Emergency department overview - Improving the dynamic capabilities using an event-driven information architecture

Kristofer Bengtsson, Elin Blomgren, Oskar Henriksson, Linnéa Johansson,
Edvard Lindelöf, Martin Pettersson, Åsa Söderlund

Abstract—**It is challenging to get an overview and understanding of what is going on at an emergency department (ED). This is due to the sometimes turbulent work environment and a large variation in patient processes. To increase the dynamic capability and responsiveness of an ED, it is important that the staff and patients have an overview of what is going on and what will happen in the coming hours. This paper presents a smart online support software that shows the current state of the ED as well as a prediction of the coming hours. The software has been developed as a case study of using agile development ideas when developing new systems for hospitals. The result shows that the use of the event-driven information architecture for healthcare (EVAH) enabled a rapid development of a successful running application, helping the nurses getting an overview of current situation and the coming hours.**

I. Introduction

Dynamic capabilities is often used to refer to an organizations ability to adapt and reconfigure resources and processes to react to changing circumstances [25]. One key to achieve dynamic capabilities in turbulent environments like an emergency department (ED) can be to utilize advanced IT support [9]. This, together with the challenge of overcrowded emergency departments (EDs) [7] has led to an increased interest from the healthcare sector in finding new and innovative support tools.

Healthcare research has been studying IT-support for decades [3], [11], including a large variety of systems and functionalities. One example is the transition from paper-based to computer based patient records [8], guideline support [19] and archetype-based information structures [21]. However, few have been studying how to support the dynamic capabilities [24] of an ED, i.e. how to support a reactive and adaptive patient process control at chaotic emergency departments.

Bengtsson et al. [5] introduced an event-driven architecture for healthcare (EVAH) as a backbone to support dynamic capabilities. The first application was to visualize the current and future situation and behavior at an ED, especially for the nurses working with the actual care [4]. EVAH is inspired by event-driven architectures [16] which has been shown to support dynamic capabilities in health care [26].

EVAH is event-driven, 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 easily introduce new calculation and visualization algorithms not only based on new, but also on historical data. This paper presents a dashboard tools for the nurses at an emergency department including information about current situation as well as a prediction of the coming hours.

Modern development paradigms like agile software development [15] is hard to use when developing new hospital applications. This is often due to restrictive polices and non flexible information technologies. This paper will therefore also show that EVAH supports modern development paradigms like agile software development. As a case study, six students developed a dashboard for nurses at an emergency department using agile development ideas. The result show that the use of the event-driven information architecture for healthcare (EVAH) enabled the students to rapidly develop a running application helping the nurses getting an overview of current situation and the coming hours.

Two key enablers were the use of transformations of real-time events into understandable information and a simple approach for creating new services, like a predictive algorithm. This made it possible to continuously include the end users and try out a variety of ideas.

This paper demonstrated the power of EVAH and the ease of developing new support tools when having a modern architecture. In section II, EVAH is introduced and in Section III, the dashboard is described. The data handling and the key performance indicator calculation is described in Section IV and the prediction algorithm is presented in Section V.

