for each message activity, in which the `@portType` is wrong.

SA00006

"The <rethrow> activity MUST only be used within a faultHandler (i.e. <catch> and <catchAll> elements)."[3, p. 194]

The correctness of the implementation of this rule can be determined by detecting `<rethrow>` in every wrong place. As this is unfeasible, we only test this condition in every activity that can contain other activities, namely, `<if>`, `<else>`, `<elseIf>`, `<flow>`, `<onAlarm>`, `<onMessage>`, `<repeatUntil>`, `<scope>`, `<sequence>`, `<while>`, `<compensationHandler>`, and `<terminationHandler>`. This list of activities does not include `<catch>` and `<catchAll>` due to the rule logic. We do not test any nesting of various activities, as this would result in test explosion. The accepted opposite is expressed as existential quantification in [2, p. 42]. Hence, we get twelve test cases, one for each containing activity in which we place `<rethrow>`.

SA00007

"The <compensateScope> activity MUST only be used from within a faultHandler, another compensationHandler, or a terminationHandler."[3, p. 194]

The correctness of the implementation of SA00007 can be determined by detecting `<compensateScope>` in every wrong place. As this is unfeasible, we test this condition in every activity that can contain other activities (analogously to SA00006), namely, `<if>`, `<else>`, `<elseIf>`, `<flow>`, `<onAlarm>`, `<onMessage>`, `<repeatUntil>`, `<scope>`, `<sequence>` and `<while>`. This list of activities does not include `<catch>`, `<catchAll>`, `<compensationHandler>`, and `<terminationHandler>` due to the rule logic. Thus, we require ten tests and the accepted opposite is modeled with an existential quantification by [2, p. 59] as well.

SA00008

"The <compensate> activity MUST only be used from within a faultHandler, another compensationHandler, or a terminationHandler."[3, p. 194]

Analogous to SA00007, SA00008 requires the same amount of test cases and the accepted opposite is modeled with an existential quantification by [2, p. 59] as well.

SA00009

"In the case of mandatory extensions declared in the <extensions> element not supported by a WS-BPEL implementation, the process definition MUST be rejected."[3, p. 194]

This rule depends on engine implementations, therefore, it is postponed to future work.

SA00010

"A WS-BPEL process definition MUST import all XML Schema and WSDL definitions it uses. This includes all XML Schema type and element definitions, all WSDL port types and message types as well as property and property alias definitions used by the process."[3, p. 195]

All elements introducing constructs (depending on XML schema or WSDL definition) must refer an imported file, e.g., a `<variable>` requires an *include check* for these defining documents. SA00010 is comprehensive and is not modeled by [2, p. 65] because it requires a formalization of WSDL and XML schema. Eleven elements need to be checked with WSDL definitions. Additionally, attributes of `<variable>` and `<correlationSet>` can reference constructs defined in XSDs directly and transitively. While the rule text states the `@propertyAlias` explicitly, it is implicitly tested every time a `@property` is resolved. Because of this, we did not include `@propertyAlias` in our formalization. Moreover, `@portType` is checked as part of SA00005.

[`<reply>`, `<receive>`, `<invoke>`, `<onMessage>`, `<onEvent>`]
$\times$
[`@operation`, `@partnerLink`]

Formalization 3: SA00010, for message activities

In addition to the rules from the previous listing, `<onEvent>` implicitly defines its own scope with its own variable, hence, the `@messageType` or `@element` can be set optionally.

[`<onEvent>`] $\times$ [`@messageType`, `@element`]

Formalization 4: SA00010, for `<onEvent>`

[`<catch>`] $\times$ [`@faultMessageType`, `@faultElement`]

Formalization 5: SA00010, for the `<catch>` activity

[`<variable>`] $\times$ [`@messageType`, `@type`, `@element`]

Formalization 6: SA00010, for `<variable>`

[`<partnerLink>`] $\times$ [`@partnerLinkType`]

Formalization 7: SA00010, for `<partnerLink>`

[<to>, <from>] × [@property]

Formalization 8: SA00010, for <to> and <from>

[<correlationSet>] × [@properties]

Formalization 9: SA00010, for <correlationSet>

In total, we have $(2 * 5) + 2 + 2 + 3 + 1 + 2 + 1 = 21$ combinations, which are the test cases.

SA00011

"If a namespace attribute is specified on an <import> then the imported definitions MUST be in that namespace."[3, p. 195]

To violate SA00011 the document that is imported via <import> in the BPEL process definition, the @namespace of the <import> has to differ from the @targetNamespace of the imported document. Hence, only a single test case is required. This rule is explicitly excluded from the model of [2, p. 65].

SA00012

"If no namespace is specified then the imported definitions MUST NOT contain a targetNamespace specification."[3, p. 195]

SA00012 is similar to SA00011. To violate it, we require an <import> with a namespace="" and a non-empty @targetNamespace in the imported document. Thus, a single test is sufficient. [2, p. 65] exclude this rule as well.

SA00013

"The value of the importType attribute of element <import> MUST be set to http://www.w3.org/2001/XMLSchema when importing XML Schema 1.0 documents, and to http://schemas.xmlsoap.org/wsdl/ when importing WSDL 1.1 documents. "[3, p. 195]

The import of both XSD and WSDL files using <import> requires specifying the correct @importType, i.e., the namespace of the imported document, as stated in SA00013. Therefore, two tests are required: a) one for XSD <import> and b) one for WSDL <import>. [2, p. 65] exclude this rule from their model.

SA00014

"A WS-BPEL process definition MUST be rejected if the imported documents contain conflicting definitions of a component used by the importing process definition (as could be caused, for example, when the XSD redefinition mechanism is used)."[3, p. 195]

[2, p. 65] do not model SA00014. We, however, determine the necessary tests by permuting the combinations of doubled definitions in WSDL and schema files.

Regarding the redefinition of an element, the original version has to be defined within the XSD, whereas the redefinition may occur in the WSDL under `<types>`, or in a separate XSD. For that purpose, we have created an additional XSD that contains also `<simpleType>`, `<group>`, and `<attributeGroup>` in addition to `<complexType>` and `<element>` definitions. Please note that `<element>`s cannot be redefined.

XSD with [`<simpleType>`, `<complexType>`, `<group>`, `<attributeGroup>`]
×
redefinition in [WSDL, XSD, none]

Formalization 10: SA00014 for XSD redefinition

From the $4*3 = 12$ combinations, only $4*2 = 8$ are invalid, because when they are not redefined, there is no problem.

[`<simpleType>`, `<complexType>`, `<element>`, `<group>`, `<attributeGroup>`]
×
definition in [WSDL, XSD]
×
in [WSDL, XSD, none]

Formalization 11: SA00014 for XSD elements, types, etc.

From the $5 * 2 * 3 = 24$ combinations, only $(5 * 2 * 2) - 5 = 15$ are invalid, because when they are not defined again, there is no problem.

WSDL and XSD files combined have the same effects in any import order.

[`<operation>`, `<message>`, `<portType>`, `<partnerLinkType>`, `<property>`]
×
definition in [WSDL]
×
in [WSDL, none]

Formalization 12: SA00014 for WSDL messages, operations, etc.

From the $5 * 1 * 2 = 10$ combinations, only $5 * 1 * 1 = 5$ are invalid, because when they are not defined in another WSDL, there is no problem.

In summary, we get $8 + 15 + 5 = 28$ test cases.

SA00015

"To be instantiated, an executable business process MUST contain at least one <receive> or <pick> activity annotated with a createInstance='yes' attribute."[3, p. 195]

In line with SA00015 model by [2, p. 61], we test `<receive>`s and `<pick>`s. In erroneous processes, `createInstance="no"` must hold in all `<receive>` and `<pick>` activities. This can be achieved in two ways: 1) explicitly set the attribute or 2) use the default value.

[only `<receive>`, only `<pick>`] in process × with [`createInstance="no"`, default value of `@createInstance` which is `"no"`, `createInstance="yes"`]

Formalization 13: SA00015

If `createInstance="yes"` holds, the process is valid. Four tests are required to cover the violations of SA00015, two for either `<receive>` and `<pick>` activities.

SA00016

"A partnerLink MUST specify the myRole or the partnerRole, or both."[3, p. 195]

By negating the SA00016 definition of [2, p. 25], we identify the single test case for this rule, which is a `<partnerLink>` with neither `@myRole` nor `@partnerRole`.

SA00017

"The initializePartnerRole attribute MUST NOT be used on a partnerLink that does not have a partner role."[3, p. 195]

Analog to SA00016, SA00017 has a single test that is also specified by the negation of the rule definition by [2, p. 25], in this case, a `<partnerLink>` with `initializePartnerRole="yes"` with no `@partnerRole`.

SA00018

"The name of a partnerLink MUST be unique among the names of all partnerLinks defined within the same immediately enclosing scope."[3, p. 195]

A `<partnerLink>` can be declared in a `<scope>` or in `<process>`, so these are the locations where the violations of SA00018 can occur. Hence, we have created two tests, one for a duplicated `<partnerLink>` name on the `<process>` level and one on the `<scope>` level. In [2, p. 25], there

is a more complex definition than is required for executable BPEL processes, as they include abstract BPEL processes as well.

SA00019

"Either the type or element attributes MUST be present in a <vprop:property> element but not both."[3, p. 195]

The violation case of SA00019 is to have an `@element` and a `@type` in a `<property>`, or neither an `@element` nor a `@type`. [2, p. 12] mention this rule, but they do not model it because the rule concerns only WSDL elements.

[`@element`, none]
×
[`@type`, none]

Formalization 14: SA00019

SA00020

"A <vprop:propertyAlias> element MUST use one of the three following combinations of attributes: messageType and part, type or element"[3, p. 195]

Even though SA00020 is about WSDL constraints similarly to SA00019, [2, p. 13] model it. We look directly at the attributes of a `<propertyAlias>` and identify following possible combinations.

[`@messageType`, none]
×
[`@part`, none]
×
[`@type`, none]
×
[`@element`, none]

Formalization 15: SA00020

The single `@element`, `@type` or the pair `@messageType` and `@part` are valid attributes, i.e., three out of the $2 * 2 * 2 * 2 = 16$ combinations are valid. Hence, $16 - 3 = 13$ combinations are erroneous, requiring 13 tests.

The derived test cases import all a modified WSDL that have all combinations of `@messageType`, `@part`, `@type`, and `@element` except the three allowed combinations.

SA00021

"Static analysis MUST detect property usages where propertyAliases for the associated variable's type are not found in any WSDL definitions directly imported by the WS-BPEL process."[3, p. 196]

This rule is our of scope. The formalization must be considered experimental and incomplete.

SA00021 is experimental, as we do not handle detecting the correct property usage via the `getVariableProperty` function.

