Periodica Polytechnica Electrical Engineering and Computer Science, 63(4), pp. 282–294, 2019

Semantic Data Management in IT Service Performance Assurance

Imre Kocsis[1*], András Pataricza[1]

[1] Department of Measurement and Information Systems, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, H-1117 Budapest, Magyar tudósok krt. 2., Hungary
[*] Corresponding author, e-mail: ikocsis@mit.bme.hu

Received: 15 October 2018, Accepted: 19 April 2019, Published online: 25 July 2019

Abstract

In today's dynamic and highly composed environments, IT service performance and dependability assurance require efficient reasoning about the performance and dependability effects of faults and the countermeasures to choose, using limited knowledge. Model- and observation-based qualitative error propagation analysis methods can be applied to this end; however, providing support for the human, as well as conceptually structured machine interpretation of sets of competing error propagation hypotheses is an open problem. This paper proposes the application of Formal Concept Analysis (FCA) for these tasks. A natural way to represent error propagation hypothesis sets as formal contexts is proposed, and the visual diagnostic exploration of formal context lattices is introduced. On this basis, potential applications of FCA in performance and dependability assurance activities are characterized.

Keywords

error propagation analysis, formal concept analysis, system test and diagnosis

1 Introduction

The capacity of IT systems to provide services at the agreed level of performance – commonly recorded in legally binding Service Level Agreements (SLAs) [1] – can diminish due to a range of factors. These include overload faults (spare capacity exhaustion), capacity reduction due to "hard" faults as node crashes, and performance interferences of workloads in shared resource systems [2].

The local error state effects of faults can lead to component failures; and these failures become external faults of connected elements in composed systems. This way, error modes that are initially local to specific components can propagate across the topology, change their nature depending on the failure response of components and may lead to service failures.

This phenomenon of *error propagation* (see Fig. 1) is a central concept in dependable computing [3]. To prevent, mitigate, or at least to recover from service level failures, error propagation in a system has to be analyzed at design time and countermeasures have to be deployed in the system. For performance failures, these mechanisms include admission control points and continuously maintained spare capacities. Active dependability mechanisms rely on monitoring and diagnosing the system at runtime and deciding on the appropriate action to take, based on policies that are derived from error propagation analysis.

In the general case, error propagation analysis problems entail a set of competing error propagation solution hypotheses: a set of error propagation scenarios that are all consistent with the underlying model. Multiple solutions can arise due to:

1. limited observability,
2. nondeterministic manifestation of errors in distributed systems and
3. limited diagnosability of the system.

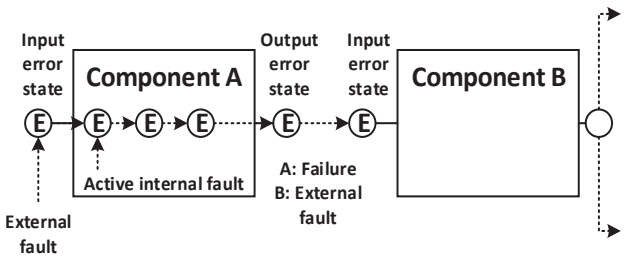

Fig. 1 The concept of error propagation in composed systems

Cite this article as: Kocsis, I., Pataricza, A. "Semantic Data Management in IT Service Performance Assurance", Periodica Polytechnica Electrical Engineering and Computer Science, 63(4), pp. 282–294, 2019. https://doi.org/10.3311/PPee.13305

The efficient expert assessment and iterative refinement of *error propagation hypothesis sets* are largely open problems.

This paper explores the novel application of Formal Concept Analysis (FCA) [4] on error propagation hypothesis sets. FCA is a mathematical approach to derive concept hierarchies of objects, based on their shared properties; in the proposed application, it is used to form hierarchies of sets of faults, based on their commonalities in error propagation effects.

The lattices of fault sets which FCA creates, ordered by increasing / decreasing commonality in effects, enable an expressive visualization of error propagation hypothesis sets. Visualization interactions, as relation projections and filterings, support visual exploration. FCA can provide a tool for:

1. *guided diagnostics*,
2. assessing the need for *additional observations* and
3. hypothesis set *refinements*.

It will be shown that the core FCA concepts have direct diagnostic interpretations. This way, on the one hand, it provides a common platform to assess the results of different *diagnostic inference* approaches, and on the other hand, FCA itself can directly deliver diagnostic inference rules.

FCA is a proven tool in ontology engineering, both for creating and assessing ontologies. While the current paper focuses on its direct application in design for dependability, it carries the promise of creating a semantic bridge between diagnostic inference and observations, and dependability knowledge and requirement modeling.

As a simple example, the paper uses a business process model, with inference over the propagation of performance errors and task-activation errors. However, the presented techniques are expected to efficiently support the qualitative performance and dependability analysis of Systems of Systems and Cyber-Physical Systems [5] – two contemporary domains where error propagation analysis with limited knowledge are key challenges.

2 Error Propagation Analysis

System-level inference over error propagation classically used such standard methods as Fault Tree Analysis (FTA) and Failure Mode and Effects Analysis (FMEA). Today, so-called Error Propagation Analysis (EPA) employs more sophisticated, highly automated techniques that perform system-level inference using system models and component error propagation rules. The source of the latter can be a formal analysis of system components, domain expert knowledge as well as observations.

The fault (and failure) models used in EPA are typically based on domain-specific, standard (either actual or de facto) dictionaries. In composed computing systems, it is customary to distinguish component service timing failures (EARLY and LATE), value failures (SUBTLE and COARSE – differentiated based on detectability) and so-called "provision" failures (OMISSION and COMMISSION) [6]. It is also a standard technique to use multi-dimensional failure modes by associating, for instance, a timing, value and provision aspect with failure modes at the same time ([7] established the basic logic of such classifications).

Purpose-built languages exist to describe the way system components may transform their "incoming" qualitative error signals into "outgoing" ones in various internal fault modes. The "Fault Propagation and Transformation Calculus" (FPTC) [8] provides a practical formalism in the form of *(incoming pattern, outgoing pattern)* component error propagation rule clauses. The "Formalism for Incompletion, Inconsistency, Interference and Impermanence Failures" (FI^{4}FA) [6] extends the above outlined taxonomy with work unit processing related failure modes, and modifies FPTC to accommodate these.

Modern EPA approaches rely on system and service models, reuse and compose rules describing component error propagation characteristics and are highly automated. The description of error propagation characteristics has been formulated as a specialization of the UML MARTE profile [9], partially inspired by [10]; can be consistently expressed through "views" in SysML (see e.g. [11]); and was standardized for the Architecture Analysis and Design Language (AADL) [12]. In a broader context, [13] presents an in-depth survey of the dependability modeling and analysis of software systems specified with UML. [14] defines an intermediate model that is applicable for a range of dependability analyses, including EPA.

Analytically, EPA can be approached in a number of fundamentally different ways. Classic models as FTA support a combinational style of modeling, which is ill suited for incorporating error propagation dynamics. Connected automata of nominal and faulty behavior can be subjected to model checking (as supported by, e.g., xSAP [15], and the COMPASS Toolset [16]), with the obvious limitations. One notable way for analytically resolving dynamics is determining the maximum possible error / failure sets on component connections as an EPA result, using fixpoint computations (see HiP-HOPS [17]).

