Patient coordination in emergency departments using an event-based information architecture

Kristofer Bengtsson, Bengt Lennartson

***Abstract*—It is challenging to get an overview and understanding of the activities and their relations at an emergency department (ED). This is due to the complicated relations among activities – called operations in this paper – and the always changing system behavior. Two key enablers to get this overview are the transformation of real-time events into understandable information and an operation-based behavior description. This paper presents an event-based information architecture for healthcare (EVAH) for gathering state changing events in real-time at an ED and how these are transformed into an operation-based representation. EVAH together with the tool Sequence Planner, will be used for visualization, online prognosis and optimization.**

I. Introduction

The problem of overcrowded emergency departments (EDs) [10] has led to increasing interest from the healthcare sector in studying industrial production methods and ideas. One example is the lean production philosophy [13], though the impact has so far been limited [16]. One of the main reasons for this is that production ideas assume a repeatable process and static operation behavior but where the ED operation behavior is highly changeable and variating. It it therefore important to give personal and patients an overview and understanding of the current and future operation behavior during the actual care as well as when planning and improving the work [6].

Healthcare research has been studying IT-support for decades [2], [15], includes a large variety of systems and functionality. One example is how to manage the patient record and the transition from paper-based to computer based IT and architype-based information structures [26]. However, few have been studying how to support the dynamic capabilities [30] of an ED, i.e. how to support an reactive and adaptive patient process control at chaotic emergency departments.

Dynamic capabilities is often referred to an organizations ability to adapt and reconfigure resources and processes to react to changing circumstances [31]. One key to achieve dynamic capabilities in turbulent environments like an ED is to utilize advanced IT support [11].

Bengtsson et al. [4] introduced a new approach for how to visualize the current and future situation and behavior at an ED, especially for the nurses working with the actual care. The current problem when trying to model health care behavior using standard workflow tools, Mans et al. [23] saw that these tools' inability to describe flexibility made it difficult to apply them in practice. Malhotra et al. [22] stated that: "an attempt to represent visually the workflow of a complex work environment such as that of a critical care setting is like working on a jigsaw puzzle with no pictures to guide you." This was managed by Bengtsson et al using simple operation building blocks and visualization algorithms introduced for industrial design in [3], [5]. Part of this approach is currently being implemented in a tool called Sequence Planner (SP).

One challenges however, when implementing this approach at an ED using SP, is how to gather the required real time data and how this data is transformed into usable information. Events and raw data need to be transformed into live information that can be used for making decisions.This paper therefore presents an event-based architecture for healthcare (EVAH) that will be used together with SP.

EVAH is event-based, has formalized transformation patterns, uses stream-based aggregation, and prototype-oriented information models. This makes EVAH able to handle the complex and changing processes at an ED and allowing a large diversity of communication devices and interaction with multiple IT-systems. Furthermore, EVAH gives the possibility to introduce new calculation and visualization algorithms not only based on new, but also on historical data. EVAH is inspired by event-driven architectures [24] which has been shown to support dynamic capabilities in health care [32].

In the next section, EVAH events and how they are sent out is defined. In section III, one part of EVAH called service endpoints is discussed and how they are enable dynamic capabilities. EVAH will be used for organizing the plan of the patient which is shown in section IV.

II. Event-based information architecture for healthcare (EVAH)

EVAH is based on a set of simple building blocks: A message bus, a message specification, service and communication endpoints, and EVAH events. These building blocks enable, in a modular and loosely coupled way, the creation and transformation of events into usable information. EVAH uses a message bus called Apache ActiveMQ [1] for sending and receiving events and messages. ActiveMQ is an Enterprise Service Bus (ESB) that supplies transformation and routing of data/information throughout several distributed applications.

A. EVAH Events

When something happens, for example when a patient is examined or someone in the staff goes for lunch, an event

K. Bengtsson, is with Sekvensa AB, Göteborg, Sweden, e-mail: kristofer@sekvensa.se

B. Lennartson is with the Automation Research Group, Department of Signals and Systems, Chalmers University of Technology, SE-412 96 Göteborg

can be sent out with information about the change. A EVAH event, which can only happen once, is defined as:

Definition 1 (EVAH events): $e = \langle id, t, KV \rangle$, where id is a unique identifier of the event, t is a timestamp, and $KV = \{attr_1 : value_1, \ldots, attr_k : value_k\}$ is a set of ordered attribute – value pairs describing the event and the state change. □