SA00021 is not mentioned in the model of Kopp et al. [2]. According to the rule, we have to identify the property usages in the message activities with `<correlation>`, and in the data flow activity `<assign>` in `<from>` or `<to>` directives. As the `<propertyAlias>` links a `<property>` to either a `<message>`, an XSD type or XSD `<element>` and vice versa, we assume that both are available for our various combinations because there are other rules covering their absence.

[`<receive>`, `<reply>`, `<onMessage>`, `<invoke>`, `<onEvent>`, `<from>`, `<to>`]
$\times$
[correct `<propertyAlias>` exists, correct `<propertyAlias>` is missing]

Formalization 16: SA00021

Seven test cases are required that have no `<propertyAlias>` for the `<property>` that is used in the `<correlation>` or assignment.

SA00022

"A WS-BPEL process definition MUST NOT be accepted for processing if it defines two or more propertyAliases for the same property name and WS-BPEL variable type."[3, p. 196]

Each `<propertyAlias>` element refers to either an XSD `<element>` via `@element`, an XSD type via `@type`, or a WSDL `<message>` and its `<part>` via `@messageType` and `@part`. For modeling this rule, we can omit the part, as a `<propertyAlias>` links a specific `<message>` to a specific `<property>` regardless of their part because correlation works on the message level and not on the part level. To test violations against SA00022, we require duplicate `<propertyAlias>` elements for each of the three possibilities. The rule model of [2, p. 13] can be negated to define the error cases.

SA00023

"The name of a variable MUST be unique among the names of all variables defined within the same immediately enclosing scope."[3, p. 196]

Like all uniqueness checks, SA00023can be tested by means of duplicated names. The `@name` of a variable has to be unique within its variable container (`<variables>`). This container can be within a `<scope>` or `<process>` element, hence, we require two test cases, one for each of the different locations of the container with duplicate `<variable>`s. The negation of this rule model by [2, p. 22] reveals the two necessary tests that we mentioned above.

SA00024

"Variable names are BPELVariableNames, that is, NCNames (as defined in XML Schema specification) but in addition they MUST NOT contain the '.' character."[3, p. 196]

The `@name` is possible for almost every element in a BPEL process. Each test for SA00024 has a BPEL element with a `@name` containing a ".", which can be identified by negating the rule model by [2, p. 22]. This rule can be checked by means of a schema validation with the official XSD of the BPEL specification. This is the only rule that is already covered as part of the schema validation, which we assume the engines already use extensively. Because of this, instead of creating a plethora of tests, we have created a single test case that reveals whether this rule is checked or not. The test case uses bad variable names because they are crucial for any BPEL process.

SA00025

"The messageType, type or element attributes are used to specify the type of a variable. Exactly one of these attributes MUST be used."[3, p. 196]

SA00025 is not defined precisely in [2, p. 11], but the required tests can be easily identified by permuting the three attributes.

[`@element`, none]
$\times$
[`@type`, none]
$\times$
[`@messageType`, none]

Formalization 17: SA00025

Three of the $2 * 2 * 2 = 8$ combinations are valid; each has just one attribute. Therefore, we require $8 - 3 = 5$ tests in total.

SA00026

"Variable initialization logic contained in scopes that contain or whose children contain a start activity MUST only use idempotent functions in the from-spec."[3, p. 196]

Because the rule deals with general expression parsing, it is postponed to future work.

SA00027

"When XPath 1.0 is used as an expression language in WS-BPEL there is no context node available. Therefore the legal values of the XPath Expr (http://www.w3.org/TR/xpath#NT-Expr) production must be restricted in order to prevent access to the context node. Specifically, the 'LocationPath' (http://www.w3.org/TR/xpath#NT-LocationPath) production rule of 'PathExpr' (http://www.w3.org/TR/xpath#NT-PathExpr) production rule MUST NOT be used when XPath is used as an expression language."[3, p. 196]

The rule deals with the parsing of XPath expressions and is postponed to future work.

SA00028

"WS-BPEL functions MUST NOT be used in joinConditions."[3, p. 196]

Because the rule deals with general expression parsing, it is postponed to future work.

SA00029

"WS-BPEL variables and WS-BPEL functions MUST NOT be used in query expressions of propertyAlias definitions."[3, p. 196]

Because the rule deals with general expression parsing, it is postponed to future work.

SA00030

"The arguments to bpel:getVariableProperty MUST be given as quoted strings. It is therefore illegal to pass into a WS-BPEL XPath function any XPath variables, the output of XPath functions, a XPath location path or any other value that is not a quoted string."[3, p. 196]

The rule deals with the parsing of XPath expressions and is postponed to future work.

SA00031

"The second argument of the XPath 1.0 extension function bpel:getVariableProperty(string, string) MUST be a string literal conforming to the definition of QName in [XML Namespaces] section 3."[3, p. 197]

The parsing of BPEL functions, as required for this rule, is postponed to future work.

SA00032

“For <assign>, the <from> and <to> element MUST be one of the specified variants. The <assign> activity copies a type-compatible value from the source (’from-spec’) to the destination (’to-spec’), using the <copy> element. Except in Abstract Processes, the fromspec MUST be one of the following variants:

```
<from variable='BPELVariableName' part='NCName'?>
    <query queryLanguage='anyURI'?>?queryContent </query>
</from>

<from partnerLink='NCName'  endpointReference='myRole|partnerRole' />

<from variable='BPELVariableName'  property='QName' />

<from expressionLanguage='anyURI'?> expression </from>

<from>
    <literal>literal value</literal>
</from>

<from/>
```

In Abstract Processes, the from-spec MUST be either one of the above or the opaque variant described in section 13.1.3. Hiding Syntactic Elements The to-spec MUST be one of the following variants:

```
<to variable='BPELVariableName' part='NCName'?>
    <query queryLanguage='anyURI'?>?queryContent </query>
</to>

<to partnerLink='NCName' />

<to variable='BPELVariableName'  property='QName' />

<to expressionLanguage='anyURI'?> expression </to>

<to/>
```

”[3, p. 197]

SA00032 model by [2, p. 37] ignores the `<from>`, and the definition is vague and more textual than the original rule. To get the tests, we modified each of the six variants of `<from>` (and five forms of `<to>`) by adding elements or attributes. The empty version of both `<from>` and `<to>` is ignored, as it is meaningless. As a result, we only have five variants of `<from>` and four of `<to>`. Because we have combined the two versions using `@variable`, we are down to four variants of `<from>` and three for `<to>`. In the following, first, the variants for the `<from>` element are stated, directly followed by the variants for the `<to>` element.

The from Variants

The <from> element consists of four variants, each of which are explained with a formalization and a textual description.

[<from> with @variable]
×
[@property, @part, <query>, @part and <query>, @property and @part, @property and @query,]
×
one additional one out of [none, @partnerLink, @endpointReference, expression, @expressionLanguage, <literal>]

Formalization 18: SA00032 for <from> with @variable

Out of the $1 * 6 * 6 = 36$ combinations, only four (@property, @part, <query> and @part with <query> combined with no additional element) are valid, hence, we have $36 - 4 = 32$ faulty combinations. However, we ignore $2 * 5 = 10$ of them because @property and @part as well as @property and @query already violates this rule and adding additional errors does not help. Moreover, as the combination of <query> and <literal> is caught as part of the XML schema validation (there are two cases for which this would happen), we get $32 - 10 - 2 = 20$ actual tests.

[<from> with @partnerLink and @endpointReference]
×
one additional one out of [none, @variable, expression, @expressionLanguage, <literal>, @property, @part, <query>]

Formalization 19: SA00032 for <from> with @partnerLink and @endpointReference

Hence, we have 8 combinations with only a single valid one, hence $8 - 1 = 7$ test cases.

[<from> with expression]
×
[expression, @expressionLanguage, expression and @expressionLanguage,]
×
one additional one out of [none, @partnerLink, @variable, @part, <query>, @property, @endpointReference, <literal>]

Formalization 20: SA00032 for <from> with expression

Hence, we have $1 * 3 * 8 = 24$ combinations. Only two (none with expression or expression and @expressionLanguage) are valid. Using @expressionLanguage without an expression is always invalid, hence, we only create a single test case, ignoring the other seven tests. This results in $24 - 2 - 7 = 15$ test cases.

[<from> with <literal>]
×
one additional one out of [none, @partnerLink, @variable, expression, @expressionLanguage, @endpointReference, @property, @part, <query>]

Formalization 21: SA00032 for <from> with <literal>

Hence, we have $1 * 9 = 9$ combinations with eight test cases, as one case (no addition) is valid. However, as the combination with @expressionLanguage or expression is already covered by the previous formalization, we have two duplicate test cases. Moreover, as the combination of <query> and <literal> is caught as part of the XML schema validation, we get five test cases in total.

In total, we have $20 + 7 + 15 + 5 = 47$ test cases for the four <from> variants.

The to Variants

The <to> element consists of three variants, each of which are explained with a formalization and a textual description.

[@property, @part, <query>, @part and <query>, @property and @part, @property and @query,]
×
one additional one out of [none, @partnerLink, expression, @expressionLanguage]

Formalization 22: SA00032 for <to> with @variable

Hence, we have $6 * 4 = 24$ combinations, but $24 - 4 = 20$ test cases. This is because @property, or @part, or @part with <query>, and <query> with none are valid ($1+1+1+1 = 4$), and every other combination is invalid. Six cases can be ignored because using @property with something else is always an error, resulting in $24 - 4 - 6 = 14$. We did not consider the @queryLanguage attribute of the <query>, as this would cause additional testing, resulting in more test explosion.

one additional one out of [none, @variable, expression, @expressionLanguage, @property, @part, <query>]

Formalization 23: SA00032 for <to> with @partnerLink

Hence, we have 7 combinations, and six test cases as one is valid.

[expression, @expressionLanguage, expression and @expressionLanguage,]
×
one additional one out of [none, @partnerLink, @variable, @part, <query>, @property]

Formalization 24: SA00032 for <to> with expression

Hence, we have $3 * 6 = 18$ combinations, of which there are $18 - 2 - 5 - 2 = 9$ test cases, because adding nothing for either expression or expression and `@expressionLanguage` ($1 * 2 = 2$) is valid, as every combination with `@expressionLanguage` is always bad we can ignore five test cases, and one test case is duplicated in the previous formalization, namely the combination of `@partnerLink` and expression.

In total, we have $14 + 6 + 10 = 30$ test cases for the three `<to>` variants.

Summary

In summary, we get $47 + 30 = 77$ test cases for all the `<from>` and `<to>` variants.

SA00033

"The XPath expression in <to> MUST begin with an XPath VariableReference."[3, p. 198]

The rule deals with the parsing of XPath expressions and is postponed to future work.

