Patient coordination in emergency departments using visualization of operation behavior

Patient coordination in emergency departments using visualization of operation behavior

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. This paper presents a method to represent and visualize operations during the actual work at the ED that enables flexibility during modeling and actual care. By specifying operation behavior using only execution restrictions in transition conditions for each operation instead of fixed sequences, the method and related visualization tool can describe the ever changing and complex ED flows. Using the proposed approach, the ED staff will therefore have the possibility to in a better way coordinate the operations due to the increased overview.**

I. Introduction

The problem of overcrowded emergency departments (EDs) [7] has led to increasing interest from the healthcare sector in studying industrial production methods and ideas, such as lean production philosophies [9], though the impact has so far been limited [12]. 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. Therefore, this paper presents a method to give personal and patient an overview and understanding of the complex operation behavior during the actual care as well as when planning and improving the work.

When trying to model health care behavior using a number of workflow tools, Mans et al. [19] saw that these tools' inability to describe flexibility made it difficult to apply them in practice. Malhotra et al. [18] 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." A number of papers have tried to utilize information and control technology to model and automate the control of operations in healthcare systems, e.g. [11], [6], [4], but most solutions had problems handling the need for flexibility.

Instead this paper focuses on how to visualize and give an overview of the current and future situation at an ED, especially for the nurses working with the actual care. This overview and understanding is crucial when trying to coordinate the work. A method is presented in this paper that describes operations and automatically visualizes them during planning, actual execution and when studying the event history. The method is an adaptation of the modeling method presented by Bengtsson et. al [3], [16]. That modeling method initially targeted the design process to develop automation system and control code, but is also useful for keeping track of all the patients at an emergency department as will be presented here.

Operations are referred to by various names in literature, like activities, tasks or actions. In the rest of this paper, the name operation is used to comply with previous work by the authors. Hence, operation, task, action or activity can be used interchangeable in this paper.

The modeling in [3] includes a set of operations and their transition conditions, together with entities, resources and variables - where the execution restrictions are specified using transition conditions for each operation. This enables the possibility to create multiple projections or multiple visual views of the operation relations to enable better understanding. This is accomplished by creating various Sequences of Operations including a multiplicity of sequences and operation relations. The key discoveries that accomplishes this, are the state-based operation relation identification and visualization methods. Most of the presented models and algorithms presented in this paper have been implemented in an prototype tool called sequence planner, see fig 1.

This paper defines a generalization of the modeling method to also include the "online" execution behavior that will be used at an ED. In the next section, related work is discussed together with a synthesis of the fundamental challenges. The extended modeling method is presented in Section III together with the visualization methods in Section IV. A case study from an ED is presented in Section V.

II. Modeling operation processes

The most widespread tool for specifying operations is probably the Gantt chart [27], which is easy to use and understand and intuitive to work with [14]. However, it was soon recognized that planning complex, large-scale systems was too complicated for the Gantt chart [27]. The chart does not really represent logical relations among system elements, but rather their durations and start and stop times. For example, Gantt charts cannot handle alternatives and arbitrary order or represent interdependent and network-like relations.

Since the relation among operations are fundamentally logical, planning using Gantt charts alone can worsen planning and execution problems. This lack of flexibility is also found in other process planning tools, such as PERT charts [17], statecharts [10], sequential function charts (SFCs) [13], Petri

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

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

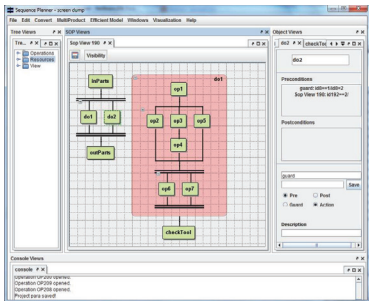

Fig. 1. The prototyp tool sequence planner

nets (PNs) [28], and workflow management tools [8]. All of these attempt to describe operation behavior using process models in various ways. However, as will be argued in this paper, that is an inflexible and limiting approach.