EVAH does not strictly define types or classes of events. Instead, a prototype-oriented approach is used [29]. Inheritance is managed by cloning an event, and similarities among events are identified based on the attribute – value pairs. This makes the event creation, identification and filtering more flexible and easier to change and update, since the strict hierarchical relation enforced by a class structure is removed. To reason about this, let us make the following definitions:

Definition 2 (Similar events): If $e_1 = \langle id_1, t_1, KV_1 \rangle$ and $e_2 = \langle id_2, t_2, KV_2 \rangle$, then e_1 and e_2 are similar if $KV_1 \cap KV_2 \neq \emptyset$. □

Definition 3 (Attribute pattern): An attribute pattern $ap = \langle KV_{ap}, AT_{ap} \rangle$ is a tuple including a set of ordered attributes – value pairs KV_{ap} and a set of attributes AT_{ap}. If $e_1 = \langle id_1, t_1, KV_1 \rangle$ such that $KV_{ap} \subseteq KV_1$ and $AT_{ap} \subseteq A_1$, where A_1 denotes all the attributes found in KV_1, then e_1 is matched by ap. This is denoted $e_1 \leftharpoonup ap$, □

An attribute pattern is used for matching, identifying, filtering, and creating events. In this article, a pattern is denoted e.g. $ap = \{attr_1 : value_1, attr_2 : value_2, attr_3 : \_\}$, where $KV_{ap} = \{attr_1 : value_1, attr_2 : value_2\}$ and $AT_{ap} = \{attr_3\}$. When the value is replaced with "_", then that attribute can have any value in the implementing event. Values can also be a list of attribute – value pairs or a list of values, hence a hierarchical data structure can be constructed.

Patterns can be defined freely by the user and are not enforced by EVAH. However, the receivers of the events will match events based on patterns, which makes the definitions important. Observe that this definition of events is somewhat modified compared to how the Discrete Event System (DES) community defines events [8] for Deterministic Finite Automata (DFA) [17]. It can be seen as a DES sample path extended with variables [8], where the variables are here called attributes. Before we study events and how they are used in EVAH, let us look at an example:

Example: In this example, a small emergency department section $yellow$ is studied. It includes patients $P_i, i = 1, \ldots, n$, a waiting room WR_1, a room R_1, a nurse N_1 and a doctor D_1. The patients are waiting for an examination (an $exam_i$ operation) and a blood test (a $test_i$ operation). Before these operations, the patient i needs to be located in the room. The execution of the operations are traced by the start and stop events $exam_i^{\uparrow}$, $exam_i^{\downarrow}$, $test_i^{\uparrow}$ and $test_i^{\downarrow}$, where $operation^{\uparrow}$ and $operation^{\downarrow}$ denote the start and stop event, respectively.

Other events are not directly related to the execution of an operation. These so called state events can for example be fired when a room is occupied, when a blood sample is completed, when someone in the staff is going for lunch or when the plan or record of a patient is updated. In the example there is an event fired when a patient is placed in a new location $patLoc_i$, when the staff is changed at the section $resA$, when a patient is assigned the section, $patA_i$, and when new operations are added to the patient plan $plan_i$. The events in this example are created based on some of the attribute patterns shown in Table I. □

TABLE I
SOME OF THE ATTRIBUTE PATTERNS USED FOR CREATING AND MATCHING THE EXAMPLE EVENTS

$exam_i^{\uparrow}$	name: $exam^{\uparrow}$ location: [$yellow$, R_1] resources: [D_1] patID: _	$test_i^{\downarrow}$	name: $test^{\downarrow}$ location: [$yellow$, R_1] resources: [D_1] patID: _ testDescr: _
$resA^{\uparrow}$	name: $resA$ assignTo: $yellow$ resources: _	$patA_i$	name: $patA$ assignTo: $yellow$ patID: _
$patLoc_i$	name: $patLoc$ location: _ rfid: _	$plan_i$	name: $plan$ patID: _ resources: [D_1] planUpdate: _

B. Communication Endpoints

The personnel, the patients and various information systems have limited knowledge about the surroundings and they communicate with the outside world in very different ways. The communication endpoints are responsible for transforming events – which are generated by humans or systems – into messages, and for sending events and receiving commands as events via the ESB.

