Modeling Software Agent Awareness of Physical-Informatical Essence Duality

Yaniv Mordecai, Ori Orhof
Faculty of Industrial Engineering and Management
Technion – Israel Institute of Technology
Haifa, Israel
yanivmor@technion.ac.il, ori@orhof.com

Dov Dori
Engineering Systems Division
Massachusetts Institute of Technology
Cambridge MA, USA;
Faculty of Industrial Engineering and Management
Technion – Israel Institute of Technology
Haifa, Israel
dori@mit.edu

Abstract— **Physical-informatical essence duality (PIED) is the parallel existence of an entity as both the original, usually physical source, and its informatical representation, as held by each agent interacting with the entity. Software systems interacting with physical and real-world entities have to maintain internal representations of these entities in order to handle them appropriately. The importance of modeling the dual essence of these entities in software controlled systems is critical to the execution of software-integrated processes that affect physical real-world entities. The implications of PIED must therefore be recognized and accounted for. This paper introduces a PIED-aware conceptual modeling framework for software control agents acting in cyber-physical settings. We discuss the utilization of UML and OPM for this purpose and demonstrate the applicability of our OPM-based framework through the case of airport baggage handling processes. In particular, we show that PIED-aware modeling and design of system controls can significantly improve treatment of inherent anomalies such as the common lost baggage problem.**

Keywords— Model-based Systems Engineering; Physical-Informatical Essence Duality; Agent-Oriented Architecture; Object-Process Methodology; Epistemic Logic; Knowledge Representation; Cyber-Physical Systems

I. Introduction

Cyber-physical systems (CPS) are software-based systems comprising virtual and physical segments of various controllers and actuators that interact with each other and with the environment to accomplish virtual and physical goals [1]. Interactions and mutual effects between software segments and real-world segments pose significant challenges to system designers, developers, and users. Realizing that what the system "knows" about a significant portion of the surrounding environment must be accounted for is a fundamental concept in the systems engineering and cybernetics domains [2], [3].

A holistic systems thinking approach advocates the consideration of the system in the broadest sense, including the environmental segment which affects it or is affected by it. This approach extends the software-centered approach, which considers only the software segment as the system and everything outside of it as the environment [4]. The meaning of this paradigm shift, which is critical for CPS design and development, is the consideration of environmental entities, which traditionally had not been considered parts of software-based systems. Successful design and delivery of software-controlled processes mandates distinguishing "closed" software systems from "open" ones and specifying open software systems based on their environment.

Interactions and mutual effects between a subsystem of the CPS and systemic or environmental entities rely on the awareness of the subsystem's software agents of these internal and external entities. The entities can be human or inanimate, sentient or indifferent to external stimuli, cognizant or incognizant, contactable or unreachable, and controllable or incontrollable. For any one of these distinctions, some of these entities can perceive each other and refer to each other symmetrically or asymmetrically. Software agents interacting among themselves and with their external environment have to maintain and constantly update a representation of the system-environment relationships. This representation must support the agent's capability to interact with the environment and with other parts of the system, facilitating seamless agent and system-environment interactions.

Realizing that both the original entity and its representation co-exist independently is critical for deep understanding of this duality and modeling it correctly. The dual existence of each significant entity as both "real" and "conceived" is called Physical-Informatical Essence Duality (PIED) [5]. Each manifestation—the original (usually physical) embodiment and each one of its software representations—has its own characteristics, and these may not be holistically aligned. In previous papers [5]–[7] we have thoroughly analyzed this challenging problem and its implications on complex CPS processes and interactions.

Contemporary and legacy popular software modeling languages, including the structured approach and the UML approach, do not provide appropriate semantics and a referential framework for PIED-aware modeling, design, and realization. Consequently, software models fail to provide high-fidelity support for complex software-based interactions in unreliable settings. The model does not satisfactorily reflect of the actual system, so PIED effects and PIED-affected situations are missed or misunderstood until they manifest themselves through inconsistencies revealed during system realization and activation. Sometimes dire consequences to the system and those who interact with it follow. For example, failure to realize that there are often differences between the perceived and actual location of moving devices can cause inaccuracies and errors in various location-based processes, such as navigation, positioning, localization, location-based response, and physical interaction [8].

A prominent example is the search for missing Malaysia Airlines flight MH370. For a whole week, great efforts and resources were spent to search an area some 1500 miles south-west off Australia's coast. On March 28, 2014, the search was shifted almost 700 miles north east of the old search area based solely on outcomes of new analysis of already-known information. As of writing of these lines search is still underway but findings seem to be promising [9]. PIED awareness is manifested starkly in this example because the location of the missing airplane is unknown though it is clear that it must be somewhere in that area, and all that search teams can do is rely on the best information available, which is vague and sketchy at best, and its best analysis.

In this paper we extend the notion of PIED by highlighting the importance of PIED-aware modeling and design of complex software-based interactions in cyber-physical settings. We also demonstrate and discuss the challenges associated with the identification, consideration, and regulation of PIED instances and occurrences and their derivatives. We focus on PIED-aware modeling of the issuance of monitoring and control signals—informatical entities—for the purpose of changing the state of physical entities in the environment or within the system itself.

The rest of this paper is organized as follows: Section II is a brief literature review of theoretical foundations of PIED-aware software modeling and design. Section III discusses PIED-aware modeling with UML, while Section IV focuses on an OPM-based approach. While the former is more common and largely adopted by the software community, the latter has important advantages for complex systems modeling and design: OPM's ability to concurrently represent physical and informatical objects and processes. In Section V we present and analyze a PIED-aware model of an airport baggage handling system, accounting for the all too familiar problem of lost baggage. In Section VI we review the benefits and results of applying PIED-aware design considerations, we summarize the paper and discuss future research.

II. Physical-Informatical Essence Duality

Physical-Informatical Essence Duality (PIED) is the dual existence of environmental or systemic entities as physical embodiments and the informatical representations thereof. Software agents interacting with an external entity have to maintain a model of the entity in order to manage, control, and facilitate the interaction. The interaction has to rely on information about the entity's state and attribute values that is as close to real time as possible.

PIED has been discussed in the literature on epistemic logic [10] and knowledge representation [11]. The way an entity is captured, understood, or perceived by agents in a system is defined by Mizzaro as "epistemic information" [12], i.e., information derived from analysis and understanding of the entity's nature, state, and history. Bar-Hillel's Semantic Information Theory distinguishes the meaning of content payload from the meaning of the physical carriers, namely, the distinction of information from data [13]. Hayles [14] provides an in-depth discussion on the implications of information-matter duality and dualism, the abstraction of information from matter, the separation of information from the material source it reflects, and the separation of content and medium. Dori [15] discusses the informatics hierarchy from data to information to knowledge and ingenuity, and the realization that any informatical object must be recorded on a physical medium, be it a stone tablet, a digital medium, or the human brain.

