Towards a Specification-based Quality Guarantee for Geo Raster Web Services

By **Jinsongdi Yu**

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in Computer Science

Approved, Thesis committee:
Prof. Dr. Peter Baumann, Chair
Jacobs University, Germany

Dr. Heinrich Stamerjohanns
Jacobs University, Germany

Prof. Dr. Peijun Du
Nanjing University, China

Date of defense: January 11, 2012

School of Engineering and Science

I hereby declare that this thesis has been written independently except where sources and collaborations are acknowledged and has not been submitted at another university for the conferral of a degree.

Abstract

Geo raster web services provide access to detailed and rich sets of geospatial information used in multidisciplinary earth system science research, such as solar, atmosphere, ocean, cryosphere, solid earth, and biosphere research. However, the heterogeneity of services deployed in these disciplines tends to weaken the interoperability of today's highly multidisciplinary earth system science research. Several syntactical and semantic approaches have been investigated, such as similarity assessment, inconsistency resolution and standardization, to overcome this heterogeneity obstacle. Among them, standardization is a promising approach, which has the potential to achieve interoperability by combing methods of best practices and making them generally accepted within communities at large. Families of geo service standards have been developed and maintained by international organizations, such as the Open Geospatial Consortium (OGC) and the International Standardization Organization (ISO). To ensure the interoperability of implementations which claim to adhere to those standards, conformance testing programs have been set up by some of the standardizing bodies. For instance, the OGC has set up a conformance testing program with open test for each OGC standard. This constitutes a more comprehensive conformance testing in the field of geo web services than human inspection and closed source approaches. The approach follows a specification-based black box approach. This approach organizes specification requirements as assertions and groups them by functional modules. However, there is no methodology to coherently classify these assertions; neither the maturity of derived tests nor their completeness can be proven based on this approach.

The goal of this thesis is to establish a model to evaluate an existing standardization approach and to evaluate the extent to which the specification itself supports the evaluation of confidence in service implementations, tests and test results. We propose a Request Parameter Relationship Analysis approach to master service testing complexity and maturity. Deductive reasoning is used to address reference output adoption according to the corresponding conditional statements and service facts. By

addressing the relations among individual conformance statements, we propose the integrated dependency and goal model. We present a sequential evaluation schedule and prove the stable status in an isolated testing environment. These approaches provide a basis for analyzing service compliance, test maturity, and the validity of global statements. As our study example, we use the OGC WCS standard series as it offers a suitably formalized model for conformance testing. These approaches are applied to a real-life application, concretely testing the OGC WCS 2.0 geo service standard. The outcomes of this thesis contribute to the development of specifications, services and test suites for several standards whose normative conformance tests are specified using our approach. As a result, improved interoperation of geo raster web services under evolving implementations is expected in the future.

Acknowledgments

I would like to express my gratitude to my Ph.D. supervisor, Dr. Peter Baumann, whose expertise, understanding, and patience, added considerable support to my research work. His guidance helped me during all the time of research and writing of this thesis. I would like to thank the other members of my committee, Dr. Peijun Du and Dr. Heinrich Stamerjohanns, for their encouragement and insightful comments. My sincere thanks also goes to Dr. Klaus Grosfeld from Alfred-Wegener Institute (AWI) for his kind guidance during the soft-skill training activities. A very special thanks goes to Dr. Hairong Zhang, Dr. Junqin Lu and Dr. Jiuyun Sun for their encouragement and spiritual backup. I would also like to thank my friends in the Large-Scale Scientific Information Systems Research Group (LSIS), particularly Dr. Angelica Garcia, Michael Owonibi, Salahaldin Juba, Constantin Jucovschi, Xia Wang and Dimitar Misev, for our philosophical debates and exchanges of knowledge and skills which helped enrich the experience. I would also like to thank my family for the support they have provided me through my entire life and in particular, I must acknowledge my wife Ruiju, without whose love, encouragement and logistical support, I would not have finished this thesis.

My thesis research has been supported by the European Space Agency (ESA) through the HMA-FO project, China Scholarship Council (CSC) through a stipend and the Helmholtz Earth Science System Research School (ESSReS). I would like to express my gratitude to these agencies.

Contents

1 Introduction

Geo raster web services provide access to detailed and rich sets of geospatial information used in the multidisciplinary earth system science research, such as solar, atmosphere, ocean, cryosphere, solid earth, and biosphere research. The use of internet and related technologies for accessing such geospatial data, as well as for performing basic queries and processing on these data, allows us to find, share, combine and process geospatial information more easily. However, the heterogeneity of services constitutes a barrier to service interoperability. To ensure better global sharing of geospatial information in a heterogeneous world of geospatial web services, a series of corresponding principles that have been applied to geo service standards have been defined by international organizations such as the Open Geospatial Consortium (OGC), in collaboration with ISO, which are currently the main driving forces in open standards for interoperable geo services. The corresponding conformance testing programs have been set up to ensure interoperability of implementations that claim to adhere to the specifications. While such testing poses challenges in itself, the extent to which the specifications themselves support testing of implementations is also significant. For example, the designed test cases may be too many or too few to test a specified requirement; the truth evaluation of a conformance statement is undecidable when its dependency is unknown.

1.1 Motivation

Consider the following simple cutout retrieval. Assume that a request is sent to a geo raster web service to obtain a subset area of a 2-D image (see **Figure 1**).

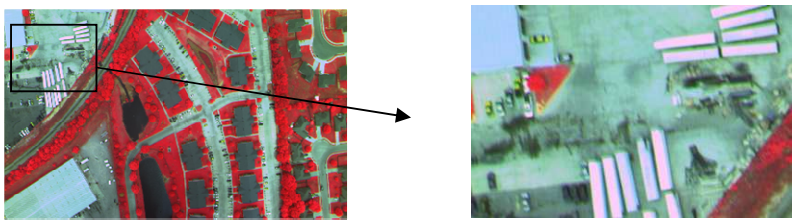
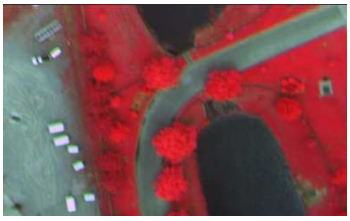
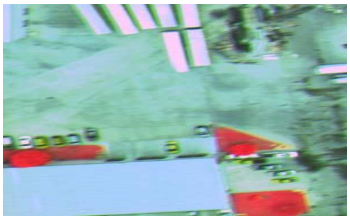


Figure 1 Subset area obtained from a false color city image
(source: http://www.earthlook.org/)

The result is a cutout image in a specified encoding format. The question arises how a machine can verify that the service does not misinterpret the corresponding specifications and provides the data correctly. Obviously, a successful cutout operation depends on the proper cutout action, format encoding and transfer. Unless a test machine is intelligent enough to digest the underlying geospatial semantic, it will accept images which actually are pseudo compliance results. For example, the above image may represent a 2-D image of the same size as the request but in a different spatial area, a rotated coordinate system, false pixel values or any other falsified image result (see **Figure 2**).

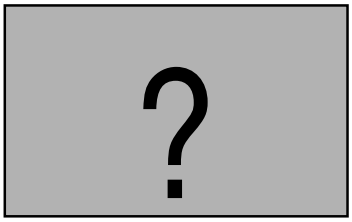


(a) result of different area



(b) result in a rotated coordinate system

(c) result with wrong pixel intensities



(d) otherwise wrong result

Figure 2 Pseudo compliance images

This scenario gives rise to the following questions:

- How much do you trust your services?
- How many tests are sufficient?
- How much do you trust the tests?
- Do they cover everything?
- How can you analyze your test results?
- How do you know you can trust the “Yes” or “No” results?

1.2 Conformance test

In software engineering, a rich variety of testing approaches has evolved, such as traditional testing technologies, formal proof [26], the semi-formal approach [83] and integration approaches [83]. Due to the inherent complexity of most systems, testing can only examine certain aspects of all possible system behaviors, as stated by

Dijkstra [21], “Program testing can be used to show the presence of bugs, but never to show their absence.” Different testing approaches test software from different aspects. Conformance testing determines whether a system meets specified requirements. The aim is to gain confidence in the behaviors of the implementation with respect to specified requirements. A conformance statement asserts which specific requirements are met.

1.2.1 Standardized conformance testing

ISO 19105 [50] is a standard that provides the framework, concepts, and methodology for conformance testing in the field of digital geographic information. Since geographic information is complex and has many particular aspects such as spatial reference systems as stated in ISO 19111 [51], metadata as stated in ISO 19115 [52] and spatial characteristics of coverages as stated in ISO 19123 [53], this International Standard specifies its framework based in part on ISO 9646-1, ISO 10303-31 and ISO 10641. ISO 9646-1 [54] describes conformance testing methodology and the framework of Open Systems Interconnection (OSI). This standard specifies a general methodology for testing the conformance of a product to OSI specifications which the product claims to implement. ISO 10303-31 [55] describes conformance and testing in industrial automation systems and integration. ISO 10641 [56] describes conformance and testing for computer graphics and image processing.

1.2.2 General approaches for conformance testing

1.2.2.1 Formal correctness proof

Formal methods use rigorous proofs of correctness in which the conformance of an implementation can be conclusively and exhaustively demonstrated [26]. For example, array algebra [65] is used to define the unambiguous and operational semantics of multi-dimensional raster operations by Gutierrez [2]. This mathematically-based technique proves that if the input values satisfy certain constraints, the produced values will satisfy certain properties. The mathematical method contributes to system reliability and robustness. However, the agreement between formal proofs and practical implementations still needs to be tested by the use of falsification testing.

1.2.2.2 Semi-formal methods

The semi-formal methods impose some formality and support refinement activities, such as the use of semi-formal UML descriptions [83]. The symbols used in graphical notations are mainly analogous to real-world objects as stated by Bauer et al. [40] and Stenning et al [37]. This enables system requirements to be specified more naturally, and thus promotes better understanding. Although a semi-formal method is easier to understand and to apply, its semantics is always ambiguous without a precisely defined translation to a formal language. As stated by Dong [28], the graphical notations are sometimes imprecise, ambiguous, unclear and lack expressive strength.

1.2.2.3 Falsification testing

Falsification testing is a practical means of conformance testing [50]. Unlike the formal correctness proof approach, the use of falsification testing does not guarantee that the testing provides complete coverage of the requirements. It detects errors in an implementation by running specific tests that test an implementation against its relevant requirements. Therefore, it can only be used to conclude that an implementation has not been implemented to conform to its corresponding requirements.

1.2.3 Suitable approaches for Web service conformance testing

A web service is on the internet and makes data retrieval or processing available to the general public. This means that the major challenges faced by the testers of web services are different compared to traditional desktop applications. The distributed nature of web services needs to be considered before exploring different types of testing techniques as stated by Prazen [45]. While white-box testing and mutation testing [77] are not recommended due to lack of source code knowledge, black-box testing provides rapid functional testing that can be used across distributed services. By leveraging the rich information presented in service descriptions, Yunus and Rizwan [82] advocate that gray-box testing is ideal for detecting defects within a Service Oriented Architecture (SOA) by considering the distributed nature of web services. A number of different approaches, including their combination, can be used

for conformance testing as stated by Razali [83], the selected approaches depend on the specified domain criteria. In this research, we deal with services which are tested only with respect to their functional requirements. The internal structures of these implementations are not accessible, therefore, we only use black-box conformance testing. Usually, an implementation is evaluated based on its testing results. Additional works, such as evaluations of test maturity and result validity, improve confidence in testing results.

1.3 Array and Raster Web Services

In earth system research, raster data incorporate the use of arrays of arbitrary sizes and dimensions, so-called Multi-dimensional Discrete Data (MDD), where a geographic area is divided into array cells. Raster is a special case of geospatial coverage as defined in ISO 19123 [53] that can be used to represent phenomena that vary continuously over space. A coverage associates a position within a spatiotemporal domain to a record of values of defined data types [53].

Array algebra is a formal framework for describing arrays and their operations for manipulating arrays. Research of Gutierrez and Baumann [2] shows that fundamental geo raster operations can be derived according to array algebra. The derived preconditions and post conditions provide a formal guarantee on pixel level correctness of the corresponding applications, such as standardized raster services [66].

A set of formal algebra and calculus approaches have been done on array modeling [65], such as AQL [39], AML [4], RAM [8], and array algebra [73]. Recently, more raster data models have emerged in the field, such as SciDB [76], AQuery system [3], Maier and Howe's ADT/blob based model [10]. Baumann and Holsten [74] present a set of mappings among these array models, such as mappings from AQL, AML and RAM models to array algebra. The research shows that array algebra can express each of these models by inspecting all relevant aspects of both data models and operations. The research also shows that the inverse does not hold.

2 State of the Art

2.1 Test search space

Test search space is the set of all possible tests. Model-based testing, such as the works done by Belinfante et al. [1] and Fraser et al. [24], creates flexible and useful automation on test generation process. The model used describes the system under test (SUT) as an abstract, partial representation of the desired SUT behavior. Tests derived from such a model are functional ones at the same level of abstraction as the model. When considering a structural model, an exhaustive search of all combinations of all variables is trivial, but impossible if a continuous variable exists. A random test generator may create many tests, but may fail to satisfy all specified requirements without formal constraints as stated by Bin et al. [20]. Constraint programming [11] can be used to select tests that satisfy specific constraints by solving a set of constraints over a set of parameters. Under this approach, a solution found by solving the set of constraint formulas can be served as tests. Furthermore, numerous works have been done on equivalence class partitioning and boundary analysis [48][88] to condense test search spaces, such as edit distance measurement [87], extreme values, and structure similarities [33]. Each variable has its equivalence classes. To assemble these separately partitioned results, the complexity of cross-product computing is exponential. To design minimum necessary and specification-consistent test cases, specification-consistent constraints, need to be used to reduce unnecessary cases.

2.2 Test oracles

In software engineering, a test oracle is a mechanism used by software testers and software engineers to determine whether a test has passed or failed [80]. There are many different test oracle approaches [16][17][18][80] that can be used to generate, capture, and compare test results. Douglas describes classes of oracles for various types of automated software verification and validation; these include true oracle, stochastic oracle, heuristic oracle, sample oracle and consistent oracle [17]. “Different

oracles may be used for a single automated test and a single oracle may serve many test cases" as stated by Douglas [17]. However, more specific domain knowledge needs to be considered when testing a standardized service, such as types of geo services. Limited by human processing capabilities, it is not satisfactory to have a human to inspect machine-oriented messages in the automated testing of web services. Obviously, machine-oriented oracles are more suitable for testing service infrastructures on the fly. Based on these investigations, several suitable approaches, such as model checking, consistency verification, feature sampling and test combination, are preliminarily discussed by Yu [36]. However, how to adopt these oracles without human intervention is still missing from these approaches. With some suitably expressed specifications which can be digested by the machines, this process can be automated without the participation of domain experts.

2.3 Conformance statement evaluation

Traditionally, conformance testing investigates whether a product or system adheres to defined properties. Two-valued logic provides truth values indicating *true* and *false* results. Finite-valued [41][30] or infinite-valued (e.g. fuzzy) logics [90] provide more expressive capabilities for evaluation results. For example, Kleene's three-valued logic [42][43] can provide such expressiveness by adding a third truth value to address unexplored facts. To derive global validity of conformance statements, statement dependency relationships are needed to keep track of consistencies. On the overall orchestration of conformance statements, so as to ensure global validity of testing results, traditional graph theory approaches or Dependency Structure Matrix [63] (DSM) helps to sequentialize the evaluation process. Among them, several projects have been done on ordering the cycles in non-singular strongly-connected components. For example, Kung et al. [15] remove a random edge to break cycles, and Le Traon et al. [91], Tai and Daniel [38], Hewett et al. [81] and Briand et al. [34] deploy removal strategies according to the number of incoming and outgoing edges. Kraft et al. [47] remove edges according to a dependency weight function. These serializations lose dependency information of the set of removal edges. These may introduce

inappropriate conclusions in the assessment of global validity of conformance statements. Similar work in the 80's and early 90's is based on truth maintenance systems (TMS). Dependency analyses in strongly-connected components (SCCs) were performed, for instance, by Goodwin [31][32] to keep track of dependencies and detect inconsistencies. However, the cases in which logically dependent relationships occur are only considered in two-valued logic. Evaluation of validity of global conformance statements among cross-platform services can substantially increase the confidence of users, especially vendors and other information professionals, that specification-based products provide reliable results.

2.4 The derived quality model

ISO 9126 provides an internationally standardized view for evaluating software quality. ISO 9126-1 standard[1] [58] classifies software quality in a structured set of quality attributes. It provides a framework for organizations to define a quality model for a software product. Based on this framework, more specific quality models can be derived for specific tasks. To address the questions raised by our motivation in Section 1.1, we consider the aspects of the provided services, designed test cases and test results in the conformance assessment process. Respectively, we consider functionality compliance, maturity and analyzability as their quality attributes. We leave out the other aspects because measurements of these fit well with our research objectives as stated in Section 2.5.

By specifying target values for quality metrics, the degree of presence of quality attributes can be measured. There are three metric categories for ISO 9126-1 quality evaluation. Specifically, they are internal, external and quality in use metrics. The internal metrics, which are specified in ISO 9126-3 [60], are used to measure the quality of the intermediate deliverables and thereby predict the quality of the final products. The external metrics, which are specified in ISO 9126-2 [59], are used to measure the quality of the software products by measuring the system behaviors and

[1] In March of 2011, ISO/IEC 25010 is released to supersede ISO/IEC 9126-1

.

can only be used during the testing stages or during any operational stages. The quality in use metrics, which are specified in ISO 9126-4 [61], are used to measure the effects of using the software products in specific context of uses.

A conformance assessment process, directly or indirectly determines that a process, product, or service meets relevant specifications during the testing stages. The external metrics [64], which are measured during such a process, quantitatively reflect quality attributes as defined in ISO 9126-1. Therefore, we derive the external metrics to measure the three quality attributes. The quality attributes are **F**unctional compliance, **T**est coverage, **A**udit trail capability (FTA). The metrics are **F**unctionality compliance Metric, **M**aturity Metric and **A**nalyzability Metric (FMA). These attributes and their metrics are detailed as below:

- **Functionality compliance Metric - Functional compliance** [59] is a metric on how compliant the functionality of the product is to applicable regulations, standards and conventions. The formula is as below:

  $$X = A / B$$

  where A is the number of the compliance requirements that have been correctly implemented and B is the total number of compliance requirements specified.

- **Maturity Metric – Test coverage** [59] is a metric on how many of the required test cases have been executed during testing. The formula is demonstrated as below:

  $$Y= C / D$$

  where C is the number of actually performed test cases representing operation scenario during testing of a requirement and D is the total test case number that is to be performed to cover the requirement.