3 Modern EPA and CSP-based analysis

This paper relies on the theoretical and analytical framework established by [18] and [19]. In [18], a core idea is that component behavior under external and internal faults can be represented through categorizing the deviations of "actual" component input and output behaviors from the specified (system-wide fault-free) case. This way, the dynamic description of error propagation in a system can be performed through connected error automata.

For situations where the complexity of state-based analysis would be prohibitive, or propagation characteristics are simply not known at such a resolution, this dynamic description can be compacted to so-called *syndromes* – scalar temporal abstractions of the infinite set of (potentially infinite) automaton input-output traces. This replaces the component error automata with simple relations ("static syndrome relations"), what, in turn, enables highly efficient analysis through finite-domain Constraint Satisfaction Problem (CSP) solving.

Contemporary EPA is typically not purely forward or backward diagnostic reasoning, as fault impact analysis and fault diagnosis are at the *conceptual* level. Rather, the goal is to characterize error propagation under a set of potentially mixed fault and service failure constraints, optimality objectives and decision variables (see, e.g., [20]).

An important complex use case is determining the dependability mechanisms that allow only a specified worst failure on the output, under some fault activation pattern. Dependability mechanisms as "watchdog", "fail-silent" or "task replication" [21] can be modelled as additional decision variables switching "on" and "off" the effect of their presence. CSP based analysis of EPA in the formulation of [18] is able to solve for such problems [20].

However, especially in these cases, the set of error propagation hypotheses can easily become unmanageable – not necessarily technically, but in the sense that an analyst will find it intractably hard to understand the logic structure and characteristics of the solution set.

As the set of all model-consistent error propagation hypotheses is essentially a relation over the same set of (component port error propagation) variables, solution sets can be expressed in a number of ways, from a tabular form to Multiple Decision Diagrams (MDD), a multi-valued variant of Binary Decision Diagrams (BDD). As a matter of fact, constraint solving itself can be implemented directly over MDDs [22]; [23] proposes using such a solver in EPA to derive error propagation hypothesis sets, instead of enumerating constraint problem solutions.

In general, these representations don't lend themselves well to efficient human analysis. As a counterpart to domain-specific visualization and assessment techniques (as for instance qualitative error propagation covers [23]), subjecting error propagation hypothesis sets to FCA is proposed – after the minimal necessary introduction of the diagnostic concepts that will be used.

4 Basic diagnostic concepts

The structural granularity of an EPA model, the coverage of error propagation on component connections and the modeled resolution of faults / errors / failures all influence *diagnostic resolution*: the precision with which EPA is capable to determine the location and nature of faults from (the observation of) errors and failures. Too low a resolution leads to the use of costly overprotective dependability mechanisms in design and at runtime. The core problem is the indistinguishability of faults that require different treatment. At the same time, too high a resolution wastes resources, too, by creating and exercising a diagnostic logic that is far finer than what is needed by the available corrective actions.

For the purposes of this paper, we define *tests* as mechanisms that binarily determine the presence or absence of a specific error mode on a specific component-connection in a system (system failures are errors propagating "out" from the system boundary). A test *detects* a fault, if its positive outcome implies the presence of a fault. In classic test theory, a fault f_j is said to dominate another fault f_i, if all tests for f_j also detect f_i. When two faults dominate each other, they are called equivalent [24, 25]. This way, tests and their combinations partition the set of possible faults into *equivalence classes*: sets of faults that are indistinguishable using a set of tests.

In system level diagnosis, selecting an optimal subset of tests is a different problem for *fault detection* and *fault localization* ("diagnosis"); as detection aims at only determining that there is *some* active fault in a system, while localization is the problem of determining the point of manifestation of an active fault. Test sets can be static as well as dynamic – in the former case, all tests are run for each diagnosis, while in the latter, the outcomes of the already-performed tests determine the next one to run. For both cases, finding minimal-size optimal test sets is NP-hard (see, e.g., [26]), but either explicit solution is tractable for rough-granular models, or good heuristics exist.

5 Formal Concept Analysis

Formal Concept Analysis (FCA) is a field of mathematics that is intimately connected to the fundamental philosophical question of "what concepts are" in human thinking and communication; and what "relations between concepts" constitute. FCA provides a formal mathematical theory of concepts and their hierarchies; at the same time, its tools and "knowledge representation" approaches are applicable in a wide variety of domains. Its dominant practical uses are increasing human understanding; communicating concept structures and various automated concept discovery and simplification tasks. In its latter roles, it's also a valuable tool in data mining and machine learning [27].

In FCA, a "concept" has two key parts: its extension – the objects "belonging to" the concept; and its intension: the set of attributes that all objects belonging to the concept have. Using the quasi-standard notation of the seminal paper of Wille [28], a formal context is a triple $\mathbb{K} = (G, M, I)$, where

- G is a set of objects (*Gegenstände*),
- M is a set of attributes or properties (*Merkmale*), and
- I (*Inzidenz*) is a binary relation over these sets, expressing whether an object "has" an attribute or not.

Informally, a formal concept in such a formal context is an (O, P) set-pair of $O \subseteq G$ and $P \subseteq M$, for which the object set is exactly the set of objects that share the specified properties; and conversely, the property set is exactly the set of properties shared by the objects.

For our purposes, it is useful to also formulate the notion of formal concepts in a somewhat more "object-oriented" style. Based on I, let us introduce a property mapping: $\Pi : 2^G \to 2^M$, which maps each object set to the properties common to the objects. This mapping unambiguously defines the property set for a concept-forming object set. Then the $C \subseteq 2^G$ set of formal concept forming sets contains such $c_i \in C$, where $\forall o_i \notin c_i : \exists m \in \Pi(c_i)$, so that $m \notin \Pi(o_i)$; and $\forall c_j \in C : \forall m_i \notin \Pi(c_j)$, it holds that $\exists o \in c_j : m_i \notin \Pi(o)$.

There is a natural subconcept-superconcept ordering between the concepts, by intent (or dually, extent) inclusion:

$$\begin{aligned}(c_1, \Pi(c_1)) \le (c_2, \Pi(c_2)) \\ \Leftrightarrow (c_1 \subseteq c_2 \Rightarrow \Pi(c_2) \subseteq \Pi(c_1)) \\ \wedge (\Pi(c_2) \subseteq \Pi(c_1) \Rightarrow c_1 \subseteq c_2).\end{aligned} \tag{1}$$

Informally, a subconcept is associated with a smaller number of objects that share across themselves a larger set of attributes. The set of all formal concepts for $\mathbb{K}$ together with this ordering is denoted $\mathfrak{B}(\mathbb{K})$.

$\mathfrak{B}(\mathbb{K})$ is a complete lattice. (Note that a concept can be empty: either object-, or attribute-wise.) Explicit derivation of $\mathfrak{B}(\mathbb{K})$ for a finite $\mathbb{K}$ is supported by a number of algorithms. For large contexts, approaches such as "Iceberg lattices" [29] are known to compute only the uppermost level(s) and only concepts with a large enough support (in the association mining sense).

The actual *analysis* of the formal concepts can involve the following key activities:

- interactive, visual lattice exploration,
- structure-preserving simplifications,
- determining association and implication rules between the attributes, and
- "attribute exploration".