Several studies have referred to precise real-world event and information detection, with applications in system safety [16], cyber security [17], and counter-terrorism [18], [19]. These occurrences of PIED are typical; they have in common the criticality of identifying threats or emergency situations by software agents embedded in large technical, organizational, or socio-technical systems. PIED-aware modeling draws on the definition of the software agent's knowledge base: what it needs to know about a physical entity in order to handle or tackle it, and what it actually knows about it. Knowledge about and awareness of a physical entity has several levels: a) existence, b) characteristics, c) state-space, and d) state.

The existence of a physical entity of interest to the system must first be recognized. Similarly, each relevant characteristic of the entity, or *refineable* in OPM terms, must also be recognized. *Refineables* include components (parts) and features—attributes and operations. The state space encompasses the various states of the entity, which are the permissible combinations of the states of its comprising components, and the value ranges (or sets) of its attributes. State awareness concerns knowledge of the actual state at which the entity is, i.e., the combination of its component states and attribute values. Each knowledge base artifact is defined in the model twice: once as pertaining to the physical, external entity (relative to the agent), and once as pertaining to the internal informatical representation of the entity.

As an example, consider a car plate number identifier for automatic toll road charging. The system automatically identifies the plate number of each passing car and charges the bill payer associated with the plate number. The system must first identify the presence of a car passing through. Using one or more sensors (optical, infra-red, RFID…), it then has to capture the plate image and use it to extract its number. The recognition should account for plate characteristics (e.g., color, size, material, figure format, viewing angle and position relative to the car), car characteristics (speed, type, size, color), and environmental conditions (light, weather). The system awareness of environmental conditions is a factor in the decision of which sensor or, if possible, combination of sensors to use. The real and extracted plate numbers are the physical and informatical entities, respectively. There is always some probability of wrong recognition, which depends on factors that include the quality of the sensors, the sophistication of the recognition algorithms and those that determine what sensor combination to select and what weight to assign to each sensor type based on external conditions and car speed. A system that can assess its own reliability can, for example, decide to avoid charging a car whose recognition reliability is below a certain threshold.

Five main processes are required to handle external entities in a PIED-aware manner: (a) entity detection, acquisition and recognition, (b) representation generating, (c) representation-based interaction, (d) interaction outcomes analysis, and (e) representation improvement. The first set of processes involves the entity's recognition and identification based on acquisition from an external source, or early definition by an external source. Following is internal representation generating, i.e., the creation of a software-based model of the entity. The next function, representation-based interaction, is the interaction of the agent's system with the external entity, which is based on how the agent perceives the entity's state prior to and during their interaction. To this end, the agent has to be aware of the state-space, which, in turn, requires dynamic characteristic awareness. Interaction outcomes analysis concerns analyzing any available information gathered by the system during and following the interaction. A desired result of this analysis is representation improvements, which may include state-space updating, corrections, and refinements.

Action coherence and reaction coherence are important concepts underlying PIED-aware entity handling. Action coherence is the measure of similarity between the actual entity state and the state perceived by the agent. Reaction coherence is the measure of similarity between the external entity's actual response to the action and the external entity's response expected by the agent. The reaction coherence may be high in spite of the agent's misconceptions about the entity's state. This might happen, for instance, if (a) the external entity is friendly and has the capability to ignore or correct actions or commands which it deems to result from an agent's misconception or (b) when several agents (acting as controllers) attempt to affect the same physical entity, which can decide to follow some controllers and ignore others. In such cases, interaction is resilient to erroneous agent actions. Action and reaction coherence imply that all possibly conceivable interactions must be accounted for, modeled, and analyzed in detail, in order to mitigate perilous incoherent actions and reactions. Insufficient system resilience can be a sign of problems with this mechanism, calling for PIED-aware modeling and realization.

III. PIED-Aware Software Modeling with UML

Conceptual modeling is highly instrumental in clarifying system function, structure, and behavior [20], [21]. However, common modeling languages like UML and SysML [22], as well as available architectures adapted for cyber-physical systems [1], lack the semantics to capture the intricate yet critical cyber-physical constructs in general [23] and PIED-related ones in particular.

The most critical shortcoming of modeling languages grounded in the information systems domain, such as UML and SysML, is their inability to distinguish physical entities from informatical ones, let alone capturing both physical and informatical aspects at the same view or within a single diagram. The model artifact which represents physical external entities is called *Actor*. Actors' interactions with the software are captured in use-case and activity diagrams. These diagrams typically provide understanding of the system's top-level. Non-human assets of the system are rarely captured as actors, and this is often a source of confusion. Other subsystems are also not defined as actors w.r.t. the software system in focus. Examples of such dubious actors are a passenger's baggage in an airport's baggage handling system, available and rented cars in an online car-rental service, and interceptors in ballistic missile defense systems.

Human actors, including users and subjects of system processes, as well as external non-human affecting or affected entities, can be captured in a static view of UML class diagram (CD). This diagram type shows the various object types (classes) comprising the system as semi-autonomous software components. While CDs are capable of capturing any external object as a class, they only capture the external entity's informatical representation, not the external physical entity itself. If, for instance, a software system controls the altitude of an aircraft, while the aircraft can be captured as a class, and its altitude as an attribute of the class, this is only a data item, which holds what can be referred to as the altitude, assuming it truly represents the physical altitude of the aircraft in real time. Recalling the MH370 Flight it is clear how problematic this can be. Many of the flight route and distance calculations rely on highly uncertain knowledge of the flight's altitude as it progressed over its presumed course during six hours.

The System Modeling Language (SysML), a UML profile, was created to support modeling and design of complex systems involving hardware and software. SysML models put less emphasis on the underlying informatical model, often assuming the existence of a separate UML model of the software segment or subsystem. Therefore, the informatical representation layer, captured in the CD, is not part of the SysML conventional model, unless a unified SysML-UML model is created. Such unification is complicated and not less problematic than separately modeling the physical system with SysML and its software portion with UML.

Since actors and their representative classes are almost invariably defined in two separate SysML and UML diagrams, typical UML models do not capture the relations between them. The ability to model subsystems as interacting actors or agents is key to complete modeling of cyber-physical systems. Yet, actor designations do not percolate to lower levels, so subsystems are not modeled as actors of other subsystems.