- **Analyzability Metric – Audit trail capability** [64] is a metric for a user to measure and identify specific operation which caused failure. The formula is as below:

  $$Z= E / F$$

  where E is the number of data actually recorded during operation and F is the

number of data planned to be recorded, sufficient to monitor the status of the software during operation.

We inherited the three metrics and used them in the conformance assessment process (see Table 1). A functional compliance metric is used to measure the functionality compliance of service interfaces. To test compliance items of service interfaces, a set of test cases is designed for the tasks. A test coverage metric is used to measure the maturity of designed tests. An audit trail capability metric is used to measure the analyzability of test results.

Table 1 Derived external metrics

Quality attribute	Metrics	Formula	Parameter description
Functionality compliance	Functional compliance	X = A / B	A is the number of the compliance requirements as specified that have been correctly implemented during testing and B is the total number of compliance requirements specified.
Maturity	Test coverage	Y = C / D	C is the number of actually performed test cases representing operation scenario during testing the requirement and D is the total test case number that is to be performed to cover the requirement.
Analyzability	Audit trail capability	Z = E /F	E is the number of data actually recorded during operation and F is the number of data planned to be recorded, enough to monitor the status of software during operation.

2.5 Research objectives

The main goal of the research presented in this dissertation is to investigate trust-worthiness issues pertinent to ensuring conformance statements of implementations that claim to adhere to the specifications. Specifically, these include functionality compliance of implemented services, maturity of designed tests and analyzability of test results. The goal gives rise to the following concrete research objectives:

1. Derive a quality model beyond the intrinsic properties of web service

conformance assessment process [50], such as repeatability, comparability and auditability [50]. The model should be able to measure functional compliance of specified compliance items, test coverage of test cases and audit trail capability of corresponding test results.

2. Establish a method to allow evaluating how much the specification itself supports testing the implementation. This gives rise to the further sub-objectives:
   2.1 How to establish a mechanism that allows evaluation of how many service requests are needed to test the specification-based service interfaces;
   2.2 How to establish a model that allows deriving the necessary reference outputs for comparison against actual responses of the system under test to enable automatic testing based on suitably expressed interface specifications;
   2.3 How to establish a model that allows addressing goals and dependency issues among the declared requirement items;
   2.4 How to establish an evaluate schedule that allows evaluation the validity of global conformance statements.
3. Collect feedback for improving specifications, test suites and service implementations.

By addressing the above objectives, improved interoperation of geo raster web services under evolving implementations is expected in the future.

3 Service testing complexities

In this section, we study how many service requests are needed to test a standardized implementation and establish a mechanism that allows doing this. We assume that an implementation service responds to an invalid request by returning an error message and responds to a valid request by returning a valid item. A request consists of a set of parameters. For each parameter, there is a set of valid inputs and a set of invalid inputs. As the continuous variable is impossible for finite test generation, we treat each equivalence class as a test input of the parameter. Valid equivalence classes make up its search space of valid inputs and invalid equivalence classes make up its search space of invalid inputs. Valid and invalid equivalence classes of a parameter comprise a test search space of this parameter. Normally, an equivalence class is tested by one of its instances. For a request with a single parameter, its equivalence classes comprise its test search space. For a request with multiple parameters, an element of its search space is a combination of separate equivalence classes of these parameters. In a service test request, some parameters are independent from each other, while others are not. We propose a Request Parameter Relationship Analysis (RPRA) approach and differentiate between independent and dependent relationships among the parameters.

3.1 Service Request Parameter Relationships

In this research, we distinguish two parameter types, Atomic Parameters (AP) and Composite Parameters (CP). An atomic parameter can not be subdivided. A composite parameter is a combination of parameters which are called *direct children* of the parameter.

A set of parameters is in an independent relationship if and only if each parameter neither proves nor refutes any of the others. For a composite parameter with independent direct children, an instance is valid if all instances of its direct children are valid and is invalid if there is at least one invalid input. A valid instance of a composite parameter is able to test n possible valid equivalence classe. An invalid

instance of a composite parameter can test only one invalid equivalence class as long as the others are stubbed.

A set of parameters is in a dependent relationship if and only if at least one parameter can be proved by the other. If A depends on B, B is called a dependency of A and A is called a dependent of B. We only consider the case of two parameters, because both dependent and dependency can be composite parameters. Furthermore, as the validity of the dependent can not be concluded if its dependency is invalid, we only consider the case when the dependency contains only one valid input and the dependent contains a set of valid and invalid inputs. A dependency contains multiple inputs are not discussed here as the cases with different inputs can be treated as alternative inputs for the parameter. For a composite parameter with its direct children in a dependent relationship, an instance is valid if both the dependent and dependency are valid and is otherwise invalid. A valid instance of the composite parameter can test only one valid equivalence class of the dependent together with its valid dependency. An invalid instance of the composite parameter can test only one invalid equivalence class of the dependent together with its valid dependency.

3.2 Service test request generation

Based on separate equivalence classes of parameters, searched spaces of composite parameters are assembled according to relationship constraints among their direct children.

- Independent relationship (IR)

  When $P_1, P_2 \ldots,$ and P_n are direct children of a composite parameter in an independent relationship, parameter P_i has a set of x_i valid equivalence classes $\{p_{i1}^+, p_{i2}^+, \ldots, p_{ix_i}^+\}$ and a set of y_i invalid equivalence classes $\{p_{i1}^-, p_{i2}^-, \ldots, p_{iy_i}^-\}$, P_i^+ is the search space of valid inputs, P_i^- is the search space of invalid inputs, P_i is the test search space of this parameter, $p_{ij}^+ \in P_i^+, p_{ij}^- \in P_i^-, P_i = \{p_{i1}, p_{i2}, \ldots, p_{i(x_i+y_i)}\}, p_{ij} \in P_i$. Specifically,

$$P_i^+ \cup P_i^- = P_i$$

$$P_i^+ \cap P_i^- = \phi$$

$< list >$, a list of direct children with cardinality m where $1 \leq m \leq n$, is an input of such a composite parameter. An instance of the composite parameter is able to test m possible equivalence classes. To test valid retrievals, in the case of cardinality m, the minimum required instance number of the composite parameter is ceiling($\frac{\sum x_i}{m}$) if ceiling($\frac{\sum x_i}{m}$) $\geq \max(x_i)$, otherwise, it is $\max(x_i)$. The search space of valid inputs is denoted as $\{< list >: \forall p \; in < list > (\exists P \in \{P_i^+\} \wedge p \in P)\}$. The invalid retrievals need to be tested for each invalid equivalence class of its direct children under the assumption that the other parameters are valid. An instance of such a composite parameter tests only one invalid equivalence class of its direct children. Therefore, to test invalid retrievals, the minimum required instance number of the composite parameter is $\sum y_i$. The search space of invalid inputs is denoted as $\{< list >: \exists p \; in < list > (\exists P \in \{P_i^-\} \wedge p \in P)\}$.

We derive some special cases in a real life service testing study [35]. When $m = 0$, the list is empty and no valid or invalid equivalence class is tested. However, the case tests a request with empty parameters and its validity depends on its real use. When $m = 1$, the list has one element. Only one valid or invalid equivalence class is tested in this case. The case can be treated as a substitution among equivalence classes. When $m = n$, an instance of the composite parameter is able to test n possible equivalence classes. $\text{Max}(x_i)$ instances of the composite parameter are able to test $n \times \max(x_i)$ possible equivalence classes of its direct children and are enough to cover all valid equivalence classes. Furthermore, the direct child with $\max(x_i)$ valid equivalence classes needs at least $\max(x_i)$ instances of the composite parameter to cover its search space of valid inputs. Therefore, to test valid retrievals, the minimum required instance number of the composite parameter is $\max(x_i)$. The invalid retrievals need to be tested for each invalid equivalence class of its direct children under the

assumption that the other parameters are valid. An instance of such a composite parameter tests only one invalid equivalence class of its direct children. Therefore, to test invalid retrievals, the minimum required instance number of the composite parameter is $\sum y_i$.

- Dependency relationship (DR)

The mono-dependency, which is a singleton, has only one valid equivalence class. Each equivalence class of the dependent is compared with the dependency for the validity in such a circumstance. A dependency, which has n valid equivalence classes, can be turn into n mono-dependencies.

If M, Q are direct children of a composite parameter in a dependent relationship, M is a mono-dependency, $m \in M$, Q has a set of x valid equivalence classes $\{q_1^+, q_2^+, \dots, q_x^+\}$ and a set of y invalid equivalence classes $\{q_1^-, q_2^-, \dots, q_y^-\}$, Q^+ is the search space of valid Q inputs, Q^- is the search space of invalid Q inputs, $Q^+ = \{q_1^+, q_2^+, \dots, q_x^+\}$, $q^+ \in Q^+, Q^- = \{q_1^-, q_2^-, \dots, q_y^-\}, q^- \in Q^-, q \in Q$, $< m, q >$ is an instance of such a composite parameter. An instance of the composite parameter is able to test one equivalence class of Q. Therefore, to test valid retrievals, the minimum required instance number of the composite parameter is x. To test invalid retrievals, the minimum required instance number of the composite parameter is y.

We differentiate complex parameters based on a real life service testing study [35], which are: Partial Elements Complex Parameter and Mono-Dependency Relationship. Specifically, they are defined as:

- **Partial elements complex parameter (PECP)**: a composite parameter in an independent relationship with a cardinality of m, where $0 \leq m \leq n$;
- **Complex parameter with mono-dependency (MDCP)**: a composite parameter in a dependent relationship between a mono-dependency and its

corresponding dependent.

Accordingly, we design a query language, namely, Service Test Request Generation Query Language (STRG-QL) to manipulate the generation of test search space. The syntax is given as below:

```
STR : Parameter

M: Parameter:stubbed

Parameter:
   AP|M|CP|Parameter:random|Parameter:random_invalid
   |(Parameter)

ParameterList:Parameter|ParameterList,Parameter

CP : PECP|MDCP

PECP: IR(ParameterList):cardinality = m|random(low:high)

MDCP: DR(M,Parameter)
```

Syntax rules are as follows [49] : underlined tokens represent literals which appear “as is” (“terminal symbols”) and other tokens represent sub-expressions to be substituted (“non-terminals”). A vertical bar (“|”) denotes alternatives. `AP` is an atomic parameter, and `M` is a stubbed parameter. When a parameter is stubbed, it contains only one valid input. `CP` is a complex parameter. `IR` means the parameters are in an independent relationship. `DR` means the parameters are in a dependent relationship; `cardinality = m|random(low:high)` indicates that a constant number *m* or a random number of children participate in test requests. The random number is constrained by the given *low* and *high* boundary. `Parameter:stubbed` indicates that the parameter is stubbed. `Parameter:random` indicates that the parameter uses a random input from its test search space. `Parameter:random_invalid` indicates that the parameter uses a random invalid input from its search space of invalid inputs.

More special cases are derived according to real uses. These include: Empty

Complex Parameter, Single Element Complex Parameter, Random Element Complex Parameter and All Elements Complex Parameter. These are detailed as below:

- **Empty complex parameter (ECP)**: is a composite parameter in an independent relationship with a cardinality of 0; its syntax is denoted as:

  ECP: *IR*(ParameterList): *cardinality = 0*

- **Single element complex parameter (SECP)**: is a composite parameter in an independent relationship with a cardinality of 1; its syntax is denoted as:

  SECP: *IR*(ParameterList):*cardinality = 1*

- **Random elements complex parameter (RECP)**: is a composite parameter in an independent relationship with a random cardinality; its syntax is denoted as:

  RECP:*IR*(ParameterList): *cardinality = random(low:high)*

  where *cardinality* is a random integer bounded by a pair of *low* and *high* values.

- **All elements complex parameter (AECP)**: is a composite parameter in a dependent relationship with a cardinality of *n*; its syntax is denoted as:

  AECP:*IR*(ParameterList): *cardinality = n*

3.3 Test maturity evaluation

In our quality model, the test coverage is used to measure test maturity as mentioned in Section 2.4. When e is an equivalence class of an atomic parameter, i is an instance of the equivalence class, $instance(e,i)$ mean i is an instance of e, the equivalence classes with no empty instances are denoted as $\{e: |\{i: instance(e,i)\}| > 0\}$. Correspondingly, test coverage [59] can be measured as Y= C / D, where C is the number of equivalence classes with no empty instances and D is the total equivalence class number.

For a composite parameter, the test coverage is measured by test coverages of its direct children. We assume that each equivalence class of its direct children is equally-weighted. Specifically, they are discussed below:

- Independent relationship evaluation

  When $c_1, c_2 \dots, c_n$ are test coverages on valid inputs of its direct children, $c_1', c_2', \dots, c_n'$ are test coverages on invalid inputs, $x_1, x_2 \dots, x_n$ are valid equivalence class numbers of its direct children, $y_1, y_2 \dots, y_n$ are invalid equivalence class numbers, the test coverage on valid inputs of the composite parameter is $\sum_{i=1}^{n} \frac{c_i * x_i}{\sum_{j=1}^{n} x_j}$ and the test coverage on invalid inputs of the composite parameter is $\sum_{i=1}^{n} \frac{c_i * y_i}{\sum_{j=1}^{n} y_j}$.

- Dependency relationship evaluation

  Q is *mono-dependent* on M, and M has only one valid equivalence class. Q has x valid equivalence classes and y invalid equivalence classes. When c_Q is test coverage on valid inputs of Q, c_Q' is test coverage on invalid inputs of Q, the test coverage on valid inputs of the composite parameter is c_Q and the test coverage on invalid inputs of the composite parameter is c_Q'.

3.4 Summary

We have discussed the minimally necessary and specification-based test request based on the dependent and in dependent relationships. Accordingly, we have designed STRG-QL for the generation of test search spaces. We have also discussed test maturity evaluations based on these relationships. The result can help to evaluate the existing tests.

4 Systematic reference outputs

In this section, we study how to establish a model that allows adopting the necessary reference outputs. Based on suitably expressed interface specifications, the process can be automated without manual intervention. Often, such a conceptualization is described by conditional statements. A conditional statement contains a set of preconditions and a set of post conditions. Preconditions are always associated with a set of facts which are extracted during operational stages. To deduce adopt reference outputs in accordance with post conditions, preconditions need to be satisfied.

4.1 Statement truth and three evaluation assumptions

We define a *statement* as a truth-bearer proposition. We distinguish three possible truth values of a statement, namely T, F, and U, following a three-valued logic [43]. We differentiate between atomic and composite statements. In logic, an *atomic statement* is a logic statement which cannot be broken down into smaller statements; a *composite statement* is a logic statement having two or more statements connected by logical *conjunction* (“∧”) and *disjunction* (“∨”) operators. We express such a statement by a logical expression. Syntactically, it is represented by this grammar:

TS: "*T*"|"*F*"|"*U*"

CDOP: " ∧ "|" ∨ "

TRE: TS|"("TRE")"|TRE CDOP TRE

where TS is a truth status and TRE is a logical truth result expression. This kind of expression is similar to the AND/OR graph as used in software engineering.

We distinguish *three evaluation assumptions* to evaluate unknown features: open world assumption (OWA), closed world assumption (CWA) [14] and stub assumption (SA). The open world assumption (OWA) states that the truth value of a statement that is not included in or inferred from the knowledge explicitly recorded in the system shall be considered unknown. The closed world assumption (CWA) is the assumption that any statement that is not known to be true is false. We distinguish yet

another case in software testing. Frequently, there are test stubs which simulate the behaviors of the dependent test modules. In this case, the dependent test modules are always assumed to be true when their results are not available. We call this a stub assumption (SA).

In Kleene's logic [42], a conjunction produces a value of T if both of its operands are T, an F if one of its operands is F, and otherwise U. Disjunction delivers a value of T if one of its operands is T, an F if both of its operands are F, and otherwise a U. Obviously, Kleene's approach [42] is an open world assumption. We extend this approach and evaluate traditionally logical expressions under the three assumptions (see Table 2). The corresponding syntax, namely 3A-TRE, is detailed in Annex A.

Table 2 Logic assumptions on unknown results

Assumption	Operand	Result
CWA	U	F
OWA	U	U
SA	U	T

4.2 Reference output adoption

We use deductive reasoning [78] to deduce suitable reference outputs for a single test. According to the law of detachment, also called modus ponens [46], when a conditional statement is made and a hypothesis (P) is stated as true, the conclusion (Q) is deduced from the hypothesis and the statement. The format is:

conditional statement: $P \rightarrow Q$

where P is the hypothesis and Q is the conclusion. The adopted reference outputs depend on the hypothesis that must be met. When P is evaluated as true, the actions in Q are carried out.

We model both P and Q with logical expressions (see Section 4.1). The conceptualization, which is always explicitly or implicitly specified, for example included in specifications as default tacit knowledge, is modeled by conditional statements. The set of extracted facts are used to evaluate the truth value of the

hypothesis as stated in P. With proper evaluation (see Section 4.1), we can conclude the adopted reference outputs. The corresponding example is in Section 6.2.

4.3 Summary

For automatic testing, having good reference outputs is indispensable. Systematic generation of tests and their outputs is one way to meet this requirement. In order to address reference output adoption, we propose using deductive reasoning based on specifically expressed conditional statements.

5 Validity of global statements

In this section, we propose the integrated dependency and goal model (IDGM) to address dependency issues among service components under a test process. We prove the conditions of stable results and provide the corresponding evaluation schedule.

5.1 Sub-goal and subordination

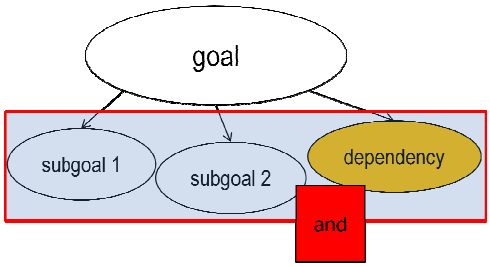
A *goal* is a result that an activity is trying to achieve. *Subgoals* are lesser goals that form part of a greater goal. Sometimes several subgoals must be achieved to claim success of the greater goal. This establishes a logical “and” prerequisite condition. Sometimes, at least one subgoal must have been achieved; for example, at least one service protocol must have passed the test before the service as such can be assessed. This leads to a logical “or” combination. Therefore, we treat a goal as a logical statement as stated in Section 4.1.

Subordination is a relationship in which one or more statements are dependent on each other. Sometimes a goal contains subordinates. An unknown or false subordinate will introduce an over-optimistic or wrong result. Therefore, such a goal’s subordinate can be treated as a subgoal which needs to be evaluated for consistency. A goal that is defined by the conjunction of its subgoals and subordinates is called a well-designed goal (WDG).