The EVAH message is designed to be as simple as possible and will not enforce any type of specific attributes or structure. It consists of a header, with information related to the sending and routing of the message, and a body. The body is a key-value map between attributes (the keys) and their values. Values are usually of primitive data types like strings or integers, but can also be lists or maps, hence hierarchical structures can be built and sent in a EVAH message.

Example continued: In the example, there are two communication endpoints. One is connected to an RFID reader located in the room R_1 used for registering patient locations and start and stop of operations and the other is connected to the the patient record system. The order of events fired for patient 1 is shown in Fig. 1.

The patient is first assigned to the section $yellow$ notified with the event $patA_1$. Then first $patLoc_1$ is fired when the patient is assigned R_1 by N_1. After that the nurse performance a veni puncture and sends the blood sample of to the laboratory, identified by $test_1^{\uparrow}$ and $test_1^{\downarrow}$. The doctor examines the patient, $exam_1^{\uparrow}$ and $exam_1^{\downarrow}$ and decides that the patient need

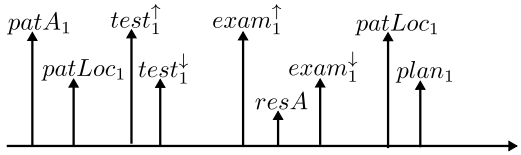

Fig. 1. The events fired related to patient 1. Note that the first and second $PatLoc$ are different events with the same name. They have different id, t and different attribute values.

an x-ray, $plan_1$. After the examination the patient is moved to the waiting room WR_1, identified by the $patLoc_1$ event. During the examination, the nurse N_1 is replaced by nurse N_2, identified by res_A. □

III. Service Endpoints

One big challenge when trying to create an information system for healthcare is to manage all the various types of events. Sometimes events are registered afterward, some events only include limited information, or operations are only defined by a single events after completion. To be able to use all these events for example to calculate various KPIs, it is therefore necessary to transform, update, and aggregate events.

A. Fill, Fold and Map

EVAH uses three fundamental types of transformations: Fill, Map, and Fold. Fill and Map are used for adding missing information to an event and Fold is used for transforming events sequences into messages or new events.

Definition 4 (Fill): A Fill transformation is a function that transforms $e = \langle id, t, KV \rangle$ by appending a set of new attribute – value pairs, i.e. $\langle id, t, \acute{KV} \rangle = Fill(e, ap)$, where $\acute{KV} = KV \cap KV_{ap}$ and $ap = \langle KV_{ap}, AT_{ap} \rangle$. □

The Fill transformation fetches information from a database or other type of static information that do not change over time based on the attribute template. The most common use cases are to fetch and include patient and staff information based on an id tag, or to fetch and include extra information about the sender of the event. The function will always return the same result unrelated to what has happen before.

In many cases, an event does not only need static information, but also values that are based on the current state of the system. If we study a system as a DES, a state can be identified based on an initial state and a sequence of events [8]. This is also true in the EVAH architecture. Let Σ^* be the set of all finite sequences of events over the set of all EVAH events Σ. Then, given a finite sequence $s \in \Sigma^*$ the state $q \in Q$ of the system is defined by $q = \delta(q^0, s)$, where q^0 is the initial state of the system and δ is the transition function of the system, defined as $\delta : Q \times \Sigma^* \to Q : (q^0, s) \mapsto \delta(q^0, s)$.

The state of a specific part of the system R, such as a patient or a resource, can also be identified by an event sequence. If we define R using an attribute pattern ap^R , then the current state of R is $q_R = \delta(q_R^0, s_R)$, where only events that matches ap^R are included in the sequence s_R. The Map transformation defined below permits to refine an event according to the current state of the system.

Definition 5 (Map): A Map transformation is a function that transform $e = \langle id, t, KV \rangle$ by appending a set of new attribute – value pairs based on the current state q, i.e. $\langle id, t, \acute{KV} \rangle = Map(e, q)$, where $KV \subset \acute{KV}$. □

Fill and Map can be used to transform events in multiple steps to simplify the implementation and to increase the changeability. But they do not change the unique identifier id of the event or the timestamp t. However, the transformation history and the event version are stored as attributes. The last transformation type is Fold, which takes a sequence of events s and transforms them into a new event or a single message that is sent out onto the message bus.