The sequence diagram (SD), common to UML and SysML, is the only diagram type in which actors and their respective classes can be modeled together. Sequence diagrams are therefore useful for PIED-aware modeling. SD's major drawbacks are their linearity, case-specificity, and inability to display conditional branching in a structured manner. They therefore serve mostly for modeling specific scenarios of transactions which are neither too simple nor too complex. Simple transactions are often too trivial to capture with SDs, while support of highly complex transactions with branching conditions is problematic. Some of the difficulties in SDs were amended in UML 2.0, including support for alternative and parallel routes and for SD chaining [24].UML modelers can choose which aspect to describe and what diagram type to use. Hence, if they find capturing agent interactions with the system challenging, they might choose not to use this type of diagram.

Capturing PIED aspects in UML or SysML models requires dedicating a sequence diagram for each of the five processes required for PIED-aware external entity handling. Each scenario that might involve action or reaction incoherence has to be captured in a dedicated sequence diagram that involves the perceived and actual entity states. The sequence diagram must capture the system's function and mechanisms to mitigate incoherencies. A set of sequence diagrams of these processes is illustrated in Fig. 1 and Fig. 2. Fig. 1 is a simple representation generating transaction, in which the external entity (the actor) signals its existence to the (agent's) representation generator, which, in turn, creates the internal representation.

Fig. 2 captures the agent's action, based on its internal representation state of the external entity. The action is a message (signal) to the external entity. The agent initially assumes that the action is coherent, and therefore updates the representation to the state into which the external entity is supposed to enter. From this point there are two possible branches, which are shown in the same sequence diagram using diagram frames. In the upper frame, a coherent response is assumed, so the agent's result analysis process leads to the termination of the interaction. In the lower frame is an incoherent response. Assuming the agent is able to detect the updated state of the external entity, the interaction outcomes analysis (marked in the diagram as "result analysis") process leads to representation improvement. Importantly, the use of frames in a sequence diagram is for display purposes only; it does not translate to conditional branching in the model's database.

IV. PIED-Aware Modeling with OPM

Data integration and fusion techniques, which account for multiple representations [25], [26], often avoid verification of the information with the actual state of the physical entity to which information refers. This complicates the problem of information integration due to the need for but lack of aligning statically stored information with real-world, dynamic and volatile information. Data integration, fusion and reconciliation with real-world embodiments must be conducted [21], [27]. System safety studies have shown that systemic misconceptions and incapability to identify information biases with respect to the actual state of affairs have the potential to lead to dire consequences and catastrophes, and these have occurred multiple times [28].

Object-Process Methodology (OPM) [15] meets many of the challenges posed by PIED-aware modeling. A holistic framework for complex systems modeling, specification, design, and verification, OPM captures the functional, structural, and procedural aspects of the modeled system in a single unified, dual graphical-textual view. An OPM model consists of a set of hierarchically organized and interconnected object-process diagrams (OPD). Each OPD extends and elaborates the OPD above it, providing for inherent complexity management and reduction. Each OPD is accompanied by a textual computer-generated description in Object-Process Language (OPL) – a subset of English.

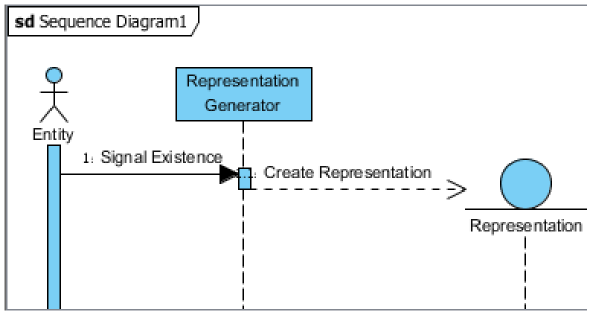

Fig. 1. Representation generating UML Seq. Diagram

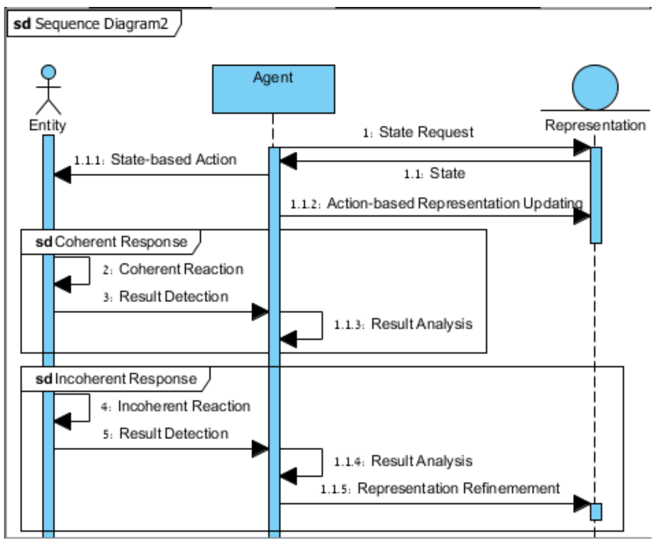

Fig. 2. SD of representation-based interaction, interaction outcomes analysis, and representation improvement.

Entity, or *thing* in OPM terms, generalizes an object and a process. OPM's inherent support of two semantically important metamodel entity attributes – essence and affiliation – is essential for PIED-aware modeling. *Essence* enables specifying an entity as either physical or informatical. *Affiliation* enables specifying the entity as either environmental (external to the system) or systemic (part of the system). OPM's graphical notation for these attributes is shading to denote physical entities and contour dashing to denote environmental entities. Non-shaded things are informatical, and solid contoured things are systemic (see Fig. 3).

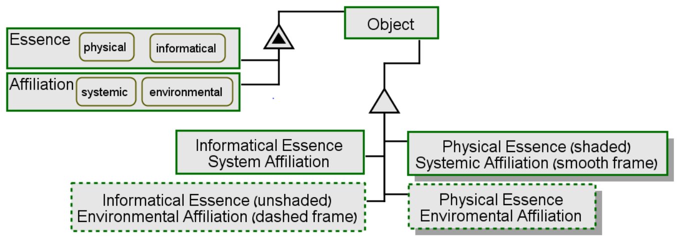
Essence
physical
informatical
Affiliation
systemic
environmental
Object
Informatical Essence System Affiliation
Physical Essence (shaded) Systemic Affiliation (smooth frame)
Informatical Essence (unshaded) Environmental Affiliation (dashed frame)
Physical Essence Enviromental Affiliation

Fig. 3. OPM Distniction of Affiliation (contour) and Essence (shade)

Each essence-affiliation combination is feasible and useful. External and physical entities include human and machine actors, beneficiaries, and *affectees*, i.e., objects which are potentially affected (undergo state change) by some process in the system. External informatical entities can include external sources of information, as well as intangible informatical real-world entities, such as facts, events, measurements, objectives, decisions, requests, results, algorithms, instructions, and control signals. Systemic physical entities are usually subsystems, components and parts of the system, while systemic informatical entities represent stored and managed data, information, and knowledge. Any informatical object is also physical since it needs a physical medium to record the information it stores and manages.