5.2 Integrated dependency and goal model

A well-designed goal depends on both its subgoals and subordinates. We model dependence between two statements with a directed edge. We now develop the model based on the directed graph for tracing dependencies between statements. We start with a directed dependency graph G=(V,E). The vertex set V contains statements as vertices, the edge set E consists of dependencies; an edge e = (s_1,s_2) denotes that vertex s_1 depends on vertex s_2. s_2 is said to be a direct successor of s_1 while s_1 is said to be a direct predecessor of s_2. Generally, if a path in G is made up one or more successive edges leading from vertex s_1 to vertex s_2, then s_2 is said to be a successor

of s_1 and s_1 is said to be a predecessor of s_2. If there is a path from s_1 to s_2 and also a return path from s_2 to s_1, then s_1 and s_2 are strongly connected. A strongly-connected component (SCC) [89] is a maximal strongly-connected subgraph of G. To evaluate a well-designed goal, its result is evaluated according to its direct successors, specifically, subgoals and subordinates in a logical expression. The integrated goal and dependency model (IDGM) is a model for handling such logical statements with each of its statements represented as a well-designed goal. For example, a goal has subgoal1 s_1 and subgoal2 s_2 as its sub goals, and a dependency d; its corresponding well-designed goal is denoted as $s_1 \wedge s_2 \wedge d$ (see Figure 3).



\ d

Figure 3 Sample IDGM

5.3 Isolated testing environment

An *isolated testing environment* is an environment that consists of a decidable system to be tested and a test with a partial test function that has no interaction with its external environment. The partial test function *f* is defined as:

$$\forall x(x \in X \rightarrow f(x) = T)$$

$$\forall x(x \in Y \rightarrow f(x) = F)$$

$$\forall x(x \in S \wedge x \notin X \wedge x \notin Y \rightarrow f(x) = U)$$

where *S* is the set of test inputs, the test returns *true* if the input is in set *X*, and returns *false* if the input is in set *Y*; otherwise, it returns an *unknown* result.

5.4 Global Validity

To ensure global validity of conformance statements, it is necessary to obtain stable results. Dependencies among statements may introduce evaluation deadlocks. For

example, let us assume that statement s_1 depends on s_2 and s_3, s_3 depends on s_4 and s_4 depends on s_1. This means that s_1, s_3 and s_4 are strongly-connected as shown in Figure 4. The cycle introduces a deadlock in the evaluation process.

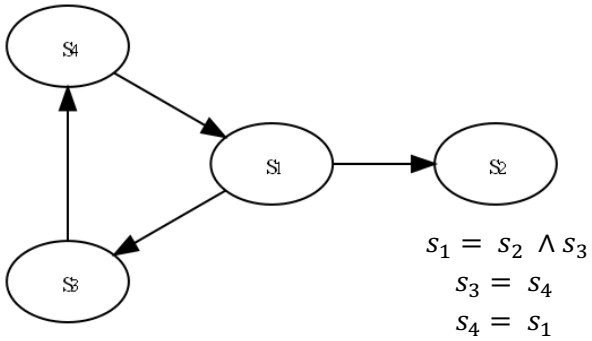


Figure 4 Sample deadlock

An evaluation action is an evaluation of a statement by its corresponding logical expression. The integration of goals and subordinates avoids inconsistent results which may be introduced when a statement's subgoals and subordinates are evaluated separately.

When $s_1, s_2, \ldots,$ and s_n are statements, $a_1, a_2, \ldots,$ and a_n are the corresponding evaluation actions of these statements, $x_1, x_2, \ldots,$ and x_n are truth values of these statements. An *evaluation step* is a sequential or concurrent execution of a set of evaluation actions; a *concurrent evaluation* is an evaluation step, which consists of a set of concurrent evaluation actions; a *sequential evaluation* is an evaluation step, which consists of a set of sequential evaluation actions; an evaluation process is a recursive evaluation step. The step takes the previous result as an input and it is denoted as $f_{step}: E \to E$, where $e \in E$ and $e = (x_1, x_2, \ldots, x_n)$. The process is terminable if there exists a fixpoint such that $e = f_{step}(e)$; an evaluation schedule of a terminable process is a *terminable schedule*, otherwise, it is an *interminable schedule*.

5.4.1 Fixpoint in IDGM

We prove that a set of stable results exists, including statements in non-singular strongly-connected components. We set up three *confidence levels* for a statement. Specifically, they correspond to three decreasing confidence degrees which are T (true), U (unknown) and F (false), respectively. We use an ordering relation of confidence and denote it as $x_1 > x_2$. Consequently, the confidence precedence order is defined as T > U > F.

Table 3 Extended truth table of the logic operations for Kleene's logic with CWA and SA

P	Q	P and Q (CWA)	P and Q (Kleene's logic, OWA)	P and Q (SA)	P or Q (CWA)	P or Q (Kleene's logic, OWA)	P or Q (SA)
T	T	T	T	T	T	T	T
T	U	F	U	T	T	T	T
T	F	F	F	F	T	T	T
U	T	F	U	T	T	T	T
U	U	F	U	T	F	U	T
U	F	F	F	F	F	U	T
F	T	F	F	F	T	T	T
F	U	F	F	F	F	U	T
F	F	F	F	F	F	F	F

We define the truth table of conjunction and disjunction operations with open world assumption (OWA), closed world assumption (CWA) and stub assumption (SA) (see Table 3). Then, we prove that each evaluation action, each according to its logical expression, is order preserving with respect to certain constrains and the logical precedence below. The proof steps are:

- Conjunction: $a \leq b$ implies $f_{conj}(a) \leq f_{conj}(b)$, where $f_{conj}(x)$ is a conjunction clause that contains any occurrence of variable x, with the others

hold constant;

- Disjunction: $a \leq b$ implies $f_{disj}(a) \leq f_{disj}(b)$, where $f_{disj}(x)$ is a conjunction clause that contains any occurrence of variable x, with the others hold constant;
- DNF: $a \leq b$ implies $f_{DNF}(a) \leq f_{DNF}(b)$, where $f_{DNF}(x)$ is a conjunction clause that contains any occurrence of variable x, with the others hold constant;

The expression which can be expressed by a formula in DNF, satisfies the order preserving property. Hence, each evaluation action is order preserving. An evaluation action, which contains a set of variables, is order preserving:

- DNF: $e_1 \leq e_2$ implies $f_{DNF}(e_1) \leq f_{DNF}(e_2)$, where $e \in E$, $e = (x_1, x_2, \dots, x_n)$ and $f_{DNF}(e)$ is a conjunction clause that contains any occurrence of variables $x_1, x_2, \dots$ and x_n.

If x_i is a statement evaluation result and e is the evaluation result of the evaluation step $f_{step}: E \to E$, where $e \in E$ and $e = (x_1, x_2, \dots, x_n)$, the set of all inputs E is a complete lattice. A complete lattice [23] is a partially-ordered set [25] in which all subsets have both a supremum and an infimum. This proof is:

- Firstly, we prove that E is partially ordered along the confidence precedence relation:
  - **Reflexivity**, $(x_1, x_2, \dots, x_n) \geq (x_1, x_2, \dots, x_n)$, where x_i is the truth value of conformance statement s_i;
  - **Antisymmetry**, if $(x_1, x_2, \dots, x_n) \geq (y_1, y_2, \dots, y_n)$ and $(y_1, y_2, \dots, y_n) \geq (x_1, x_2, \dots, x_n)$, $(x_1, x_2, \dots, x_n) = (y_1, y_2, \dots, y_n)$, where x_i and y_i is are corresponding truth values of conformance statement s_i;
  - **Transitivity**, if $(x_1, x_2, \dots, x_n) \geq (y_1, y_2, \dots, y_n)$ and $(y_1, y_2, \dots, y_n) \geq (z_1, z_2, \dots, z_n)$, $(x_1, x_2, \dots, x_n) \geq (z_1, z_2, \dots, z_n)$,

where x_i, y_i and z_i are the corresponding truth values of conformance statement s_i.

- Secondly, we prove that each subset *T* of the partially-ordered set *E* has both a *supremum* and an *infimum*:

  - If *t* is an element of *T*, $t_j = (x_1^{t_j}, x_2^{t_j}, \dots, x_n^{t_j})$; $x_i^{t_j}$ is the truth value of statement s_i of evaluation t_j;

$$T = \{t_1, t_2, \dots, t_m\}$$

$$= \{(x_1^{t_1}, x_2^{t_1}, \dots, x_n^{t_1}),$$

$$(x_1^{t_2}, x_2^{t_2}, \dots, x_n^{t_2}),$$

$$\dots,$$

$$(x_1^{t_m}, x_2^{t_m}, \dots, x_n^{t_m})\};$$

  low$(x_i^{t_1}, x_i^{t_2}, \dots, x_i^{t_m})$ is a lower bounder of x_i;

  the lower bounder of *T* is ρ = (low$(x_1^{t_1}, x_1^{t_2}, \dots, x_i^{t_m})$,

  low$(x_2^{t_1}, x_2^{t_2}, \dots, x_2^{t_m})$,

  …,

  low$(x_n^{t_1}, x_n^{t_2}, \dots, x_n^{t_m})$);

  - Then, we can derive that:

    1. $\forall t(\rho \leq t)$
    2. $\forall e(\forall t(e \leq t) \rightarrow e \leq \rho))$, where *e* is an element of *E*;
    3. *infimum* (*T*) = ρ;

  - Similarly, we can derive:

  supremum (*T*) = (high$(x_1^{t_1}, x_1^{t_2}, \dots, x_i^{t_m})$,

$$\text{high}(x_2^{t_1}, x_2^{t_2}, \ldots, x_2^{t_m}),$$

$$\ldots,$$

$$\text{high}(x_n^{t_1}, x_n^{t_2}, \ldots, x_n^{t_m}));$$

- Therefore, each subset T of the partially-ordered set E has both a *supremum* and an *infimum*.

- Therefore, E is a complete lattice.

The Knaster–Tarski theorem [22] states that there is a set of fixpoints of f in E if E is a complete lattice and $f: E \to E$ is an order preserving function. A fixpoint of f in E is denoted as $e = f(e)$.

However, there are dependent relationships among variables $x_1, x_2, \ldots$ and x_n. This will result in the case of neither non-decreasing nor non-increasing if these statements are not properly evaluated. Specifically, they are discussed in the below subsections.

5.4.1.1 Concurrent evaluation

A concurrent evaluation simultaneously executes several evaluation actions in an evaluation step. However, the concurrent evaluation is not always an order preserving function due to the oscillating results in non-singular strongly-connected components; such as the example shown in Table 4. Oscillation is the repetitive variation among different states. The oscillating results will result in an interminable evaluation process.

Table 4 Concurrent evaluation on nodes of a non-singular strongly-connected component

statement	logical expression	initial value	$step_1$	$step_2$	…
s_1	s_2	T	F	T	…
s_2	s_1	F	T	F	…

5.4.1.2 Sequential evaluation

A *sequential evaluation* sequentially executes several evaluation actions in one evaluation step. An improper sequence, in which a statement evaluation is scheduled prior to its dependencies, may result in oscillating results. Thus, the sequential evaluation is not an order preserving function in this case, such as the example shown in Table 5.

Table 5 Sequential evaluation of a non-singular strongly-connected component with an improper sequence

statement	logical expression	initial value	$step_1$	$step_2$	…
s_1	s_2	T	F	T	…
s_2	s_3	F	T	F	…
s_3	s_1	T	F	T	…

Therefore, we consider a *serialized sequence*, in which each dependency is scheduled prior to a statement itself. An IDGM can either be a directed acyclic graph (DAG), a non-singular strongly-connected component, or a directed graph with non-singular strongly-connected components. For a non-singular strongly-connected component, if there is only one path from each node in the non-singular strongly-connected component to every other node without crossing itself, the component is a *simple non-singular strongly-connected component*; otherwise, it is a *complex non-singular strongly-connected component*. In a serialized sequence, the evaluation is an order preserving function. The proof is:

(1) For a directed acyclic graph (DAG), a topological sorting [89] avoids an improper evaluation sequence; the evaluation step can be equivalently defined as a sequence of functions as:

$$\begin{cases} f: x_1 \to x_2 \\ f: (x_1, x_2) \to x_3 \\ \dots \\ f: (x_1, x_2, \dots, x_{n-1}) \to x_n \end{cases}$$

where x_1 is the initial input.

We can derive that:

1. Each evaluation action is order preserving; therefore, f is order preserving because each evaluation action feeds its result to its next action and shares the same order preserving property; and such an evaluation result is stable and can be derived in one evaluation step;
2. If the fixpoint derivation is defined as a function $f_{fixpoint}: x_1 \to P$, where x_1 is the initial input and P is the fixpoint, the function is order preserving.

(2) For a simple non-singular strongly-connected component, the deadlock can be disentangled by initiating one of its nodes. If e_i is an evaluation step, the evaluation step can be equivalently defined as a sequence of functions as:

$$\begin{cases} f_{e_i}: x_1^{e_i} \to x_2^{e_i}\ (i \geq 1) \\ f_{e_i}: (x_1^{e_i}, x_2^{e_i}) \to x_3^{e_i}\ (i \geq 1) \\ \dots \\ f_{e_i}: (x_1^{e_i}, x_2^{e_i}, \dots, x_{n-1}^{e_i}) \to x_n^{e_i}\ (i \geq 1) \\ f_{e_i}: (x_2^{e_i}, x_3^{e_i}, \dots, x_n^{e_i}) \to x_1^{e_i}\ (i > 1) \end{cases}$$

where $x_1^{e_1}$ is the initial input.

We can derive that:

1. Each evaluation action is order preserving; therefore, f_{e_i} is order preserving because each evaluation action feeds its result to its next action and share the same order preserving property.
2. If the fixpoint derivation is defined as a function $f_{fixpoint}: x_1^{e_1} \to P$, where $x_1^{e_1}$ is the initial input and P is the fixpoint, the function is order preserving.
3. $x_1^{e_1} \in \{T, U, F\}$; hence, $e = f_{step}(e)$ can be derived in three steps, where

$e = (x_1, x_2, \dots, x_n)$. Only one fixpoint can be derived for an initial input. However, this fixpoint may be different for a different initialization.

(3) For a *complex non-singular strongly-connected component*, one of its nodes is initialized by a given input $x_1^{\prime e_i'}$. This way, the node is disentangled from its dependencies. In the rest of the component, the non-singular strongly-connected components are condensed to atomic nodes. This makes a directed acyclic graph (DAG).

When $s_1', s_2', \dots, \text{and } s_m'$ are the derived nodes, including the initial node s_1', $e_1', e_2', e_3', \dots$ are evaluation steps, $x_1^{\prime e_i'}, x_2^{\prime e_i'}, \dots, \text{and } x_m^{\prime e_i'}$ are the corresponding evaluation results, a evaluation step, which consists of a set of evaluation actions of these nodes, and can be equivalently defined as a sequence of functions as below:

$$\begin{cases} f_{e_i'}: x_1^{\prime e_i'} \to x_2^{\prime e_i'} \ (i \geq 1) \\ f_{e_i'}: (x^{\prime e_i'}_1, x_2^{e_i'}) \to x_3^{\prime e_i'} \ (i \geq 1) \\ \dots \\ f_{e_i'}: (x_1^{\prime e_i'}, x_2^{\prime e_i'}, \dots, x^{\prime e_i'}_{m-1}) \to x^{\prime e_i'}_m \ (i \geq 1) \\ f_{e_i'}: (x_2^{\prime e_i'}, x_3^{\prime e_i'}, \dots, x_m^{\prime e_i'}) \to x_1^{\prime e_i'} \ (i > 1) \end{cases}$$

where $x_1^{\prime e_1'}$ is the initial input of s_1'. If each node evaluation action is order preserving, we can derive that:

1. $f_{e_i'}$ is order preserving because each evaluation action feeds its result to its next action and shares the same order preserving property;
2. If the fixpoint derivation is defined as a function $f_{fixpoint}: x_1^{\prime e_1'} \to P$, where $x_1^{\prime e_1'}$ is the initial input and P is the fixpoint, the function is order preserving;
3. $x_1^{\prime e_1'} \in \{T, U, F\}$; hence, $e' = f_{step}(e')$ can be derived in three steps, where $e' = (x_1', x_2', \dots, x_m')$. One and only one fixpoint can be derived for

an initial input and this fixpoint may be different for a different initialization.

However, if a derived node is a deadlock, its evaluation result is not stable. Thus, its order preserving is impossible. Further initialization is needed to disentangle this deadlock. In this case, the derived node's evaluation can be defined as $f_{e_i'}: (x_1'^{e_i'}, x_2'^{e_i'}, \dots, x_{j-1}'^{e_i'}, x_1''^{x_j'}) \to x_j'^{e_i'}$ $(i \geq 1, m \geq j > 1)$, where $x_1''^{x_j'}$ is the initial input of a node of x_j'. Hence, we define the set of sequential evaluation action functions as:

$$\begin{cases} f_{e_i'}: (x_1'^{e_i'}, x_1''^{x_2'}) \to x_2'^{e_i'} \ (i \geq 1) \\ f_{e_i'}: (x'^{e_i'}_1, x_2'^{e_i'}, x_1''^{x_3'}) \to x_3'^{e_i'} \ (i \geq 1) \\ \dots \\ f_{e_i'}: (x_1'^{e_i'}, x_2'^{e_i'}, \dots, x'^{e_i'}_{m-1}, x_1''^{x_m'}) \to x'^{e_i'}_m \ (i \geq 1) \\ f_{e_i'}: (x_2'^{e_i'}, x_3'^{e_i'}, \dots, x_m'^{e_i'}) \to x_1'^{e_i'} \ (i > 1) \end{cases}$$

where $x_1''^{x_j'}$ is the initial input of node x_j'. If each evaluation result is stable, the fixpoint derivation $f_{fixpoint}: x_1'^{e_i'} \to P$ is order preserving because each evaluation action feeds its result to its next action and shares the same order preserving property. According to (1), if node x_j' is an atomic node, its evaluation result is stable with truth values of $x_1'^{e_i'}, x_2'^{e_i'}, \dots,$ and $x_{j-1}'^{e_i'}$ which hold constant. According to (2), if x_j' is a simple non-singular strongly-connected component, its evaluation result, which is initialized by $x_1''^{x_j'}$, is stable with truth values of $x_1'^{e_i'}, x_2'^{e_i'}, \dots,$ and $x_{j-1}'^{e_i'}$ which hold constant. According to (3), if the node is a complex non-singular strongly-connected component, further initialization, disentanglement and condensation are needed. These need to be repeated recursively until only atomic nodes or simple non-singular strongly-connected components can be found. Hence, the stable status is derived by recursive initializations. Obviously, this can be achieved in finite steps.

Therefore, for a given initialization of deadlocks of non-singular strongly-connected components, the stable result can be derived through a serialized schedule. The evaluation in a serialized sequence with respect to the given initialization is order preserving.