SA00034

"When the variable used in <from> or <to> is defined using XML Schema types (simple or complex) or element, the part attribute MUST NOT be used."[3, p. 198]

[2, p. 35] define the positive model of SA00034 for `<from>`, but not for `<to>` elements. The other attributes combined with the `@part` are the tests for this rule.

[`<from>`, `<to>`]
×
[`<variable> @element`, `<variable> @type`, `<variable> @messageType`, `<onEvent> @variable @element`, `<onEvent> @variable @messageType`, `<forEach> @counterName`]
×
[`@part`, none]

Formalization 25: SA00034

Overall, we have $2 * 6 * 2 = 24$ combinations. All tests with `@messageType` and `@part` are valid, as well as all variants without `<part>`, summing up to $2 * 2 * 1 + 2 * 6 * 1 = 4 + 12 = 16$. Thus, we need four tests for each `<from>` and `<to>`, leading to eight tests in total to test all aspects of SA00034.

SA00035

“In the from-spec of the partnerLink variant of <assign> the value "myRole" for attribute endpointReference is only permitted when the partnerLink specifies the attribute myRole.”[3, p. 198]

If there is no `@myRole` in the referenced `<partnerLink>` used in an `<assign>`, but `endpoint-Reference="myRole"` holds, then SA00035 is violated, which can be evaluated in a single test. In the model of [2, p. 36], the concrete usage of the `<partnerLink>` is unmentioned, however, the negation of the model indicates the test case.

SA00036

“In the from-spec of the partnerLink variant of <assign> the value "partnerRole" for attribute endpointReference is only permitted when the partnerLink specifies the attribute partnerRole.”[3, p. 198]

SA00036 resembles SA00035, so does the model by [2, p. 36]. They differ in the role only, as SA00036 refers to the `partnerRole` instead of the `myRole`.

SA00037

“In the to-spec of the partnerLink variant of assign only partnerLinks are permitted which specify the attribute partnerRole.”[3, p. 198]

Negating SA00037 model by [2, p. 38] shows the single test required: The `<partnerLink>` shall have no `@partnerRole` when referenced from a `<to>`.

SA00038

“The literal from-spec variant returns values as if it were a from-spec that selects the children of the <literal> element in the WS-BPEL source code. The return value MUST be a single EII or Text Information Item (TII) only.”[3, p. 198]

The rule is postponed to future work because it deals with expression parsing of arbitrary literal expressions.

SA00039

“The first parameter of the XPath 1.0 extension function bpel:doXslTransform(string, node-set, (string, object)*) is an XPath string providing a URI naming the style sheet to be used by the WS-BPEL processor. This MUST take the form of a string literal.”[3, p. 198]

The parsing of BPEL functions, as required for this rule, is postponed to future work.

SA00040

"In the XPath 1.0 extension function bpel:doXslTransform(string, node-set, (string, object)*) the optional parameters after the second parameter MUST appear in pairs. An odd number of parameters is not valid."[3, p. 198]

The parsing of BPEL functions, as required for this rule, is postponed to future work.

SA00041

"For the third and subsequent parameters of the XPath 1.0 extension function bpel:doXslTransform(string, node-set, (string, object)*) the global parameter names MUST be string literals conforming to the definition of QName in section 3 of [Namespaces in XML]."[3, p. 198]

The parsing of BPEL functions, as required for this rule, is postponed to future work.

SA00042

"For <copy> the optional keepSrcElementName attribute is provided to further refine the behavior. It is only applicable when the results of both from-spec and to-spec are EIIs, and MUST NOT be explicitly set in other cases."[3, p. 198] To identify the types of all `<from>` and `<to>` expression parsing is required.

Because the rule deals with general expression parsing, it is postponed to future work.

SA00043

"For a copy operation to be valid, the data referred to by the from-spec and the to-spec MUST be of compatible types. The following situations are considered type incompatible:

- the selection results of both the from-spec and the to-spec are variables of a WSDL message type, and the two variables are not of the same WSDL message type (two WSDL message types are the same if their QNames are equal).
- the selection result of the from-spec is a variable of a WSDL message type and that of the to-spec is not, or vice versa (parts of variables, selections of variable parts, or endpoint references cannot be assigned to/from variables of WSDL message types directly)."[3, p. 199]

To identify the types of all `<from>` and `<to>` expression parsing is required. Because the rule deals with general expression parsing, it is postponed to future work.

SA00044

"The name of a <correlationSet> MUST be unique among the names of all <correlationSet> defined within the same immediately enclosing scope."[3, p. 199]

SA00044 requires both `<scope>`s and `<process>`s to have unique `@name`s for `<correlationSet>` elements in their `<correlationSets>` container. We test this with a duplication test for both `<scope>` and `<process>`, hence, we have two test cases. These tests are also modeled by negating the formalization in [2, p. 32].

SA00045

"Properties used in a <correlationSet> MUST be defined using XML Schema simple types."[3, p. 199]

`<property>` elements referenced in a `<correlationSet>` with either complex types of the XML schema or no type violate SA00045. Such `<correlationSet>` are forbidden in the rule model of [2, p. 31], thus, this results in two test cases, one for each variant of the `<correlationSet>`.

SA00046

"The pattern attribute used in <correlation> within <invoke> is required for request-response operations, and disallowed when a one-way operation is invoked."[3, p. 199]

The SA00046 model by [2, p. 32] has an equivalence relation, so we can switch sides of the parameters and negate the original and the switched statement. In other words, the two subsequent tests have either no `@pattern` in conjunction with a request-response communication or a `@pattern` in conjunction with a one-way communication.

SA00047

"One-way invocation requires (<invoke>) only the inputVariable (or its equivalent <toPart> elements) since a response is not expected as part of the operation. Request-response invocation requires both an inputVariable (or its equivalent <toPart> elements) and an outputVariable (or its equivalent <fromPart> elements). If a WSDL message definition does not contain any parts, then the associated attributes variable, inputVariable or outputVariable, MAY be omitted,and the <fromParts> or <toParts> construct MUST be omitted."[3, p. 199]

[2, p. 29] model SA00047 for the assignment of parts using `<fromPart>` or `<toPart>` for the empty message aspect, but not for non-empty messages. To violate the rule, however, you need to consider both cases.

[empty <message>, <message> with <part>(s)]
×
[<invoke>]
×
[@inputVariable, none]
×
[<fromParts>, none]

Formalization 26: SA00047 for <invoke> input

From the $2 * 1 * 2 * 2 = 8$ combinations, we can ignore $2 * 1 * 1 * 1 = 2$ cases which are violated by other rules. Moreover, $1 * 1 * 2 * 1 = 2$ cases are valid for an empty <message> without any <parts> and $1 * 1 * 1 * 1 + 1 * 1 * 1 * 1 = 2$ cases are valid for having a <message> with <part> and a variable or part assignment. Hence, $8 - 2 - 2 - 2 = 2$ cases are invalid, hence, two test cases.

[empty <message>, <message> with <part>(s)]
×
[<invoke>]
×
[@outputVariable, none]
×
[<toParts>, none]

Formalization 27: SA00047 for <invoke> output

From the $2 * 1 * 2 * 2 = 8$ combinations, we can ignore $2 * 1 * 1 * 1 = 2$ cases which are violated by other rules. Moreover, $1 * 1 * 2 * 1 = 2$ cases are valid for an empty <message> without any <parts> and $1 * 1 * 1 * 1 + 1 * 1 * 1 * 1 = 2$ cases are valid for having a <message> with <part> and a variable or part assignment. $8 - 2 - 2 - 2 = 2$ cases are invalid, hence, two test cases.

[empty <message>, <message> with <part>(s)]
×
[<receive>, <onMessage>, <onEvent>]
×
[@variable, none]
×
[<fromParts>, none]

Formalization 28: SA00047 for receiving message activities

From the $2 * 3 * 2 * 2 = 24$ combinations, we can ignore $2 * 3 * 1 * 1 = 6$ cases which are violated by other rules. Moreover, $1 * 3 * 2 * 1 = 6$ cases are valid for an empty <message> without any <parts> and $1 * 3 * 1 * 1 + 1 * 3 * 1 * 1 = 6$ cases are valid for having a <message> with <part> and a variable or part assignment. $24 - 6 - 6 - 6 = 6$ cases are invalid, hence, six test cases.

[empty <message>, <message> with <part>(s)]
×
[<reply>]
×
[@variable, none]
×
[<toParts>, none]

Formalization 29: SA00047 for <reply>

From the $2 * 1 * 2 * 2 = 8$ combinations, we can ignore $2 * 1 * 1 * 1 = 2$ cases which are violated by other rules. Moreover, $1 * 1 * 2 * 1 = 2$ cases are valid for an empty <message> without any <parts> and $1 * 1 * 1 * 1 + 1 * 1 * 1 * 1 = 2$ cases are valid for having a <message> with <part> and a variable or part assignment. $8 - 2 - 2 - 2 = 2$ cases are invalid, hence, two test cases.

The valid combinations are either an empty <message> without any <fromParts> or <toParts>, or a <message> with <part> and a variable or part assignment. Having a part assignment implies the rule is violated if the <message> is empty, but the variable assignment has no effect then. If we have both variable and part assignment, one of the rules SA00051, SA00052, SA00055, SA00059, SA00063, or SA00085 is violated, but not SA00047. Thus, we require twelve tests in total, two for each of the five message activities and two additional tests for <invoke> as this can handle sending and receiving messages.

SA00048

"When the optional inputVariable and outputVariable attributes are being used in an <invoke> activity, the variables referenced by inputVariable and outputVariable MUST be messageType variables whose QName matches the QName of the input and output message type used in the operation, respectively, except as follows: if the WSDL operation used in an <invoke> activity uses a message containing exactly one part which itself is defined using an element, then a variable of the same element type as used to define the part MAY be referenced by the inputVariable and outputVariable attributes respectively."[3, p. 200]

The rules SA00048, SA00058, SA00087 and SA00090 are alike as they restrict different message activities with the same constraint: to have either @messageType or @element if there is exactly one <part> in a <message> as defined in the model of [2, p. 27] (on the same page SA00047 is stated with the same definition, but this is a typo and means SA00048). The following combinations violate SA00048, which refers to the message activity <invoke>.

<invoke> with [@inputVariable, @outputVariable] × [<message> with one <part>, <message> with two <part>s, <message> with no <part>s] × [<variable> with same @element, <variable> with different @element, <variable> with type @type, <variable> with same @messageType, <variable> with different @messageType]

Formalization 30: SA00048

From the $5 * 3 * 2 = 30$ combinations, $1 * 3 * 2 = 6$ cases are always valid when using the same `@messageType` as well as $1 * 1 * 2 = 2$ cases are always valid when having exactly one `<part>` in a `<message>` and using the same `@element`. What is more, we cannot use the same `@element` when referring to a message with no `<part>` or with two `<part>`s, as there is no same or correct element to choose from. Hence, we get $30 - 6 - 2 - 2 = 20$ test cases.