Interactive exploration will be introduced using a simple, hierarchical failure abstraction example in the next section.

In the original FCA theory, two lattice structure preserving (up to isomorphism) simplification operations are defined: clarification and reduction [30]. Clarification of objects and attributes is simply replacing intent-equivalent object sets with a single object and extent-equivalent attribute sets with a single attribute, respectively.

An attribute is said to be reducible – can be eliminated with preserving lattice structure – if it can be replaced by a combination of other attributes; that is, there's a set of attributes for which the set of objects having these common attributes equals the set of objects having the attribute. Objects can be reduced in a similar way; the context reached after deleting all reducible objects and attributes is called the standard context (the concept lattice of which is still isomorphic to the concept lattice of the original context).

A minimal generator set – the *Duquenne-Guigues* set of implications – can be automatically computed for the deterministic attribute-implications of the formal context. From this generator set, all valid implications can be enumerated using a set of rules.

Concept lattice computation on finite contexts may work on a context that aims to be representative of a domain, but is not necessarily complete in the sense that it contains all examples necessary to discover the "true" lattice structure valid for the whole domain. Attribute exploration interactively poses a series of targeted questions to ensure generalized (structural) validity. A domain expert could answer that question with a simple "yes", or provide a counterexample (which is incorporated into the relation).

6 Visual FCA by example: failure abstractions

The immediate utility of FCA, in the current domain as well as in others, is a kind of intelligent "knowledge exploration" through visual exploration, as we introduce it on a hierarchical failure mode abstraction example.

Fig. 2 presents a portion of a cross table of component failures and their associated "types" at multiple levels of abstraction – as could be extracted, for instance, from a bug tracker system. (CO denotes "COMMISSION or OMISSION"). The lattice structures of formal contexts are usually visualized using a so-called additive line diagram, depicted for the cross table on Fig. 3. Each node of the graph represents a formal concept of the context and can be interpreted using two simple rules.

	Data Error	Timing error	CO error	Omission
Sign bug	X			
Error code	X			
Too fast + sign bug	X	X		
Fast error code reply	X	X		
Sign bug + overload	X	X		
Late and error code	X	X		
Missing reply			X	X
Unexpected signal			X	
Too early response		X		
Slow service		X		
Out of interval	X			
Missing timed signal			X	X

Fig. 2 Cross table: component failure modes and attributes (abbrv.)

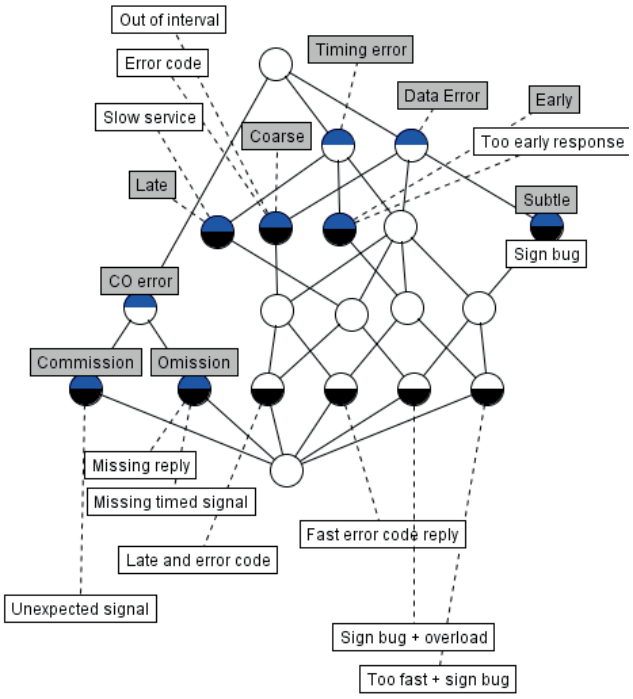

Fig. 3 Line diagram for the lattice of failure modes and attributes

1. The intent of a node is the attribute(s) directly associated with it plus all attributes that can be reached from it through upwards leading paths.
2. The extent of a node is the object(s) that are directly associated with it plus all objects from which the node can be reached through an upward leading path.

At the top of the diagram, there's the concept with no attributes and all objects; at the bottom, the concept with maximal intent and empty extent. The full lower half denotes whether a node "owns" any object; a full upper half whether it "owns" an attribute. Note that there are such unrealized concepts (no object introduced in the node as part of the concept extent) that also don't "define" an attribute directly (empty circles). In the mathematical sense, these are (formal) concepts, too – their associated extent and intent can be decoded using the above rules.

Very similarly to Exploratory Data Analysis (EDA) [31, 32], visual inspection of such a graph leads to useful observations and hypotheses. For instance, the CO category is mutually exclusive with the others, and type implications like EARLY $\Rightarrow$ TIMING_ERROR become apparent.

In general, the nodes (formal contexts) enable an easy assessment of the *failure instance coverage* of each abstract category, the trivial structural redundancies in failure modes and types, and the complexity and member-exclusivity of refinement hierarchies (trees versus complex bipartite graph patterns, edges leading "out" from a perceived hierarchy).

The main interactions with the diagram (e.g., in the Concept Explorer tool [33]) are the following:

1. filtering the included objects and/or attributes; and
2. highlighting concept nodes and their up/down neighbors (transitively; shown later).

Filtering enables discovering the impact of adding and removing assumed failure modes as well as types. Highlighting a concept node not only helps "reading off" its object and property sets, but also emphasizes the magnitude the subconcepts of the selected concept contribute to *other* concepts, and the adequacy of the selected concept to describe its superconcepts.

Such taxonomy analysis applications of FCA are standard practice, for instance in ontology

engineering [34]. However, in the EPA context these techniques have direct *diagnostic* interpretations, too, with diagnostic applications.

7 Diagnostic FCA

Error propagation hypothesis sets as relations encode the admitted valuation-combinations of component error propagation port decision variables for a system (including fault activations). To apply FCA, objects and attributes have to be identified – FCA directly over the relations is nigh useless. The key idea here is to contrast a pair from the set of core dependability aspects: faults, errors / failures, and the application of dependability mechanisms. This paper explores the case when faults are the objects, and errors / failures / mechanism decisions the attributes; grouping fault activations by their increasingly common effects, thus enabling the diagnostic capability analysis of propagating errors and failures as tests. Other configurations are also meaningful; e.g., the faults as attributes and errors / failures as objects scoping would mainly support the evaluation of fault impact magnitudes and their hierarchies.

The cross table and accompanying line graph provide a very simple example for the idea, if we assume the objects to be the internal fault modes of a complex composite component, and the attributes the testable service failure modes of that component on a single output. The next section will provide a more realistic, worked-out example for system level analysis with tests defined on component connections; however, the key insights remain the same.

Under this interpretation, the diagnostic semantics of visual line graph exploration and FCA can be summarized as presented by Table 1.