Many workflow management tools are based on Petri nets [23] and are used to model and manage business processes. They are commonly used to automate information handling in computer systems [8]. Numerous tools are available for modeling and executing workflows, tools such as Tibco Staffware, IBM Websphere, YAWL, and SAP Workflow (compared in van der Aalst et al. [25]).

A workflow describes the routing of a case by means of a sequence of tasks [24]. The term "case" refers to a specific instance of a workflow model representing a job, product, service, or item, and "task" refers to a "logical unit of work" [24] carried out by one resource. The challenge when modeling workflows is modeling the many possible routing behaviors. Van der Aalst et al. [25], [1] identify workflow patterns expressing numerous routing behaviors, for example, branching, synchronization, iteration, cancellation, multiple instances, resource, and data interaction.

The main advantage of workflow models is that routing behavior can be expressed graphically by adding lines and arrows. This is very attractive for small systems, since an explicit routing is shown graphically, enabling easy understanding. Unfortunately, these graphical descriptions soon become messy, due to the complexity of real routing behaviors. This is evident when considering the increasing number of "discovered" patterns [1].

These modeling approaches have three main problems: missing design rationale, inflexible during planning and execution and third – single view. The design rationale describes the reasoning underlying decisions made during planning and actual execution of the process [2], i.e., *why* a specific operation ordering is defined. However, when modeling the routing behavior, the design rationale is often not explicitly stated, making it impossible to understand why one particular operation is executed after another. This occurs because the sequence of operations is related to something external to the graphical model, meaning that the routing behavior is also related to something external.

Most modeling languages explicitly represent every possible route. However, during the planning the number of possible routes is enormous and, moreover, these routes frequently change during execution. This either makes the modeling tools unusable during the planning process or leads to design restrictions so that the plan fits the model. The latter is actually the most common outcome, which is evident when trying to follow the planned operation sequence in practice.

The last problem with modeling tools is that usually the modeler needs to decide what "view" to use, for example, whether the model displays the patient routing or the working order of a resource. However, to design complex systems, it is necessary to consider multiple views of various entities, resources, functionalities, etc.

These problems arise 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, entity, 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. A better approach is therefore needed when planning and executing operation and routing behavior.

All these modeling challenges becomes even more evident during the actual execution of the operations. At an emergency department, the number of possible patient routes through the system are enormous and many operations are going on at the same time [5], [15]. Therefore it is important to have an overview and understanding of the operation behavior during the actual care.

The proposed approach is to specify the operations and their execution constraints and not explicitly specifying all possible operation sequences. This retains full flexibility during the planning phase and allows the design rationale to be tracked, since the reason for a specific execution order is available in the specifications. This flexibility will also be available during the operation execution.

The problem though, with this flexible approach is that it makes it difficult or even impossible to understand the routing and operation behavior. Therefore, methods and tools that can automatically create various views of current behavior based on the constraints are presented in this paper.

III. Modeling operation behavior

The presented modeling approach is an extension of the work by Bengtsson et. al [3], [16]. Some parts of the approach is still the same as in these papers, for example how the operations can be translated into an extended finite automaton (EFA) [21], a generalized automaton including variables, guards, and actions, and is therefore left our of this paper. Formal EFA tools are available (see [22], [20], [26]). Hence, both formal verification and synthesis can be applied directly in the suggested EFA models.

The behavior of an operation model, O, can be represented by the state model depicted in Fig. 2. The initial location is denoted O^i, the executing location O^e, and the finished location O^f. The start transition between O^i and O^e is indicated by the start event, $e^{\uparrow}$, and the precondition by $O^{\uparrow}$.

Finally, the stop event and the postcondition are denoted $e^{\downarrow}$ and $O^{\downarrow}$, respectively.

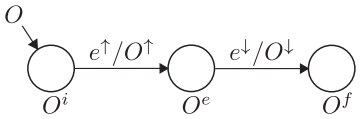

Fig. 2. A model of operation O

An operation can also be defined by a set of sub-operations describing the detailed behavior of the operation. These operations are included in a sequences of operations (SOP) structure, which will be discussed later. Before presenting more details of the operations, let us first discuss entities and resources.