SA00050

"When <toParts> is present, it is required to have a <toPart> for every part in the WSDL message definition; the order in which parts are specified is irrelevant. Parts not explicitly represented by <toPart> elements would result in uninitialized parts in the target anonymous WSDL variable used by the <invoke> or <reply> activity. Such processes with missing <toPart> elements MUST be rejected during static analysis. "[3, p. 200]

The two sending activities `<invoke>` and `<reply>` violate SA00050 with at least one unmentioned `<part>` in `<toParts>`. We need two tests, one for each sending activity. [2, p. 28] define this rule along with SA00054 by bijectively mapping the parts of the message with the toParts. But for this rule, we are solely interested in the direction $Set_{\texttt{<part>}} \hookrightarrow Set_{\texttt{<toPart>}}$, as every `<part>` has to be used in the `<toPart>`s, but not vice versa.

Concerning message activities that leaves the process for each `<part>` of the message a `<toPart>` is required.

SA00051

"The inputVariable attribute MUST NOT be used on an invoke activity that contains <toPart> elements."[3, p. 200]

The rule SA00051 model by [2, p. 29] is defined along with the one of SA00059. In these two rules, it is forbidden to use both a `<toPart>` and a variable. As this rule refers to `<invoke>`, the test has an `@inputVariable` and a `<toPart>`.

SA00052

"The outputVariable attribute MUST NOT be used on an <invoke> activity that contains a <fromPart> element."[3, p. 200]

[2, p. 29] model the rules SA00052, SA00055, SA00063, and SA00085 together. All rules deal with the exclusive use of a variable or `<fromParts>`, similar to the definitions of `<toParts>` in rules SA00051 and SA00059. SA00052 has a BPEL process with `<toParts>` and an `@outputVariable` in an `<invoke>` as test.

SA00053

"For all <fromPart> elements the part attribute MUST reference a valid message part in the WSDL message for the operation."[3, p. 200]

The rule SA00053 deals with `<fromPart>` elements and is not as strict as for `<toPart>` elements, because only one implication is checked in contrast to both directions (SA00050 and SA00054). It is mentioned by [2, p. 27], but is excluded later on [2, p. 65]. There are four tests having at least one more `<fromPart>` in either `<receive>`, `<onMessage>`, `<invoke>` or `<onEvent>`, and thereby violating rule SA00053.

SA00054

"For all <toPart> elements the part attribute MUST reference a valid message part in the WSDL message for the operation. "[3, p. 200]

SA00054 is the converse direction of SA00050 $Set_{\texttt{<part>}} \hookleftarrow Set_{\texttt{<toPart>}}$, so an erroneous test has at least one more `<toPart>` than `<message>` `<part>` elements. Therefore, two tests are required one for `<reply>`, another for `<receive>`.

SA00055

"For <receive>, if <fromPart> elements are used on a <receive> activity then the variable attribute MUST NOT be used on the same activity."[3, p. 200]

A process containing a `<receive>` with a `@variable` and `<fromParts>` violates SA00055. This is similar to SA00052 which is defined accordingly by [2, p. 29].

SA00056

"A 'start activity' is a <receive> or <pick> activity that is annotated with a createInstance='yes' attribute. Activities other than the following: start activities, <scope>, <flow> and <sequence>

MUST NOT be performed prior to or simultaneously with start activities."[3, p. 200]

This rule is our of scope. The formalization must be considered experimental and incomplete.

A starting <receive> or <onMessage> (in an <pick>) in another structured element than the permitted ones of SA00056 violates this rule. The same applies to activities executed before or in parallel to the start activities which are not start activities by themselves. We cannot identify these activities from the rule model by [2, p. 50]. In the formalizations, we ignore multiple nestings, focusing on a single nesting relation only.

[<onMessage> in a <pick> with createInstance="yes", <receive> with createInstance="yes"]
×
in [<process>, <sequence>, <flow>, <scope>, <catch>, <catchAll>, <else>, <elseIf>, <forEach>, <if>, <onMessage> in a <pick> with createInstance="no", <onAlarm>, <onEvent>, <repeatUntil>, <terminationHandler>, <compensationHandler>, <while>]

Formalization 31: SA00056 with start activities within invalid encompassing activities

For the first formalization, we get 13 test cases, as four of the 17 activities (namely, <process>, <sequence>, <flow>, <scope>) are allowed, for each start activity. Hence, there are 26 test cases in total.

[<onMessage> in a <pick> with createInstance="yes", <receive> with createInstance="yes"]
×
activity before [<onMessage> in a <pick> with createInstance="yes", <receive> with createInstance="yes", none, <assign>, <exit>, <empty>, <invoke>, <receive>, <throw>, <reply>, <wait>, <else>, <elseIf>, <forEach>, <if>, <onMessage> in a <pick> with createInstance="no", <onAlarm> in a <pick>, <repeatUntil>, <while>]

Formalization 32: SA00056 with start activities in sequence

For the second formalization, we have 16 test cases for <sequence> for each start activity, totaling to 32 test cases, as valid start activities or no activities can happen before.

[<onMessage> in a <pick> with createInstance="yes", <receive> with createInstance="yes"]
×
activities in parallel [<onMessage> in a <pick> with createInstance="yes", <receive> with createInstance="yes", none, <assign>, <exit>, <empty>, <invoke>, <receive>, <throw>, <reply>, <wait>, <else>, <elseIf>, <forEach>, <if>, <onMessage> in a <pick> with createInstance="no", <onAlarm> in a <pick>, <repeatUntil>, <while>]
×
[link from start activity to other activity, link to start activity from other activity, none]

Formalization 33: SA00056 with start activities in flow

For the third formalization, we have 16 test cases for either no links or links to start activities for each start activity, totaling in 64 test cases.

SA00057

"If a process has multiple start activities with correlation sets then all such activities MUST share at least one common correlationSet and all common correlationSets defined on all the activities MUST have the value of the initiate attribute be set to 'join'."[3, p. 200]

In a process with multiple start activities, starting `<pick>` or `<receive>` activities with at least one `<correlation>` can violate SA00057 that is modeled by [2, p. 33]. In the negation of the model we may receive an empty intersection of `<correlation>` or another `initiate="join"`, as every start activity requires at least one shared `<correlation>` with `initiate="join"`.

starting activity [`<pick>`, `<receive>`]
×
with [no `<correlation>`, `<correlation>` with `initiate="join"`]
×
additional starting activity [`<pick>`, `<receive>`]
×
with [no `<correlation>`, `<correlation>` with `initiate="join"` different `<correlationSet>`, `<correlation>` with `initiate="yes"` different `<correlationSet>`, `<correlation>` with `initiate="join"` same `<correlationSet>`, `<correlation>` with `initiate="yes"` same `<correlationSet>`]

Formalization 34: SA00057

From the total $2 * 2 * 2 * 5 = 40$ combinations, we have only eight valid ones for two start activities with either no `<correlation>` at all or both using the same `<correlationSet>` with `initiate="join"`. Thus, we have $40 - 8 = 32$ test cases which violate this rule. But because some test cases can be seen as duplicated as the order of the activities is irrelevant, hence, we have twelve test cases only.

SA00058

"In a <receive> or <reply> activity, the variable referenced by the variable attribute MUST be a messageType variable whose QName matches the QName of the input (for <receive>) or output (for <reply>) message type used in the operation, except as follows: if the WSDL operation uses a message containing exactly one part which itself is defined using an element, then a WS-BPEL variable of the same element type as used to define the part MAY be referenced by the variable attribute of the <receive> or <reply>activity."[3, p. 201]

SA00058 resembles SA00048, but deals with `<receive>` and `<reply>`. Because they are modeled alike, see SA00048 for more information on the model of Kopp et al. [2].

[<receive>, <reply>]
×
[<message> with one <part>, <message> with two <part>s, <message> with no <part>s]
×
[<variable> with same @element, <variable> with different @element, <variable> with type @type, <variable> with same @messageType, <variable> with different @messageType]

Formalization 35: SA00058

From the $2 * 3 * 5 = 30$ combinations, $1 * 3 * 2 = 6$ cases are always valid when using the same @messageType as well as $1 * 1 * 2 = 2$ cases are always valid when having exactly one <part> in a <message> and using the same @element. What is more, we cannot use the same @element when referring to a message with no <part> or with two <part>s, as there is no same or correct element to choose from. Hence, we get $30 - 6 - 2 - 2 = 20$ test cases.

SA00059

"For <reply>, if <toPart> elements are used on a <reply> activity then the variable attribute MUST NOT be used on the same activity."[3, p. 201]

The test of SA00059 has a @variable for <reply> and a <toPart>. According to [2, p. 29] SA00059 is similar to SA00051, hence, only a single test case is required.

SA00060

"The explicit use of messageExchange is needed only where the execution can result in multiple IMA-<reply> pairs (e.g. <receive>-<reply> pair) on the same partnerLink and operation being executed simultaneously. In these cases, the process definition MUST explicitly mark the pairing-up relationship."[3, p. 201]

This rule is our of scope. The formalization must be considered experimental and incomplete.

The SA00060 model by [2, p. 31] constraints the possible @messageExchange equalities. The negation indicates a test set.

start1Mex = start2Mex, start1Mex != stop1Mex and start1Mex != stop2Mex; start1Mex != stop1Mex and start2Mex != stop1Mex; start2Mex != stop2Mex and start1Mex != stop2Mex; start2Mex != stop2Mex and start2Mex != stop1Mex; startMex1 = /; stopMex1 = /; startMex2 = /; stopMex2 = /;

Formalization 36: The negation of SA00060 from Kopp et al. which is not used

However, this would require a change of a single attributes which results in a violation of SA00061. To isolate the tests we have to use proper processes and remove one of the <messageExchange> and corresponding attributes in the IMA and the <reply>.

first IMA [<receive>, <pick>]
×
second IMA [<receive>, <pick>, <onEvent>]
×
marked message Exchange [none, both, first IMA, second IMA]

Formalization 37: SA00060

Therefore, we have 18 tests in total.

SA00061

"The name used in the optional messageExchange attribute MUST resolve to a messageExchange declared in a scope (where the process is considered the root scope) which encloses the <reply> activity and its corresponding IMA."[3, p. 201]

For all SA00061 tests the message exchange definition is not in a valid location. The rule model by [2, p. 30] is not precise as they only check the uniqueness of the @names of <messageExchange>s, but the negation of the model indicates the tests as its generally uses locations instead of identifying the required elements directly.