Table 1 Diagnostic interpretation of FCA visualization, diagram interactions and algorithms

FCA	Diagnostic interpretation
Line graph interpretation (concept hierarchy)	
Objects on a downward path	Dominance ordering of a set of faults
Attributes on a downward path	Increasing fault resolution of tests
Multiple objects belonging to a concept node	Faults indistinguishable with the currently visualized attributes
Multiple attributes belonging to a concept node	Tests equivalent in fault detection and fault resolution
Node upper and lower half full	Test directly detects one or more faults (resolution depends on faults in downward neighbors)
Only node upper half full, antecedents of node with objects don't have paths leading up to this level or higher	Test fault coverage is a combination of more specific tests
Only node lower half full	Fully specific determination of node fault(s) requires combination of tests
Line graph interactions	
Node selection and highlighting of upward / downward reachable nodes (see later)	A maximal effect-equivalent (upwards attributes) fault set (downwards objects). Edges leading out from selection inform on fault uniqueness to tests and individual test resolution for faults.
Adding / removing attributes	Potentially increasing / decreasing diagnostic resolution
Adding / removing objects	Effects of changing fault model and assumptions
FCA algorithms	
Attribute reduction	Minimal size test set without reducing diagnostic resolution
Object reduction	Minimal size representative fault set without eliminating propagating effects
Implication set	Generator set for deterministic error propagation rules
Attribute exploration	Methodical checking of hypothesis set completeness

8 Theoretical justification of diagnostic FCA

Table 1 relies on equivalences between FCA and diagnostic theory that require some basic justification. The current section provides these arguments.

Let $\mathbb{K}_f = (F, E, I)$ be a formal context, where F is the set of fault-activations present in an error propagation hypothesis set; E the set of possible error / failure valuations on each component connection; and I the relation encoding the errors / failures characteristic for the different fault activations in the error propagation hypothesis set. For the sake of simplicity, this section assumes that the relation is deterministic for the fault activations. The set of fault activations may, but is not required to, include the fault-free case. In the following it is assumed that sets of tests are evaluated as a conjunction for detecting faults.

Proposition 8.1. *Each formal concept of* $\mathbb{K}_f$ *represents a test set and the set of equivalent faults for that test set. The equivalence set of faults is complete for the given test set, and the test set is maximal for the fault set in the sense that adding any further tests can only shrink the equivalence set covered.*

The proposition does not require a formal proof, as it follows from the basic properties of formal concepts and the formulation of the test notion used. For applications, some care may need to be exercised: the fault-free case(s) may be ordered in an equivalence set with "real" fault activations.

The concept order relation has an important diagnostic interpretation: it directly translates to fault dominance.

Proposition 8.2. *The order of* $\mathfrak{B}(\mathbb{K}_f)$ *is a fault dominance relation in the sense that* $\forall c_i, c_j \in \mathfrak{B}(\mathbb{K}_f), c_i < c_j$ *implies that the test set of* c_i *is also a test set for all faults in* c_j.

For the ordering to hold, the definition requires the smaller concept to have attributes that are strictly a subset of those of the larger one (and exactly reverse for the objects). This means that the smaller concept dominates the larger by detecting a larger set of faults, using a common subset of test attributes.

The higher fault *detection* power of well-chosen, smaller sets of attributes is entirely in line with the basic logic of system level test and diagnosis [35]. Detecting the mere presence of an otherwise unidentified fault activation from a set of fault activations requires testing for the "most common properties" of the fault activations. Conversely, determining the location of the fault, up to the resolution that is practical for repair actions, requires identifying the "uncommon" (i.e., distinguishing) fault effect features within a set of fault activations.

Proposition 8.3. *Attribute reduction of clarified contexts creates a minimal test set for fault localization with the maximum possible resolution.*

As a proof, we can call on the dominance-interpretation of the order relation to show that if a test set is minimal, then it's a reduced attribute set; and if an attribute set is reduced, then it defines a minimal test set.

Minimality of the test set means that taking away any of the tests, at least two distinguishable faults would become indistinguishable. In other words, taking away any attribute creates a concept lattice where at least two objects that previously were present at least in two different concepts are now only elements of the same concepts.

Removal of an attribute from a context can only force each existing concept to merge with others or stay unchanged – no "new" concept will be formed, as a new (not produced by mergers) concept would mean that we now distinguish an object set with a maximum common attribute set that we did not distinguish before; and if we only delete an attribute, then there's no reason for any such concept not be present in the original context. And if only mergers and non-changes are possible, two objects previously elements of at least two different concepts becoming part only of the same concepts means that the number of concepts has to be smaller than originally.

But in this case, taking away an attribute cannot be done without changing the lattice structure. As it is a property of reducible attributes that their removal does not change the lattice structure, there is none of them; meaning that the lattice is not further attribute-reducible.

In the other direction, a reduced attribute set means that no test is equivalent to a combination of other tests; there's no test the fault equivalence set of which could be exactly created as an intersection of the fault equivalence sets of other tests. Any combination either includes further elements, and thus the test contributes additional fault localization; or leaves one or more faults out, and thus the test contributes fault detection.

Proposition 8.4. *The minimal set of implications (Duquenne-Guigues set of implications) acts as a generator set (using, e.g., the Armstrong rules) for the deterministic error propagation rules and cross-input / output port value combination correspondences in the system. The implications have general validity for the fault activation set at hand.*

Again, this proposition holds through a diagnostic interpretation of the general-purpose FCA construct. Due to the construction and meaning of the elements of the formal context, any attribute implication either

1. describes direct or indirect forward propagation;
2. describes direct or indirect "backward propagation";
3. establishes implication in the input or the output port set of some component; or
4. establishes an implication between two ports not connected through error propagation.

As implications can be categorized mechanically based on the error propagation component model, the rule-subset membership of any base or derived implication is trivially decidable. Multiple fault activations taken into account may suppress otherwise deterministic propagation rules by making propagation seemingly nondeterministic; in this representation fault activation is encoded in the objects, so the difference in output for a component port due to an activated internal fault will manifest through the absence of certain attribute implications due to outcome-ambiguity.

However, FCA is not expected to directly support error propagation reasoning by creating propagation rules or minimalizing propagation characteristics at the system level. Rather, it may prove to be a useful tool for experts to check the validity of propagation rules, especially when performed for components fault mode-wise.

9 A business process EPA example

For demonstration purposes, a simple business process example using the EPA approach of [18] is presented.

The modeling and analytic method has a number of direct antecedents. Based on [36], [18] describes mathematically precise, data flow network based EPA of business processes, using early modeling notations. [37] refined the approach for Business Process Model and Notation (BMPN 2) [38] models. [39] introduced a problem modularization approach for EPA that the example applies at the implementation level. Relatedly, [40] discusses the impact of error refinement and nondeterminism elimination on the specificity and level of pessimistic overabstraction of the analysis results.

While this work focuses on the qualitative evaluation of business process models, it has to be noted that the *quantitative* model-based dependability and performance analysis of business processes has quite a rich literature; see e.g. [41] and [42].

Fig. 4 depicts the simple example business process model used here. The process – adapted from a real-life example – describes the steps taken during initial application for a loan. While BPMN supports constructs to express data dependencies and data flows between activities (such as data objects), in order to keep the example simple only the control flow aspect is expressed.

It is also assumed that all activities are implemented by IT resources that can fail, but are dedicated (that is, activities don't have common mode faults). The "Manual check" step is an exception; it is assumed not to be influenced by resources. (Although not discussed here, resource dependencies can be integrated directly into the model and used for EPA – see, for instance, [37].)