A. Entities and resources

Operations describe the behavior of a system, where the system is specified by entities and resources. Entities (e.g., patients, components, documents, blood samples, or customers) are objects that need to be transformed by operations to reach a complete state. Resources are defined as objects that aid in the transformation of entities; resources include machines, people, tools, computers, and places. Resources that can perform operations have a set of abilities defining what type of operation they can perform, where an ability is defined as an operation model.

An entity or a resource, E, can be represented by a set of state variables, i.e., $E = \{x_1, \ldots, x_n\}$. A single state variable represents a specific aspect of E, for example, a door can have a state variable x, where the domain of x is denoted X with the values $X = \{opened, closed\}$), representing the positioning aspect of the door. A door can also have state variables representing other aspects, for example, whether it is mounted, in what manufacturing step it is located, or whether it is locked. What state variables to use and what values they can take depend on the intended use of the system model.

The purpose of a system is to transform the state variables of the relevant entities such that a completed state is reached. A completed state is reached when every state variable is assigned a marked value, where one or many of the values in the variable domain can be marked. The marked value of state variable x is denoted X^M, where $X^M \subseteq X$. State variables that are not relevant to a completed state are called *unmarked*, though all their values are actually marked. This requires a clear definition to avoid confusion:

Definition 1 (Marked and unmarked state variables): State variable x is referred to as **marked** if $X^M \subset X, X^M \neq \emptyset$, i.e., at least one but not all variable values are included in the marked set. State variable y is referred to as **unmarked** if $Y^M = Y, Y^M \neq \emptyset$, i.e., all variable values are marked. □

The difference between entities and resources related to their marked values are that all the state variables of a resource are unmarked. Observe, though, that in some cases a resource can be an entity as well, and therefore have marked state variables, for example, a room that needs to return to empty state at the end of a cycle, which can be modeled by a marked state of the room.

A state variable can also be shared among entities and resources when it represents a shared aspect or interaction. Shared state variables are specified by an interaction, defined by $I =< I^V, I^E >$, where I^V is a set of state variables and I^E is a set of entities and resources that receives the state variables.

B. Operation modeling using transition conditions

The pre- and postconditions, i.e., $O^{\uparrow}$ and $O^{\downarrow}$, of operation O constitute the core of the operation. Requirements and constraints related to the operation are specified using these transition conditions. A guard expression can contain statements including state variables or operation states. The state variables used in the statement can refer to both the current value, x, and the next value, x'. The statements are separated by $\wedge$ (AND) or $\vee$ (OR), where a statement can include standard operators such as $=, \neq$, and $<$. A statement that refers to the state of an operation can be expressed using the current location of an operation, O^i, O^e, or O^f.

The conditions also includes transition actions that updates state variables. A state variable can be assigned a direct value (e.g., $position := closed$, $x := 4$) or the value of a function or another variable (e.g., $x := y + z$).

C. Resource abilities

A *transformation operation* is an operation that directly changes the value of a marked state variable. But to execute a transformation operation, it must be matched and merged with a resource ability. Then, a transformation operation instance is created. During this matching process, multiple operation instances can be created, since many abilities may be matched with the same transformation operation. For example, multiple resources may be able to weld two parts together, so one operation instance per resource will be created.

The created operation instances will also have other types of requirements and constraints. They will, for example, require that before examine a patient, the patient needs to be in a examination room. Therefore, other types of operations are also needed by the system, such as transportation, registration, resource motions, safety tasks, and communication. Each transformation operation instance requires a set of these other operations, based on what the instance has specified in its transition condition.

By attempting to satisfy requirements in the transition conditions, new operations can be recursively added to the system. When all requirements are satisfied by at least one other operation, it is possible to execute the operation. However, since it is difficult to completely understand what is going, it is necessary to use a graphical representation to better understand the system and its operations.