Definition 6 (Fold): A Fold transformation is a function that transform a finite sequence of events into either a single new event or a message, i.e. $message = Fold(s)$ or $e = Fold(s), s \in \Sigma^*$. □

Fold is often used to aggregate a set of events into a EVAH message. Fold transformations can also implement advanced event pattern identification algorithms like complex event processing (CEP) [21] or real-time languages [25]. CEP is a concept that tries to formalize how patterns and "knowledge" are identified from a flow of lower-level events, which are then sent out as higher-level events. [9]

Example continued: In the example, the $patLoc$ events only include the rfid value that represent the patient. These events will be updated by the Fill transformation with information from a database about the patient like name and personal security number.

Other transformations require knowledge about the current state. For example, a Map transformation service listens to the sequence of res_A and pat_A events to keep track on which staff is responsible for a section.

One Fold transformation tracks when a patient first enters the system and when it leaves, resulting in a message defining the throughput time of each patient. Another Fold transformation is tracking all operation events and combines start and stop events into an operation message which can, for example, include durations. There is also a Fold transformation that aggregates the section events over ten minutes and one hour, sending out a status message about behavior of the section including number of incoming and outgoing patient, number of operations, etc.

To summarize, the following transformations are used in the example:

- $\acute{e} = PatFromRFID(e, db)$. The transformation adds patient id from a database db to events that matches the attribute pattern $\{rfid\}$, i.e. $e \leftharpoondown \{rfid\}$ and $\acute{e} \leftharpoondown \{rfid, patID\}$.
- $\acute{e} = PatDetailFill(e, db)$. The transformation adds patient details from a database db to events that matches the attribute pattern $\{patID\}$, i.e. $e \leftharpoondown \{patID\}$ and $\acute{e} \leftharpoondown \{patID, patientDetails\}$.
- $\acute{e} = patientAssignmentMap(e, q^L)$, where $e \leftharpoondown \{assignTo : L\}$ and $\acute{e} \leftharpoondown \{assignMap\}$. For each section, found in $assignTo$ in events, a state q^L is

created, which maps each section to the nurses and patients currently assigned, $assignMap$. An event that includes the corresponding assignment, like $patA_1 \leftarrow \{assignTo : yellow\}$, will be updated with this map, i.e. after this Map transformation, $\acute{patA}_1 \leftarrow \{assignTo : yellow, assignMap : \{patients : [pat_1], resources : [N_1, D_1]\}\}$.

- $patientMessage = PatientFold(\{\forall e \in s | e \leftarrow \{patID : p_i\}\})$. This Fold transformation collects events related to the specific patient identifier p_i and, after the last event, sends out a patient message. The message includes, for example, the time of the first and last events, the sequence of all the positions visited by the patient instance P_i as well as the various waiting times, like time to doctor.
- $operationMessage = OperationFold(e_i \in \{O_i^{\uparrow}, O_i^{\downarrow}\})$. Collects the events from each operation and sends out an operation message.
- $sectionMessage = SectionFold(\{\forall e \in s | e \leftarrow \{locations : sid\}\})$. Collects events that matches a specific location sid and sends out a status message every 10 minutes and every hour about incoming and out going number of patients, number of patient movements, staff changes etc.

□

B. Dynamic capabilities using EVAH

Whang et. al [32] identifies that by using an event-based structure, the IT system enables sensing, responding, interoperability and flexibility capabilities. This leads to an increased dynamic capability.

The benefit of using Fill, Map, and Fold transformations is the increased flexibility they give for handling changes and updates. This is due to the loose coupling between sender and receiver and the possibility to allow a large variety of event structure and event generators.

Example continued: Let us extend the example with two more sections, $triage$ and $blue$. The patients in $triage$ is moved into $yellow$ or into $blue$. Patients can also be reassigned from $yellow$ to $blue$. By connecting the new sections to EVAH via communication endpoints, without changing any of the implemented service endpoints, the created messages will be correct and include the new layout. For example the $patintMessage$ will include all events from the newly added sections, including correct information about the longer lead time and new steps. Also $sectionFold$ will identify two new sections as soon as their first event will be fired and start sending out section messages for them.

Since these messages follow a structure, understood by the upper level information receivers, these upper services neither have to change. □

Even though the example may seem trivial this flexibility is in most cases not found in healthcare. Often, a point to point communication approach is used and the upper level systems require detailed understanding about the current layout. This makes it very time consuming to change the layout of the system or the process. Another important benefit of EVAH is the possibility to be flexible when calculating KPIs and tracking the plan of each patient, which is discussed in the next section.