5.4.2 False deadlocks and deadlock initialization

Actually, we do not need to recursively evaluate all non-singular strongly-connected components. In a serialized sequence evaluation schedule, some truth values of a non-singular strongly-connected component are determined by one or more elements of its dependencies. For example, when exp is a logical expression, $T \vee exp$ will always return *T* even the truth value of exp is unknown. A node with such a constant truth value disentangles the deadlock. This results in a *false evaluation deadlock*. Furthermore, a $U \vee exp$ will not return an *F* result. In this case, the statement can only be valued as either *T* or U.

A possible truth list (PTL) is a list of possible truth values for a given statement. We distinguish these initialization patterns in Table 6.

Table 6 Possible truth lists of specific initialization patterns

case	Possible truth list
$T \vee exp$	T
$F \wedge exp$	F
$U \vee exp$	T, U
$U \wedge exp$	U, F
$F \vee exp$	T, U, F
$T \wedge exp$	T, U, F
$exp \vee exp$	T, U, F
$exp \wedge exp$	T, U, F

Here are truth tables to evaluate a possible truth list under open world assumption (see Table 7), closed world assumption (see Table 8) and stubbed assumption (see Table 9).

Table 7 Possible truth lists under open world assumption

OWA	T	U	F	T,U	U,F	T,U,F	T,F	T	U	F	T,U	U,F	T,U,F	T,F
T	T	U	F	T,U	U,F	T,U,F	T,F	T	T	T	T	T	T	T
U	U	U	F	U	U,F	U,F	U,F	T	U	U	T,U	U	T,U	T,U
F	F	F	F	F	F	F	F	T	U	F	T,U	U,F	T,U,F	T,F
T,U	T,U	U	F	T,U	U,F	T,U,F	T,U,F	T	T,U	T,U	T,U	T,U	T,U	T,U
U,F	U,F	U,F	F	U,F	U,F	U,F	U,F	T	U	U,F	T,U	U,F	T,U,F	T,U,F
T,U,F	T,U,F	U,F	F	T,U,F	U,F	T,U,F	T,U,F	T	T,U	T,U,F	T,U	T,U,F	T,U,F	T,U,F
T,F	T,F	U,F	F	T,U,F	U,F	T,U,F	T,F	T	T,U	T,F	T,U	T,U,F	T,U,F	T,F

Table 8 Possible Truth Lists under closed world assumption

CWA	T	U	F	T,U	U,F	T,U,F	T,F	T	U	F	T,U	U,F	T,U,F	T,F
T	T	F	F	T,F	F	T,F	T,F	T	T	T	T	T	T	T
U	F	F	F	F	F	F	F	T	F	F	T,F	F	T,F	T,F
F	F	F	F	F	F	F	F	T	F	F	T,F	F	T,F	T,F
T,U	T,F	F	F	T,F	F	T,F	T,F	T	T,F	T,F	T,F	T,F	T,F	T,F
U,F	F	F	F	F	F	F	F	T	F	F	T,F	F	T,F	T,F
T,U,F	T,F	F	F	T,F	F	T,F	T,F	T	T,F	T,F	T,F	T,F	T,F	T,F
T,F	T,F	F	F	T,F	F	T,F	T,F	T	T,F	T,F	T,F	T,F	T,F	T,F

Table 9 Possible truth lists under stubbed assumption

SA	T	U	F	T,U	U,F	T,U,F	T,F	T	U	F	T,U	U,F	T,U,F	T,F
T	T	T	F	T	T,F	T,F	T,F	T	T	T	T	T	T	T
U	T	T	F	T	T,F	T,F	T,F	T	T	T	$T,$	T	$T,$	T
F	F	F	F	F	F	F	F	T	T	F	T	T,F	T,F	T,F
T,U	T	T	F	T	T,F	T,F	T,F	T	T	T	T	T	T	T
U,F	T,F	T,F	F	T,F	T,F	T,F	T,F	T	T	T,F	T	T,F	T,F	T,F
T,U,F	T,F	T,F	F	T,F	T,F	T,F	T,F	T	T	T,F	T	T,F	T,F	$T,,F$
T,F	T,F	T,F	F	T,F	T,F	T,F	T,F	T	T	T,F	T	T,F	T,F	T,F

Syntactically, a possible truth result expression (PTRE) is represented as given by this grammar:

PTL: "T"|"U"|"F"|"T,U"|"U,F"|"T,U,F"|"T,F"|"("PTL")"

CDOP: " ∧ "|" ∨ "

PTRE: PTL|"("PTRE")"CDOP"("PTRE")"

where PTL is a list of possible truth values, PTRE is a logical possible truth list expression. We use PTRE to calculate a statement possible truth list.

If the possible truth list cardinality $l = 1$, statement truth value can be derived directly. Therefore, this kind of statement is always used to initiate a deadlock.

5.4.3 Evaluation schedule

To provide a serialized schedule, the dependencies need to be scheduled prior to the node itself. However, non-singular strongly-connected components of an IDGM will always cause deadlocks. Therefore, we contract each such component in to a condensed node, making the resulting graph a directed acyclic graph. Then we initialize the contracted non-singular strongly-connected component and recursively apply the contraction and initiation until no further non-singular strongly-connected

components can be found. The stable result is derived for each non-singular strongly-connected component according to its initialization. The schedule algorithm is detailed in Annex C. The algorithm traverses an entire graph, taking O(|V| + |E|) time where V is the number of vertices, and E is the number of edges.

We apply the schedule algorithm to the example in Figure 5. We detect one non-singular strongly-connected component which consists of s_1 and s_3 and another which consists of s_5 and s_6. Then we condense the non-singular strongly-connected components to vertices in the IDGM and apply the schedule algorithm on the derived DAG. Finally, we arrive at the evaluation schedule (see Table 10).

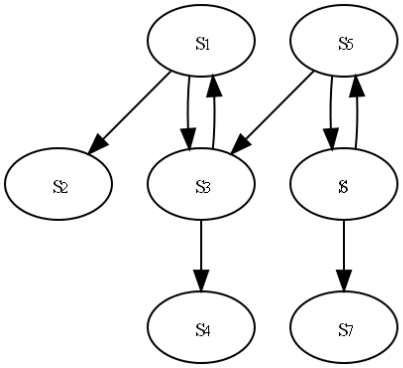
Table 10 Sample IDGM schedule result

	Schedule with cycle dependencies
IDGM	V = $\{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$, E = $\{(s_1, s_2), (s_1, s_3), (s_3, s_1), (s_3, s_4), (s_5, s_3), (s_5, s_6), (s_6, s_5), (s_6, s_7)$
DAG	V =$\{v_1\|\{s_5, s_6\}, v_2\|\{s_7\}, v_3\|\{s_1, s_3\}, v_4\|\{s_2\}, v_5\|\{s_4\}\}$, E = $\{(v_1, v_2), (v_1, v_3), (v_3, v_4), (v_3, v_5)\}$
Result	$s_2, s_4, s_7, (\boldsymbol{s_1}, s_3), (\boldsymbol{s_5}, s_6)$, ($s_1$, s_5 are initial nodes of the deadlocks)

Atomic nodes, such as s_2, s_4 and s_7, can be evaluated independently. Thus, these nodes are put in the front of the evaluation queue. To derive stable results of the deadlocks, an initial input is given to s_1. Then, s_3 is evaluated according to $s_1 \wedge s_4$. s_1 is reevaluated according to the derived s_3. The recursive process stops if the stable result is derived. Similarly, we initiate s_5 and get the stable result of the deadlock.

To initiate a node, initialization patterns, such as $F \wedge exp$ and $T \vee exp$, are used to calculate lists of possible truth values and reduce unnecessary recursions. However, possible truth list cardinality is not always “1”. In this case, logical assumptions are taken to initiate the starting nodes of deadlocks. When SA is applied, dependencies, which are unknown, are always assumed to be true. In this case, evaluation independency is achieved; however, the result is overly optimistic. When CWA is

applied, any unknown state is treated as false and the set of completely credible truth results can be traced; while OWA is applied, any unknown state is treated as unknown. In this case, the set of completely credible statements, the set of unknown statements, and the set of false statements can be distinguished.



$$s_1 = s_2 \vee s_3$$
$$s_3 = s_1 \wedge s_4$$
$$= s_3 \wedge s_6$$
$$= s_5 \wedge s_7$$

Figure 5 Sample IDGM sequential schedule

5.4 Summary

We have investigated evaluating conformance statements against complex specifications consisting of requirements with manifold subordinations and dependencies. Our novel contribution is to establish the integrated dependency and goal model (IDGM) to help in setting up a correct conformance statement evaluation schedule. This is based on graph theory and a three-valued logic properly taking into account open world assumption (OWA), closed world assumption (CWA) and stub assumption (SA). We have proven that the serialized evaluation is stable in an isolated testing environment. When a fixed logic assumption strategy is adopted for initiating nodes of deadlocks, the fixpoint location is unique. However, this fixpoint location is always indeterminable since the logic assumptions are not always explicit during the evaluation process. The logic assumption-based evaluation helps to distinguish the fully credible, questionable, and completely non-credible statements.

6 Real life study applications

Services providing geospatial information play an increasingly important role in the Web service landscape. A wide range of information is provided in what we subsume as geo services for the purpose of this research – satellite imagery, in-situ sensor measurements, map data, climate and ocean simulation data, and many more. These are used not only by scientists, but to a large and growing extent by agencies and citizens in general. Concrete tasks range from simple route planning using satellite imagery as backdrop to managing cadastral information to international disaster relief management like in Haiti in 2010 [64]. At least with the last category of use - crisis information and situational awareness - it becomes clear that correctness and reliability are fundamental requirements.

Testing is one important practical means that contributes to assessing to correctness of a service implementation. As usual, services under test are required to undergo some specific verification and validation procedures. However, the complexity of geo services establishes additional problems; one of them, which we address in this work, is related to dependency issues. Such dependencies often already exist within one specification against which a service is to be assessed. Moreover, specifications themselves often refer to other specifications – either because some pre-existing specifications are used, as in the case of relying on standards, or because the specification on hand is itself modularized into different documents between which, consequently, logical relationships exist. Further problems include interface conformance evaluation, test maturity evaluation, global validity evaluation, and the question of how to craft specifications in a way which eases rigid testing. One use case for conformance testing where this becomes particularly apparent is the Web Coverage Service (WCS) 2.0 Interface Standard issued by the Open Geospatial Consortium (OGC).

6.1 OGC Web Coverage Service

A coverage is formally defined by ISO 19123 [53] as a function f: D $\rightarrow$ R where D, the coverage's *domain*, represents the spatiotemporal extent in which coverage values can be queried and R, the coverage's *range*, specifies the values which can be associated with coverage locations. Actually, the coverage definition in ISO 19123 [53] only gives an abstract model and it is further refined to become a practical usable encoding, such as the work which has been done on different GML coverage encodings [13][84][85][67]. The standards make geospatial raster interchangeable across OGC web services and improve the interoperability of geo raster services [27][6][68].

To support retrieval of raster data as a geospatial "coverage" [53], WCS defines three operations to probe information from both metadata and data levels. These are detailed as:

- *GetCapabilities* retrieves an overview which is normally about server functionalities and a list of available coverages as defined in the Service Metadata model [7][69].
- *DescribeCoverage* retrieves a comprehensive description of the offered coverage [67] such as spatial boundaries and attribute descriptions.
- *GetCoverage* retrieves the whole or part of an identified coverage, including both space/time and attribute information. In this way, WCS gives access to multi-dimensional coverage data, rather than rendering images as it is done, e.g., with Google Maps and OGC WMS [29].

The WCS Core [69] relies on the GML Application Schema for Coverages [67] which contains standardized attribute descriptions [5]; the Core itself, which defines service functionalities mandatory for every implementation claiming to conform with the WCS standard, consists of 42 requirements which are often logically connected. Extension standards add further data structures, functionality, or usage requirements. Currently, three extensions have been adopted as standards and about ten more are in the pipeline of the standards working group. Core and extension documents internally

contain requirements classes as modularization units [79]; WCS choose to have, whenever possible, only one requirements class in the Core for reasons of simplicity. Application Profiles, finally, bundle the Core plus a custom set of extensions into packages targeted at particular application domains. The Earth Observation Application Profile candidate standard [75] is an example. Figure 6 gives an overview of the WCS suite structure [68].

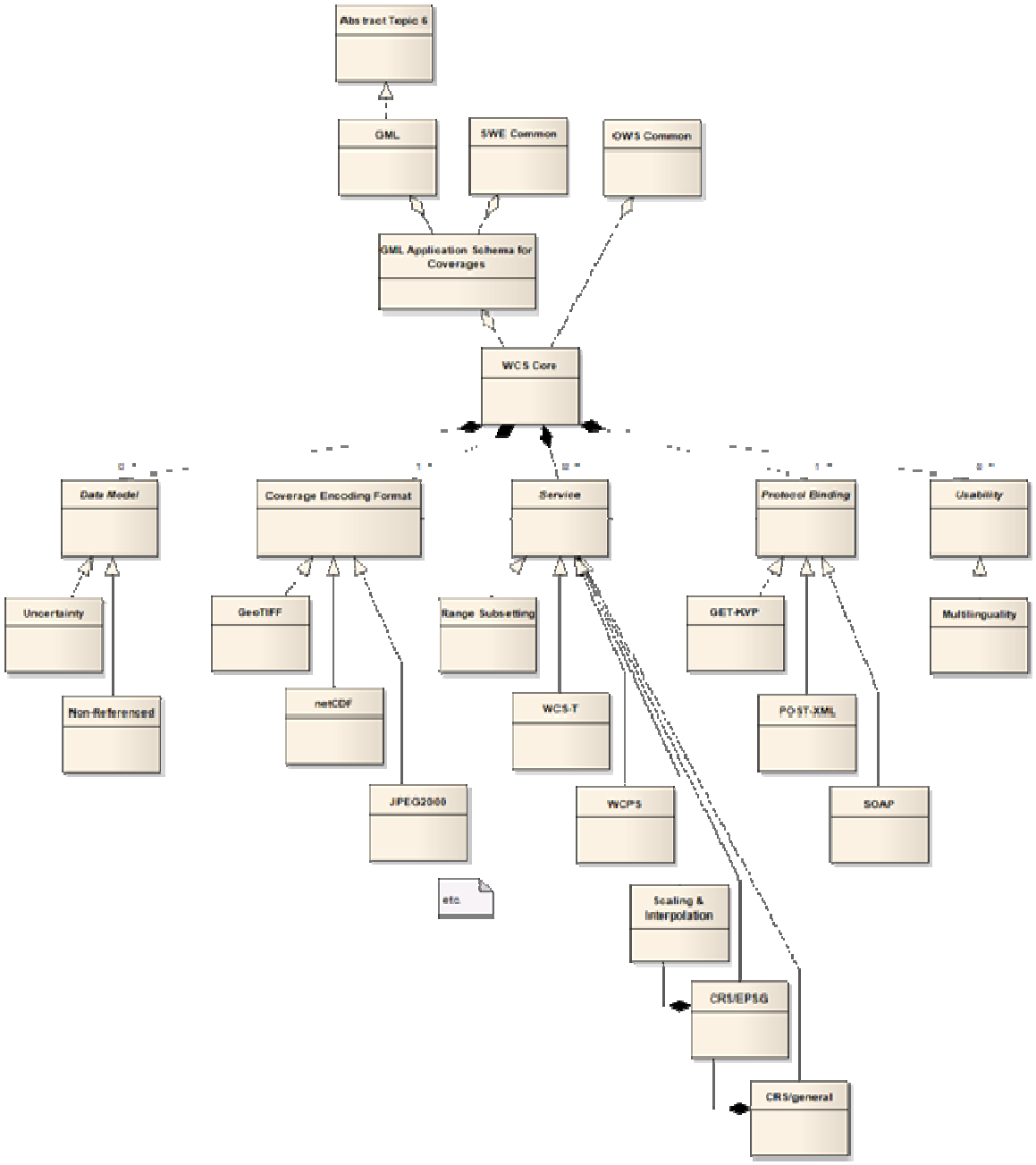


Figure 6 WCS 2.0 Core and extensions [68]

Data model extensions: This category of extensions focuses on extending or refining coverage-related data models, such as “nil” representation and domain models.

Service model extensions: This category of extensions focuses on adding further service capabilities, such as coverage scaling, editing and online processing.

Format encoding extensions: This category of extensions focuses on providing further encodings support for the transfer of coverage, such as HDF, NetCDF and GeoTIFF.

Protocol extensions: This category of extensions focuses on providing client/server communication support through existing internet transformation protocols, such as HTTP GET/KVP, HTTP POST/XML[71] and SOAP[72].

Usability extensions: This category of extensions focuses on usability of the overall service, such as a multi-lingual support.

Every specification document contains an Annex with an Abstract Test Suite (ATS) consisting of one test specification for each requirement. In sync with the requirements classes, these tests are grouped into conformance test classes. These tests are formulated abstractly in the sense that they only give a high-level description; a concrete Executable Test Suite (ETS), which is developed in Compliance Test Language [12], represents the executable counterpart of the ATS. OGC’s Test, Evaluation, and Measurement (TEAM) Engine runs this ETS and tests corresponding implementations.

Any implementation of the WCS 2.0 standard must mandatorily use the Core and can add requirements classes from extensions, thereby achieving an individual, yet interoperable functionality package; alternatively, an implementation can choose to implement one of the pre-packaged Application Profiles. Whatever set of conformance classes a service implements is announced to the client in the response to a *GetCapabilities* request.

WCS 2.0 was the first OGC standard to rigorously follow this so-called Core/Extension paradigm adopted by OGC to achieve more modular specification sets. WCS design, therefore, often had to explore pathways on the best use in face of complex dependencies. The final WCS 2.0 Core was adopted in August 2010 together with a first slate of extensions. Several extension specifications and the application profile mentioned above are currently under work.

Due to the inherent complexity of WCS 2.0, it is excellently suited to look at the problems arising in testing implementations against modular specifications, such as interface compliance, test maturity and the manifold dependencies among requirements.

6.2 Service interface conformance

To evaluate the compliance metrics of service interfaces, it is import to deduce the set of requirements which need to be tested. Core WCS specifies a set of abstract requirements on WCS domain knowledge. This core is a dependency of a set of extensions, such as communication protocols and encoding formats. To test such a core, at least one protocol and one encoding format are required. Furthermore, each protocol needs at least one encoding support and each encoding needs at least one protocol support.

We use $\{p_i\}$ to denote the set of supported communication protocols, $\{f_j\}$ to denote the set of supported encoding formats, c to denote the core, $p_i^{f_j}$ to denote the protocol p_i with encoding format f_j support, $f_j^{p_i}$ to denote the encoding format f_j with protocol p_i support, and c_{p_i,f_j} to denote the core with p_i and f_j supports.