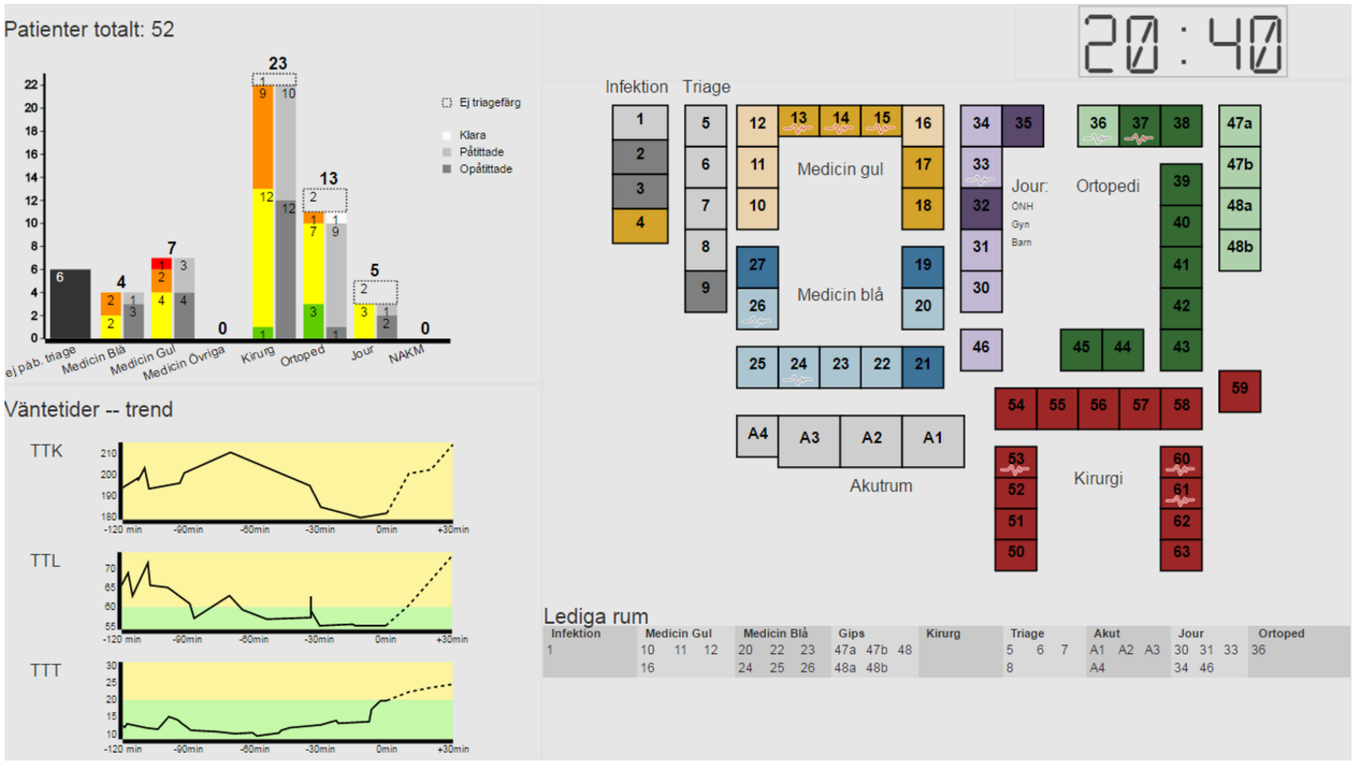
II. Event-driven information architecture for healthcare (EVAH)

EVAH is based on a set of simple building blocks: An event bus, transformation and service endpoints, and EVAH events. These building blocks enable, in a modular and loosely coupled way, the creation and transformation of events into

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

E. Blomgren, O. Henriksson, L. Johansson, E. Lindelöf, M. Pettersson, Å. Söderlund are students at Chalmers University of Technology, SE-412 96 Göteborg

978-1-5090-1314-2/16$31.00 © 2016 IEEE



Lediga rum

Infektion	Medicin Gul	Medicin Blå	Gips	Kirurg	Triage	Akut	Jour	Ortoped
1	10 11 12 16	20 22 23 24 25 26	47a 47b 48 48a 48b		5 6 7 8	A1 A2 A3 A4	30 31 33 34 46	36

Fig. 1. The coordination view of the dashboard. Upper left show patients per section, priority and current state. Current and predicted waiting times are shown lower left. The ED layout and occupied and free rooms are on shown to the right.

usable information. EVAH uses a event / message bus called Apache ActiveMQ for sending and receiving events, but can use any type of publish-subscriber solution. 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 can be sent out with information about the change. A EVAH event, 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 [23]. 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. This is described in more detail in [5]. Each event is published onto the message bus by an endpoint as a json messages so that other services can receive it.

B. Transformation endpoints

One big challenge when trying to create an information system for healthcare is to manage all the various types of events. In addition, sometimes events are not immediately registered, some events only include limited information, or activities are only defined by a single event after completion. To be able to use all these events, for example to calculate various key performance indicators, KPIs, it is necessary to transform, update, and aggregate events.

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.

The Fill transformation fetches information from a database or other type of static information that do not change over time. 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 happened 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. A Map transformation is a function that transform events by appending a set of new attribute – value pairs based on the current state of some part of the system, which the map function is tracking.