IV. VISUALIZATION

Each operation will start in its initial location and wait for its precondition to be fulfilled. If there are no preconditions, all operations will execute unrelated to each other. In practice, a number of conditions will restrict this behavior. If an operation includes the state of another operation in a transition condition, for example, $O_\ell^\uparrow \triangleq O_k^f$, then the two operations are *directly related*. For example, it is obvious from studying the two operations that O_k will always execute before O_ℓ. In addition, if $O_k^\uparrow \triangleq x = 0; x := 1$ and $O_\ell^\uparrow \triangleq x = 1$, then O_k, and O_ℓ will be directly related. However, most operations will not be directly related to each other, even though they are related in some way. Consider, for example, O_k, O_ℓ, and O_m, where $O_\ell^\uparrow \triangleq O_k^f$ and $O_m^\uparrow \triangleq O_\ell^f$. Then O_k will always precede O_m even though this is not obvious from examining only O_m and O_k; these are *indirectly related*.

A. 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 $e_\ell^\uparrow$ is enabled, the possible locations of O_k can be found. A set denoted $O_k^{e_\ell^\uparrow}$ is created, which can be one of the following seven location combinations $\{O_k^e\}$, $\{O_k^f\}$, $\{O_k^i, O_k^e\}$, $\{O_k^i, O_k^f\}$, $\{O_k^e, O_k^f\}$, and $\{O_k^i, O_k^e, O_k^f\}$. For example, if $O_k^{e_\ell^\uparrow} = \{O_k^f\}$, then operation O_ℓ will only start when operation O_k is in its final location.

To define the possible relations between operations O_k and O_ℓ, all four location sets, i.e., $O_k^{e_\ell^\uparrow}$, $O_k^{e_\ell^\downarrow}$, $O_\ell^{e_k^\uparrow}$, and $O_\ell^{e_k^\downarrow}$, must be identified and compared. Observe that $O_k^{e_\ell^\uparrow}$ is the set of possible locations of O_k when the start event $e_\ell^\uparrow$ of O_ℓ is enabled, i.e. when O_ℓ has the possibility to start. The possible combinations of these state sets can be grouped into the following relation types:

Definition 2 (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: $O_k \oplus O_\ell$
- 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. The SOP, SOP is a model that represents a set of operations, $SOP^{\mathcal{O}} \subseteq \mathcal{O}$, their relations, SOP^R, and a set of sequences, SOP^S. The $SOP^{\mathcal{O}}$ operations are grouped into a SOP for various reasons, for example, that they are executed by the same resource, related to a specific patient, or related to a diagnosis. A single operation can also be included in multiple SOPs. The main reason for grouping these operations is to be able to consider the relations among the included operations, which are defined by relations $SOP^R = \{< O_1, rel, O_2 >, < 0_1, rel, O_3 >, \ldots, < O_n, rel, O_{n-1} > \{$, where rel is a relation from Definition 2, which represents the pairwise relation between operations. The final part of the SOP is the sequences. Sequence $s \in SOP^S$ is a graph that connects set $s^{\mathcal{O}} \subseteq SOP^{\mathcal{O}}$ of operations to visualize their relations, which is done using arrows, lines, and Boolean expressions.

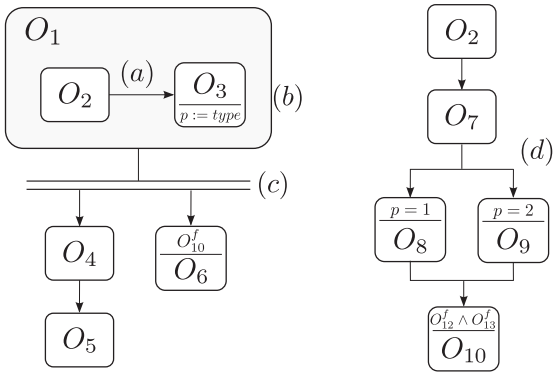

Fig. 3. Two sequences of SOP_1. Examples of SOP language notations: (a) sequence, (b) hierarchy, (c) parallel, and (d) alternative

In Fig. 3, SOP_1 is shown including two sequences. The arrow at (a) corresponds to the sequential relation, always in sequence, between O_2 and O_3. Operation O_1 consists of two sub-operations, O_2 and O_3, denoted by the gray box (b). Pre- and postconditions can also be shown as Boolean expressions, for example, as the postcondition $p := type$ for O_3, which is an action that assigns variable p the value of another variable, or like the precondition O_{10}^f for O_6.