Fault activation patterns are restricted to resource faults; activity implementation faults and execution engine faults are assumed not to be present. These would only add unnecessary complexity to the example.

For EPA, BPMN elements as well as resources are mapped to a network of error propagation components, with a number of input / output error port types (see Fig. 5). Port types have associated error mode dictionaries, as shown by Fig. 5. Splitting data and activation (provision) + timing reflects the way BPMN presents data and control flow – that is, using two intertwined dependency graphs. The error mode INACTIVE is included to differentiate the case when a workflow element is "correctly inactive" – e.g., because it belongs to an execution path that's avoided due to choices in exclusive gateways. A component can use resources and in turn,

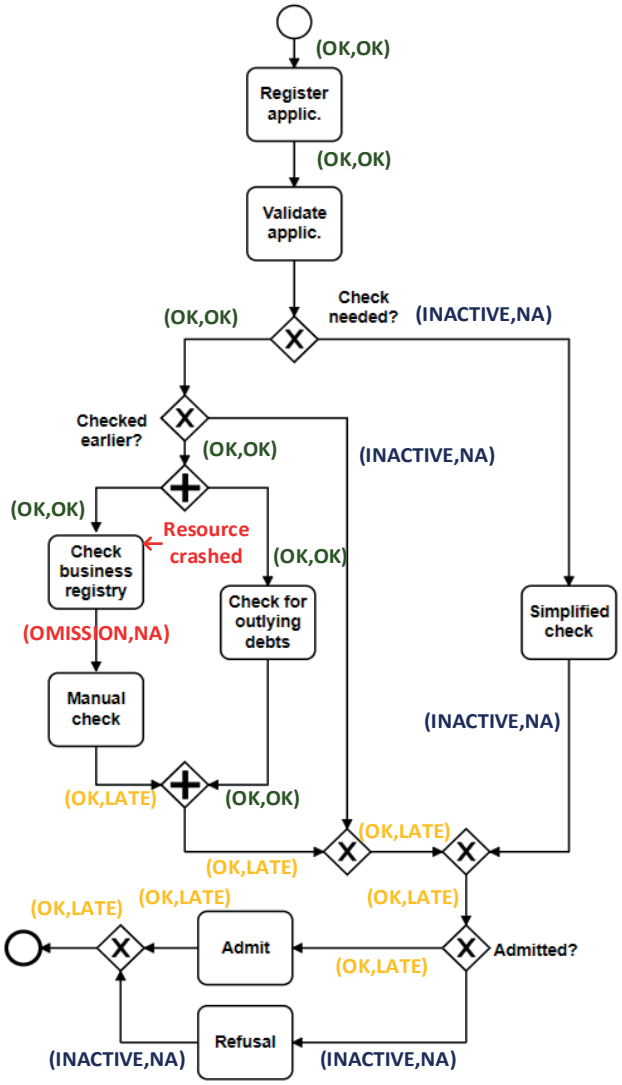

Fig. 4 An example business process model, annotated with a particular error propagation hypothesis

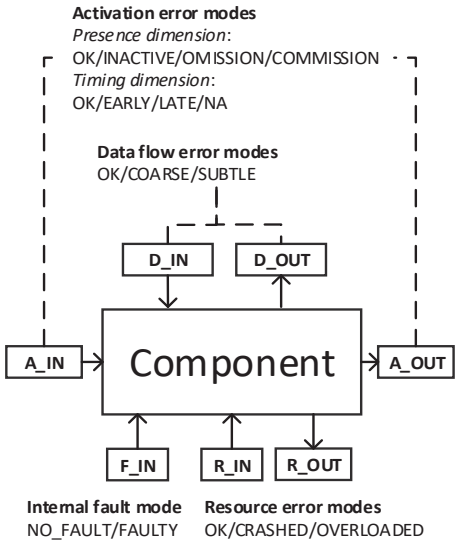

Fig. 5 Error propagation component model

act as a resource; the resource error modes used here are CRASHED and OVERLOADED. Components may also have internal faults under a binary fault model.

The component model is specialized for the three types of resource-extended BPMN model elements used here.

- BPMN elements map to error propagation components with a 1:1 mapping of modeled relationships to activation and data input / output error ports.
- Tasks – roughly, the "actual steps" of the process that perform the orchestrated business activities – also have a resource error mode input port. Data error flow is unutilized in the example.

Thirdly, resources are simply components that have an internal fault mode and a resource error mode out port.

The basic logic of error propagation model translation to mathematical representation for CSP is the following (the example does not require explicit temporal compaction through syndromes).

1. Component ports are mapped to decision variables with a finite domain corresponding to their port-dictionaries.
2. The process and resource topology is mapped into equality constraints on port-variable pairs.
3. Input-output port variable constraints express the error propagation (transformation) rules for each component.

Propagation rules for non-task BPMN elements are straightforward and reusable. The same holds for the "default" rule sets of tasks – although some cases need careful consideration. For instance, a CRASHED execution resource transforms an incoming COMMISSION activation error mode to INACTIVE, and not OMISSION.

The default model includes a number of nondeterministic choices; e.g., by default, a task-internal fault can cause almost any outgoing error modes (notably, not COMMISSION). For individual tasks, the rules can be refined, made more deterministic, or even replaced.

10 Exploratory analysis of hypothesis sets

As a propagation example, consider the error propagation annotation on Fig. 4, a single error propagation hypothesis solution to the problem of "the resource under the business registry check activity is CRASHED and all other resources remain OK". Such a (single) solution is easily tractable. However, this is not the only solution of this problem. On one hand, there are those solutions where execution simply steers clear of the crashed resource – through a different, in our data-omitting case nondeterministic, choice on the first or second exclusive gateway. On the other hand, the last exclusive gateway "doubles" each solution "unnecessarily"; from the point of view of system-level effects, it's largely unimportant which branch the execution chooses there.

It is important to note the way this style of error propagation analysis (in a broader sense, disturbance effect propagation analysis) can efficiently blend inference over distinct, but interconnected service assurance domains. In the example, the engagement of the watchdog-like dependability mechanism, while providing tolerance against a "hard" fault, also has a performance impact; that is, the experienced delay will increase. Similarly, solving for an overloaded resource for the "Check business registry" step will lead to a propagating OK, but LATE activation. At the same time, there are business process tasks that can "transform back" a propagating error that is purely a performance issue to one that falls under classic dependability (or performability) concerns. An example is activating a task LATE that is bound to a specific deadline.

Fig. 6 presents the line diagram for the resulting error hypothesis set under the constraint that zero or one resource faults may be present ("multiple instantiation" of faults is due to the fact that the activation of a specific fault can nondeterministically cause different outcomes). The diagram is a projection for the input of the "Stop" event (failure types) and the input errors of the confluence element of the "Checked earlier?" choice gateway (identified by the CH4 prefix, denoting the fourth "choice" element in the model). The latter "probing point" has been selected purely as an example; in practice, much larger diagrams can be well assessed than what can be presented here (and the filtering of variables would focus on field replaceable units and elements with applicable dependability mechanisms; an aspect that this paper does not address).