Representations of external entities are mostly captured as informatical artifacts, held by the agent. In some cases, when the external entity's behavior is critical and complex, the external entity can be represented internally as a simulated agent. This simulated agent mimics the behavior of the external entity, and is therefore capable of executing virtual processes reflecting the physical processes performed by the external entity. This is a useful design pattern in distributed systems engineering, artificial intelligence, and machine learning.

Like any OPM model, the OPM PIED-aware design pattern model consists of interconnected OPDs. The system diagram (SD), which is the top-level OPD of the OPM model of a PIED-aware system is presented in Fig. 4. This high-level view illustrates the availability of **PIED-Aware Entity Handling** operation of **Subsystem**, supporting its own **Function**, which, in turn, supports the **System**'s **Function**.

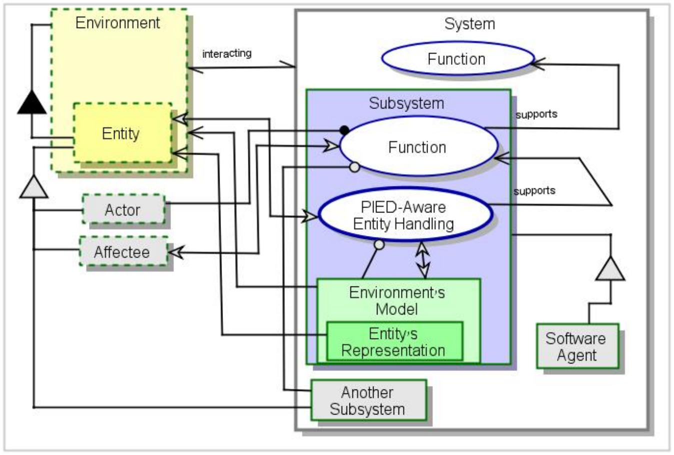

Fig. 4. System diagram of a PIED-aware system

Fig. 5 is the OPD in which **PIED-Aware Entity Handling** functional model is in-zoomed. The five subprocesses – **Entity Acquisition**, **Representation Generating**, **Interaction**, **Outcome Analysis**, and **Representation Improvement**, occur in a serial, cascading manner – i.e., the result of each subprocess conditions and leads to the next subprocess. Interaction is also considered part of the system's function. The model provides an enveloping setting for PIED-aware interaction. The interaction is based on the representation that the model maintains, i.e., what the system knows (or thinks it knows) about the external **Entity**.

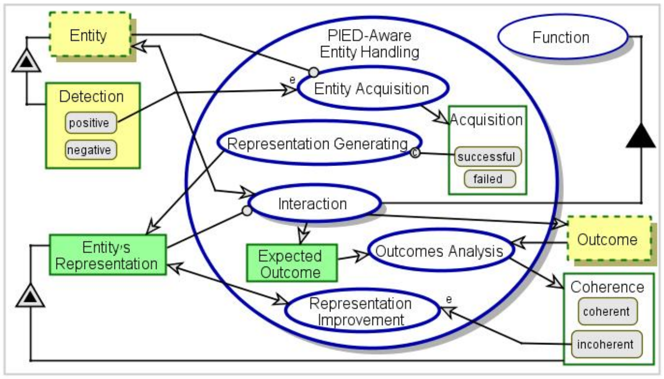

Fig. 5. PIED-aware entity handling process from Fig. 4 in-zoomed.

The **Entity Acquisition** and **Outcome Analysis** processes are zoomed into in Fig. 6 and Fig. 7. **Entity Acquisition** is critical for initial existence recognition of the **Entity**. The interaction cannot take place without preliminary **Entity** identification and its successful acquisition by the system.

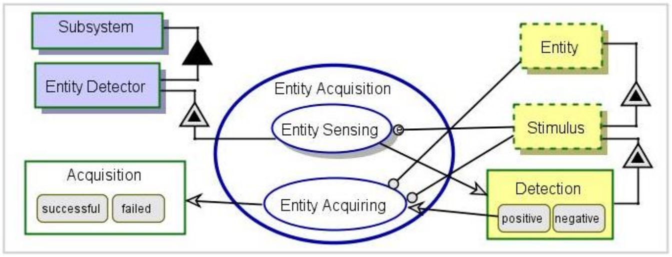

Fig. 6. Entity acquisition process diagram.

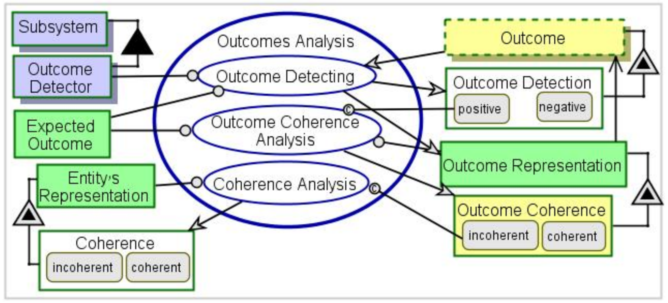

Fig. 7. Outcomes analysis process diagram

Outcome Analysis is critical for feedback and control. Outcome coherence inference is key to the preservation of the fidelity of the representation, although the entity and the outcome are not strictly coupled, i.e., the outcome does not necessarily impact the entity itself. A good example is the activation of protection means in response to identifying an intruder. The intruder is expected to be affected by the operation of the protection device. **Outcome Detecting** is critical to **Outcome Coherence Analysis**, which triggers the overall **Coherence Analysis** in case the outcome is **incoherent**. The main point is the criticality of correct detection of the outcome, internal outcome representation, and comparison of the outcome representation to the expected outcome. The outcome may not be detected immediately or reliably, or it may not be detected at all. In these cases, the system will not even