IV. Operation-based process model

The most widespread tool for specifying and showing operation processes is probably the Gantt chart [33], which is easy to use and understand and intuitive to work with [18]. However, it was soon recognized that planning complex, large-scale systems was too complicated for the Gantt chart [33]. Examples of other tools that can be used are PERT charts [20], statecharts [14], Petri nets (PNs) [34], and workflow management tools [12].

These tools and modeling approaches are also used in healthcare. But they have three main problems: missing design rationale, inflexible during planning and execution and third – single view [5]. The most important aspect of being dynamic, flexible and proactive at an ED, is to have control over the plan of each patient. This is the core of operation based process models and execution.

A. Operation-based processes

The behavior of an operation O_k, can be represented by the state model depicted in Fig. 2. The initial location is denoted O_k^i, the executing location O_k^e, and the finished location O_k^f. The start transition between O_k^i and O_k^e is indicated by the start event, $O_k^{\uparrow}$, and the precondition by $C_k^{\uparrow}$. Finally, the stop event and the postcondition are denoted $O_k^{\downarrow}$ and $C_k^{\downarrow}$, respectively. The operations can be translated into an extended finite automaton (EFA) [27], a generalized automaton including variables, guards, and actions, and is described by Bengtsson [6].

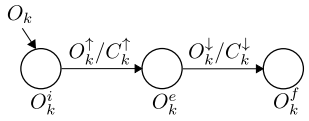

Fig. 2. A model of operation O

B. Operation relations and sequences of operations

To analyze and reason about the relations among operations, one approach is to examine the possible locations of an operation, related to when an event is enabled. An operation O_k will be located in one of its three locations when operation O_ℓ starts. By identifying the states where $O_\ell^{\uparrow}$ is enabled, the possible locations of O_k can be found. The possible relation types between two operations are:

Definition 7 (Relations between O_k and O_ℓ):

- Always in sequence: $O_k \succ O_\ell$
- Sometimes in sequence: $O_k \succsim O_\ell$
- Parallel: $O_k \| O_\ell$
- Alternative: $O_k + O_\ell$
- Arbitrary order: $O_k \oplus O_\ell$

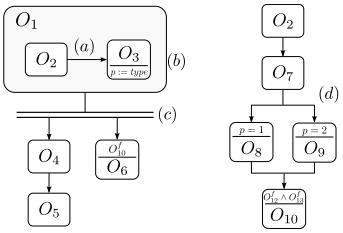

Fig. 3. Two sequences of SOP_1. Examples of SOP language notations: (a) sequence, (b) hierarchy, (c) parallel, and (d) alternative

- Hierarchy: $O_k \sqsubset O_\ell$
- Sometimes in hierarchy: $O_k \tilde{\sqsubset} O_\ell$
- Other: $O_k \curlywedge O_\ell$ □

When the relations have been pairwise identified, these relations needs to be visualized in some way. Here, a graphical language called Sequences of operations, SOP, is used, which is defined by Bengtsson et. al [3], [19]. In Fig. 3, an example SOP is shown including two sequences.

C. Sequence Planner and EVAH

Sequence Planner also include advanced features for matching operations with available resource abilities [5], creating optimization models [28] and control function generation [7]. Future functionality that is under development for EDs in SP are short term prognosis and room optimization as well as new visualization techniques for high-level overview. But to accomplish these implementation, the core behavior of tracking low level events for each patient and the patient plan need to be in place. This is accomplished using EVAH and current SP implementation.

Based on the operation model, it is possible to track the plan of the patient and current progress by listening to events from EVAH. When a $plan$ events arrive, a new operation is added including pre and post conditions to the patient. This operation is then matched with possible resource abilities and the SOP defining the plan of the patient is updated. When operation events are received by SP, the SOP is showing the current progress of the patient, including information about the performance of the ED.

In Fig. 4 various projection are shown. The top left SOP represents the possible route for patient P_1 when arriving at 11:40. On arrival, the patient is registered and a $plan$ event assign an $exam$ operation and the alternative discharge or admittance. These are matched and merged with resource abilities by SP. New operations are automatically added based on the $exam$ transition condition, for example that triage is needed due to overcrowding, and that the patient needs to be transported to one of the sections.