The selection leading to the highlighting on the figure has been performed on the node of the "STOP_IN_T_LATE" label (timing input port of the "Stop" component being evaluated to LATE) – that is, we are looking at the (pure) performance failures at the system level.

The reading of the diagram is the following. "Upwards", late execution at the end of the process entails only execution correctly reaching the "Stop" stage – maybe not interestingly, but reassuringly. "Downwards", the paths identify that

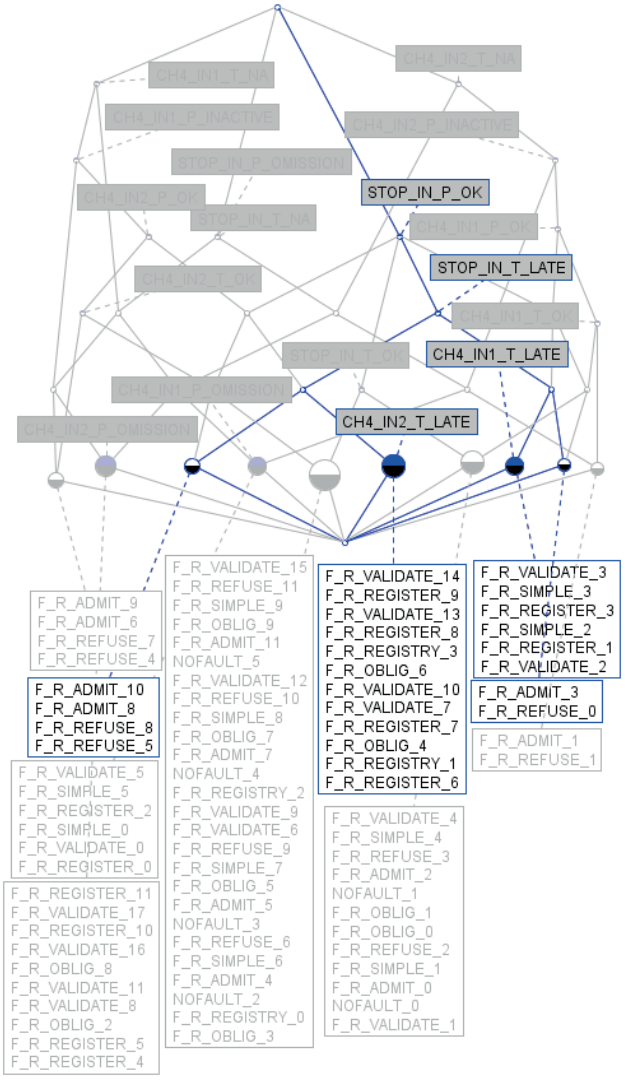

Fig. 6 Fault activation instance and error / failure port valuation formal context, with filtering and interactive selection

- the "late" nature of system-level behavior is associated either with one of the inputs of CH4 being late, or the latency introduced later in the process (in an XOR manner); and
- the possible resource fault activations leading to the late activation at the stop event – and these don't include the "no fault" cases.

In general, line diagrams make the fault-localization capability of the (full) intent of any node easy to visually assess. Furthermore, nodes with an owned attribute express a "most specific" fault-localizing attribute that makes the "upward" elements of the full intent superfluous to distinguish its extent from the remainder. Regarding the full context, line diagrams visually represent the fault equivalence classes of the finest granularity possible under the current (filtered) error / failure set; and the hierarchy of tests that make them increasingly distinguishable.

The exploration made possible by interactive filtering and highlighting not only enables the agile, visual assessment of pre-designed fault detection and localization attribute sets, but also helps in expert monitoring feature selection when it is impractical to formalize all important feature selection preferences.

11 Use cases for FCA in EPA

Two fundamental use cases categories of FCA in dependability and performance assurance are apparent: supporting dependability and performance assurance planning and application for ontology design.

For the first use case, Fig. 7 presents an overview of the techniques that the previous sections introduced. The fundamental idea here is that the visualization capabilities and algorithms for various simplifications in FCA can provide a unique insight into the models, analysis and planning artifacts at each stage of the workflow.

For visual interpretability, context sizes have to be kept moderate, thus projections have to be performed for human processing; however, this is not a major problem, as the human analyst is generally interested in validation and taking major design decisions at a low structural resolution (as intended error containment regions, replaceable / repairable components, or major functional blocks). Also, dedicated domain-specific algorithms (especially practice-proven intelligent heuristics) in the

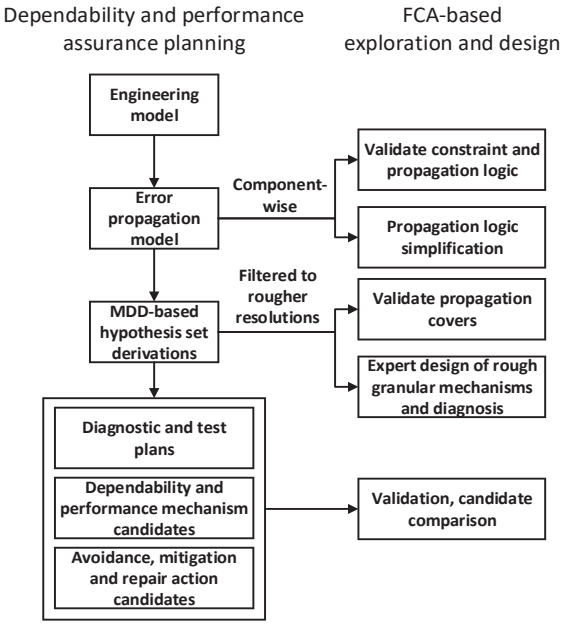

Fig. 7 Applications of FCA in assurance planning

planning workflow can be expected to have much more favorable computational complexity characteristics then the exact FCA algorithms. (Although it is an open question, whether the statement holds for FCA heuristics and partial context computation algorithms.)

At the same time, for error propagation models, propagation hypothesis sets, test plans, etc., FCA carries the promise to serve as a bridge between automated model building and the human expert – the very same way as EDA does for statistical modelling and hypothesis testing. In EDA as well as here, visualizations and descriptive abstractions don't serve to replace sophisticated algorithms, but to efficiently reach hypotheses, check a wide range of (implicit) assumptions and validate modelling results.

The second major category is ontology building. Formal concepts in general are recognized as an efficient tool for identifying concepts and relations for ontologies from data, as well as for merging and completing them (e.g. through attribute exploration) [43, 44].

There are established approaches to express core concepts, requirements and their relationships in dependability and performance (for dependability see e.g. [45]; specifically for workflow performance, [46]); however, these tend not to fully utilize the potential of ontologies. Recently, [47] introduced an ontology that explicitly captures fault-failure mechanisms and severities (through concepts, relationships and the use of description logic) and uses FMEA to determine them.

As it has been directly shown here, FCA is capable to characterize, partition and order faults by their effects – and the same is true in reverse, for failures through fault-attributes (cross table transposition is a standard FCA technique; regarding the diagnosis and test interpretation, similar results can be derived). This way, FCA supports deriving the fault and failure hierarchies and relationships that emerge from data – in the here discussed case from computed error propagation hypothesis sets, but the same holds for (classified / quantized) experimental and monitoring data. For an existing ontology capturing fault types through their effects, it also enables checking the effect-wise equivalence or compatibility of observed effects with pre-modelled ones, and checking the ontology for completeness and freedom from (formal) conceptual redundancy.