IMA is [<receive>, <pick>, <onEvent>]
×
location of <messageExchange> [no, <process>]
×
set @messageExchange for [IMA, <reply>, both, none]

Formalization 38: SA SA00061

We have $3 * 2 * 4 = 24$ combinations. When no @messageExchange is set, every case is valid ($3 * 2 * 1 = 6$). The same holds when the <messageExchange> is located in the <process> and both message activities have set @messageExchange ($3 * 1 * 1 = 1$). Hence, we require $24 - 6 - 3 = 15$ test cases.

IMA is [<receive>, <pick>, <onEvent>]
×
set @messageExchange for [IMA, <reply>, both, none]
×
is <messageExchange> reachable for [none, IMA, <reply>, both]

Formalization 39: SA SA00061 for <scope>

We have $3 * 4 * 4 = 48$ combinations. When no @messageExchange is set, every case is valid ($3 * 1 * 4 = 12$). The same holds when both message activities have set @messageExchange and both can reach to <messageExchange> ($3 * 1 * 1 = 3$). As the reachability can occur for message

activities with `@messageExchange` only ($3 * 4 * 1 = 12$), we require $48 - 12 - 3 - 12 = 21$ test cases for this part of the formalization.

SA00062

"If `<pick>` has a `createInstance` attribute with a value of yes, the events in the `<pick>` MUST all be `<onMessage>` events."[3, p. 201]

To violate SA00062 a sole test is sufficient that has an `<onAlarm>` in a starting `<pick>`. This constraint of SA00062 is also modeled by [2, p. 46].

SA00063

"The semantics of the `<onMessage>` event are identical to a `<receive>` activity regarding the optional nature of the `variable` attribute or `<fromPart>` elements, if `<fromPart>`elements on an activity then the `variable` attribute MUST NOT be used on the same activity (see SA00055)."[3, p. 201]

A process containing an `<onMessage>` with a `@variable` and `<fromParts>` violates SA00063. This is similar to SA00055 which is defined accordingly by [2, p. 29].

SA00064

"For `<flow>`, a declared link's `name` MUST be unique among all `<link>` names defined within the same immediately enclosing `<flow>`."[3, p. 201]

SA00064 is violated with a doubled `<link>` `@name` in the same `<flow>`. [2, p. 47] model this rule and the negation identifies the test, but the definition contains a typo in second declareLink: the l has to be a m.

SA00065

"The value of the `linkName` attribute of `<source>` or `<target>` MUST be the `name` of a `<link>` declared in an enclosing `<flow>` activity."[3, p. 201]

Negating the SA00065 model of [2, p. 50] we can identify two tests: a `<source>` that is not in the same `<flow>` as the `<link>`, and a `<target>` also not within the `<flow>`. Even the negation has always a `<link>` element, so in addition, we derive two tests if the `<link>` element is missing.

[<source>, <target>]
×
[<link> undefined, <link> out of <flow>, <link> in <flow>]

Formalization 40: SA SA00065

A process is valid if the <link> is in the <flow>.

SA00066

"Every link declared within a <flow> activity MUST have exactly one activity within the <flow> as its source and exactly one activity within the <flow> as its target."[3, p. 201]

If there is not exactly one <source> with a single <target> for a <link>, SA00066 is violated. In the rule model by [2, p. 48], the constraint of the <link> tuples is given by the definition af a <link> relation.

any activity in <flow> [<source>, <target>]
×
any other activity in <flow> [<source>, none]
×
yet another activity in <flow> [<target>, none]

Formalization 41: SA SA00066

We have $2 * 2 * 2 = 8$ combinations. It is valid to have a single activity as a <source> with another single element as a <target>, resulting in two valid cases. The remaining $8 - 2 = 6$ combinations are invalid, resulting in six test cases.

SA00067

"Two different links MUST NOT share the same source and target activities; that is, at most one link may be used to connect two activities."[3, p. 202]

A single test is required for SA00067. The negation of the rule model by [2, p. 48] points to have two <link> elements connecting the same activities.

SA00068

"An activity MAY declare itself to be the source of one or more links by including one or more **source** elements. Each **source** element MUST use a distinct link name."[3, p. 202]

Violating SA00068 can be tested by a @linkName duplicate in the <source> elements of an activity. [2, p. 48] provide a rule model that shows the test if it is negated.

SA00069

"An activity MAY declare itself to be the target of one or more links by including one or more `<target>` elements. Each `<target>` element associated with a given activity MUST use a link name distinct from all other `<target>` elements at that activity."[3, p. 202]

Similar to SA00068, SA00069 is violated by a `@linkName` duplicate in the `<target>` elements of an activity, and the model (by [2, p. 48]) negation provide a rule model that shows the test.

SA00070

"A link MUST NOT cross the boundary of a repeatable construct or the `<compensationHandler>` element. This means, a link used within a repeatable construct (`<while>`, `<repeatUntil>`, `<forEach>`, `<eventHandlers>`) or a `<compensationHandler>` MUST be declared in a `<flow>` that is itself nested inside the repeatable construct or `<compensationHandler>`."[3, p. 202]

SA00070 can be violated by crossing a) `<compensationHandler>`, b) repeatable constructs (`<while>`, `<repeatUntil>`, `<forEach>`) and c) `<eventHandlers>`. The rule model for (a) by [2, p. 55] can be negated to receive tests, a `<target>`/ `<source>` out of the handler. For (b), the rule model by [2, p. 51] can be negated as well and the tests are the same for each repeatable construct. There is no model for (c) and the `<link>` could be also outside of the construct.

[`<compensationHandler>`, `<while>`, `<repeatUntil>`, `<forEach>`, `<eventHandlers>`]
$\times$
[`<source>` outside boundary, `<target>` outside boundary, `<link>` outside boundary, all elements in the boundary]

Formalization 42: SA SA00070

We have $5 * 4 = 20$ combinations. If all elements are in the boundary, the process is valid, resulting in $5 * 1 = 5$ valid cases. Thus, we have $20 - 5 = 15$ tests in total.

SA00071

"A link that crosses a `<catch>`, `<catchAll>` or `<terminationHandler>` element boundary MUST be outbound only, that is, it MUST have its source activity within the `<faultHandlers>` or `<terminationHandler>`, and its target activity outside of the scope associated with the handler."[3, p. 202]

SA00071 resembles SA00070, but is more lax. For 1) `<catch>`, 2) `<catchAll>`, and 3) `<terminationHandler>`, incoming links violate the rule, resulting in three test cases. The rule model by [2, p. 55] can be negated to show these tests, yet, there is a typo a' has to be b.

SA00072

"A `<link>` declared in a `<flow>` MUST NOT create a control cycle, that is, the source activity must not have the target activity as a logically preceding activity."[3, p. 202]

To develop the tests for SA00072 we negate the rule model by [2, p. 50]. In the set of clan defined in [2, p. 50], we find the action itself and all descendants. Thus, a self linked activity and two cross linked activities are sufficient to test SA00072, totaling in two test cases.

SA00073

"The expression for a join condition MUST be constructed using only Boolean operators and the activity's incoming links' status values."[3, p. 202]

The rule deals with the parsing of XPath expressions and is postponed to future work.

SA00074

"The expressions in `<startCounterValue>` and `<finalCounterValue>` MUST return a TII (meaning they contain at least one character) that can be validated as a `xsd:unsignedInt`. Static analysis MAY be used to detect this erroneous situation at design time when possible (for example, when the expression is a constant)."[3, p. 202]

Because the rule deals with general expression parsing, it is postponed to future work.

SA00075

"For the `<forEach>` activity, `<branches>` is an integer value expression. Static analysis MAY be used to detect if the integer value is larger than the number of directly enclosed activities of `<forEach>` at design time when possible (for example, when the branches expression is a constant)."[3, p. 202]

Because the rule deals with general expression parsing, it is postponed to future work.

SA00076

"For `<forEach>` the enclosed scope MUST NOT declare a variable with the same name as specified in the `counterName` attribute of `<forEach>`."[3, p. 203]

The SA00076 model by [2, p. 52] can be negated to show the single test for the rule which is a process with `<variable>` defined with the `@name` of the `<forEach>` `@counterName`.

SA00077

"The value of the `target` attribute on a `<compensateScope>` activity MUST refer to the `name` of an immediately enclosed `scope` of the `scope` containing the FCT-handler with the `<compensateScope>` activity. This includes immediately enclosed scopes of an event handler (`<onEvent>` or `<onAlarm>`) associated with the same `scope`."[3, p. 203]

This rule is our of scope. The formalization must be considered experimental and incomplete.

[2, p. 59] model SA00077 and one test type can be directly derived. If the `<compensateScope>` targets a `<scope>` or `<invoke>` outside its own enclosing `<scope>` than SA00077 is violated. Other violations target `<scope>` elements inside other constructs. Therefore, they are not direct children of the enclosing `<scope>`.

targets [`<invoke>`, `<scope>`]
×
location of `<compensateScope>` [nested in `<catch>`, nested in `<catchAll>`, nested in `<compensationHandler>`, nested in `<terminationHandler>`,]
×
location of target [`<if>`, `<else>`, `<elseIf>`, `<flow>`, `<onAlarm>`, `<onEvent>`, `<onMessage>`, `<repeatUntil>`, `<sequence>`, `<while>`, `<catch>`, `<catchAll>`, `<compensationHandler>`, `<terminationHandler>`,
`<scope>` in enclosing `<scope>`, sibling of enclosing `<scope>`, enclosing `<scope>`]

Formalization 43: SA SA00077

If the target is within the enclosing `<scope>`, the process is valid. Thus, we have 128 $(2 * 4 * 16)$ tests for SA00077.

SA00078

"The `target` attribute of a `<compensateScope>` activity MUST refer to a `scope` or an `invoke` activity with a fault handler or compensation handler."[3, p. 203]

The SA00078 model by [2, p. 58] does not model the rule. It solely checks unique name of the children, but the SA00078 requires `<faultHandlers>` or `<compensationHandler>` in a targeted `<scope>` that is referenced by a `<compensateScope>`.

targets [`<invoke>`, `<scope>`]
×
elements of targets [none, `<compensationHandler>`, `<faultHandler>`, both]

Formalization 44: SA SA00078

Thus, we require two tests, one for `<invoke>` and one for `<scope>` with no handler at all.

SA00079

"The root `scope` inside a FCT-handler MUST not have a compensation handler."[3, p. 203]

To violate SA00079, the `<scope>` inside a 1) `<catch>`, 2) `<catchAll>`, 3) `<compensationHandler>`, or 4) `<terminationHandler>` has to carry a `<compensationHandler>`. The rule model by [2, p. 60] can be negated to show the structure of those tests, but it has a typo: the last } must be a |. The structure is a `<compensationHandler>` in one of the mentioned elements. In total, there are four test cases.