The $trige.*$ operation is marked red to show that the patient is waiting for that task and the $*$ define that the task can be executed by multiple resources, i.e. the task consist of the alternative between the two operation instances $trige.T1$ (Triage Team 1) and $trige.T2$ (Triage Team 2). This SOP will evolve based on the events read from EVAH. The SOP

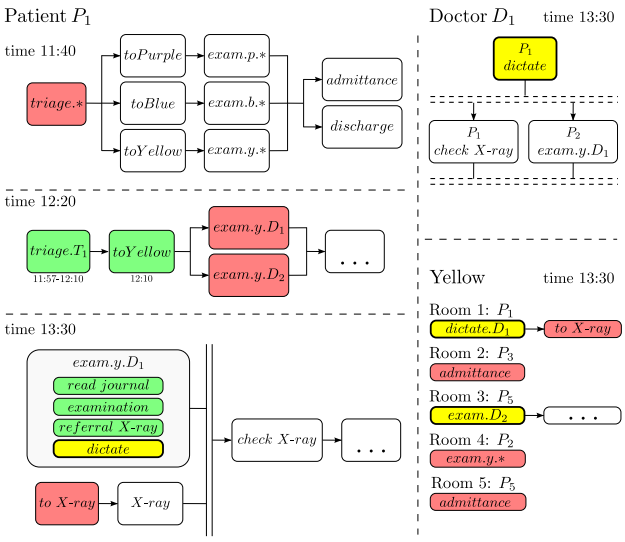

Fig. 4. Overview of Yellow section and Patient P_1

below show the status at 12:20 and at the bottom at 13:30. The green marked operations have been completed and their start and stop time is shown below. The patient was placed in the yellow section, which removes the other possible routes in the first SOP.

In the 13:30 SOP, the patient is currently being examined by doctor D_1, who has just written an referral to an x-ray examination and is currently dictating the examination. When the referral was written two new operations were added, $X\text{-}ray$ and $check\ X\text{-}ray$, and based on the $X\text{-}ray$ requirements, also $to\ X\text{-}ray$ was added. Since D_1 is not currently with the patient, it is possible to start $to\ X\text{-}ray$ directly.

The SOP at the top right shows the current and coming operations for D_1. To understand why a specific operation is not started, it is necessary to understand what the various resources are doing. The SOP will also give the personal guidance on what to do next. But maybe the most important projection to give an overview is the SOP in the lower right showing the patient in each room.

It is also possible to identify why an operation is not starting by studying its transition condition. By visualizing the relations between the studied operation and the sequences of operations that satisfies these condition, it is easy to understand what a patient is actually waiting for.

V. Conclusions

One important tool to handle overcrowding in emergency departments is to visualize the current system state and the future possible operation behavior. This visualization is highly complicated because the routing behavior is an indirect consequence of the requirements to start executing an operation, which involve, for example, the state of a resource, patients, or another operation. These operation requirements can result in many types of routing behavior, which will be almost impossible to describe in a graphical model.

This paper presents EVAH, which is an event-based architecture that offers flexibility and scalability and give the emergency department personnel a dynamic capability to react to variations.

The next important step is to use the gathered information to also calculate prognosis for the coming hours at the ED. This will help the staff to rapidly reorganize and adapt the resources and processes to better meet the patient needs.

REFERENCES

[1] The Apache Software Foundation. Apache ActiveMQ.

[2] M J Ball. An overview of total medical information systems. *Methods Inf. Med*, 10:73–82, 1971.

[3] K. Bengtsson, P. Bergagård, C. Thorstensson, B. Lennartson, K. Åkesson, C. Yuan, S. Miremadi, and P. Falkman. "sequence planning using multiple and coordinated sequences of operations". *IEEE Transactions on Automation Science and Engineering*, 9(2):308–319, 2012.

[4] K. Bengtsson and B. Lennartson. Patient coordination in emergency departments using visualization of operation behavior. In *2013 IEEE Symposium on Computational Intelligence, Singapore*, pages 58–63, April 2013.

[5] K. Bengtsson and B. Lennartson. Flexible specification of operation behavior using multiple projections. *Automation Science and Engineering, IEEE Transactions on*, 11(2):504–515, 2014.

[6] Kristofer Bengtsson. *Flexible design of operation behavior using modeling and visualization*. PhD thesis, Department of Signals and Systems, Chalmers University of Technology, Göteborg, Sweden, 2012.

[7] Kristofer Bengtsson, Bengt Lennartson, Oscar Ljungkrantz, and Chengyin Yuan. Developing control logic using aspect-oriented programming and sequence planning. *Control Engineering Practice*, 21(1):12 – 22, 2013.