Assumed is that $\{p_i|1 \leq i \leq m\}$ and $\{f_j|1 \leq j \leq n\}$ are the declared supported protocol sets and encoding format sets. Then, each protocol can be tested against the declared formats and each format can be tested against the declared protocols. Respectively, these tests are denoted as $\{test(p_i^{f_j})|1 \leq i \leq m, 1 \leq j \leq n\}$ and

$\{test(f_j^{p_i})|1 \leq i \leq m, 1 \leq j \leq n\}$. Furthermore, the core should be respectively tested by the declared formats and protocols. These are denoted as $\{test(c_{p_i,f_j})|1 \leq i \leq m, 1 \leq j \leq n\}$. We denote the declared protocol and encoding format supports as:

$$\{p_1, p_2, p_3, \dots, p_m\}, \{f_1, f_2, f_3, \dots, f_n\}$$

The conditional statement is denoted as:

$$p_i \wedge f_j \rightarrow test(p_i^{f_j}) \wedge test(f_j^{p_i}) \wedge test(c_{p_i,f_j})$$

Therefore, the adopted tests are denoted as:

$$\{test(p_i^{f_j})|1 \leq i \leq m, 1 \leq j \leq n\},$$

$$\{test(f_j^{p_i})|1 \leq i \leq m, 1 \leq j \leq n\}$$

and $\{test(c_{p_i,f_j})|1 \leq i \leq m, 1 \leq j \leq n\}$

Without considering dependency relationships, the certification processes are denoted as:

$$testresult(p_i^{f_j}, T) \rightarrow certify(p_i^{f_j})$$

$$testresult(f_j^{p_i}, T) \rightarrow certify(f_j^{p_i})$$

$$testresult(c_{p_i,f_j}, T) \rightarrow certify(c_{p_i,f_j})$$

However, to declare global validity, dependencies should be considered. Then, the certification processes should be denoted as:

$$testresult(p_i^{f_j}, T) \wedge testresult(f_j^{p_i}, T) \rightarrow certify(p_i^{f_j})$$

$$testresult(f_j^{p_i}, T) \wedge testresult(p_i^{f_j}, T) \rightarrow certify(f_j^{p_i})$$

$$testresult(c_{p_i,f_j}, T) \wedge testresult(p_i^{f_j}, T) \wedge testresult(f_j^{p_i}, T) \rightarrow certify(c_{p_i,f_j})$$

If $test(c_{p_i,f_j})$ consists of m requirements tests, $test(p_i^{f_j})$ consists of n requirements tests and $test(f_j^{p_i})$ consists of k requirements tests, to certify c_{p_i,f_j}, the number of requirements which need to be implemented is $m + n + k$. Its corresponding compliance metric can be derived accordingly as $x/(m + n + k)$, where x is the number of conformant requirements.

6.3 Test maturity analysis

We address the core WCS2.0 functionalities in this application and evaluate the maturity of the designed tests. This is based on the approach as proposed in Section 3.

In the core WCS 2.0, there are requirements on different functionalities such as general service and data models, service requests, service responses, information coherence, and exception cases. One implementation of the requirements is designating the server as a black box to the test agent which does not have access to source code. In this case, these functionalities can only be validated by the test agent via sending a set of service requests and checking the corresponding responses. Therefore, to derive the set of minimal necessary service test requests which matches the specified requirements, it is of critical importance to design comprehensive tests.

ISO's rules for the structure and drafting of International standards [57] states that "expression in the content of a document conveying criteria to be fulfilled if compliance with the document is to be claimed and from which no deviation is permitted". This kind of statement uses verbal forms, "shall" and "shall not" to indicate requirements that strictly followed in order to conform to the document and from which no deviation is permitted. WCS 2.0 requirements are specified in such normative statements. To aid in the understanding and precise communication of the criteria, further non-normative explanatory presentations are used to paraphrase or supplement requirements, for example, informal natural language statements, tables, semi-formal UML diagrams, XML schemas and XML schematrons. To safeguard against potential internal inconsistencies resulting from design errors, precedence is

set up in the standard [69], specifically, that when multiple representations of the same information are given in a specification document, the rules specified in the machine-oriented XML schema take precedence over all others. However, the employed UML models, XML schemas and XML schematrons are structural and have limited power to express parameter relationships within a service request. Although these constraints can be manually derived from natural language descriptions and applied in test design, the process is not publicly observable and auditable. Hence, the test maturity evaluation is not available. Therefore, we use the approach as stated in Section 3.2 to express parameter relationships and discuss the correspondingly minimal necessary service test requests. We chose two sample requirements to demonstrate this type of evaluation. They are detailed as follows:

- Case 1: OGC 09-110r3 A.1.26

  Requirement: "The id parameter value in a *GetCoverage* request shall be equal to the identifier of one of the coverages offered by the server addressed."

  Abstract Test Suite: "Send valid *GetCoverage* requests to server under test addressing existing and non-existing coverages, resp. Check if appropriate results or exceptions, resp., are delivered."

Table 11 Valid and invalid parameter inputs for the studied case 1

Input	`Service:`*`stubbed`*	`Version:`*`stubbed`*	`id`
Valid	*Stubbed*	*Stubbed*	*id*
Invalid	-	-	*id_bogus*

A minimal valid *GetCoverage* request consists of a `Service` parameter, a `Version` parameter and an `id` parameter, where `Service` is the service type, `Version` is the service version and `id` is the identified coverage. Although auxiliary parameters are not specified in the requirement, these inputs are mandatory and should be valid for testing the specified `id` parameter. Otherwise, the service under

test will respond with an exception even when an existing coverage is identified. Therefore, `Service` and `Version` inputs are stubbed. Accordingly, inputs of `id` are partitioned into the cases of addressing existing and non-existing coverages (see Table 11). The query for generating both valid and invalid service test requests is given as:

STRG-QL: `IR(Service:`*`stubbed`*`, Version:`*`stubbed`*`, id): `*`cardinality`*` = 3`

where `id` parameter is assumed to be independent of `Service` and `Version` parameters. Its search space is given in Table 12.

Table 12 Corresponding service test requests for the Studied Case 1

Search space	`Service:`*`stubbed`*	`Version:`*`stubbed`*	`id`
Valid	*Stubbed*	*Stubbed*	*id*
Invalid	*Stubbed*	*Stubbed*	*id_bogus*

- Case 2: OGC 09-110r3 A.1.28

Requirement: "Every `dimension` value in a *GetCoverage* request shall be equal to one of the `axisLabels` dimension names specified in the `gml:SRSInformationGroup` of the coverage's `gml:Envelope`, unless the server offers a WCS CRS extension which overrides this requirement."

Abstract Test Suite: "If a CRS extension is implemented by the server under a test which overrides this requirement, do nothing. Otherwise, send otherwise valid *GetCoverage* requests with all `dimension` values appearing in the `axisLabel` of the coverage addressed, with some of the `dimension` values appearing there, and with none of the dimension names provided appearing there. Verify that coverage response is returned if and only if `dimension` occurring in the `axisLabel` attribute are used, and an exception is reported otherwise."

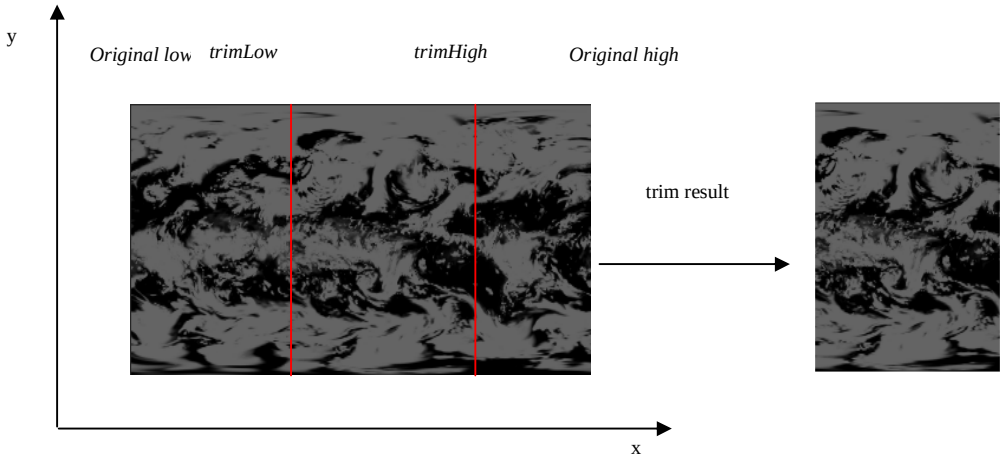


Figure 7 Trim on earth cloud
(source: http://www.earthlook.org/)

A valid *GetCoverage* request without specifying a dimension retrieves the whole coverage; otherwise, a *GetCoverage* request retrieves a partial coverage. Each dimension indicates a *subsetting* operation which may be either a *trim* or *slice*. A valid coverage *trim* (see **Figure 7**) is a geometric cutout in one dimension given by a pair of valid *low* and *high* boundaries. A valid coverage *slice* (see **Figure 8**) is a geometric cutout by a *slicePoint*.

Without a CRS extension which overrides the requirement, these dimension values

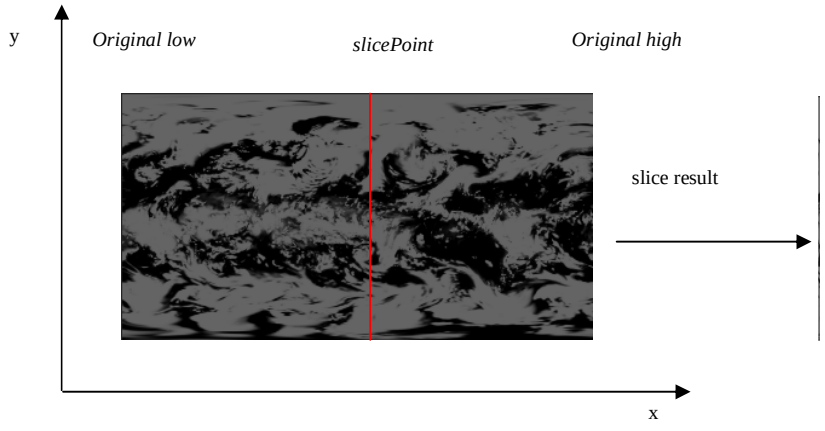


Figure 8 Slice on earth cloud
(source: http://www.earthlook.org/)

are extracted from the `axisLabel` of the coverage addressed. To test such a request, we create inputs for each parameter. Although auxiliary parameters are not specified in the requirement, these inputs are mandatory and should be valid for testing the specified `dimension` parameter. Otherwise, the service under test will respond with an exception even when correct dimensions are identified. Therefore, `Service`, `Version`, `id` and *subsetting* inputs are stubbed. Accordingly, dependence and independent relationships among these inputs are modeled. Inputs of `dimension` parameter are partitioned into the cases of addressing existing and non-existing dimensions (see Table 13). We use d_i to address the coverage `dimension` values. The queries for generating service test requests are given as below:

STRG-QL:

- This query is used for generating the valid *GetCoverage* request with none of the dimension values appearing in the `axisLabel` of the coverage addressed:

```
IR(Service:stubbed,
   Version:stubbed,
   id:stubbed):cardinality = 3
```

where `id` parameter is assumed to be independent of `Service` and `Version` parameters. Its search space is given in Table 14.

- This query is used for generating the valid *GetCoverage* request with some of the `dimension` values appearing in the `axisLabel` of the coverage addressed:

```
IR(Service:stubbed,
   Version:stubbed,
   MD(
      id:stubbed,
```

```
IR(MD(d1:stubbed,
      IR(
         MD(
             TrimL1:stubbed,
             TrimH1:stubbed
             ):stubbed,
         SlicePoint1:stubbed
        ):cardinality = 1
      ):stubbed,
   MD(d2:stubbed,
      IR(
         MD(
             TrimL2:stubbed,
             TrimH2:stubbed
             ):stubbed,
         SlicePoint2:stubbed
        ): cardinality = 1
      ):stubbed,
    …,
   MD(dn:stubbed,
      IR(
         MD(
             TrimLn:stubbed,
             TrimHn:stubbed
```

```
                    ):stubbed,
                SlicePoint_n:stubbed
               ):cardinality = 1
            ):stubbed,
         ):cardinality = random(1:n-1)
      ):stubbed
   ):cardinality = 3
```

where `id` parameter is assumed to be independent of `Service` and `Version` parameters, `id` parameter is assumed to be the dependency of dimension *subsettings*, dimension *subsettings* are assumed to be independent from each other on each dimension, `dimension` parameter is assumed to be the dependency of the *trim* or *slice* input and `TrimL` parameter is assumed to be the dependency of `TrimH`. *`cardinality=random`*`(1:`*`n`*`-1)` means the number of dimension *subsettings* is less than *`n`*`-1` but larger than `1`. Its search space is given in Table 14.

- This query is used for generating the valid *GetCoverage* request with all `dimension` values appearing in the `axisLabel` of the coverage addressed:

```
IR(Service:stubbed,
   Version:stubbed,
   MD(
      id:stubbed,
      IR(MD(d_1:stubbed,
            IR(
               MD(
                  TrimL_1:stubbed,
```

```
            TrimH1:stubbed
            ):stubbed,
        SlicePoint1:stubbed
       ):cardinality = 1
    ):stubbed,
MD(d2:stubbed,
    IR(
        MD(
            TrimL2:stubbed,
            TrimH2:stubbed
            ):stubbed,
        SlicePoint2:stubbed
       ):cardinality = 1
    ):stubbed,
 …,
MD(dn:stubbed,
    IR(
        MD(
            TrimLn:stubbed,
            TrimHn:stubbed
            ):stubbed,
        SlicePointn:stubbed
       ):cardinality = 1
    ):stubbed,
```

```
      ):cardinality = n
    ):stubbed
  ):cardinality = 3
```

where `id` parameter is assumed to be independent of `Service` and `Version` parameters, `id` parameter is assumed to be the dependency of dimension *subsettings*, dimension *subsettings* are assumed to be independent from each other on each dimension, `dimension` parameter is assumed to be the dependency of the *trim* or *slice* input and `TrimL` parameter is assumed to be the dependency of `TrimH` parameter. *`Cardinality = n`* means the number of dimension *subsettings* is *n*. Its search space is given in Table 14.

- The query for generating the invalid *GetCoverage* request with some of the `dimension` values not appearing in the `axisLabel` of the coverage addressed is given as below:

```
IR(Service:stubbed,
  Version:stubbed,
  MD(
    id:stubbed,
    IR(IR(d1,
          IR(
            MD(
              TrimL1:stubbed,
              TrimH1:stubbed
              ):stubbed,
            SlicePoint1:stubbed
           ):cardinality = 1
```

```
        ):cardinality = 2,
    IR(d2,
        IR(
            MD(
                TrimL2:stubbed,
                TrimH2:stubbed
                ):stubbed,
            SlicePoint2:stubbed
           ):cardinality = 1
        ):cardinality = 2,
     …,
    IR (dn,
        IR(
            MD(
                TrimLn:stubbed,
                TrimHn:stubbed
                ):stubbed,
            SlicePointn:stubbed
           ):cardinality = 1
        ):cardinality = 2,
      ):cardinality = random(1:n)
    ):random_invalid
  ):cardinality = 3
```

where `id` parameter is assumed to be independent of `Service` and `Version`

parameters, `id` parameter is assumed to be the dependency of dimension *subsettings*, dimension *subsettings* are assumed to be independent from each other on each dimension, `dimension` parameter is assumed to be independent of the *trim* or *slice* input and `TrimL` parameter is assumed to be the dependency of `TrimH` parameter. *`Cardinality`* `=` *`random`*`(1:`*`n`*`)` means the number of dimension *subsettings* is less than *`n`* but larger than `1`. *`random_invalid`* means only one random invalid request is adopted. Its search space is given in Table 14.

Table 13 Valid and invalid parameter inputs for the studied case 2

Input	`Service`	`Version`	`id`	d_1	$TrimL_1$	$TrimH_1$
Valid	*Stubbed*	*Stubbed*	*stubbed*	d_1	*stubbed*	*stubbed*
Invalid	-	-	-	*d_bogus*	-	-

Input	$SlicePoint_1$	d_2	$TrimL_2$	$TrimH_2$	$SlicePoint_2$
Valid	*Stubbed*	d_2	*stubbed*	*stubbed*	*Stubbed*
Invalid	-	*d_bogus*	-	-	-

Input	…	d_n, when n>=2	$TrimL_n$	$TrimH_n$	$SlicePoint_n$
Valid	…	d_n	*stubbed*	*stubbed*	*stubbed*
Invalid	…	*d_bogus*	-	-	-

Table 14 Corresponding service test requests for the studied case 2

Search space	*Service*	*Version*	*id*	*dimensions*	(*TrimL*, *TrimH*) or *SlicePoint*
Valid	*stubbed*	*stubbed*	*stubbed*	-	-
Valid	*stubbed*	*stubbed*	*stubbed*	*some valid dimension values*	*stubbed*
Valid	*stubbed*	*stubbed*	*stubbed*	*all valid dimension values*	*stubbed*
Invalid	*stubbed*	*stubbed*	*stubbed*	*a list of dimensions with one invalid value*	*stubbed*

To evaluate the test request maturity we designed, we used STRG-QL results which meet the specified criteria according to the specifications studied and compared them with hard-coded, random, and exhaustive approaches.

The hard-coded approach sends a constant number of requests with hard-coded parameters to the service entries. The random approach sends a constant number of requests with random valued parameters to the service entries and the exhaustive approach sends requests of exponential complexity to the service entries.

To assess the maturity metrics [59] of the requests considered, we distinguish three maturity levels:

- *immature* means the designed test requests are always less than needed;
- *over-mature* means the designed test requests meet all specified criteria, but

are more than needed;

- *fitting* means the designed test requests meet all specified criteria and the number of requests is minimal.

If c_1 is the number of valid requests of the hard-coded approach, c_2 is the number of invalid requests of the hard-coded approach, r_1 is the number of valid requests of the random approach, r_2 is the number of invalid requests of the random approach, n is the number of atomic parameters, x_i is the number of inputs for each atomic parameter, v_1 is the number of valid requests of STRG-QL results, and v_2 is the number of invalid requests of STRG-QL results, we compared the numbers of generated tests of these approaches in Table 15.

Table 15 Comparison on the numbers of generated test requests

Approach / EQC	*Hard-coded*	*Random approach*	*Exhaustive approach*	*RPRA approach*
Valid	c_1	r_1	$\prod_i^n x_i$	v_1
Invalid	c_2	r_2		v_2

The hard-coded inputs may match the specified criteria. However, it is easy to create false service messages to pass the test. Random approach makes such forged activities impossible. However, such a testing is either immature when only a small number of requests are generated or over-mature when a large number of requests are generated. By coincidence, the designed tests match the requirement exactly. Exhaustive approach is always over mature due to trivial inputs. The fitting case is hard to achieve. Request Parameter Relationship Analysis (RPRA) uses STRG-QL to generate a search space which contains the minimally necessary requests which, at the same time, are specification consistent.