System exhibits PIED-Aware Entity Handling, and Function.
System consists of Subsystem, Entity's Representation, and Environment's Model.
System and Environment are interacting.
Software Agent is a Subsystem.
Actor handles Function.
Subsystem exhibits PIED-Aware Entity Handling and Function.
PIED-Aware Entity Handling exhibits Acquisition and Expected Outcome.
PIED-Aware Entity Handling consists of Entity Acquisition, Representation Generating, Interaction, Outcomes Analysis, and Representation Improvement.
PIED-Aware Entity Handling supports Function of Subsystem.
PIED-Aware Entity Handling requires Environment's Model.
PIED-Aware Entity Handling affects Environment's Model and Entity.
Acquisition can be positive or negative.
Entity Acquisition consists of Entity Sensing and Entity Acquisition.
Entity Acquisition requires Entity.
Entity Acquisition consumes positive Detection.
Entity Acquisition yields Acquisition.
Entity Sensing requires Stimulus.
Entity Sensing yields Detection .
Entity Acquisition requires Stimulus and Entity.
Entity Acquisition consumes positive Detection.
Entity Acquisition yields Acquisition.
Representation Generating occurs if Acquisition is successful.
Representation Generating yields Entity's Representation.
Interaction requires Entity's Representation.
Interaction affects Entity.
Interaction yields Expected Outcome and Outcome.
Outcomes Analysis consists of Outcome Detecting, Coherence Analysis, and Outcome Coherence Analysis.
Outcomes Analysis consumes Expected Outcome and Outcome.
Outcomes Analysis yields Coherence.
Outcome Detecting requires Expected Outcome and Outcome Detector.
Outcome Detecting consumes Outcome.
Outcome Detecting yields Outcome Representation and Outcome Detection.
Outcome Coherence Analysis occurs if Outcome Detection is positive.
Outcome Coherence Analysis requires Outcome Representation and Expected Outcome.
Outcome Coherence Analysis yields Outcome Coherence.
Coherence Analysis occurs if Outcome Coherence is incoherent. Coherence Analysis requires Entity's Representation.
Coherence Analysis yields Coherence.
Representation Improvement affects Entity's Representation.
Representation Improvement consumes incoherent Coherence.
Function of Subsystem is physical.
Function of Subsystem consists of Interaction.
Function of Subsystem supports Function.
Function of Subsystem requires Another Subsystem.
Function of Subsystem affects Affectee.
Subsystem consists of Environment's Model, Entity Detector, and Outcome Detector.
Environment's Model consists of many Entity's Representations.
Entity's Representation exhibits Coherence.
Coherence can be coherent or incoherent.
Coherence triggers Representation Improvement when it enters incoherent.
Entity's Representation relates to Entity.
Environment's Model relates to Environment.
Entity Detector exhibits Entity Sensing.
Function is physical.
Environment consists of Entity.
Entity is environmental and physical.
Entity exhibits Detection and Stimulus.
Detection can be positive or negative.
Detection triggers Entity Acquisition when it enters positive.
Stimulus exhibits Detection.
Stimulus triggers Entity Sensing.
Outcome Detection can be positive or negative.
Outcome Representation exhibits Outcome Coherence.
Outcome Coherence can be coherent or incoherent.
Outcome Representation relates to Outcome.

Fig. 8. PIED-aware metamodel – OPL specification

be able to understand that the interaction did not result in a desirable outcome, and will not be able to analyze the coherence of the entity.

External entities may overcome erroneous representation-based actions. If the external entity is friendly, it might provide a coherent response in spite of incoherent representation-based action. The system will not be able to know this, unless it is manifested by additional indicative effects. These effects must also be detected and handled. Recognizing representation-entity inconsistencies in spite of the entity's coherent response is an interesting challenge for future research.

OPM's freely available CASE tool, OPCAT [1] [29] automatically generates OPL sentences in response to visual model edits. The textual description is equivalent to the graphical view, allowing for integrated model understanding. Any model fact that appears at least once in any diagram of the model is valid throughout the entire model. Fig. 8 lists a partial OPL textual specification for the PIED-aware design pattern. This set of statements is machine-generated and machine-readable. Specifications appearing in several diagrams are defined only once, with no contradictions.

[1] Downloadable for free from http://esml.iem.technion.ac.il/

V. PIED-Aware Modeling of Anomaly Handling Mehcanisms: the Lost Baggage Handling Problem

In this section we exemplify PIED-aware modeling of a simplified airport baggage handling system. This is an example of a generic cyber-physical software-based system in which software agents control physical actuators and affect physical objects. It demonstrates the critical role of PIED-aware modeling in tackling complex cyber-physical anomalies, and can be implemented in various cases of similar nature.

PIED-aware modeling is critical for various scenarios deviating from the nominal "sunny day" scenario, in which everything works reliably and deterministically as planned. While the nominal scenario is what the system is built for, and what it is mandated to obtain, it cannot truly function appropriately without taking care of all the various anomalies that might arise during system operation, due to an ever-increasing number of failures, assumptions, misconceptions, or unpredictable events. Indeed, the larger part of system analysis work usually focuses on mitigating those rogue branches that might fail the system's function.

Lost baggage handling, a problem that many of us have experienced, cannot be adequately modeled without a PIED-awareness component. A naïve model fails to capture the baggage item's dual essence and consequently also a lost baggage situation. The baggage is 'lost' only from the passenger's point of view due to some incoherence between the whereabouts of the actual baggage and its record in the information system. It is quite common for baggage items to find their way into a wrong airplane and arrive in a completely different destination. In such cases, the misrouted baggage must be identified at the wrong destination and returned to its appropriate destination, where the passenger expects it.

Interestingly, when this problem was presented to groups of undergraduate and graduate students in information systems engineering courses, they did not make the PIED distinction at all; they only thought about baggage as an informatical object. This should not come as a surprise, though, because the PIED notion is not yet recognized by most systems and information engineering professionals, let alone students. And conversely, a mechanical engineer tasked to design the baggage conveying system in an airport could not care less about the informatical baggage object once it left the airport.

Baggage items are tagged with unique identity during the passenger check-in process. The baggage is then routed through several stations to the airplane the passenger is supposed to board. International flight regulations prohibit baggage leaving the airport when it is not on the same airplane as the passenger owning it, yet this happens all the time. Baggage can be stuck, delayed, or misrouted at any point along its journey. When baggage is reported as lost or found in a wrong destination, it is reported in the SITA World Tracer, an international baggage tracing IT service, accessible to all airports and airlines. If a passenger is the first to report the lost baggage (which is usually the case), the description and the baggage tag, which is also attached to the passenger's passport, are searched in a worldwide pool of lost baggage.

Tasked with modeling the lost baggage handling subsystem, we soon realized that it cannot be decoupled from the larger baggage handling system. We then realized that a single baggage entity cannot capture the composite state of the baggage in both the physical and informatical levels. Designing a software-controlled cyber-physical system requires deep understanding of the wider problem context. In the baggage handling case, the relevant context includes the entire airport and aviation operation. To avoid the need to design the whole airport so we can place a single process in its context, we must select the aspects that are critical to baggage handling. We also assume that PIED is applicable only to the baggage, and not to other cyber-physical objects in the environment, such as passengers and airplanes.

Fig. 9 is the top-level system diagram, SD, of the baggage handling system. It comprises both the physical segment, **Baggage Routing System**, and software segment, **Baggage Management Information System**. SD captures the physical interaction with the **Baggage** through the **Baggage Routing** process and the informatical interaction through the **Baggage Management** process. We also specify **iBaggage**, the informatical representation of the baggage.