The parallel lines at (c) represent a parallel relation, meaning that O_4 and O_6 have a parallel relation and that they are both always in sequence with O_1. An alternative branch is shown at (d), where either O_8 or O_9 is executed. The same operation can occur in multiple sequences, but only once per sequence, for example, O_2. More details of the SOP language and notations can be found in [16], where also the graphical notation for arbitrary order and loops is defined. Algorithms to create sequences based on operation relations are presented in [3].

V. PATIENT COORDINATION STUDY

An emergency department at the hospital in Trollhättan, Sweden, was studied during one week. The conclusion was that it was hard but important to get an understanding of the operation behavior and that the care of patients at an emergency department includes an aspect of development, i.e. the care of a patient needs to be developed uniquely for that patient.

The ever repeating design and improvement cycle consist of four stages; planning, doing, checking and adjusting. Currently different tools and operation behavior representations

are used in each stage, often with limited interconnection between the representations. To better understand the operation behavior during the planning phase of the ED work, sometimes value stream mapping [9] or workflow models are created. But due to the limitations of these tools, the created models are hard to use for aiding the actual work with the patients in the doing stage. And the few tools that can be used during the doing phase, they are not subtitle to use for follow up and improvement work in the checking and adjusting phase.

In the following example, which is a modified real life situation, the operation model and the projection tools are used in all four stages.

Planning: The emergency department (ED) at the hospital in Trollhättan receives around 150 patient per day. This study was conducted at the internal medicine specialty consisting of the three sections Purple, Blue and Yellow. Each section has two physicians, two nurses and a set of rooms. A section examines the patients, initiate treatment and conducts testing and patient care until the patient either is discharged from the hospital or admitted for hospital care. Before a patient is assigned to a section, a triage team tests the vital parameters of the patient and assign an emergency priority. There are two triage teams at the ED.

During the planning phase, managers and ED coordinators plan for the coming months and updates and improvements of the work. Various patient types are considered and assigned a set of fundamental operations. For example, most patients will always have the operations registration, triage, examination, and discharge or admittance. To these fundamental operations, a large set of other operations are also added or updated, like tests, X-ray, patient care, and various treatments, based on the decisions taken by the ED personal.

All the resources and their abilities are also defined and updated during the planning phase. Based on the defined data it can be possible to simulate the patient flow in the ED by assigning estimated operation durations. But most importantly, the model can be used by the personal at the ED when they are doing the real work.

Doing: One of the most obvious observations during the study at the ED, was the challenge for the personal to get an overview of the patient flow, especially when the ED was overcrowded with patients. During the peak time of the day, each of the three sections could have 10-15 patients at the same time, making the work very fragmented. Getting an overview of the all these patients, not only by the personal at each section but also the coordinators at the ED, is important both for current work and for improvements. It is also important for the patients to better understand what is going on and why they are waiting.

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 assigned an $exam$ operation and the alternative discharge or admittance. These are matched and merged with resource abilities. 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.

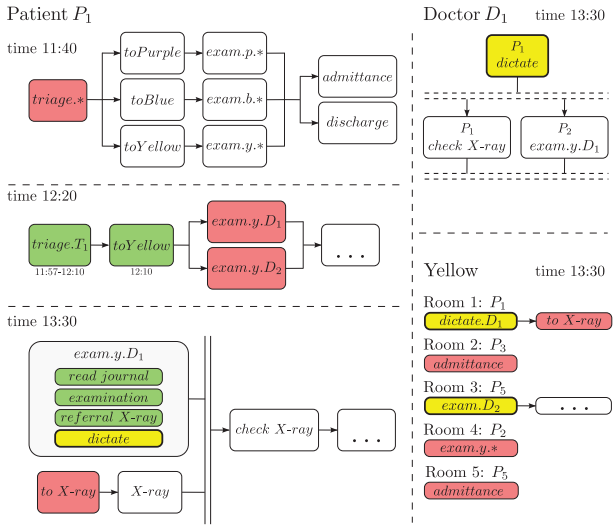

Fig. 4. Overview of Yellow section and Patient P_1

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 with the patient and operations will be added and removed based on the patient examination. The SOP 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.