[8] Christos G. Cassandras and Stéphane Lafortune. *Introduction to Discrete Event Systems, second edition*. Springer, 2008.

[9] Gianpaolo Cugola and Alessandro Margara. Processing flows of information: From data stream to complex event processing. *ACM Comput. Surv.*, 44(3):15:1–15:62, 2012.

[10] Robert W. Derlet and John R. Richards. "overcrowding in the nation's emergency departments: Complex causes and disturbing effects". *Annals of emergency medicine*, 35(1):63–68, 2000.

[11] Omar A El Sawy and Paul A Pavlou. It-enabled business capabilities for turbulent environments. *MIS Quarterly Executive*, 7(3), 2008.

[12] Diimitrios Georgakopoulos, Mark Hornick, and Amit Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3:119–153, 1995.

[13] Mark Graban. *"Lean Hospitals"*. Taylor & Francis Group, 2009.

[14] D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.

[15] Reinhold Haux. Health information systems - past, present, future. *International journal of medical informatics*, 75(3):268–2281, 2006.

[16] Richard J. Holden. "lean thinking in emergency departments: A critical review". *Annals of emergency medicine*, 57(3):265–278, 2011.

[17] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison Wesley, 2006.

[18] H. Kerzner. *Project Management: A Systems Approach to Planning, Scheduling, and Controlling, Ninth Edition*. J. Wiley & Sons, 2006. ISBN 0471741876.

[19] B. Lennartson, K. Bengtsson, C. Yuan, K. Andersson, M. Fabian, P. Falkman, and K. Åkesson. Sequence planning for integrated product, process and automation design. *IEEE Transactions on Automation Science and Engineering*, 7(4):791–802, 2010.

[20] R. Levin and C. Kirkpatrick. *Planning and Control with PERT/CRM*. McGraw-Hill, 1966.

[21] David Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Professional, 2002.

[22] Sameer Malhotra, Desmond Jordan, Edward Shortliffe, and Vimla L. Patel. Workflow modeling in critical care: Piecing together your own puzzle. *Journal of Biomedical Informatics*, 40(2):81 – 92, 2007.

[23] R. Mans, W. van der Aalst, , and N. Russell. "implementation of a healthcare process in four different workflow systems". Technical report, Technische Universiteit Eindhoven, 2009.

[24] Brenda M Michelson. Event-driven architecture overview. *Patricia Seybold Group*, 2, 2006.

[25] J. Perez, J. Jimenez, A. Rabanal, A. Astarloa, and J. Lazaro. FTL-CFree: A fuzzy real-time language for runtime verification. *Industrial Informatics, IEEE Transactions on*, 2014.

[26] P Schloeffel, T Beale, G Hayworth, S Heard, and H Leslie. The relationship between cen 13606, hl7, and openehr. In *HIC 2006 and HINZ 2006 Proceedings*, pages 24–28, Brunswick East, Vic.: Health Informatics Society of Australia, 2006.

[27] M. Sköldstam, K. Åkesson, and M. Fabian. Modelling of discrete event systems using finite automata with variables. In *Proc. 46th IEEE Conference on Decision and Control*, New Orleans, USA, Dec 2007.

[28] Nina Sundström, Oskar Wigström, Petter Falkman, and Bengt Lennartson. Optimization of operation sequences using constraint programming. In *14th IFAC Symposium on Information Control Problems in Manufacturing, INCOM'12*, Bucharest, 2012.

[29] A. Taivalsaari and I. Moore. *Prototype-Based Object-Oriented Programming: Concepts, Languages, and Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 2001.

[30] David J Teece, Gary Pisano, and Amy Shuen. Dynamic capabilities and strategic management. *Strategic Management Journal*, 18(7):509–533, 1997.

[31] Catherine L. Wang and Pervaiz K. Ahmed. Dynamic capabilities: A review and research agenda. *International Journal of Management Reviews*, 9(1):31–51, 2007.

[32] Y Whang, L Kung, and T Byrd. Leveraging event-driven it architecture capability for competitive advantage in healthcare industry: A mediated model. In *Thirty Fourth International Conference in Information Systems*, Milan, 2013.

[33] James M. Wilson. Gantt charts: A centenary appreciation. *European Journal of Operational Research*, 149(2):430 – 437, 2003.

[34] M. C. Zhou and F. DiCesare. *Petri net synthesis for discrete event control of manufacturing systems*. Kluwer Academic Publishers, 1993.