Several **iBaggage** attributes are derived from the physical **Baggage** and from the **Passenger**. The relations between **Baggage** and **iBaggage** are elaborated in Fig. 10. The identity of the owner of **Baggage** and its origin and destination are derived from the **Passenger**. **Known Location reflects** the **Physical Location**. The **Identification Tag** of **Baggage** helps control it, via the **Identification Code** it displays, enriching the physical entity with informatical attributes.

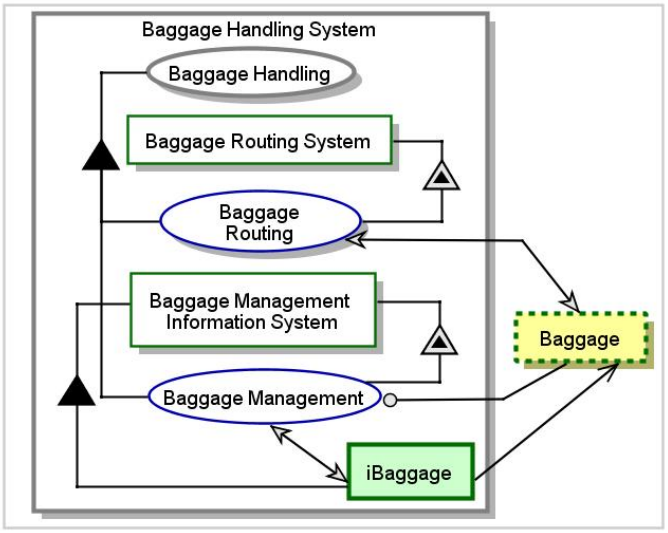

Fig. 9. Baggage Handling System

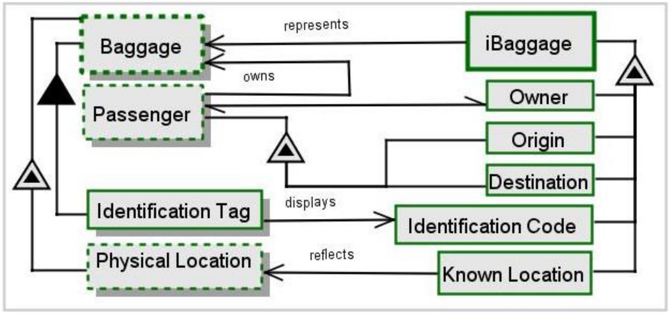

Fig. 10. Entity-representation relations for baggage

In order to create a PIED-aware model for baggage handling we have to capture the five PIED design pattern processes: **Entity Acquisition**, **Representation Generating**, **Interaction**, **Outcomes Analysis**, and **Representation Improvement**. These processes can be renamed to be more meaningful to the domain and system being modeled. The **Passenger Check-In** process at the origin airport, for instance, replaces ***Entity Acquisition***. This is where a ground attendant associates the baggage with the passenger in the airline information system and issues the identification tag, which is attached to the baggage. We can assume that at this point the physical and informatical manifestations are coherent. The **Baggage Registering** process, which is part of **Baggage Reception** shown in Fig. 11, replaces ***Representation Generating***. Here, **iBaggage**, the informatical representation of the physical **Baggage** entity, is created.

Baggage will henceforth be handled by the system according to its informatical representation and **known location**, which corresponds to the **physical location**. The **known location** attribute value will be updated by readings from detectors and scanners, which are positioned along the baggage routes and operated by baggage handling workers.

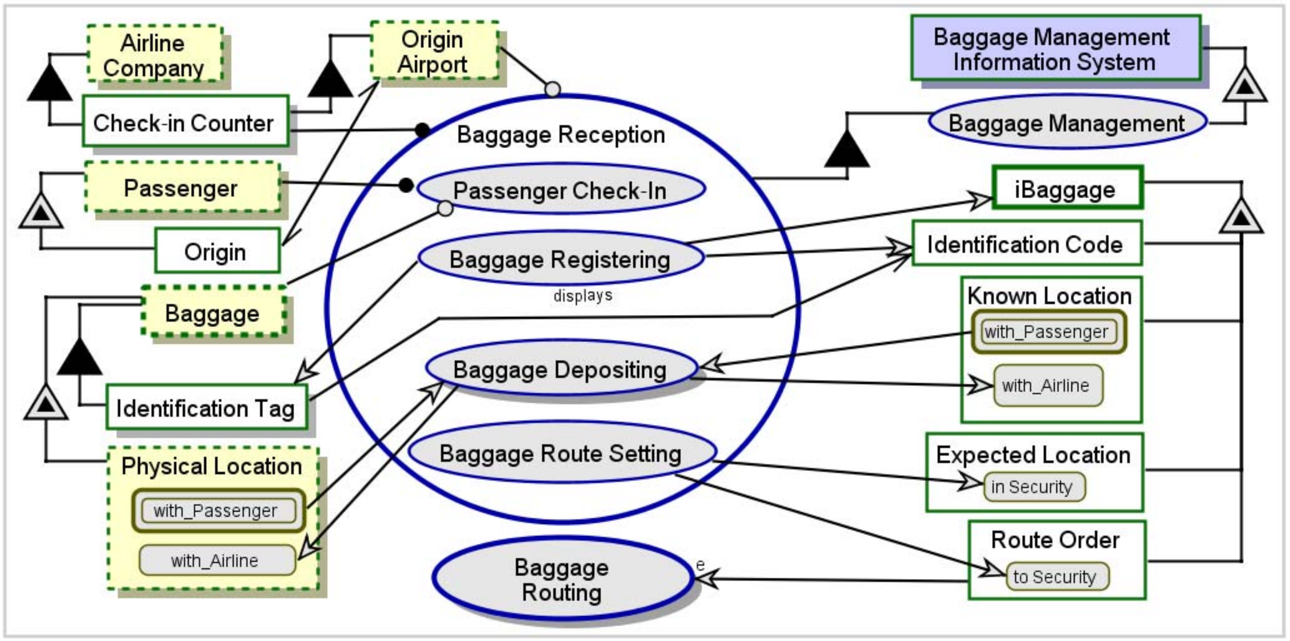

Fig. 11. Baggage Reception process – OPD

Baggage items may be missed by scanners due to various reasons, adversely affecting the fidelity of the location representation.

Each baggage reading activates a routing validation process, which is intended to make sure that the baggage is routed to its next designated station or destination and issue the appropriate routing orders to the operators of the physical segment of the system. The system has to determine if the baggage advances according to the prescribed routing. If the routing is in order, the system has no need to interfere. If, however, the baggage is routed wrongly, then corrective routing orders must be issued to the physical routing system. These can be implemented automatically or manually.