More generally, FCA is also capable to hierarchically conceptualize error propagation behaviors (e.g. through their error containment / creation or dampening / amplification capability attributes) and error mitigation requirements, opening up the possibility for data-driven, rich semantic support of all major qualitative aspects of design for dependability and performance. Future research will target these challenges.

Acknowledgement

The work was created in commission of the National University of Public Service under the priority project KOFOP-2.1.2-VEKOP-15-2016-00001 titled "Service Development Establishing Good Governance" in the Workshop for Science of Public Governance 2017/162 BME-VIK "Smart City - Smart Government".

References

[1] Lloyd, V., Rudd, C. "Service Design", The Stationary Office, London, United Kingdom, 2007.

[2] Kocsis, I., Pataricza, A., Micskei, Z., Kövi, A., Kocsis, Z. "Analytics of resource transients in cloud-based applications", International Journal of Cloud Computing, 2(2-3), pp. 191–212, 2013. https://doi.org/10.1504/IJCC.2013.055267

[3] Avizienis, A., Laprie, J.-C., Randell, B., Landwehr, C. "Basic concepts and taxonomy of dependable and secure computing", IEEE Transactions on Dependable and Secure Computing, 1(1), pp. 11–33, 2004. https://doi.org/10.1109/TDSC.2004.2

[4] Wille, R. "Formal Concept Analysis as Mathematical Theory of Concepts and Concept Hierarchies", In: Ganter, B., Stumme, G., Wille, R. (eds.) Formal Concept Analysis: Foundations and Applications, Lecture Notes in Computer Science, vol. 3626, Springer, Berlin, Heidelberg, Germany, 2005, pp. 1–33. https://doi.org/10.1007/11528784_1

[5] Rajkumar, R., Lee, I., Sha, L., Stankovic, J. "Cyber-physical Systems: The Next Computing Revolution", In: 47th Design Automation Conference, Anaheim, USA, 2010, pp. 731–736. https://doi.org/10.1145/1837274.1837461

[6] Gallina, B., Punnekkat, S. "FI4FA: A Formalism for Incompletion, Inconsistency, Interference and Impermanence Failures' Analysis", In: 37th EUROMICRO Conference on Software Engineering and Advanced Applications, Oulu, Finland, 2011, pp. 493–500. https://doi.org/10.1109/SEAA.2011.80

[7] Bondavalli, A., Simoncini, L. "Failure classification with respect to detection", In: Second IEEE Workshop on Future Trends of Distributed Computing Systems, Cairo, Egypt, 1990, pp. 47–53. https://doi.org/10.1109/FTDCS.1990.138293

[8] Wolforth, I., Walker, M., Papadopoulos, Y., Grunske, L. "Capture and reuse of composable failure patterns", International Journal of Critical Computer-Based Systems, 1(1-3), pp. 128–147, 2010. https://doi.org/10.1504/IJCCBS.2010.031710

[9] Bernardi, S., Merseguer, J., Petriu, D. C. "Adding Dependability Analysis Capabilities to the MARTE Profile", In: Czarnecki, K., Ober, I., Bruel, J.-M., Uhl, A., Völter, M. (eds.) Model Driven Engineering Languages and Systems, Lecture Notes in Computer Science, vol. 5301, Springer, Berlin, Heidelberg, Germany, 2008, pp. 736–750. https://doi.org/10.1007/978-3-540-87875-9_51

[10] Pataricza, A. "From the General Resource Model to a General Fault Modeling Paradigm?", In: Workshop on Crititcal Systems Development with UML at UML 2002, Dresden, Germany, 2002, pp. 163–171.

[11] Andrews, Z., Fitzgerald, J., Payne, R., Romanovsky, A. "Fault modelling for systems of systems", In: IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS), Mexico City, Mexico, 2013, pp. 1–8. https://doi.org/10.1109/ISADS.2013.6513445

[12] Delange, J., Feiler, P. "Architecture Fault Modeling with the AADL Error-Model Annex", In: 40th EUROMICRO Conference on Software Engineering and Advanced Applications, Verona, Italy, 2014, pp. 361–368. https://doi.org/10.1109/SEAA.2014.20

[13] Bernardi, S., Merseguer, J., Petriu, D. C. "Dependability Modeling and Analysis of Software Systems Specified with UML", ACM Computing Surveys, 45(1), pp. 2:1–2:48, 2012. https://doi.org/10.1145/2379776.2379778

[14] Montecchi, L., Lollini, P., Bondavalli, A. "An Intermediate Dependability Model for state-based dependability analysis", University of Firenze, Dip. Sistemi e Informatica, Florence, Italy, Rep. RCL101115, 2011.

[15] Bittner, B., Bozzano, M., Cavada, R., Cimatti, A., Gario, M., Griggio, A., Mattarei, C., Micheli, A., Zampedri, G. "The xSAP Safety Analysis Platform", In: Chechik, M., Raskin, J.-F. (eds.) Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science, Vol. 9636, Springer, Berlin, Heidelberg, Germany, 2016, pp. 533–539. https://doi.org/10.1007/978-3-662-49674-9_31

[16] Bozzano, M., Cimatti, A., Katoen, J.-P., Nguyen, V. Y., Noll, T., Roveri, M. "Safety, Dependability and Performance Analysis of Extended AADL Models", The Computer Journal, 54(5), pp. 754–775, 2011. https://doi.org/10.1093/comjnl/bxq024

[17] Wallace, M. "Modular Architectural Representation and Analysis of Fault Propagation and Transformation", Electronic Notes in Theoretical Computer Science, 141(3), pp. 53–71, 2005. https://doi.org/10.1016/j.entcs.2005.02.051

[18] Pataricza, A. "Model-based dependability analysis", DSc Thesis, Hungarian Academy of Sciences, 2008.

[19] Pataricza, A. "Systematic Generation of Dependability Cases from Functional Models", presented at FORMS/FORMAT Symposium on Formal Methods for Automation and Safety in Railway and Automotive Systems, Budapest, Hungary, Oct. 9-10, 2008.

[20] Pataricza, A., Urbán, P. "A Combination of Petri-Nets and Linear Programming in Design for Dependability", Budapest University of Technology and Economics, Budapest, Hungary, technical report, 1997. [online] https://pdfs.semanticscholar.org/7864/dd8f57ba1942b23d15a8ef687ede1f1434cd.pdf [Accessed: 07 October 2018]

[21] Hanmer, R. "Patterns for Fault Tolerant Software", 1st ed., John Wiley & Sons, Chichester, England, 2007.

[22] Molnár, V., Majzik, I. "Constraint Programming with Multi-valued Decision Diagrams: A Saturation Approach", In: 24th PhD Mini-Symposium, Budapest, Hungary, 2017, pp. 54–57.