SA00080

"There MUST be at least one `<catch>` or `<catchAll>` element within a `<faultHandlers>` element."[3, p. 203]

[2, p. 20] mention SA00080 but do not provide a model. The tests are a `<process>` or `<scope>` with an empty `<faultHandlers>`, requiring two test cases in total.

SA00081

"For the `<catch>` construct; to have a defined type associated with the fault variable, the `faultVariable` attribute MUST only be used if either the `faultMessageType` or `faultElement` attributes, but not both, accompany it. The `faultMessageType` and `faultElement` attributes MUST NOT be used unless accompanied by `faultVariable` attribute."[3, p. 203]

SA00081 tests have a combination out of three attributes. The negation of the rule model by [2, p. 58] also shows the tests.

[`@faultVariable`, none]
×
[`@faultMessageType`, none]
×
[`@faultElement`, none]

Formalization 45: SA SA00081

Three combinations are valid, namely, no attribute selected, and `@faultVariable` with either `@faultMessageType` or `@faultElement`, resulting in five test cases in total.

SA00082

"The peer-scope dependency relation MUST NOT include cycles. In other words, WS-BPEL forbids a process in which there are peer scopes S1 and S2 such that S1 has a peer-scope

dependency on S2 and S2 has a peer-scope dependency on S1."[3, p. 203]

The negation of the SA00082 model by [2, p. 51] models the single test. Two `<scope>` elements with activities "linking" them in a cycle.

SA00083

"An event handler MUST contain at least one `<onEvent>` or `<onAlarm>` element."[3, p. 203]

The two test cases for SA00083 consist of either a `<process>` or `<scope>` with an empty `<eventHandlers>`. The SA00083 is not modeled by [2, p. 56] as the model does not contain the wrapper element.

SA00084

"The `partnerLink` reference of `<onEvent>` MUST resolve to a partner link declared in the process in the following order: the associated scope first and then the ancestor scopes."[3, p. 203]

This rule is explicitly excluded from the model of [2, p. 65] because it is about runtime behavior. The only way to test such rules during static analysis is violating other rules, in particular, by defining a wrong `<partnerLink>` inside with the same `@name` as the correct `<partnerLink>` outside the `<onEvent>`. Hence, we have one test case.

SA00085

"The syntax and semantics of the `<fromPart>` elements as used on the `<onEvent>` element are the same as specified for the `receive` activity. This includes the restriction that if `<fromPart>` elements are used on an `onEvent` element then the `variable`, `element` and `messageType` attributes MUST NOT be used on the same element."[3, p. 203]

A process containing an `<onEvent>` with a `@variable` and `<fromParts>` violates SA00085. This is similar to SA00063 which is defined accordingly by [2, p. 29], but differs as it considers the `<onEvent>` specific `@element` and `@messageType` as well.

$$[\texttt{@variable}, \text{none}] \times [\texttt{@element}, \text{none}] \times [\texttt{@messageType}, \text{none}]$$

Formalization 46: SA SA00085

Out of the $2 * 2 * 2 = 8$ total combinations, only the absence of all three attributes is the valid case, resulting in seven test cases. As `@variable` can only exist with either `@element` or

@messageType according to SA00090, we can ignore these two test cases as they are already covered, totaling in five test cases.

SA00086

"For <onEvent>, variables referenced by the variable attribute of <fromPart> elements or the variable attribute of an <onEvent> element are implicitly declared in the associated scope of the event handler. Variables of the same names MUST NOT be explicitly declared in the associated scope."[3, p. 204]

SA00086 has two tests that are also shown in the negation of the rule model by [2, p. 55]. Both have a <variable> definition and an additional @name occurrence in the same <onEvent> first in the @variable of <onEvent> or second as @toVariable of a <fromPart>.

SA00087

"For <onEvent>, the type of the variable (as specified by the messageType attribute) MUST be the same as the type of the input message defined by operation referenced by the operation attribute. Optionally the messageType attribute may be omitted and instead the element attribute substituted if the message to be received has a single part and that part is defined with an element type. That element type MUST be an exact match of the element type referenced by the element attribute."[3, p. 204]

SA00087 resembles SA00048 for <onEvent>. See SA00048 for more information on the model of Kopp et al. [2].

[<message> with one <part>, <message> with two <part>s, <message> with no <part>s]
×
[@variable with same @element, @variable with different @element, @variable with same @messageType, @variable with different @messageType]

Formalization 47: SA SA00087

From the $3 * 4 = 12$ combinations, $3 * 1 = 3$ cases are always valid when using the same @messageType as well as $1 * 1 = 1$ case is always valid when having exactly one <part> in a <message> and using the same @element. Hence, we get $12 - 3 - 1 = 8$ test cases.

SA00088

"For <onEvent>, the resolution order of the correlation set(s) referenced by <correlation> MUST be first the associated scope and then the ancestor scopes."[3, p. 204]

Because SA00088 describes engine behavior, it is not part of the model of [2, p. 65]. The sole test is a `<correlation>` of the wrong type inside of an `<onEvent>` and a correct `<correlation>` outside.

SA00089

"For `<onEvent>`, when the `messageExchange` attribute is explicitly specified, the resolution order of the message exchange referenced by `messageExchange` attribute MUST be first the associated scope and then the ancestor scopes."[3, p. 204]

Because SA00089 relates to an element that just carries a string, the rule is totally covered by SA00061. The difference is in positive cases where the resolution detects the `<messageExchange>` inside as well, but this cannot be modeled in a negative scenario. Thus, there is a single test case without any `<messageExchange>` that can be resolved from `<onEvent>`. [2, p. 65] exclude this rule from their model, as it is execution dependent.

SA00090

"If the `variable` attribute is used in the `<onEvent>` element, either the `messageType` or the `element` attribute MUST be provided in the `<onEvent>` element."[3, p. 204]

The BPEL standard does not state in the related section of SA00090 if the element variable can be applied on single `<part>` `<messages>` exclusively. Yet, Kopp et al. [2] used the same restriction as modeled for SA00048. The two tests derived for SA00090, as defined in the standard, have all two additional attributes with `@variable` or just the sole attribute.

SA00091

"A `scope` with the `isolated` attribute set to `"yes"` is called an isolated scope. Isolated scopes MUST NOT contain other isolated scopes."[3, p. 204]

The SA00091 has a single test and is modeled by [2, p. 57] such that the negation of the rule model shows the test. But the model has a typo: Before the $\Rightarrow$ the ")" is false.

SA00092

"Within a scope, the name of all named immediately enclosed scopes MUST be unique."[3, p. 204]

The negation of the SA00092 model by [2, p. 57] shows the two test cases that have two `<scope>` elements with equal `@name` directly enclosed in the same `<scope>` or `<process>`.

SA00093

"Identical `<catch>` constructs MUST NOT exist within a `<faultHandlers>` element."[3, p. 204]

Each test of SA00093 has a `<catch>` duplicate in the same `<faultHandlers>` element. This can be seen in the negation of the rule model by [2, p. 59]. However, a `<catch>` can be distinguished by three attributes, thus we need one test per combination.

[`<scope>`, `<process>`]
×
[`@faultElement`, none]
×
[`@faultMessageType`, none]
×
[`@faultName`, none]

Formalization 48: SA SA00093

We have $2 * 2 * 2 * 2 = 16$ combinations. There shall always be at least one attribute in a `<catch>` ($2*1*1*1 = 2$). In addition the combination of `@faultMessageType` and the attribute `@faultElements` violate SA00081 ($2 * 1 * 1 * 2 = 4$), so we require $16 - 2 - 4 = 10$ tests in total.

SA00094

"For `<copy>`, when the `keepSrcElementName` attribute is set to `"yes"` and the destination element is the Document EII of an element-based variable or an element-based part of a WSDL message-type-based variable, the name of the source element MUST belong to the substitutionGroup of the destination element. This checking MAY be enforced through static analysis of the expression/query language."[3, p. 204]

Because the rule deals with general expression parsing, it is postponed to future work.

SA00095

"For `<onEvent>`, the variable references are resolved to the associated scope only and MUST NOT be resolved to the ancestor scopes."[3, p. 205]

SA00095 is mentioned by [2, p. 26] but is not modeled properly. It directs to another section [2, p. 22], but the definition is more like a part of SA00085. As SA00095 describes runtime behavior, a single test is required that can resolve a `<variable>` usage to the ancestor `<scope>`, only.

4 Tests

The tests are based on the formalizations in section 3. Due to their large number, we did not list them here but present links to the information. The mapping of which static analysis test is based on what feature test is given in the csv file hosted at `https://github.com/uniba-dsg/soca2014/tree/master/mapping-satest2featuretest.csv` whereas the differences between the static analysis test and its corresponding feature test can be found in the folder hosted at `https://github.com/uniba-dsg/soca2014/tree/master/differences` by using the naming conventions. The difference is the description of the static analysis test case, as it marks the minimal change to make a valid test fail.

References

[1] S. Harrer, C. Preißinger, and G. Wirtz. BPEL Conformance in Open Source Engines: The Case of Static Analysis. In *Proceedings of the 7th IEEE International Conference on Service-Oriented Computing and Applications (SOCA'14)*, pages 33–40, Matsue, Japan, Nov. 2014. IEEE.

[2] O. Kopp, R. Mietzner, and F. Leymann. Abstract Syntax of WS-BPEL 2.0. Technical Report 6, Universität Stuttgart, Oct. 2008.

[3] OASIS. *Web Services Business Process Execution Language*, Apr. 2007. v2.0.

[4] W3C. *Web Services Description Language (WSDL) 1.1*, Mar. 2001. v1.1.