Fill and Map can be used to transform events in multiple steps to simplify the implementation and to increase the changeability. The last transformation type is Fold, which takes a sequence of events and transforms them into a new event or that is sent out onto the message bus. Fold transformations can also implement advanced event pattern identification algorithms like complex event processing (CEP) [14] or real-time languages [18]. 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. [6]

III. An ED Dashboard using EVAH

The ED dashboard that was developed as a case study includes two views, one view, which is studied in this paper, for supporting the coordination of resources and patients at the ED Fig. 1 and one view focusing on the patients in one part of the ED. The dashboard shows information aggregated for each patient, including state as well as the history of each patient. It also shows various resource aggregations related to rooms, ED sections and performance indicators like waiting time.

A. Aggregated information

The information in the dashboard represents aggregations of specific patient information gathered from the events. With each new event related to a view, the dashboard is updated and redrawn. This leads to an accurate and continuously updated visual presentation of the present state at the ED.

The views were iteratively developed in collaboration with the end users. During the case study it became apparent that what information the staff thought they wanted to see changed rapidly. Therefore, it was necessary to rapidly evaluate new ideas and enable a system that could evolve and easily be updated.

B. Visualizing information

In the upper left in Fig. 1, a bar chart visualizes where patients are currently located. Each bar group shows the patient at that section of the ED, together with their priorities and if they have met a doctor. The three charts in the lower left show the last hours as well as a future prediction for three patient waiting time metrics: waiting time until triage, waiting time until meeting a doctor, and total time spent at the ED. The map and table on the right shows every room at the ED and whether each room is occupied or availible.

One example is how to measure the performance of the ED. A common performance indicator is the waiting time from arrival until the first physician evaluation. However, due to patient priority, the patient type, as well as inaccurate registrations, there are large variations in registered waiting time as can be seen in Fig. 2. The figure shows registered waiting times for patients as well as four different calculation methods of gliding average waiting times, between 08:00 and 18:00 a day at an ED.

The *10% new* method in Fig. 2 shows a low-pass filter implemented as the weighted average $T_{t+1} = 0.1 * T_r + 0.9 * T_t$, where T_t is the previously calculated average time and T_r

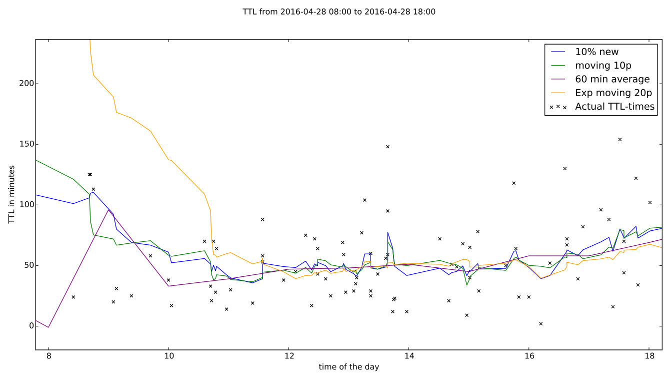


Fig. 2. Time to doctor (TTL), showing four methods for calculating average waiting time as well as registered waiting time for each patient

is the latest registered time. *Moving 10p* shows the average including the 10 last registered times, and *60 min average* shows the average for the last 60 minutes unrelated to number of registered times. The last, *Exp moving 20p* shows the moving average including a weight. These methods just show the need for evaluating a variety of measuring methods.

To have a flexibility in what information can be shown, it is necessary to have a good architecture and information management. EVAH is one of the core pieces of this.

IV. Creating information from events

The core concept of EVAH is that many systems can generate events unrelated to each other and publish them to the bus. Some systems can only send simple numbers, and others can send out complex messages. EVAH will handle them all, since small transformation services transforms and standardizes the messages so that higher level systems can understand them. The studied ED however, has only a simple database system storing the current state of each patient, including e.g. where the patient is located. To be able to track the history of each patient and to use the ideas of EVAH, the database is polled every second and changes to a patient are identified and an event is published.

A. Raw events

In the case study, three types of events were available:

1) NewPatient
2) RemovedPatient
3) UpdatedPatient

The *NewPatient* event includes just the arrival time $t_{arrival}$ and a patient id ID_p. After that event, every update to that patient generates a new event. Examples of events are: priorityChange, triageStart, locationChange, sectionChange, doctorStart, careStart, careEnd, etc. Finally, a RemovedPatient event is published when a patient is removed from the ED database. This event is analogous to the NewPatient event but also includes the final state of the removed patient. Each event includes a timestamp, the id of the patient and an event id. All these events are published on the bus where a number of services transforms and creates information.