6.4 Global statement validity

6.4.1 Dependencies among requirements and requirement modules

We derive 103 conformance statements from the WCS Core [69], the GML Application Schema for Coverages [67] and the three protocol extensions [70][71][72]. In these specifications, beyond the explicit dependencies among requirements, there are some implicit dependencies. As stated in the specification model [79] of OGC, a requirements class is an aggregation of all requirement modules that must all be satisfied to satisfy the corresponding conformance test class; a requirements module is an aggregation of requirements and recommendations of a specification. Therefore, a goal which declares conformance of a **requirements class**, contains subgoals which are to declare conformance of its corresponding requirement modules. A goal which declares conformance of a **requirements module**, contains subgoals which are to declare conformance of its corresponding requirements. Recommendations, which consist of non-testable and non-mandatory statements, are not considered in this study case.

There are explicit and implicit dependencies among requirements. These include dependencies between requirements of service requests and responses, dependencies between service contents and their models and dependencies between service operations and their communication protocols. Dependencies are considered as statement subordinations. We model these by IDGM. Firstly, we derive atomic conformance statements for each requirement; secondly, we derive composite conformance statements for each requirement module and each requirement class; thirdly, we treat each conformance statement as a test goal and derive well-designed goals by integrating logical subordinations (See Figure 9 and Annex B.1 for the results).

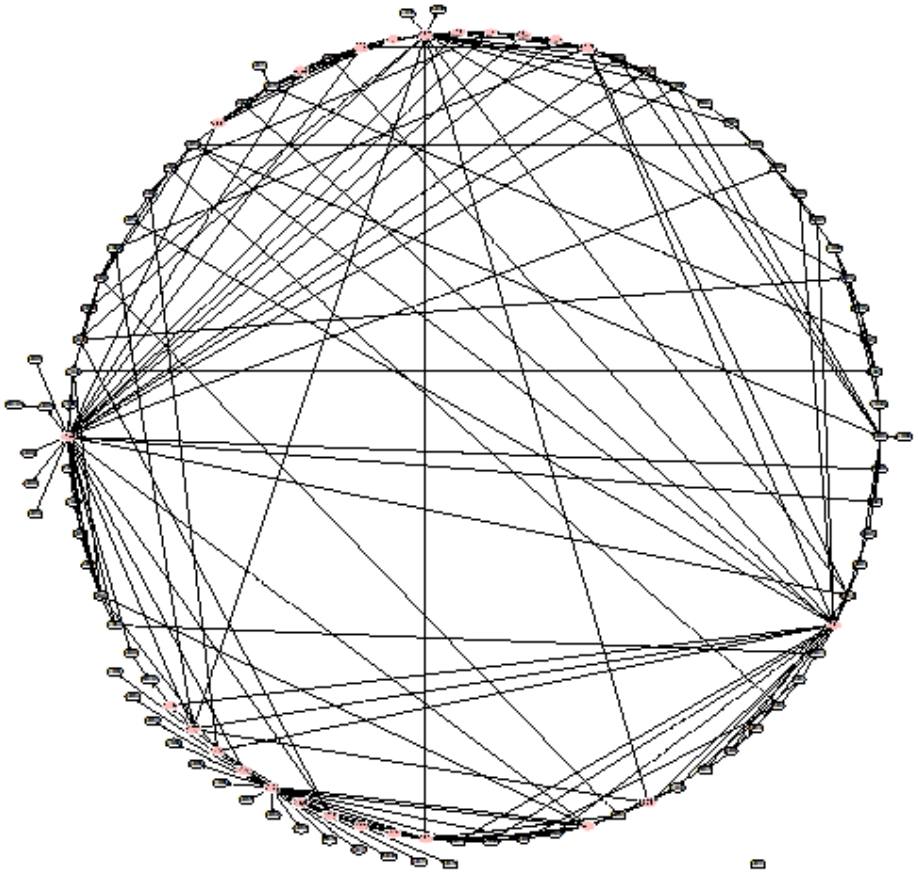


Figure 9 IDGM of the study case

6.4.2 A sequential evaluation schedule

We condense the non-singular strongly-connected components to vertices in the IDGM and apply the schedule algorithm on the derived DAG (see Figure 11 Annex B.2 for their logical relationships). To detect these strongly-connected components, we apply the Kosaraju algorithm [89] in the IDGM. There are two non-singular strongly-connected components found in the IDGM (see Figure 10 and Table 16).

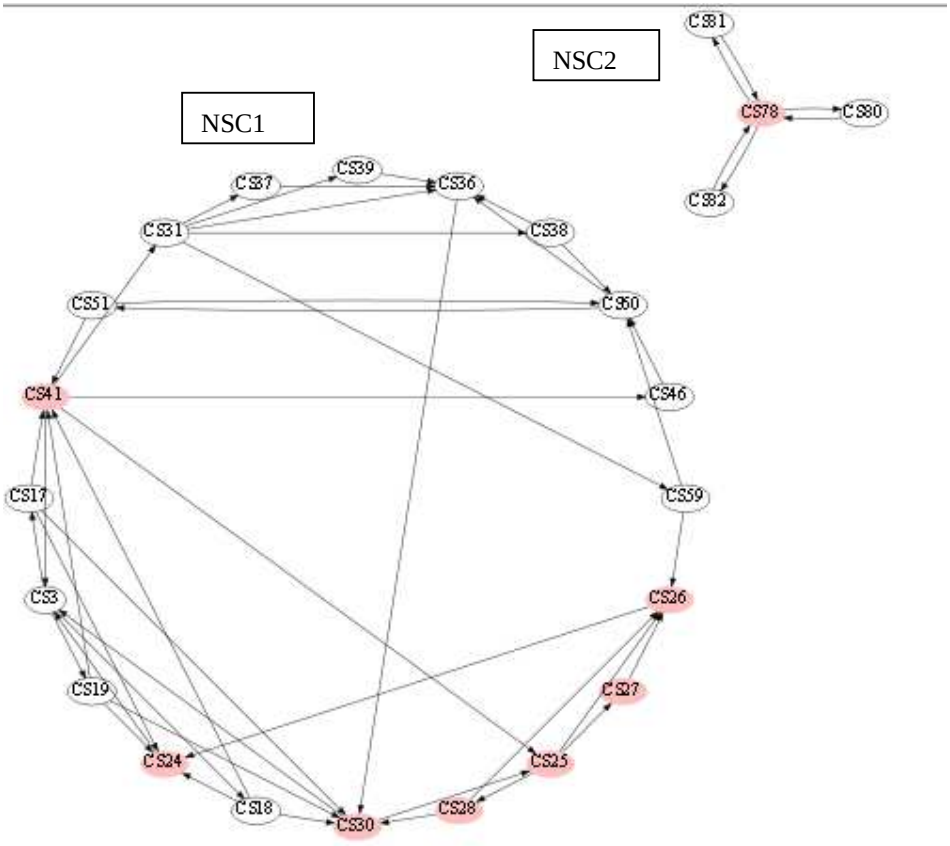


Figure 10 Non-singular strongly-connected components (NSC) of the studied case

Table 16 Non-singular strongly-connected components (NSC) of the studied case

NSC	Vertices
1	19, 18, 46, 59, 39, 51, 60, 38, 37, 36, 31, **41**, **28**, **27**, **25**, **30**, 17, 3, **24**, **26**
2	82 81 80 **78**

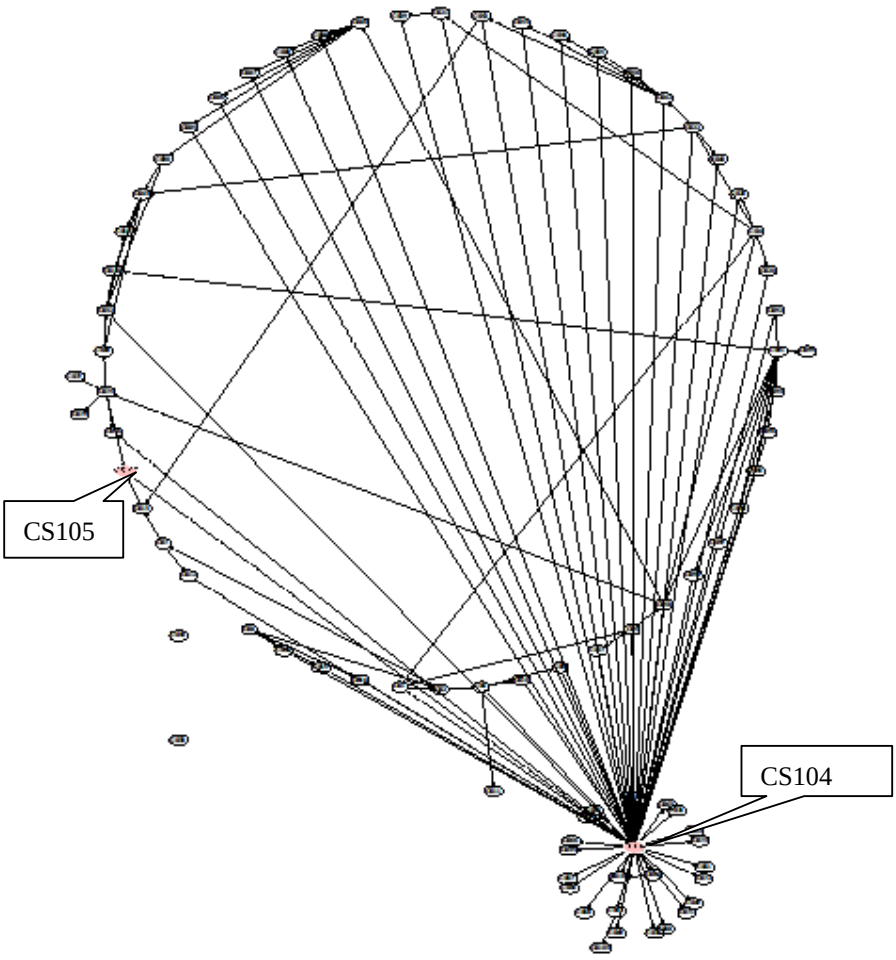
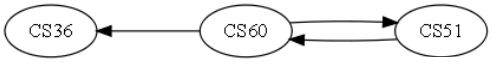


Figure 11 Condensed DAG of the study case

To disentangle the non-singular strongly-connected components, we apply the domain knowledge to initiate the starting vertices. For example, conformance statements about requests of the three respective WCS operations and WCS service capabilities should be evaluated before evaluating further conformance statements. In this way, we sequentialize the schedule. Firstly, we initiate the corresponding conformance statements 24, 25, 26, 27, 28, 30 and 41 in NSC1 (see Figure 12). Secondly, we disentangle the NSC1 according to the possible truth result expression in Section 5.4.2. Then, we find that statements of 51 and 60 form a false deadlock when CS36 is evaluated as *true* (see Figure 12). Therefore, we initiate statement 60 and schedule it ahead of statement 51. In NSC2, we initiate statement 78 as its outdegree is larger than statement 80, 81 and 82. The result is shown in Table 17.



$CS60 = \mathrm{CS36} \vee CS51$

Figure 12 False deadlock of the study case

Table 17 Topological sorting results

	Output
DAG	101 102 103 11 29 34 35 40 45 48 49 61 62 63 64 65 66 67 68 69 70 71 72 73 74 77 79 99 44 43 47 **104 105** 98 97 96 95 94 93 92 89 88 87 86 85 76 57 56 55 54 53 32 23 21 20 16 15 14 12 10 9 13 100 90 83 8 6 52 7 91 84 75 42 2 58 22 5 4 1
NSC1(104)	24 25 26 27 28 30 41 **36** 19 18 17 **60** 39 37 3 59 **51** 46 38 31
NSC2(105)	78 80 81 82

6.4.3 Evaluate global statement validity based on 3A-TRE

We implement 3A-TRE language to evaluate validity of global statements (see Annex A). For example, conformance statement CS1 is evaluated by

OWA: CS2 ∧ CS3 ∧ CS4 ∧ CS5

and CS2 is evaluated by

CWA: CS6 ∧ CS7 ∧ CS8

Then, CS1 is evaluated by

OWA: (CWA: CS6 ∧ CS7 ∧ CS8) ∧ (OWA: CS3 ∧ CS4 ∧ CS5)

An iteration process on compound statements will find all statements which are necessary to be evaluated. If the actual derived statement result number is *m* and the necessary result number is *n*, the audit trail capability can be measured by *m*/*n*.

6.4.4 A statement validity evaluation case

As a thread of OGC Web Services, Phase 8 (OWS-8) Interoperability Initiative

activity, Observation Fusion thread combines the OGC WCS 2.0 standard and architecture with the results of the recent OGC development of WCS 2.0 ATS and ETS. Meanwhile, Rasdaman, the Scalable Multi-Dimensional Array Analytics Server, implements this standard. The development of the test suite follows a cross-testing approach. The implementation and the test suite, which are derived from the same standards suite, are developed in parallel and promote each other in a conformance testing round (see Figure 12).

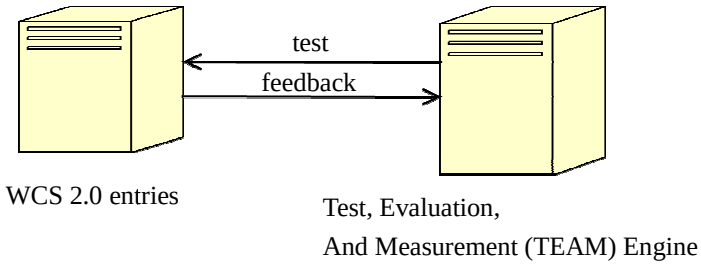


Figure 13 A conformance testing round example

In a round of conformance testing, the test tests whether the implementation is conformant with the declared requirements. Test results are used to instantiate the modeled IDGM, in which each statement is valued as T, U or F. Then, statement validity evaluation is carried out based on this IDGM. Test results and evaluation results are visualized by Graphviz, T as green, U as yellow and F as red. We use one of the conformance testing rounds to illustrate this application. Test results of this round are shown in Figure 14 .

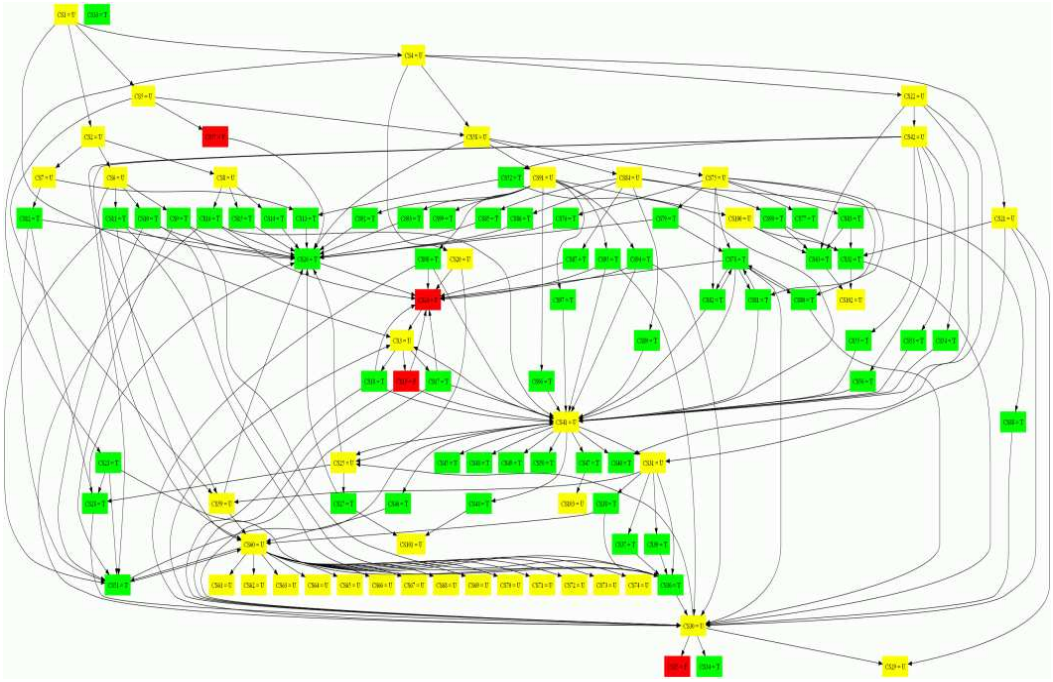


Figure 14 Test results of a conformance testing round

CWA, OWA and SA evaluations are carried out based on the sequential evaluation schedule algorithm (see Annex C). The results are specifically shown in Figure 15, Figure 15 **and** Figure 17. The fully credible, questionable, and completely non-credible statements are distinguished accordingly.