[23] Kocsis, I. "Design for Dependability Through Error Propagation Space Exploration", In: 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), Luxembourg, Luxembourg, 2018, pp. 172–178. https://doi.org/10.1109/DSN-W.2018.00059

[24] McCluskey, E. J., Clegg, F. W. "Fault Equivalence in Combinational Logic Networks", IEEE Transactions on Computers, C–20(11), pp. 1286–1293, 1971. https://doi.org/10.1109/T-C.1971.223129

[25] Agrawal, V. D., Prasad, A. V. S. S., Atre, M. V. "Fault collapsing via functional dominance", In: International Test Conference, Charlotte, NC, USA, 2003, pp. 274–280. https://doi.org/10.1109/TEST.2003.1270849

[26] Rish, I., Brodie, M., Ma, S., Odintsova, N., Beygelzimer, A., Grabarnik, G., Hernandez, K. "Adaptive diagnosis in distributed systems", IEEE Transactions on Neural Networks, 16(5), pp. 1088–1109, 2005. https://doi.org/10.1109/TNN.2005.853423

[27] Kuznetsov, S. O. "Machine Learning and Formal Concept Analysis", In: Eklund, P. (ed.) Concept Lattices, Lecture Notes in Computer Science, Vol. 2961, Springer, Berlin, Heidelberg, Germany, 2004, pp. 287–312. https://doi.org/10.1007/978-3-540-24651-0_25

[28] Wille, R. "Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts", In: Rival, I. (ed.) Ordered Sets, NATO Advanced Study Institutes Series (Series C — Mathematical and Physical Sciences), vol. 83, Springer, Dordrecht, The Netherlands, 1982, pp. 445–470. https://doi.org/10.1007/978-94-009-7798-3_15

[29] Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L. "Computing iceberg concept lattices with Titanic", Data & Knowledge Engineering, 42(2), pp. 189–222, 2002. https://doi.org/10.1016/S0169-023X(02)00057-5

[30] Dias, S. M., Vieira, N. J. "Concept lattices reduction: Definition, analysis and classification", Expert Systems with Applications, 42(20), pp. 7084–7097, 2015. https://doi.org/10.1016/j.eswa.2015.04.044

[31] Behrens, J. T. "Principles and procedures of exploratory data analysis", Psychological Methods, 2(2), pp. 131–160, 1997. https://doi.org/10.1037/1082-989X.2.2.131

[32] Morgenthaler, S. "Exploratory data analysis", Wiley Interdisciplinary Reviews: Computational Statistics, 1(1), pp. 33–44, 2009. https://doi.org/10.1002/wics.2

[33] Tane, J., Kaiser, T., Objedkov, S. "Concept Explorer", [computer program] Available at: http://conexp.sourceforge.net/ [Accessed: 07 October 2018]

[34] Fu, G. "FCA based ontology development for data integration", Information Processing & Management, 52(5), pp. 765–782, 2016. https://doi.org/10.1016/j.ipm.2016.02.003

[35] Simpson, W. R., Sheppard, J. W. "System Test and Diagnosis", 1st ed., Springer Science & Business Media, Boston, USA, 1994. https://doi.org/10.1007/978-1-4615-2702-2

[36] Csertán, Gy., Pataricza, A., Harang, P., Dobán, O., Biros, G., Dancsecz, A., Friedler, F. "BPM Based Robust E-business Application Development", In: Bondavalli, A., Thevenod-Fosse, P. (eds.) Dependable Computing EDCC-4, Lecture Notes in Computer Science, Vol. 2485, Springer, Berlin, Heidelberg, Germany, 2002, pp. 32–43.
https://doi.org/10.1007/3-540-36080-8_4

[37] Urbanics, G., Gönczy, L., Urbán, B., Hartwig, J., Kocsis, I. "Combined Error Propagation Analysis and Runtime Event Detection in Process-Driven Systems", In: Majzik, I., Vieira, M. (eds.) Software Engineering for Resilient Systems, Lecture Notes in Computer Science, Vol. 8785, Springer, Cham, Switzerland, 2014, pp. 169–183.
https://doi.org/10.1007/978-3-319-12241-0_13

[38] Object Management Group "BPMN 2.0", [online] Available at: http://www.omg.org/spec/BPMN/2.0/ [Accessed: 27 July 2017]

[39] Bonfiglio, V., Brancati, F., Rossi, F., Bondavalli, A., Montecchi, L., Pataricza, A., Kocsis, I., Molnár, V. "Composable Framework Support for Software-FMEA through Model Execution", In: Bondavalli, A., Brancati, F. (eds.) Certifications of Critical Systems - The CECRIS Experience, 1st ed., River Publishers, Gistrup, Denmark, 2017, pp. 183–200.
https://doi.org/10.13052/rp-9788793519558

[40] Pataricza, A., Kocsis, I., Brancati, F., Vinerbi, L., Bondavalli, A. "Lightweight Formal Analysis of Requirements", In: Bondavalli, A., Brancati, F. (eds.) Certifications of Critical Systems - The CECRIS Experience, 1st ed., River Publishers, Gistrup, Denmark, 2017, pp. 143–166.
https://doi.org/10.13052/rp-9788793519558

[41] Gilmore, S., Gönczy, L., Koch, N., Mayer, P., Tribastone, M., Varró, D. "Non-functional properties in the model-driven development of service-oriented systems", Software and Systems Modeling, 10(3), pp. 287–311, 2011.
https://doi.org/10.1007/s10270-010-0155-y

[42] Gönczy, L., Chiaradonna, S., Di Giandomenico, F., Pataricza, A., Bondavalli, A., Bartha, T. "Dependability Evaluation of Web Service-Based Processes", In: Horváth, A., Telek, M. (eds.) Formal Methods and Stochastic Models for Performance Evaluation, Lecture Notes in Computer Science, vol. 4054, Springer, Berlin, Heidelberg, Germany, 2006, pp. 166–180.
https://doi.org/10.1007/11777830_12

[43] Ignatov, D. I. "Introduction to Formal Concept Analysis and Its Applications in Information Retrieval and Related Fields", In: Braslavski, P., Karpov, N., Worring, M., Volkovich, Y., Ignatov, D. (eds.) Information Retrieval, Communications in Computer and Information Science, Vol. 505, Springer, Cham, Switzerland, 2015, pp. 42–141.
https://doi.org/10.1007/978-3-319-25485-2_3

[44] Cimiano, P., Hotho, A., Stumme, G., Tane, J. "Conceptual Knowledge Processing with Formal Concept Analysis and Ontologies", In: Eklund, P. (ed.) Concept Lattices, Lecture Notes in Computer Science, Vol. 2961, Springer, Berlin, Heidelberg, Germany, 2004, pp. 189–207.
https://doi.org/10.1007/978-3-540-24651-0_18

[45] Dobson, G., Sawyer, P. "Revisiting Ontology-Based Requirements Engineering in the age of the Semantic Web", presented at International Seminar on Dependable Requirements Engineering of Computerised Systems at NPPs, Halden, Norway, Nov. 27-29, 2006.

[46] Truong, H.-L., Fahringer, T., Nerieri, F., Dustdar, S. "Performance metrics and ontology for describing performance data of grid workflows", In: IEEE International Symposium on Cluster Computing and the Grid, Cardiff, UK, 2005, pp. 301–308.
https://doi.org/10.1109/CCGRID.2005.1558569

[47] Sanislav, T., Mois, G., Miclea, L. "An approach to model dependability of cyber-physical systems", Microprocessors and Microsystems, 41, pp. 67–76, 2016.
https://doi.org/10.1016/j.micpro.2015.11.021