B. *Transformations*

A number of transformations are used in the case study. To the right in Fig. 1, a map of the ED shows if a room is occupied or not. This information is created by a service that listens to UpdatedPatient events that include a location change. Every time such an event arrives, the service adds that patient to the room, and if the patient was located somewhere else, removes it from the old room. The service then sends out a new event, RoomChange, which includes the state of the changed room as well as, for example, how long the room has been occupied during the last hour. The service also handles registration "errors" if more than one patient is in the same room, and emits a registration error event.

Since the RoomChange service keeps track of the state of all the rooms, it is also the source for the user interface in the figure. If, in the future, the ED for example installs a tracking system for all patients, the new data can be used directly by the same service, making the system handle it without any changes beyond the scope of the RoomChange service.

Most of the services used in the case study are patient fold services. They listen to patient events to update their internal state, keeping track of the patients. Each service focuses only on one task, for example calculating various waiting times or tracking patient movements.

In Fig. 1, to the left, a fold service is used to aggregate information related to each section of the ED. In the figure, 23 patients are located at section kirurg (surgery), 9 have priority orange, 12 yellow and 1 green, 10 have met a doctor and 12 are still waiting for the doctor. This simple bar chart quickly gives the staff an overview of current state of the ED, which is based on low level events.

C. *Elastic search and event sourcing*

Most of the information shown in Fig. 1 is based on the current state, which is stored in the memory of the services. If a service crashes, it can recreate its state from persisted events, stored in a journal. To store the history of events with the purpose to persist an application's state is called event sourcing [13]. Compared to persisting the state itself, there are some notable differences. With event sourcing, the exact same application behavior can be replayed and analyzed in detail. For example, if an application is found to be in an incorrect state it is possible to step through the replay of events to find out which event processing introduced the error. Another advantage of event sourcing is that it is possible to apply the event history to new applications.

When studying the history of the ED however, a search database is used instead of an event journal. In this case study, elasticsearch [12] has been used for indexing and fast search of historical information.

V. Predicting the coming hours

In the bottom left of Fig. 1, the last 2 hours together with a prediction of the coming 30 minutes are shown for time to triage, time to doctor and time to finished. Due to the nature of event driven architecture, it was easy to integrate and evaluate various prediction methods.

To predict the waiting time, a learning algorithm is needed that uses a training set based on historical data. What algorithm that fits best, what input parameters to use and how to tune the learning will always be a design task. Hence a flexible information system is necessary to be able to develop learning algorithms.

Researchers have evaluated some prediction methods for emergency departments, e.g. [22], [1]. In the case study, similar methods was evaluated as well as standard prediction methods like: artificial neural networks [10], linear regression [17], k-Nearest Neighbors regression [2] and LASSO [20]. The training data for predicting time to doctor includes the following tuple of variables $\mathcal{V}$ that are retrieved from the bus:

- Average waiting time last 30 min
- Average waiting time last 60 min
- Average waiting time last 120 min
- Number of new patients last 60 min
- Number of patients assigned doctor last 60 min
- Number of patients waiting for doctor
- Average waiting time for patients waiting for doctor

The value of the variables in $\mathcal{V}$ at time instance t, denoted $\mathcal{V}_t$, are the input values when training the prediction models. The output for the prediction is the time series $\langle w_t, w_{t+10}, w_{t+20}, w_{t+k}\rangle$, denoted $\mathcal{W}$, where w_t is the waiting time at time t and $w_{t+10}, w_{t+20}, w_{t+k}$ are the waiting times after t with ten minutes intervals until k. The prediction models are trained with the input-output pair $\langle \mathcal{V}_t, \mathcal{W}_t\rangle$. The training set includes a time range of pairs, and the result is evaluated using a different time range. In Fig. 3, the evaluation of linear regression for time to doctor is shown, using an average waiting time for patients seeing a doctor in the next 120 minutes (i.e. the average of $\mathcal{W}$ where $k = 120$). Each dot illustrates the difference between the predicted and measured value. As can be seen, the prediction is accurate up to 200 minutes. After that, the variation between patients is too large for a good prediction. To improve the accuracy it would probably be necessary to separate the patients based on priority and predict each group individually.