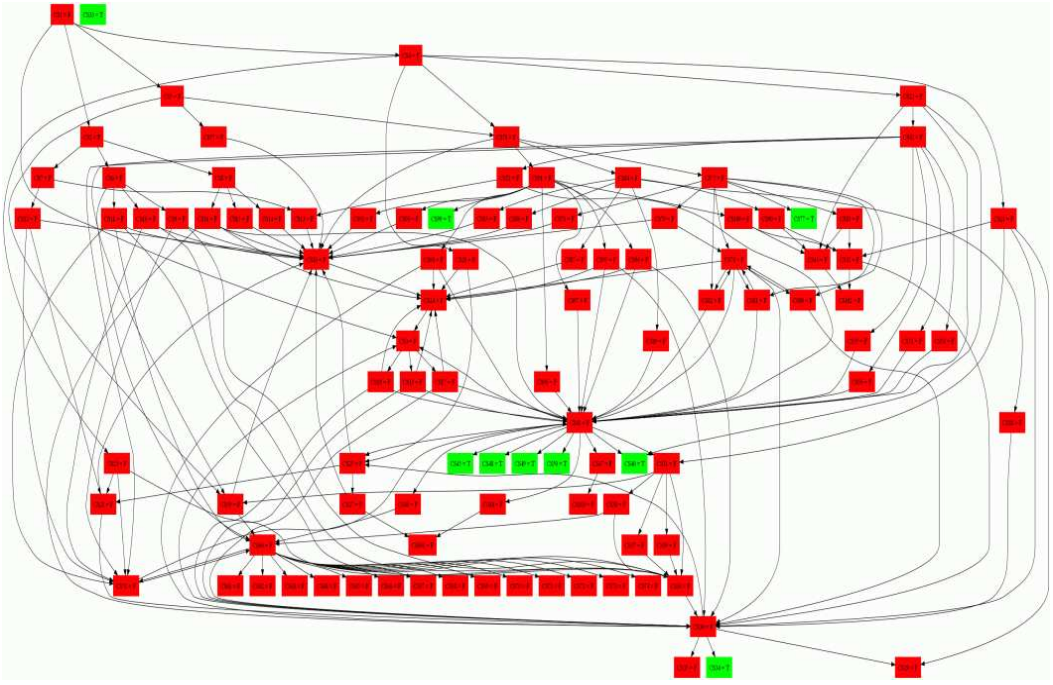


Figure 16 CWA evaluation result

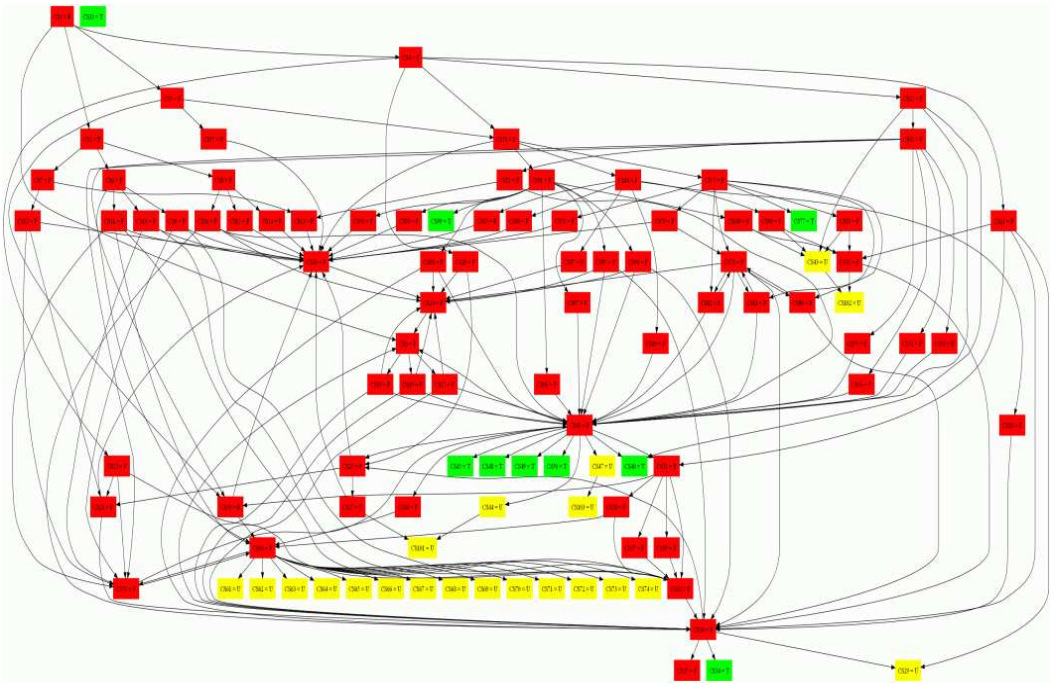


Figure 16 OWA evaluation result

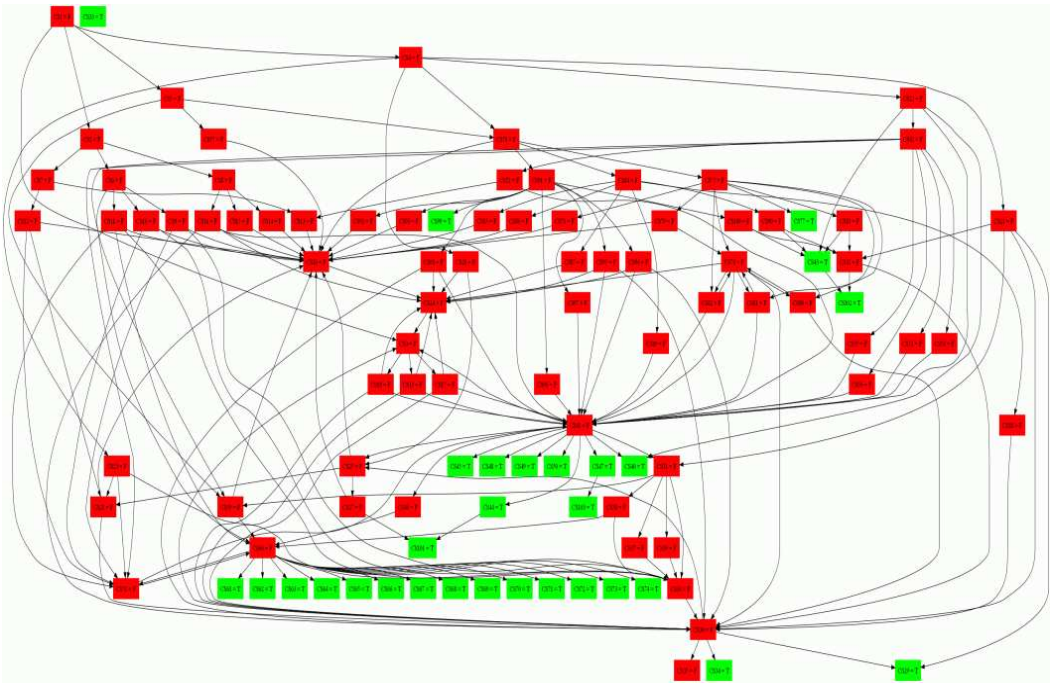


Figure 17 SA evaluation result

6.5 Summary

Our study example is geo service standardization, specifically: geo raster services based on OGC WCS 2.0. The Executable Test Suite (ETS) which has become the conformance test suite of WCS in OGC's Compliance and Interoperability Testing Initiative (CITE) program in 2011 as decided by OGC. The theoretical results described in the earlier sections form the basis to generate the evaluation schedule of the validity of the corresponding conformance statements. The fixpoint in Section 5.4.1 proves that it is safe to have mutual dependencies among requirements with proper evaluation schedule. However, these mutual dependencies may result in different fixpoint locations according to different initial statements. These may introduce extra testing and implementation work when truth values of dependencies are reversed. Therefore, it is important to have domain experts involved and indentify these initial statements. When the corresponding starting statements are stable, the following works, implementations, testing or specification revisions can continue as scheduled. When service implementations and test suites are fixed, the evaluation schedule can help to distinguish the fully credible, questionable and completely non-credible statements. A completely non-credible statement means the corresponding service implementation and test suite need to be coherent; a questionable statement means that the corresponding requirement or test suite is not complete and needs further works to cover the untouched constraints; and a fully credible statement means the corresponding requirement, service implementation and test suite are coherent. Obviously, conformance statements generation is important in this process. This requires that the corresponding well-designed goals, which are modeled by domain experts, should match the specified requirements and their corresponding modules and dependencies.

7 Conclusions and contribution

We have investigated testing Web services against complex specifications consisting of requirements with manifold dependencies among them. Here are our novel contributions.

- A specification-based quality model has been derived to address service interface compliance, test maturity and global conformance statement validity. Functional compliance is used to measure the compliance items of the service interfaces. To measure the maturity of the designed tests, a metric of test coverage based on RPRA is proposed to evaluate the test coverage of the design tests. To analyze the test results, 3A-TRE, a three-valued logic expression, was modeled to evaluate the validity based on the proposed integrated dependency and goal model (IDGM).
- Based on the modeled parameter relationships, a RPRA-based search space generation approach is proposed to derive minimal necessary and specification-consistent service test requests. The approach is also applied to evaluate the maturity of the designed tests.
- Based on the oracle study, a deductive reasoning approach is proposed to address reference output adoption.
- Proof has been provided, based on fixpoint theory, that the serialized evaluation is stable in an isolated testing environment. When a fixed logic assumption strategy is adopted for initiating nodes of deadlocks, the fixpoint location is unique. However, this fixpoint location is always indeterminable since the logic assumptions are not always explicit during the evaluation process. The evaluation helps to distinguish the fully credible, questionable and completely non-credible statements.
- A method has been established to create a correct sequence for checking the conformance statements of a given specification. This is based on a three-valued logic, properly taking into account open world assumption (OWA), closed world assumption (CWA) and stub assumption (SA).

We feel, however, that these results transcend geospatial service testing and allow improved and more efficient Web service testing in general based on specifications which make their interdependencies explicit. From this perspective, our research also considers the question of how to craft specifications – and, hence, standards – in a way which eases rigid testing. Request Parameter Relationship Analysis (RPRA) helps to identify relationships among parameters and can be used to generate specification-consistent test requests. The proposed deductive reasoning approach, which helps to adopt suitable reference outputs according to the corresponding service facts, should be adopted to certify proper service capabilities. Integrated dependency and goal model (IDGM), which identifies well-designed goals and provides global validity conformance statements, should be used to model relationships among requirements and their corresponding modules. The derived quality model can be used to assess these approaches. However, there is still work to be done for an even more trustworthy interoperable service environment, for example, semantic-based parameter relationship extraction, systematic reference outputs ordering and concurrent evaluation of multi-source conformance statements under a distributed testing environment.

References

[1] A. Belinfante, L. Frantzen and C. Schallhart, Tools for Test Case Generation. In Model-Based Testing of Reactive Systems (LNCS 3472), M. Broy, B. Jonsson, J. Katoen, M. Leucker, A. Pretschner, Eds. Berlin Heidelberg: Springer-Verlag, pp. 391-438 (2005)

[2] A.G. Gutierrez and P. Baumann, Modeling Fundamental Geo-Raster Operations with Array Algebra. In *Proceedings of the Seventh IEEE International Conference on Data Mining Workshops* (ICDMW '07). IEEE Computer Society, Washington, DC, USA, 607-612. DOI=10.1109/ICDMW.2007.66 http://dx.doi.org/10.1109/ICDMW.2007.66 (2007)

[3] A. Lerner and D. Shasha, AQuery: query language for ordered data, optimization techniques, and experiments. In *Proceedings of the 29th international conference on Very large data bases - Volume 29* (VLDB '2003), Freytag J.C, Lockemann P.C, Abiteboul S, Carey M.J, Selinger P.G, and Heuer A (Eds.), Vol. 29. VLDB Endowment 345-356 (2003)

[4] A.P. Marathe and K. Salem, A Language for Manipulating Arrays. In *Proceedings of the 23rd International Conference on Very Large Data Bases* (VLDB '97), M. Jarke, M.J. Carey, K.R. Dittrich, F.H. Lochovsky, P. Loucopoulos, and M.A. Jeusfeld (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 46-55 (1997)

[5] A. Robin (ed.), OGC SWE Common Data Model Encoding Standard, OGC 08-094 (2010)

[6] A. Whiteside and J.D. Evans (eds.), OGC Coverage Service (WCS) Implementation Standard, OGC 07-067r5 (2007)

[7] A. Whiteside and J. Greenwood, OGC Web Services Common(OWS) Standard, OGC 06-121r9 (2010)

[8] A. van Ballegooij, Ram: A multidimensional array dbms. In EDBT Workshops, Lindner W, Mesiti M, Türker C, Tzitzikas Y, and Vakali A(eds.). Volume3268 of Lecture Notes in Computer Science, pages 154–165. Springer-Verlag, London, UK (2004)

[9] B.A. Davey and H.A. Priestley, Introduction to Lattices and Order. Cambridge University Press, Cambridge (2005)

[10] B. Howe and M. Maier, Algebraic manipulation of scientific datasets. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30* (VLDB '04), Nascimento M.A, Özsu M.T, Kossmann D, Miller R.J, Blakeley J.A, and Schiefer K.B(Eds.), Vol. 30. VLDB Endowment 924-935 (2004)

[11] B.M. Smith, I.F. Rossi, P.van Beek, and T. Walsh (eds.), *Handbook of*

Constraint Programming, Chapter 11, pages 377-406. Elsevier (2006)

[12] C. Morris, OGC Compliance Test Language (CTL) Best Practice, OGC 06-126r2 (2006)

[13] C. Portele (ed.), OpenGIS Geography Markup Language (GML) Encoding Standard, OGC 07-036r1(2007)

[14] C.V. Damásio, A. Analyti, G. Antoniou, and G. Wagner, Supporting open and closed world reasoning on the Web. In Proceedings of the 4th International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR). 149-163 (2006)

[15] D.C. Kung, J. Gao, P. Hsia, Y. Toyoshima, and C. Chen, On Regression Testing of Object-Oriented Programs, The Journal of Systems and Software. Vol.32(1), 21-40 (1996)

[16] D. Coppit and J. Haddox-Schatz, On the Use of Specification-based Assertions as Test Oracles, Proc. SEW '05 Proceedings of the 29th Annual IEEE/NASA on Software Engineering Workshop, pp 305-314 (2005)

[17] D. Hoffman, A taxonomy for test oracles, QualityWeek ’98, Available: www.softwarequalitymethods.com/Papers/OracleTax.pdf (1998)

[18] D. Richardson, S. Leif Aha, and T. O'Malley, Specification-based Test Oracles for Reactive Systems, Proc. 14th Int'l. Conf.on Software Engineering, pp. 105–118 (1992)

[19] D. Zwillinger (ed.), Affine Transformations, §4.3.2 in CRC Standard Mathematical Tables and Formulae. Boca Raton, FL: CRC Press, pp. 265-266 (1995)

[20] E. Bin, R. Emek, G. Shurek, A. Ziv, Using a constraint satisfaction formulation and solution techniques for random test program generation. In IBM Systems Journal, IBM Corp. Riverton, NJ, USA, vol. 41, no. 3, pp. 386-402 (2002)

[21] E.W. Dijkstra, Notes on Structured Programming. In: Structured Programming by O.J. Dahl, E.W. Dijkstra and C.A.R. Hoare, *Academic Press, London* (1972)

[22] F. Echenique, A Short and Constructive Proof of Tarski’s Fixed-point Theorem. Int J Game Theory 33: 215–218 (2005)

[23] G. Birkhoff, Lattice Theory, 3rd ed, Vol. 25 of AMS Colloquium Publications. American Mathematical Society (1967)

[24] G. Fraser, F. Wotawa and P.E. Ammann, Testing with model checkers: a survey, in Software Testing, Verification and Reliability, New York: Wiley, pp. 215–261 (2009)

[25] H. M. MacNeille, Partially Ordered Sets, Transactions of the American Mathematical Society, Vol. 42, No. 3, pp. 416–460 (1937)

[26] I. Sommerville, Software Engineering (8th Edition), Addison-Wesley. ISBN 0-321-31379-8 (2007)

[27] J.D. Evans (ed.), OGC Web Coverage Service (WCS), OGC 03-065r6 (2003)

[28] J. Dong, UML Extensions for Design Pattern Compositions. In: *Journal of Object Technology, 1(5), pp. 151-163.*(2002)

[29] Jeff de la Beaujardiere (ed.), OGC Web Map Service (WMS) Implementation Specification, OGC 06-042r2 (2006)

[30] J. Garcia-Duque, M. Lopez-Nores, J. Pazos-Arias, A. Fernandez-Vilas, R. Diaz-Redondo, A. Gil-Solla, Y. Blanco-Fernandez, M. Ramos-Cabrer: A Six-valued Logic to Reason about Uncertainty and Inconsistency in Requirements Specifications. Journal of Logic and Computation 16(2), 227–255 (2006)

[31] J. Goodwin, An improved algorithm for non-monotonic dependency net update, Research report MAT-R-82-23, Linköping university (SE) (1982)

[32] J. Goodwin, A theory and system for non monotonic reasoning, Linköping studies in science and technology 165 (1987)

[33] J. Tekli, R.Chbeir and K.Yétongnon, An overview on XML similarity: Background, current trends and future directions. Computer Science Review 3(3): 151-173 (2009)

[34] J.-Y. Béziau, What is many-valued logic? Proceedings of the 27th International Symposium on Multiple-Valued Logic. IEEE Computer Society, Los Alamitos, 117–121 (1997)

[35] J. Yu, P. Baumann and X. Wang, RPRA: A Novel Approach to Mastering Geospatial Web Service Testing Complexity, ICSDM2011&IWGIS2011 (2011)

[36] J. Yu and P. Baumann, On the Systematic Generation of Reference Output for Geo Image Services, 19th International Conference on Geoinformatics (2011)

[37] K. Stenning and J. Oberlander, A Cognitive Theory of Graphical and LinguisticReasoning: Logic and Implementation. Cognitive Science, 19, pp. 97-140 (1995)

[38] K. Tai and F. Daniels, Interclass test order for object-oriented software, J. Object-oriented Prog., 18-35 (1999)

[39] L. Libkin, R. Machlin, and L. Wong, A query language for multidimensional

arrays: design, implementation, and optimization techniques. *SIGMOD Rec.* 25, 2 (June 1996), 228-239. DOI=10.1145/235968.233335 http://doi.acm.org/10.1145/235968.233335 (1996)

[40] M. Bauer, and P. Johnson-Laird, How Diagrams Can Improve Reasoning, Psychological Science, 4, pp.372-378 (1993)

[41] M. Belguidoum, F. Dagnat, Dependency Management in Software Component Deployment. FACS 2006. Electr. Notes Theor. Comput. Sci. 17-32 (2007)

[42] M. Fitting, Kleene's Logic, Generalized. Journal of Logic and Computation 1(6), 797–810 (1991)

[43] M. Fitting, Kleene's Three-valued Logics and Their Children. Fundamenta Informaticae 20, 113–131 (1994)

[44] M. Forsberg, A. Ranta, The User Manual for BNF Converter, available at: http://www.cse.chalmers.se/alumni/markus/BNFC/LBNF-report.pdf (2005)

[45] M.L. Prazen, Web services testing, *Web Service*s (2006)

[46] M.M. Dagli, Modus Ponens, Modus Tollens, and Likeness, retrieved March 20, 2011, available at: http://www.bu.edu/wcp/Papers/Logi/LogiDagl.htm (2011)

[47] N.A. Kraft, E.L. Lloyd, B.A. Malloy and P.J. Clarke, The implementation of an extensible system for comparison and visualization of class ordering methodologies. J. Syst. Softw. 79(8), 1092-1109 (2006)

[48] N. Kosmatov, B. Legeard, F. Peureux, and M. Utting, Boundary Coverage Criteria forTest Generation from Formal Models, ISSRE '04, pp. 139–150. IEEE (2004)

[49] n.n, Hypertext Transfer Protocol — HTTP/1.1, IETF RFC 2616 (1999)

[50] n.n, Geographic information — Conformance and Testing, ISO 19105:2000 (2000)

[51] n.n, Geographic information — Spatial referencing by coordinates (OGC Abstract Specification Topic 2, Spatial referencing by coordinates) ISO 19111:2003 (2003)

[52] n.n, Geographic information — Metadata, 19115:2003 (2003)

[53] n.n, Geographic Information - Schema for Coverage Geometry and Functions, ISO 19123:2005 (2005)

[54] n.n, Information technology — Open Systems Interconnection — Conformance testing methodology and framework — Part 1: General concepts,

ISO/IEC 9646-1:1994 (1994)

[55] n.n, Industrial automation systems and integration — Product data representation and exchange — Part 31: Conformance testing methodology and framework: General concepts, ISO 10303-31:1994 (1994)

[56] n.n, Information technology — Computer graphics and image processing -- Conformance testing of implementations of graphics standards, ISO/IEC 10641:1993 (1993)

[57] n.n, Rules for the structure and drafting of International Standards, ISO Directives Part 2, (2004)