Baggage often shows up on a different airplane than the one it should have boarded on. If this is discovered after the intended aircraft already took off, then the system has to find the next flight to the intended destination and route the baggage to that flight, assuming the bound aircraft has room and time is not too tight. This is often done manually to ensure that the baggage boards as soon as possible. A baggage discovered only after landing must also be routed to the correct destination airport. A third option is that no reading triggers the detection of the problem, but the passenger reports the lost baggage at the destination airport counter.

Due to the decentralized nature of the baggage handling systems across airports, reports often do not automatically reach all the relevant nodes, and baggage might be located only after inquiry is set up in the SITA World Tracer system. If baggage is delayed at the origin airport, it is usually possible to issue corrective routing orders and have the baggage arriving on the next airplane. However, when baggage is transferred to a different airport or delayed in a connection airport, these airports may not be able to issue corrective routing orders before the baggage is reported as lost by the destination airport through, again via the SITA World Tracer system.

The **Baggage Routing Control** process is modeled in Fig. 12. The process can only take place if a **Physical Location Reading** of the baggage has occurred. Once such a reading arrives at the system, it updates the value of the **Known Location** attribute and compares it to the value of the **Expected Location** attribute. If the comparison yields a **coherent** result, the planned route continues. If the result is **unknown**, the system queries the **SITA World Tracer** for a **Lost Baggage** status. If the baggage is not reported as lost, the system sends the baggage directly to the local lost & found counter for further handling. It may include notification to SITA World Tracer and waiting for a complaint from the passenger, who is not yet aware of the baggage problem. If the baggage is indeed lost, the system retrieves the **Expected Destination** and triggers the **Corrective Route Setting** function, which takes all the known aspects into consideration, including the allocation of the baggage to the next feasible flight to the expected destination. **Corrective Route Setting** is also carried out immediately if the location is **incoherent** and **Expected Location** is already known, in which case the corrective routing can be made without SITA inquiries.

The various scenarios described here demonstrate the problem of knowledge-based decision making and entity handling in cyber-physical systems. Further complication arises from the distributed and decentralized nature of the system. The **Baggage Routing Control** includes **Outcome Coherence Analysis**, **Representation Improvement**, and entity handling (i.e., **Interaction**) aspects, all of which must be further described in lower levels or parallel diagrams. This process should apply to baggage in any knowledge-based context. The various contexts are combinations of the various

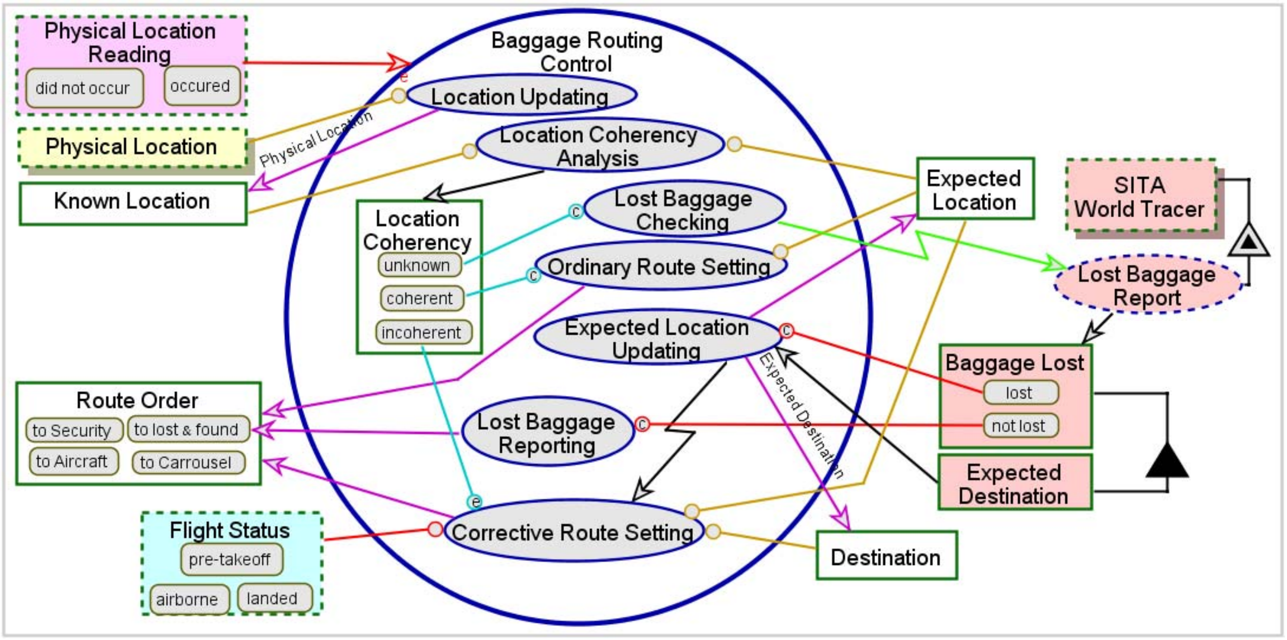

Fig. 12. Baggage Routing Control process – OPD

state and input variables affecting the dynamics of the process and eventually its control signals.

Due to space limitation, we cannot discuss issues which must be addressed as part of the overall system specification and design. We focused on processes which deliver the most significant value with respect to PIED-aware systems. The model is based on actual baggage handling processes with additional PIED-aware modeling notions. It can serve as a reference and a basis for further extension and elaboration in order to improve design accuracy and resolution. Such design should take into consideration the representation reliability and manifestation gaps introduced by the PIED problem.

VI. Summary

We address the problem of capturing cyber-physical knowledge gaps stemming from the physical-informatical essence duality (PIED) of cyber-physical entities from the perspective of constituent software control and monitoring agents. In order to cope with this problem, we propose to extend the system model to include the physical segment in addition to the software segment along with a well-defined distinction between an entity and its representation. This approach facilitates representation-based interaction between the software system and the external entity—a user or actor, another subsystem, an asset, or a resource.

The modeling includes representation generating, representation-based interaction, interaction outcomes analysis, and representation improvement. These are complex processes which consist of various activities, constitute parts of higher-level processes, and require a strong integrated functional-structural modeling capability. With this in mind, the difficulty of capturing complex cyber-physical interactions with UML or SysML-based models was demonstrated and discussed.

The OPM model we had previously proposed was shown to accommodate complex cyber-physical software-controlled processes. It is designed to resolve various complications and challenges stemming from integrating PIED awareness into already-complex models. The baggage handling example shows that in order to handle inherent system anomalies, such as lost or untracked physical objects, we have to model the entire system in a feasible and useful PIED-aware manner. When the cyber-physical environment tends to experience reliability issues and inconsistencies, the design pattern we propose becomes essential during system development and deployment.