The prediction models can be trained at one time and used to predict future waiting times, or trained again at a specific rate, e.g. once every week. The predicted segments (indicated by dashed lines) in the lower left in Fig. 1, were generated using polynomial regression. The polynomial terms were taken as all possible products of the prediction variables described above.

The prediction service makes a prediction once a minute as well as each time any of the prediction variables changes. The model predicts four points: waiting time now, 10 minutes ahead, 20 minutes ahead and 30 minutes ahead. The different algorithms evaluated showed similar results and were possible to use for online prediction, on the dashboard. However, further evaluations and experiments are needed to find a prediction technique that helps the staff at the ED in taking good decisions. It could for example be better to predict

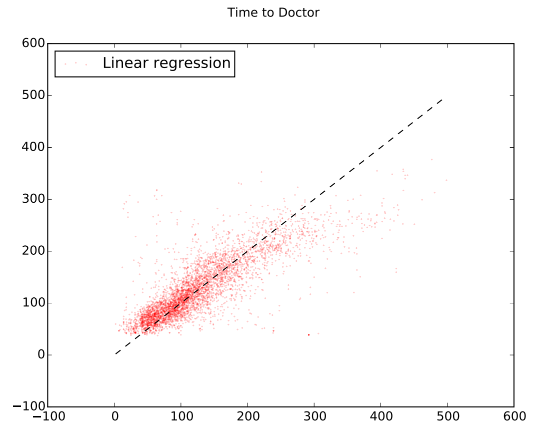


Fig. 3. Cross validation of predictingTime to doctor (TTL), linear regression. Predicted value on y axis and measured value on x axis.

patient throughput instead of the waiting time. When using EVAH it is easy to evaluate a large number of prediction techniques.

VI. Conclusions

One important tool to handle overcrowding in emergency departments is to visualize the current system state and the future possible behavior. Using an event-driven architecture is an enabler for rapid development and evaluation of various visualization techniques and algorithms. This paper shows that the Event-driven information architecture, EVAH, made it possible for a group of students to rapidly develop a prototype tool that the emergency department (ED) could use.

To increase the dynamic capabilities of an ED, new and innovative computer aided techniques are necessary, especially prediction and analysis algorithms to help the staff in making informed decisions. After this case study, EVAH will be integrated at the ED and new techniques for how to define and analyze the plan for each patient will be evaluated. When a plan for each patient is available, it will increase the possibility for the staff to make even better decisions to increase the dynamic capabilities.

References

[1] Accurate emergency department wait time prediction. *Manufacturing & Service Operations Management*, 18(1):141–156, 2016.

[2] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

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

[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. Patient coordination in emergency departments using an event-based information architecture. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–6, Sept 2014.

[6] 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.

[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] Richard S Dick, Elaine B Steen, Don E Detmer, et al. *The Computer-Based Patient Record:: An Essential Technology for Health Care.* National Academies Press, 1997.

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

[10] Daniel Graupe. *Principles of artificial neural networks*, volume 7. World Scientific, 2013.

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

[12] Rafal Kuc and Marek Rogozinski. *Elasticsearch Server*. Packt Publishing Ltd, 2013.

[13] Roland Kuhn and Jamie Allen. *Reactive Design Patterns*. Manning Publications, MEAP 1 edition, 2014.

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

[15] Robert Cecil Martin. *Agile software development: principles, patterns, and practices*. Prentice Hall PTR, 2003.

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

[17] Douglas C Montgomery, Elizabeth A Peck, and G Geoffrey Vining. *Introduction to linear regression analysis*. John Wiley & Sons, 2015.

[18] 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.

[19] Luca Piovesan, Gianpaolo Molino, and Paolo Terenziani. An ontological knowledge and multiple abstraction level decision support system in healthcare. *Decision Analytics*, 1(1):1–24, 2014.

[20] Volker Roth. The generalized lasso. *Neural Networks, IEEE Transactions on*, 15(1):16–28, 2004.

[21] 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.

[22] Yan Sun, Kiok Liang Teow, Bee Hoon Heng, Chee Kheong Ooi, and Seow Yian Tay. Real-time prediction of waiting time in the emergency department, using quantile regression. *Annals of emergency medicine*, 60(3):299–308, 2012.

[23] 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.

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

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

[26] 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.