Checking and Adjusting: When something happens at the ED, for example if the waiting time dramatically increased a

specific day, it is important to be able to go back and check all the operations. By storing the start and stop times for each operation, it will be possible to generate gantt charts based on patient flow or resource work. It is also important to be able to "play-back" the changes over time in the same format used during the doing phase.

Since the same format are used in all phases it is easier for the working personal to understand and improve their work by updating the model and their working routines. This has not be tested during the study since it requires a well developed user interface and interconnected tool with other information systems. But the study seem to confirm that it will be possible to use the methods and tools presented in this paper.

VI. 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 a flexible modeling and online execution model that only defines operations and their execution constraints. The operation sequences and routing behavior are then automatically visualized from multiple perspectives, e.g. the operation sequence related to a specific patient or the operations available for a doctor.

The next important step is to have a good user interface and intuitive tools. Therefore this research will continue with developing a tool called Sequence Planner, which already have implemented most of these methods.

References

[1] Workflow patterns. On the WWW, Jun 2012. http://www.workflowpatterns.com.

[2] F. Andersson. *"The dynamics of requirements and product concept management"*. PhD Thesis, Chalmers University of technology, Göteborg, Sweden, 2003.

[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] Morad Benyoucef, Craig Kuziemsky, Amir Afrasiabi Rad, and Ali Elsabbahi. Modeling healthcare processes as service orchestrations and choreographies. *Business Process Management Journal*, 17(4):568–597, 2011.

[5] L. Berg, A. Ehrenberg, J. Florin, J. Östergren, and K. Göransson. An observational study of activities and multitasking performed by clinicians in two swedish emergency departments. *European Journal of Emergency Medicine*, 19(4):246–251, 2012.

[6] Jiangbo Dang, Amir Hedayati, Ken Hampel, and Candemir Toklu. An ontological knowledge framework for adaptive medical workflow. *J. of Biomedical Informatics*, 41(5):829–836, 2008.

[7] 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.

[8] 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.

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

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

[11] R. Haux, C. Seggewies, W. Baldauf-Sobez, P. Kullmann, H. Reichert, L. Luedecke, and H. Seibold. "soarian–workflow management applied for health care". *Methods Inf Med*, 42(1):25–36, 2003.

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

[13] IEC 61131-3:2003. Programmable controllers—part 3: Programming languages. Technical report, International Electrotechnical Commission, 2003.

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

[15] Archana Laxmisan, Forogh Hakimzada, Osman R. Sayan, Robert A. Green, Jiajie Zhang, and Vimla L. Patel. The multitasking clinician: Decision-making and cognitive demand during and after team handoffs in emergency care. *International journal of medical informatics*, 76(11):801–811, 2007.

[16] 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.

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

[18] 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.

[19] 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.

[20] Sajed Miremadi, Knut Åkesson, and Bengt Lennartson. Symbolic computation of reduced guards in supervisory control. *IEEE Transactions on Automation Science and Engineering*, 8:754–765, 2011.

[21] 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.

[22] Supremica. http://www.supremica.org.

[23] van der Aalst. The application of petri nets to workflow management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

[24] Wil v. d. van der Aalst and Kees v. van Hee. *Workflow Management: Models, Methods, and Systems (Cooperative Information Systems)*. The MIT Press, 2002.

[25] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14:5–51, 2003.

[26] A. Voronov and K. Åkesson. Verification of process operations using model checking. In *Proc. 2009 IEEE Conference on Automation Science and Engineering*, Bangalore, India, August 2009.

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

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