[58] n.n, Software engineering — Product quality — Part 1: Quality model, ISO/IEC 9126-1:2001 (2001)

[59] n.n, Software engineering — Product quality — Part 2: External metrics, ISO/IEC TR 9126-2:2003 (2003)

[60] n.n, Software engineering — Product quality — Part 3: Internal metrics, ISO/IEC TR 9126-3:2003 (2003)

[61] n.n, Software engineering — Product quality — Part 4: Quality in use metrics, ISO/IEC TR 9126-4:2004 (2004)

[62] N. Pippenger, The shortest disjunctive normal form of a random Boolean function. Random Struct. Algorithms, 22(2), 161-186.(2003)

[63] N. Sangal, E. Jordan, V. Sinha, D. Jackson, Using dependency models to manage complex software architecture. In OOPSLA, 167-176 (2005)

[64] OGC, http://www.ogcnetwork.net/networks/haiti (2010)

[65] P. Baumann, A Database Array Algebra for Spatio-Temporal Data and Beyond. *The Fourth International Workshop on Next Generation Information Technologies and Systems (NGITS '99), July 5-7, 1999, Zikhron Yaakov, Israel, Lecture Notes on Computer Science 1649, Springer Verlag, pp. 76 – 93* (1999)

[66] P. Baumann, Beyond Rasters: Introducing The New OGC Web Coverage Service 2.0, Proc. ACM SIGSPATIAL GIS, pp. 320 - 329 (2010)

[67] P. Baumann (ed.), OGC GML Application Schema – Coverages, OGC 19-146r1 (2010)

[68] P. Baumann (ed.), WCS 2.0 Overview: Core and Extensions, OGC 19-153 (2010)

[69] P. Baumann (ed.), OGC WCS 2.0 Interface Standard - Core, OGC 19-110r3 (2010)

[70] P. Baumann (ed.), OGC Web Coverage Service 2.0 Interface Standard -KVP Protocol Binding Extension, OGC 19-147r1 (2010)

[71] P. Baumann (ed.), OGC Web Coverage Service 2.0 Interface Standard -XML/POST Protocol Binding Extension, OGC 19-148r1 (2010)

[72] P. Bauman (ed.), OGC Web Coverage Service 2.0 Interface Standard -XML/SOAP Protocol Binding Extension, OGC 19-149r1 (2010)

[73] P. Baumann, On the Management of Multidimensional Discrete Data. VLDB Journal 4(3), 1994, Special Issue on Spatial Database Systems, pp. 401-444 (1994)

[74] P. Baumann, S. Holsten, A comparative analysis of array models for databases. Technical report, Jacobs University Bremen (2010)

[75] P. Baumann, S. Meissl, J. Yu (eds.), OGC® Web Coverage Service 2.0 Interface Standard -Earth Observation Application Profile, OGC 10-140 (2011)

[76] P. Cudre-Mauroux, H. Kimura, K.-T Lim, J. Rogers, R. Simakov, E. Soroush, P. Velikhov, D.L. Wang, M. Balazinska, J. Becla, D. DeWitt, B. Heath, D. Maier, S. Madden, J. Patel, M. Stonebraker, and S. Zdonik, A demonstration of SciDB: a science-oriented DBMS. *Proc. VLDB Endow.* 2, 2 (August 2009), 1534-1537 (2009)

[77] P.G. Frankl, S.N. Weiss, and C. Hu. All-uses versus mutation testing: An experimental comparison of effectiveness. *Journal of Systems and Software, 38, pp. 235-253* (1997)

[78] P.N. Johnson-Laird, DEDUCTIVE REASONING. *Annual Review of Psychology*, 50, pp.109-135 (1999)

[79] Policy SWG, The Specification Model — A Standard for Modular specifications. OGC 08-131r3 (2009)

[80] R. Binder, Testing Object-Oriented Systems: Models, Patterns, and Tools, chapter 18, Object Tech nologies, pp. 943-951, first ed. Addison Wesley (1999)

[81] R. Hewett, and P. Kijsanayothin, Automated Test Order Generation for Software Component Integration Testing. In Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering (ASE '09). IEEE Computer Society, Washington, DC, USA, 211-220 (2009)

[82] R. Mallal and M. Yunus, SOA Testing using Black, White and Gray Box Techniques, Crosscheck Networks, http://www.crosschecknet.com/resources/white_papers/SOA_Testing_Techniques.pdf (2006)

[83] R. Razali, Usability of Semi-formal and Formal Methods Integration - Empirical Assessments, *PhD thesis, University of Southampton* (2008)

[84] S. Cox, P. Daisey, R. Lake, C. Portele, and A. Whiteside (eds.), “OpenGIS

Geography Markup Language (GML) Encoding Standard," OGC 02-023r4 (2002)

[85] S. Cox, P. Daisey, R. Lake, C. Portele, and A. Whiteside (eds.), "OpenGIS Geography Markup Language (GML) Encoding Standard," OGC 03-105r1 (2004)

[86] S.G. Krantz, Discrete Sets and Isolated Points, §4.6.2 in Handbook of Complex Variables. Boston, MA: Birkhäuser, pp. 63-64 (1999)

[87] S.V. Rice, H. Bunke, and T.A. Nartker, Classes of Cost Functions for String Edit Distance, Algorithmica, 18(2), pp. 271-280 (1997)

[88] S. Weißleder and B.-H. Schlingloff, Quality *of Automatically Generated Test Cases based on OCL Expressions*, ICST 2008. pp.517-520. IEEE (2008)

[89] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill. Section 22, 540–557 (2001)

[90] V. Le, Q. Tran, A Fuzzy Logic Based Method for Analyzing Test Results. ASEAN Journal on Science and Technology for Development, Vol.23(3), 217-222 (2006)

[91] Y. Le Traon, T. Jeron, J.-M. Jezequel, and P. Morel, Efficient object-oriented integration and regression testing, IEEE Transactions on Reliability, Vol.49(1), 12-25 (2000)

Annex A 3A-TRE Syntax

See the corresponding description in Section 4.1

For example, the result of CWA:T ∧ U is F

```
EASumExpUnary. ASUMExp ::= ASUM ":" TRE;
EASumExpParenthesis. TRE ::= "(" ASUMExp ")" ;
ETREBinary. TRE ::= TRE OP TRE;
ETREParenthesis. TRE ::= "(" TRE ")" ;
ETREUnary. TRE ::= TS ;
EASumOWA. ASUM ::= "OWA";
EASumSA. ASUM ::= "SA";
EASumCWA. ASUM ::= "CWA";
ESTrue. TS ::= "T" ;
ESFalse. TS ::= "F";
ESUnknown. TS ::= "U";
EOPAnd. OP ::= "∧";
EOPOr. OP ::= "∨";
```

Annex B Conformance statements

B.1 Statements with non-singular strongly-connected components

See the corresponding description in Section 6.4.1

```
CS1= CS2 ∧ CS3 ∧ CS4 ∧ CS5
CS2= CS6 ∧ CS7 ∧ CS8
CS3= CS17 ∧ CS18 ∧ CS19
CS4= CS20 ∧ CS21 ∧ CS22 ∧ CS23 ∧ CS58
CS5= CS57 ∧ CS58 ∧ CS59
CS6= CS9 ∧ CS10 ∧ CS11 ∧ CS60
CS7= CS12 ∧ CS13
CS8= CS14 ∧ CS15 ∧ CS16
```

CS9= CS26 ∧ CS36 ∧ CS51

CS10= CS26 ∧ CS36 ∧ CS51

CS11= CS26 ∧ CS36 ∧ CS51 ∧ CS60

CS12= CS26 ∧ CS28 ∧ CS51

CS13= CS26

CS14= CS26

CS15= CS26

CS16= CS24 ∧ CS26 ∧ CS30 ∧ CS41

CS17= CS24 ∧ CS30 ∧ CS41

CS18= CS24 ∧ CS30 ∧ CS41

CS19= CS24 ∧ CS30 ∧ CS41

CS20= CS24 ∧ CS25

CS21= CS29 ∧ CS30 ∧ CS31 ∧ CS32

CS22= CS40 ∧ CS41 ∧ CS42 ∧ CS43

CS23= CS28 ∧ CS36 ∧ CS51

CS24= CS3

CS25= CS26 ∧ CS27 ∧ CS28

CS26= CS24

CS27= CS26 ∧ CS101

CS28= CS26 ∧ CS30

CS30= CS3 ∧ CS25 ∧ CS29 ∧ CS34 ∧ CS35

CS31= CS36 ∧ CS37 ∧ CS38 ∧ CS39 ∧ CS59

CS32= CS30 ∧ CS102

CS36= CS30

CS37= CS36

CS38= CS36 ∧ CS60

CS39= CS36

CS41= CS3 ∧ CS25 ∧ CS31 ∧ CS40 ∧ CS44 ∧ CS45 ∧ CS46 ∧ CS47 ∧ CS48 ∧ CS49 ∧ CS50

CS42= CS51 ∧ CS52 ∧ CS53 ∧ CS54 ∧ CS55 ∧ CS56 ∧ CS59

CS43= CS102

CS44= CS101

CS46= CS60

CS47= CS103

CS51= CS41 ∧ CS60

CS52= CS13 ∧ CS41

CS53= CS41

CS54= CS41

CS55= CS41

CS56= CS41

CS57= CS26

CS58= CS26 and (CS75 ∨ CS84 ∨ CS91)

CS59= CS26 ∧ CS60

CS60: (CS36 ∨ CS51) ∧ CS61 ∧ CS62 ∧ CS63 ∧ CS64 ∧ CS65 ∧ CS66 ∧ CS67 ∧ CS68 ∧ CS69 ∧ CS70 ∧ CS71 ∧ CS72 ∧ CS73 ∧ CS74

CS75= CS76 ∧ CS77 ∧ CS78 ∧ CS79 ∧ CS80 ∧ CS81 ∧ CS82 ∧ CS83

CS76= CS26

CS78= CS24 ∧ CS30 ∧ CS41 ∧ CS80 ∧ CS81 ∧ CS82

CS79= CS26 ∧ CS78

CS80= CS30 ∧ CS78

CS81= CS41 ∧ CS78

CS82= CS41 ∧ CS78

CS83= CS32 ∧ CS43

CS84= CS85 ∧ CS86 ∧ CS87 ∧ CS88 ∧ CS89 ∧ CS90

CS85= CS26

CS86= CS26

CS87= CS24

CS88= CS30

CS89= CS41

CS90= CS32 ∧ CS43

CS91= CS92 ∧ CS93 ∧ CS94 ∧ CS95 ∧ CS96 ∧ CS97 ∧ CS98 ∧ CS99 ∧ CS100

CS92= CS26

CS93= CS26

CS94= CS24 ∧ CS30 ∧ CS41

CS95= CS24 ∧ CS30 ∧ CS41

CS96= CS41

CS97= CS41

CS98= CS24 ∧ CS30 ∧ CS41

CS100= CS32 ∧ CS43

CS101

CS102

CS103

B.2 Statements without non-singular strongly-connected components

See the corresponding description in Section 6.4.2

CS1= CS2 ∧ CS4 ∧ CS5 ∧ CS104

CS2= CS6 ∧ CS7 ∧ CS8

CS4= CS20 ∧ CS21 ∧ CS22 ∧ CS23 ∧ CS58

CS5= CS57 ∧ CS58 ∧ CS104

CS6= CS9 ∧ CS10 ∧ CS11 ∧ CS104

CS7= CS12 ∧ CS13

CS8= CS14 ∧ CS15 ∧ CS16

CS9= CS104

CS10= CS104

CS12= CS104

CS13= CS105

CS14= CS104

CS15= CS104

CS16= CS104

CS20= CS104

CS21= CS29 ∧ CS104

CS22= CS40 ∧ CS42 ∧ CS43 ∧ CS104

CS23= CS104

CS32= CS102 ∧ CS104

CS42= CS52 ∧ CS53 ∧ CS54 ∧ CS55 ∧ CS56 ∧ CS104

CS43= CS102

CS44= CS101

CS47= CS103

CS52= CS13 ∧ CS104

CS53= CS104

CS54= CS104

CS55= CS104

CS56= CS104

CS57= CS104

CS58= CS75 ∧ CS84 ∧ CS91 ∧ CS104

CS75= CS76 ∧ CS77 ∧ CS79 ∧ CS83 ∧ CS105

CS76= CS104

CS83= CS32 ∧ CS43

CS84= CS85 ∧ CS86 ∧ CS87 ∧ CS88 ∧ CS89 ∧ CS90

CS85= CS104

CS86= CS104

CS87= CS104

CS88= CS104

CS89= CS104

CS90= CS32 ∧ CS43

CS91= CS92 ∧ CS93 ∧ CS94 ∧ CS95 ∧ CS96 ∧ CS97 ∧ CS98 ∧ CS99 ∧ CS100

CS92= CS104

CS93= CS104

CS94= CS104

CS95= CS104

CS96= CS104

CS97= CS104

CS98= CS104

CS100= CS32 ∧ CS43

CS104= CS29 ∧ CS34 ∧ CS35 ∧ CS40 ∧ CS44 ∧ CS45 ∧ CS47 ∧ CS48 ∧ CS49 ∧ CS61 ∧ CS62 ∧ CS63 ∧ CS64 ∧ CS65 ∧ CS66 ∧ CS67 ∧ CS68 ∧ CS69 ∧ CS70 ∧ CS71 ∧ CS72 ∧ CS73 ∧ CS74 ∧ CS101

CS105= CS104

Annex C IDGM evaluation schedule algorithm

See the corresponding description in Section 5.4.3

```
Boolean Evaluation_Schedule (Node S, Graph G)

// G is a Directed Acyclic Diagram (DAG). S is the starting node of G with an initial truth value. The function reevaluates the truth value of S. The root S, which is disentangled from its dependencies, is the predecessor of all nodes of the DAG and needs to be reevaluated to check whether the derived truth value is consistent with the initial one.

{
  Node P;
  NodeQueue L;

   //L is the queue (First-In-First-Out) of nodes with all of their dependencies been explored.

   L.Add(S);

   //add the starting node into the queue
```

```
while not L.IsEmpty() Do
   P = L.Pop();
   //get the node at the front of the queue
   // evaluate the node that have not yet been explored
   If not P.IsCondensed()
      evaluate(P);
      //the function evaluates an atomic node according to
      its dependencies if it is not the root
   Else
      Graph G' = V.GetGraph();
      //derive the Graph from the condensed node
      Node I = G'.GetInitalNode();
      //get the initial node of the derived Graph
      t = ini();
      //get the initial truth value
      I.truthValue = t;
      //initiate the node with the given truth value
      GetStableResult(t, I, G');
      //the function derives the stable result of the
      derived Graph
   EndIf
   lable(P);
   //label the node
   For each V in P.GetNext();
      // P.GetNext() retrieves unexplored nodes that have
      all of their dependencies been explored
      L.Add(V);
   EndFor
   L.Remove(P);
   //remove P from the front of the queue
```

```
  EndDo
   Return root_evaluate(S);
   // reevaluate the truth value of the root according to its
   dependencies which are disentangled from S
}

Int GetStableResult (Boolean t, Node I, Graph G)
//G is a non-singular strongly-connected component and Node
I is one of its nodes, which is initialized by a given truth
value t. The function evaluates the stable result of the Graph
{
  Graph G' = G.Disentangle (I);
  //disentangle node I from its dependencies and derive a new
graph
  G' = G'.Condense(G');
  //condense the graph to get a Directed Acyclic Graph (DAG)
  Boolean t' = Evaluation_Schedule (I, G');
   //evaluate truth value of node I to check whether it is
   consistent with the initial input
   If t' == t
      Return OK;
      //if the derived result is consistent with the initial
      input, the stable result is derived
   Else
      t = t';
      GetStableResult(t, I, G);
      //otherwise, the derived result is used to evaluate the
      stable result
   EndIf
}
```

List of publications:

J. Yu, X. Wang, P. Baumann: *A Specification-Oriented Geospatial Coverage Ontology Study*. Proc. Ontology, Conceptualization, and Epistemology for Information Systems, Software Engineering, and Service Science (ONTOSE'2010), Hammamet, Tunisia, June 7 - 8, 2010, Lecture Notes in Business Information Processing (LNBIP), Volume 62, Springer, pp. 63-74 (ISTP)

J. Yu, P. Baumann, X. Wang: RPRA: A Novel Approach to Mastering Geospatial Web Service Testing Complexity, ICSDM2011&IWGIS2011, pp. 252-256, DOI: 10.1109/ICSDM.2011.5969042 (EI)

J. Yu, P. Baumann: On the Systematic Generation of Reference Output for Geo Image Services. Geoinformatics 2011, pp. 1-6, DOI: 10.1109/GeoInformatics.2011.5980753 (EI)

X. Wang, **J. Yu**, and P. Baumann: A Web Coverage Ontology for Geospatial Web Applications, Fifth IEEE International Conference on Semantic Computing, September 19-21, Stanford University, Palo Alto, CA, USA, 2011 (EI)

J. Yu, P. Baumann: Specification-Based Geo Service Testing -Strategies for Increasing Confidence in Geo Services (presentation and abstract). AGILE 14th PTB workshop, Utrecht, Netherland. April 18th, 2011

Standards (OGC):

P. Baumann, S.Juba. **J.Yu**, etc.: WCS 2.0 Core Interface Standard, OGC 09-110r2 (as a contributor)

P. Baumann, A.Aiordachioaie, **J.Yu**: GML 3.2 Application Schema for WCS 2.0, OGC 09-146r1 (as a contributor)

P. Baumann, S. Meissl, **J. Yu** (eds.): OGC® Web Coverage Service 2.0 Interface Standard Earth Observation Application Profile, OGC 10-140 (as an editor)

Oral presentations:

J.Yu: Semantic-based Quality of Service Availability for the Guarantee of Geo Raster Processing. Workshop: Innovation in Geo Services: Challenges and

Prospects, Bremen, Germay, February 3rd, 2009.

J.Yu: Toward a Specification-based Quality Guarantee for Geo Raster Web Services. 70th OGC Technical Committee, Darmstadt, Germany, October 1st, 2009.

J.Yu: A Specification-Oriented Geospatial Coverage Ontology Study. Proc. ONTOSE, Hammamet, Tunisia, June 7 - 8, 2010

Contributions:

2009 OGC WCPS queries for CCIP2009 demo (see www.earthlook.org)

2010 Test scenarios by OGC WCPS in VAROS (ESA) project

2009-2010 Rasdaman system test patch (see www.rasdaman.org)

2009-2011 Conformance testing for geo raster web services in HMA-FO (ESA)

2011 OGC OWS-8 WCS 2.0 ATS&ETS

PhD graduate program:

2008-2011 graduate training programme offered at the Helmholtz Research School on Earth System Science (ESSRES) directed by Alfred Wegener Institute for Polar and Marine Research, the University of Bremen, and the Jacobs University.