5 List of previous University of Bamberg reports

Bamberger Beiträge zur Wirtschaftsinformatik

Nr. 1 (1989) Augsburger W., Bartmann D., Sinz E.J.: Das Bamberger Modell: Der Diplom-Studiengang Wirtschaftsinformatik an der Universität Bamberg (Nachdruck Dez. 1990)

Nr. 2 (1990) Esswein W.: Definition, Implementierung und Einsatz einer kompatiblen Datenbankschnittstelle für PROLOG

Nr. 3 (1990) Augsburger W., Rieder H., Schwab J.: Endbenutzerorientierte Informationsgewinnung aus numerischen Daten am Beispiel von Unternehmenskennzahlen

Nr. 4 (1990) Ferstl O.K., Sinz E.J.: Objektmodellierung betrieblicher Informationsmodelle im Semantischen Objektmodell (SOM) (Nachdruck Nov. 1990)

Nr. 5 (1990) Ferstl O.K., Sinz E.J.: Ein Vorgehensmodell zur Objektmodellierung betrieblicher Informationssysteme im Semantischen Objektmodell (SOM)

Nr. 6 (1991) Augsburger W., Rieder H., Schwab J.: Systemtheoretische Repräsentation von Strukturen und Bewertungsfunktionen über zeitabhängigen betrieblichen numerischen Daten

Nr. 7 (1991) Augsburger W., Rieder H., Schwab J.: Wissensbasiertes, inhaltsorientiertes Retrieval statistischer Daten mit EISREVU / Ein Verarbeitungsmodell für eine modulare Bewertung von Kennzahlenwerten für den Endanwender

Nr. 8 (1991) Schwab J.: Ein computergestütztes Modellierungssystem zur Kennzahlenbewertung

Nr. 9 (1992) Gross H.-P.: Eine semantiktreue Transformation vom Entity-Relationship-Modell in das Strukturierte Entity-Relationship-Modell

Nr. 10 (1992) Sinz E.J.: Datenmodellierung im Strukturierten Entity-Relationship-Modell (SERM)

Nr. 11 (1992) Ferstl O.K., Sinz E. J.: Glossar zum Begriffsystem des Semantischen Objektmodells

Nr. 12 (1992) Sinz E. J., Popp K.M.: Zur Ableitung der Grobstruktur des konzeptuellen Schemas aus dem Modell der betrieblichen Diskurswelt

Nr. 13 (1992) Esswein W., Locarek H.: Objektorientierte Programmierung mit dem Objekt-Rollenmodell

Nr. 14 (1992) Esswein W.: Das Rollenmodell der Organsiation: Die Berücksichtigung aufbauorganisatorische Regelungen in Unternehmensmodellen

Nr. 15 (1992) Schwab H. J.: EISREVU-Modellierungssystem. Benutzerhandbuch

Nr. 16 (1992) Schwab K.: Die Implementierung eines relationalen DBMS nach dem Client/Server-Prinzip

Nr. 17 (1993) Schwab K.: Konzeption, Entwicklung und Implementierung eines computergestützten Bürovorgangssystems zur Modellierung von Vorgangsklassen und Abwicklung und Überwachung von Vorgängen. Dissertation

Nr. 18 (1993) Ferstl O.K., Sinz E.J.: Der Modellierungsansatz des Semantischen Objektmodells

Nr. 19 (1994) Ferstl O.K., Sinz E.J., Amberg M., Hagemann U., Malischewski C.: Tool-Based Business Process Modeling Using the SOM Approach

Nr. 20 (1994) Ferstl O.K., Sinz E.J.: From Business Process Modeling to the Specification of Distributed Business Application Systems - An Object-Oriented Approach -. 1st edition, June 1994

Ferstl O.K., Sinz E.J. : Multi-Layered Development of Business Process Models and Distributed Business Application Systems - An Object-Oriented Approach -. 2nd edition, November 1994

Nr. 21 (1994) Ferstl O.K., Sinz E.J.: Der Ansatz des Semantischen Objektmodells zur Modellierung von Geschäftsprozessen

Nr. 22 (1994) Augsburger W., Schwab K.: Using Formalism and Semi-Formal Constructs for Modeling Information Systems

Nr. 23 (1994) Ferstl O.K., Hagemann U.: Simulation hierarischer objekt- und transaktionsorientierter Modelle

Nr. 24 (1994) Sinz E.J.: Das Informationssystem der Universität als Instrument zur zielgerichteten Lenkung von Universitätsprozessen

Nr. 25 (1994) Wittke M., Mekinic, G.: Kooperierende Informationsräume. Ein Ansatz für verteilte Führungsinformationssysteme

Nr. 26 (1995) Ferstl O.K., Sinz E.J.: Re-Engineering von Geschäftsprozessen auf der Grundlage des SOM-Ansatzes

Nr. 27 (1995) Ferstl, O.K., Mannmeusel, Th.: Dezentrale Produktionslenkung. Erscheint in CIM-Management 3/1995

Nr. 28 (1995) Ludwig, H., Schwab, K.: Integrating cooperation systems: an event-based approach

Nr. 30 (1995) Augsburger W., Ludwig H., Schwab K.: Koordinationsmethoden und -werkzeuge bei der computergestützten kooperativen Arbeit

Nr. 31 (1995) Ferstl O.K., Mannmeusel T.: Gestaltung industrieller Geschäftsprozesse

Nr. 32 (1995) Gunzenhäuser R., Duske A., Ferstl O.K., Ludwig H., Mekinic G., Rieder H., Schwab H.-J., Schwab K., Sinz E.J., Wittke M: Festschrift zum 60. Geburtstag von Walter Augsburger

Nr. 33 (1995) Sinz, E.J.: Kann das Geschäftsprozeßmodell der Unternehmung das unternehmensweite Datenschema ablösen?

Nr. 34 (1995) Sinz E.J.: Ansätze zur fachlichen Modellierung betrieblicher Informationssysteme - Entwicklung, aktueller Stand und Trends -

Nr. 35 (1995) Sinz E.J.: Serviceorientierung der Hochschulverwaltung und ihre Unterstützung durch workflow-orientierte Anwendungssysteme

Nr. 36 (1996) Ferstl O.K., Sinz, E.J., Amberg M.: Stichwörter zum Fachgebiet Wirtschaftsinformatik. Erscheint in: Broy M., Spaniol O. (Hrsg.): Lexikon Informatik und Kommunikationstechnik, 2. Auflage, VDI-Verlag, Düsseldorf 1996

Nr. 37 (1996) Ferstl O.K., Sinz E.J.: Flexible Organizations Through Object-oriented and Transaction-oriented Information Systems, July 1996

Nr. 38 (1996) Ferstl O.K., Schäfer R.: Eine Lernumgebung für die betriebliche Aus- und Weiterbildung on demand, Juli 1996

Nr. 39 (1996) Hazebrouck J.-P.: Einsatzpotentiale von Fuzzy-Logic im Strategischen Management dargestellt an Fuzzy-System-Konzepten für Portfolio-Ansätze

Nr. 40 (1997) Sinz E.J.: Architektur betrieblicher Informationssysteme. In: Rechenberg P., Pomberger G. (Hrsg.): Handbuch der Informatik, Hanser-Verlag, München 1997

Nr. 41 (1997) Sinz E.J.: Analyse und Gestaltung universitärer Geschäftsprozesse und Anwendungssysteme. Angenommen für: Informatik '97. Informatik als Innovationsmotor. 27. Jahrestagung der Gesellschaft für Informatik, Aachen 24.-26.9.1997

Nr. 42 (1997) Ferstl O.K., Sinz E.J., Hammel C., Schlitt M., Wolf S.: Application Objects – fachliche Bausteine für die Entwicklung komponentenbasierter Anwendungssysteme. Angenommen für: HMD – Theorie und Praxis der Wirtschaftsinformatik. Schwerpunkheft ComponentWare, 1997

Nr. 43 (1997): Ferstl O.K., Sinz E.J.: Modeling of Business Systems Using the Semantic Object Model (SOM) – A Methodological Framework - . Accepted for: P. Bernus, K. Mertins, and G. Schmidt (ed.): Handbook on Architectures of Information Systems. International Handbook on Information Systems, edited by Bernus P., Blazewicz J., Schmidt G., and Shaw M., Volume I, Springer 1997

Ferstl O.K., Sinz E.J.: Modeling of Business Systems Using (SOM), 2nd Edition. Appears in: P. Bernus, K. Mertins, and G. Schmidt (ed.): Handbook on Architectures of Information Systems. International Handbook on Information Systems, edited by Bernus P., Blazewicz J., Schmidt G., and Shaw M., Volume I, Springer 1998

Nr. 44 (1997) Ferstl O.K., Schmitz K.: Zur Nutzung von Hypertextkonzepten in Lernumgebungen. In: Conradi H., Kreutz R., Spitzer K. (Hrsg.): CBT in der Medizin – Methoden, Techniken, Anwendungen -. Proceedings zum Workshop in Aachen 6. – 7. Juni 1997. 1. Auflage Aachen: Verlag der Augustinus Buchhandlung

Nr. 45 (1998) Ferstl O.K.: Datenkommunikation. In. Schulte Ch. (Hrsg.): Lexikon der Logistik, Oldenbourg-Verlag, München 1998

Nr. 46 (1998) Sinz E.J.: Prozeßgestaltung und Prozeßunterstützung im Prüfungswesen. Erschienen in: Proceedings Workshop „Informationssysteme für das Hochschulmanagement". Aachen, September 1997

Nr. 47 (1998) Sinz, E.J.:, Wismans B.: Das „Elektronische Prüfungsamt". Erscheint in: Wirtschaftswissenschaftliches Studium WiSt, 1998

Nr. 48 (1998) Haase, O., Henrich, A.: A Hybrid Respresentation of Vague Collections for Distributed Object Management Systems. Erscheint in: IEEE Transactions on Knowledge and Data Engineering

Nr. 49 (1998) Henrich, A.: Applying Document Retrieval Techniques in Software Engineering Environments. In: Proc. International Conference on Database and Expert Systems

Applications. (DEXA 98), Vienna, Austria, Aug. 98, pp. 240-249, Springer, Lecture Notes in Computer Sciences, No. 1460

Nr. 50 (1999) Henrich, A., Jamin, S.: On the Optimization of Queries containing Regular Path Expressions. Erscheint in: Proceedings of the Fourth Workshop on Next Generation Information Technologies and Systems (NGITS'99), Zikhron-Yaakov, Israel, July, 1999 (Springer, Lecture Notes)

Nr. 51 (1999) Haase O., Henrich, A.: A Closed Approach to Vague Collections in Partly Inaccessible Distributed Databases. Erscheint in: Proceedings of the Third East-European Conference on Advances in Databases and Information Systems – ADBIS'99, Maribor, Slovenia, September 1999 (Springer, Lecture Notes in Computer Science)

Nr. 52 (1999) Sinz E.J., Böhnlein M., Ulbrich-vom Ende A.: Konzeption eines Data Warehouse-Systems für Hochschulen. Angenommen für: Workshop „Unternehmen Hochschule" im Rahmen der 29. Jahrestagung der Gesellschaft für Informatik, Paderborn, 6. Oktober 1999

Nr. 53 (1999) Sinz E.J.: Konstruktion von Informationssystemen. Der Beitrag wurde in geringfügig modifizierter Fassung angenommen für: Rechenberg P., Pomberger G. (Hrsg.): Informatik-Handbuch. 2., aktualisierte und erweiterte Auflage, Hanser, München 1999

Nr. 54 (1999) Herda N., Janson A., Reif M., Schindler T., Augsburger W.: Entwicklung des Intranets SPICE: Erfahrungsbericht einer Praxiskooperation.