PIED-aware modeling, design, and realization add a layer of complexity to the system. Yet, accounting for PIED contributes to increased performance level of safety-critical systems in terms of accuracy, fidelity, and reliability. As is the case with many other desired system '*ilities'*, the extent of a system's PIED-awareness is prone to be eroded by design compromises and managerial considerations.

Integrating PIED-aware design elements into existing system models that are not PIED-aware is challenging. OPM facilitates extending and modifying existing models to make them PIED-aware. Model adjustments include the addition of informatical representations for external entities and the processes that transform them during the system's operation. Interaction with external entities is based on entities' states as recorded by the interacting system's internal representation rather than on the actual state of the external entities, so incoherence issues must be addressed. These adjustments to existing models and implementations might require design and development changes in scope, and lead to program planning problems. However, failure to address these problems during design and development is a recipe for these problems to present themselves during system operation, often in the worst time possible and with potentially devastating consequences.

The more automated the system, the more it has to be mitigated by PIED-aware design so as to avoid scenarios that are unfortunately all too frequent in current systems.

Future research will include extensions of PIED-aware modeling and design and further analysis of real-world problems that might benefit from this approach. The ultimate goal is to enhance systems with software intelligence to compensate for reliability issues stemming from uncertainties that dominate the physical environment. Our vision is that PIED will become intuitive to designers just like notions like cardinality and inheritance are.

ACKNOWLEDGMENT

This research was partially supported by EU FP7 VISIONAIR Project under Contract # 262044. The authors would like to thank the IEEE CS anonymous referees for their useful comments.

REFERENCES

[1] Y. Tan, S. Goddard, and L. C. Pérez, "A prototype architecture for cyber-physical systems," ACM SIGBED Rev., vol. 5, no. 1, pp. 1–2, Jan. 2008.

[2] Y. Wang, W. Kinsner, and D. Zhang, "Contemporary cybernetics and its facets of cognitive informatics and computational intelligence.," IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics, vol. 39, no. 4. pp. 823–33, Aug-2009.

[3] K. Kolin, "Philosophy of Information and the Fundamentals of Informatics," in The Third International Conference "Problems of Cybernetics and Informatics," 2010, pp. 290–293.

[4] I. Hayes, M. Jackson, and C. Jones, "Determining the specification of a control system from that of its environment," in Lecture Notes in Computer Science: FME 2003: Formal Methods, K. Araki, S. Gnesi, and D. Mandrioli, Eds. Springer, 2003, pp. 154–169.

[5] Y. Mordecai, C. Chapman, and D. Dori, "Conceptual Modeling Semantics for the Physical-Informatical Essence Duality Problem," in IEEE International Conference on Systems, Man, and Cybernetics - SMC2013, 2013.

[6] Y. Mordecai and D. Dori, "I 5□: A Model - Based Framework for Architecting System - of - Systems Interoperability , Interconnectivity , Interfacing , Integration , and Interaction," in INCOSE Internaional Symposium, 2013, pp. 1–22.

[7] Y. Mordecai, P. Raju, C. Chapman, and D. Dori, "Physical-Informatical Essence-Duality-Aware Generic Modeling of Threat Handling Processes," in European Modeling Symposium - EMS2013, 2013.

[8] R. Tesoriero, R. Tebar, J. a. Gallud, M. D. Lozano, and V. M. R. Penichet, "Improving location awareness in indoor spaces using RFID technology," Expert Syst. Appl., vol. 37, no. 1, pp. 894–898, Jan. 2010.

[9] "Flight MH370: Chinese ships searching for objects in new area," BBC.com, 2014.

[10] J. Hintikka, "Individuals, possible worlds, and epistemic logic," Nous, vol. 1, no. 1, pp. 33–62, 1967.

[11] F. Van Harmelen, V. Lifschitz, and B. Porter, Eds., Handbook of knowledge representation. Elsevier, 2008.

[12] S. Mizzaro, "Towards a theory of epistemic information," Inf. Model. Knowl. Bases, 2001.

[13] Y. Bar-Hillel and R. Carnap, "Semantic Information," Br. J. Philos. Sci., vol. 4, no. 14, pp. pp. 147–157, 1953.

[14] K. Hayles, How we became posthumans. The University of Chicago Press, 1999.

[15] D. Dori, Object-Process Methodology: A Holistic Systems Approach. Berlin, Heidelberg, New York: Springer, 2002.

[16] C. Tomlin, S. Member, G. J. Pappas, and S. Sastry, "Conflict Resolution for Air Traffic Management□: A Study in Multiagent Hybrid Systems," vol. 43, no. 4, pp. 509–521, 2000.

[17] V. Jaiganesh, S. Mangayarkarasi, and P. Sumathi, "Intrusion Detection Systems□: A Survey and Analysis of Classification Techniques," vol. 2, no. 4, pp. 1629–1635, 2013.

[18] M. Chmielewski, A. Gałka, P. Jarema, K. Krasowski, and A. Kosiński, "Semantic Knowledge Representation in Terrorist Threat Analysis for Crisis Management Systems," pp. 460–471, 2009.

[19] G. J. Victor, M. S. Rao, and V. C. Venkaiah, "Intrusion Detection Systems - Analysis and Containment of False Positives Alerts," Int. J. Comput. Appl., vol. 5, no. 8, pp. 27–33, 2010.

[20] J. Estefan, "Survey of model-based systems engineering (MBSE) methodologies," 2007.

[21] D. Embley and B. Thalheim, Eds., Handbook of conceptual modeling: theory, practice, and research challenges. Springer, 2012.

[22] L. Favre, UML and the Unified Process. IRM Press, 2003.

[23] E. a. Lee, "Cyber Physical Systems: Design Challenges," 2008 11th IEEE Int. Symp. Object Component-Oriented Real-Time Distrib. Comput., pp. 363–369, May 2008.

[24] Z. Micskei and H. Waeselynck, "The many meanings of UML 2 Sequence Diagrams: a survey," Softw. Syst. Model., vol. 10, no. 4, pp. 489–514, Apr. 2010.

[25] F. Saltor, M. Castellanos, and M. García-Solaco, "Suitability of datamodels as canonical models for federated databases," ACM SIGMOD Rec., pp. 44–48, 1991.

[26] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati, "Information integration: Conceptual modeling and reasoning support," in 3rd IFCIS International Conference on Cooperative Information Systems, 1998.

[27] J. Bleiholder and F. Naumann, "Data fusion," ACM Comput. Surv., vol. 41, no. 1, pp. 1–41, Dec. 2008.

[28] N. Leveson, Engineering a safer world. MIT Press, 2011.

[29] D. Dori, C. Linchevski, and R. Manor, "OPCAT – An Object-Process CASE Tool for OPM-Based Conceptual Modelling," in 1st International Conference on Modelling and Management of Engineering Processes, 2010, pp. 1–30.