Nr. 55 (2000) Böhnlein M., Ulbrich-vom Ende A.: Grundlagen des Data Warehousing. Modellierung und Architektur

Nr. 56 (2000) Freitag B, Sinz E.J., Wismans B.: Die informationstechnische Infrastruktur der Virtuellen Hochschule Bayern (vhb). Angenommen für Workshop "Unternehmen Hochschule 2000" im Rahmen der Jahrestagung der Gesellschaft f. Informatik, Berlin 19. - 22. September 2000

Nr. 57 (2000) Böhnlein M., Ulbrich-vom Ende A.: Developing Data Warehouse Structures from Business Process Models.

Nr. 58 (2000) Knobloch B.: Der Data-Mining-Ansatz zur Analyse betriebswirtschaftlicher Daten.

Nr. 59 (2001) Sinz E.J., Böhnlein M., Plaha M., Ulbrich-vom Ende A.: Architekturkonzept eines verteilten Data-Warehouse-Systems für das Hochschulwesen. Angenommen für: WI-IF 2001, Augsburg, 19.-21. September 2001

Nr. 60 (2001) Sinz E.J., Wismans B.: Anforderungen an die IV-Infrastruktur von Hochschulen. Angenommen für: Workshop „Unternehmen Hochschule 2001" im Rahmen der Jahrestagung der Gesellschaft für Informatik, Wien 25. – 28. September 2001

Änderung des Titels der Schriftenreihe *Bamberger Beiträge zur Wirtschaftsinformatik* in *Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik* ab Nr. 61

Note: The title of our technical report series has been changed from *Bamberger Beiträge zur Wirtschaftsinformatik* to *Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik* starting with TR No. 61

Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik

Nr. 61 (2002) Goré R., Mendler M., de Paiva V. (Hrsg.): Proceedings of the International Workshop on Intuitionistic Modal Logic and Applications (IMLA 2002), Copenhagen, July 2002.

Nr. 62 (2002) Sinz E.J., Plaha M., Ulbrich-vom Ende A.: Datenschutz und Datensicherheit in einem landesweiten Data-Warehouse-System für das Hochschulwesen. Erscheint in: Beiträge zur Hochschulforschung, Heft 4-2002, Bayerisches Staatsinstitut für Hochschulforschung und Hochschulplanung, München 2002

Nr. 63 (2005) Aguado, J., Mendler, M.: Constructive Semantics for Instantaneous Reactions

Nr. 64 (2005) Ferstl, O.K.: Lebenslanges Lernen und virtuelle Lehre: globale und lokale Verbesserungspotenziale. Erschienen in: Kerres, Michael; Keil-Slawik, Reinhard (Hrsg.); Hochschulen im digitalen Zeitalter: Innovationspotenziale und Strukturwandel, S. 247 – 263; Reihe education quality forum, herausgegeben durch das Centrum für eCompetence in Hochschulen NRW, Band 2, Münster/New York/München/Berlin: Waxmann 2005

Nr. 65 (2006) Schönberger, Andreas: Modelling and Validating Business Collaborations: A Case Study on RosettaNet

Nr. 66 (2006) Markus Dorsch, Martin Grote, Knut Hildebrandt, Maximilian Röglinger, Matthias Sehr, Christian Wilms, Karsten Loesing, and Guido Wirtz: Concealing Presence Information in Instant Messaging Systems, April 2006

Nr. 67 (2006) Marco Fischer, Andreas Grünert, Sebastian Hudert, Stefan König, Kira Lenskaya, Gregor Scheithauer, Sven Kaffille, and Guido Wirtz: Decentralized Reputation Management for Cooperating Software Agents in Open Multi-Agent Systems, April 2006

Nr. 68 (2006) Michael Mendler, Thomas R. Shiple, Gérard Berry: Constructive Circuits and the Exactness of Ternary Simulation

Nr. 69 (2007) Sebastian Hudert: A Proposal for a Web Services Agreement Negotiation Protocol Framework . February 2007

Nr. 70 (2007) Thomas Meins: Integration eines allgemeinen Service-Centers für PC-und Medientechnik an der Universität Bamberg – Analyse und Realisierungs-Szenarien. February 2007 (out of print)

Nr. 71 (2007) Andreas Grünert: Life-cycle assistance capabilities of cooperating Software Agents for Virtual Enterprises. März 2007

Nr. 72 (2007) Michael Mendler, Gerald Lüttgen: Is Observational Congruence on μ-Expressions Axiomatisable in Equational Horn Logic?

Nr. 73 (2007) Martin Schissler: out of print

Nr. 74 (2007) Sven Kaffille, Karsten Loesing: Open chord version 1.0.4 User's Manual. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 74, Bamberg University, October 2007. ISSN 0937-3349.

Nr. 75 (2008) Karsten Loesing (Hrsg.): Extended Abstracts of the Second *Privacy Enhancing Technologies Convention* (PET-CON 2008.1). Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 75, Bamberg University, April 2008. ISSN 0937-3349.

Nr. 76 (2008) Gregor Scheithauer, Guido Wirtz: Applying Business Process Management Systems – A Case Study. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 76, Bamberg University, May 2008. ISSN 0937-3349.

Nr. 77 (2008) Michael Mendler, Stephan Scheele: Towards Constructive Description Logics for Abstraction and Refinement. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 77, Bamberg University, September 2008. ISSN 0937-3349.

Nr. 78 (2008) Gregor Scheithauer, Matthias Winkler: A Service Description Framework for Service Ecosystems. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 78, Bamberg University, October 2008. ISSN 0937-3349.

Nr. 79 (2008) Christian Wilms: Improving the Tor Hidden Service Protocol Aiming at Better Performances. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 79, Bamberg University, November 2008. ISSN 0937-3349.

Nr. 80 (2009) Thomas Benker, Stefan Fritzemeier, Matthias Geiger, Simon Harrer, Tristan Kessner, Johannes Schwalb, Andreas Schönberger, Guido Wirtz: QoS Enabled B2B Integration. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 80, Bamberg University, May 2009. ISSN 0937-3349.

Nr. 81 (2009) Ute Schmid, Emanuel Kitzelmann, Rinus Plasmeijer (Eds.): Proceedings of the ACM SIGPLAN Workshop on *Approaches and Applications of Inductive Programming* (AAIP'09), affiliated with ICFP 2009, Edinburgh, Scotland, September 2009. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 81, Bamberg University, September 2009. ISSN 0937-3349.

Nr. 82 (2009) Ute Schmid, Marco Ragni, Markus Knauff (Eds.): Proceedings of the KI 2009 Workshop *Complex Cognition*, Paderborn, Germany, September 15, 2009. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 82, Bamberg University, October 2009. ISSN 0937-3349.

Nr. 83 (2009) Andreas Schönberger, Christian Wilms and Guido Wirtz: A Requirements Analysis of Business-to-Business Integration. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 83, Bamberg University, December 2009. ISSN 0937-3349.

Nr. 84 (2010) Werner Zirkel, Guido Wirtz: A Process for Identifying Predictive Correlation Patterns in Service Management Systems. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 84, Bamberg University, February 2010. ISSN 0937-3349.

Nr. 85 (2010) Jan Tobias Mühlberg und Gerald Lüttgen: Symbolic Object Code Analysis. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 85, Bamberg University, February 2010. ISSN 0937-3349.

Nr. 86 (2010) Werner Zirkel, Guido Wirtz: Proaktives Problem Management durch Eventkorrelation – ein *Best Practice* Ansatz. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 86, Bamberg University, August 2010. ISSN 0937-3349.

Nr. 87 (2010) Johannes Schwalb, Andreas Schönberger: Analyzing the Interoperability of WS-Security and WS-ReliableMessaging Implementations. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 87, Bamberg University, September 2010. ISSN 0937-3349.

Nr. 88 (2011) Jörg Lenhard: A Pattern-based Analysis of WS-BPEL and Windows Workflow. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 88, Bamberg University, March 2011. ISSN 0937-3349.

Nr. 89 (2011) Andreas Henrich, Christoph Schlieder, Ute Schmid [eds.]: Visibility in Information Spaces and in Geographic Environments – Post-Proceedings of the KI'11 Workshop. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 89, Bamberg University, December 2011. ISSN 0937-3349.

Nr. 90 (2012) Simon Harrer, Jörg Lenhard: Betsy - A BPEL Engine Test System. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 90, Bamberg University, July 2012. ISSN 0937-3349.

Nr. 91 (2013) Michael Mendler, Stephan Scheele: On the Computational Interpretation of CKn for Contextual Information Processing - Ancillary Material. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 91, Bamberg University, May 2013. ISSN 0937-3349.

Nr. 92 (2013) Matthias Geiger: BPMN 2.0 Process Model Serialization Constraints. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 92, Bamberg University, May 2013. ISSN 0937-3349.

Nr. 93 (2014) Cedric Röck, Simon Harrer: Literature Survey of Performance Benchmarking Approaches of BPEL Engines. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 93, Bamberg University, May 2014. ISSN 0937-3349.

Nr. 94 (2014) Joaquin Aguado, Michael Mendler, Reinhard von Hanxleden, Insa Fuhrmann: Grounding Synchronous Deterministic Concurrency in Sequential Programming. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 94, Bamberg University, August 2014. ISSN 0937-3349.

Nr. 95 (2014) Michael Mendler, Bruno Bodin, Partha S Roop, Jia Jie Wang: WCRT for Synchronous Programs: Studying the Tick Alignment Problem. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 95, Bamberg University, August 2014. ISSN 0937-3349.

Nr. 96 (2015) Joaquin Aguado, Michael Mendler, Reinhard von Hanxleden, Insa Fuhrmann: Denotational Fixed-Point Semantics for Constructive Scheduling of Synchronous Concurrency. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 96, Bamberg University, April 2015. ISSN 0937-3349.

Nr. 97 (2015) Thomas Benker: Konzeption einer Komponentenarchitektur für prozessorientierte OLTP- & OLAP-Anwendungssysteme. Bamberger Beiträge zur Wirtschafts-informatik und Angewandten Informatik Nr. 97, Bamberg University, Oktober 2015. ISSN 0937-3349.

Nr. 98 (2016) Sascha Fendrich, Gerald Lüttgen: A Generalised Theory of Interface Automata, Component Compatibility and Error. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 98, Bamberg University, March 2016. ISSN 0937-3349.

Nr. 99 (2014) Christian Preißinger, Simon Harrer: Static Analysis Rules of the BPEL Specification: Tagging, Formalization and Tests. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 99, Bamberg University, August 2014. ISSN 0937-3349.