Patterns to Enable Mass-Customized Business Process Monitoring

Marco Comuzzi[1], Samuil Angelov[2], Jochem Vonk[3]

[1]Eindhoven University of Technology, Eindhoven, The Netherlands
m.comuzzi@tue.nl
[2]Fontys University of Applied Sciences, Eindhoven, The Netherlands
s.angelov@fontys.nl
[3]Logica Nederland BV, Eindhoven, The Netherlands
jochem.vonk@logica.com

Abstract. Mass-customization challenges the one-size-fits-all assumption of mass production, allowing customers to specify the options that best fit their requirements when choosing a product or a service. In business process management, to achieve mass-customization, providers offer to their customers the opportunity to customize the way in which a process will be enacted. We focus on monitoring as a specific customization aspect. We propose a multi-dimensional classification of modeling patterns for customized monitoring infrastructures. Patterns enable the provider to offer a set of customizable options to customers and design a monitoring infrastructure that fits the preferences specified by customers on such options. An example in the online advertising industry demonstrates how our framework can improve the services currently offered by providers.

Keywords: monitoring framework, monitoring patterns, business process monitoring, process customization.

1 Introduction

Mass-customization is defined as "developing, producing, marketing and delivering affordable goods and services with enough variety and customization that nearly everyone finds exactly what they want" [20]. The aim of mass customization is to challenge the one-size-fits-all assumption of mass production, allowing customers choosing a product or a service to also specify the options that best fit their requirements. As such, it has been successfully achieved by many organizations particularly in the manufacturing industry, with examples ranging from the BMW's online product configurator to Dell's hardware configuration services.

In this paper, we consider mass-customization in business process management and, specifically, how to support process providers in implementing mass-customized business processes. In B2B, provider companies perform a set of activities, i.e. a

business process, to satisfy a customer company. Mass-customization, in this context, should result in the customers' ability to customize the way in which providers will execute a process on their behalf. Given a standard version of a process, customization ranges from process control [18], [21], e.g. the opportunity for the customer to skip, redo, or cancel activities, to QoS customization [27], e.g. the opportunity for the customer to pick a certain level of guaranteed security in financial transactions, or resources customization [24], e.g. the opportunity for customers to directly choose which resources will be used in the process.

We focus on a management aspect that has not yet been extensively investigated by the literature on BPM and customization, i.e. monitoring. Generally, business entities need information about the activities taking place in the business landscape (at internal units, partner organizations, or third parties) so that they can react and/or adapt to them. Thus, business entities need mechanisms that ensure the collection of relevant data from the environment (referred to as *monitoring* [1]). In B2B, customers may use monitoring information, for instance, to synchronize their own internal processes or to assess the provider behavior, especially in dynamic relationships where customers and providers are often not likely to have conducted business together in the past. In this context, each customer is likely to have specific monitoring requirements in terms, for instance, of what information the provider should made available and how this has to be communicated, e.g. along which channel, with which frequency. Customization of monitoring requires focus on both control flow aspects, specifying the way to capture monitoring information and make it available, and resources, specifying what has to be monitored and at which stage of the process.

A successful mass-customizer should develop three capabilities [19]: (i) *solution space development*, i.e. understanding customer needs and options over which those are likely to diverge, (ii) *robust process design*, i.e. design an infrastructure to offer such options, making sure that customization does not hinder the company's ability to execute its processes in an efficient and effective way, and (iii) *choice navigation*, i.e. support customer selection of options minimizing the burden of choice.

Focusing on the provider designing the monitoring infrastructure, this paper aims at supporting the capabilities (i) and (ii). To support the provider in the identification of the *solution space*, in fact, we provide a set of patterns for the conceptual design of mass-customizable business process monitoring infrastructures. The patterns are positioned in a framework providing a multidimensional classification space. The multidimensional space is constructed by reasoning on the literature for software monitoring, Web service, and business process monitoring and by using existing, context-specific solutions as verification and illustration techniques. To support the *robust design* of customized monitoring infrastructures, we show how to combine patterns depending on specific contextual requirements using an example in online advertising.

The paper is organized as follows. In Section 2, we introduce the multidimensional space of our framework. In Section 3, we provide the patterns for the options defined within the multi-dimensional space. Section 4 discusses an example of the application of the framework, showing how it can be used to improve current practice in online

advertising to achieve mass-customizable business process monitoring. In Section 5, we discuss relevant literature on business process customization and monitoring. The paper ends with our conclusions and plans for future work.

2 Monitoring Dimensions and Options

This paper considers monitoring of business processes supported by information systems. In Fig. 1(a), we sketch the general outline of a monitoring architecture. In the business process provider domain, a *monitoring infrastructure* (also called sensor or observer [2-3]) capturing relevant information is built around a *business process engine*. A business process engine is meant in a broad sense - it is a component producing process information (activity or process states, data values, etc.). Although the monitoring infrastructure can be built in (or even be an integral part of) the business process engine [2], conceptually, it is a separate entity [3]. Considering it as a separate entity promotes separation of concerns with respect to the underlying process engine, leading to a more focused and context-independent analysis of the monitoring infrastructure component and its possible customization. In the customer (also called controller [2-3]) domain, a *monitoring client* obtains the information captured by the monitoring infrastructure. It processes the information obtained and if desired exerts control over the business process engine.

In many scenarios, the monitoring client can be an intermediary for the actual business entity that requires the monitoring information. Monitoring may take place within an organization (between independent business units) or in a cross-organizational setting. Furthermore, the client of the monitoring infrastructure may be an autonomous application or a human user. As these cases do not introduce any specifics in our work, in the sequel of the paper, we abstract from them and discuss the general scenario depicted in Fig.1(a).

The dimensions of our framework identify what parts of the scenario in Fig.1(a) can be customized by the monitoring client, i.e. they define the monitoring *solution space* for the customer of the business process.

The first two dimensions concern the context in which monitoring has to occur. In particular, we consider the *monitoring variable* and the *anchoring points*, defined as follows:

Monitoring Variable (MV): The MV specifies the object of monitoring, i.e. the process information that the *Monitoring Infrastructure* (MI) obtains from the *Business Process Engine* (BPE) and makes available to the *Monitoring Client* (MC). It has a domain, which specifies the (range of) values that it can assume, and a unit of measure, if needed. In the world of software programs monitoring, the monitoring variable is the target, for instance, of a watch for debugging. In business process monitoring, the monitoring variable may range from infrastructure-level data, such as timestamps of service calls [5], to application-level process data, such as the status of an activity or domain specific data produced by an activity [6-7]. Note that a value of the monitoring variable represents a single data element captured by MI during the execution of the business process, e.g. the timestamp of an order, the warehouse level

at a specific point in time, the unique id of a user executing a specific activity. Captured values can be stored by MI during the execution of a business process and made available in batches to MC.

Anchoring Point (AP). The MI is enabled within a specific scope of the process to be monitored. Anchoring points specify the scope of the process within which the MI is enabled. The notion of anchoring point derives from the literature on software program monitoring, where running the monitoring program in the same memory space as the monitored program can be costly and, therefore, the monitoring program has to be enabled only when strictly necessary [8]. In a business setting, while in many cases we can make the hypothesis that monitoring is permanently enabled, defining anchoring points may be helpful when capturing and making available the values of MV to MC is very costly. Intrusive process monitoring [5] is an example of this scenario, since the execution of monitoring statements blocks and, therefore, delays, the execution of the process. In such a scenario, the MC may want to enable monitoring only when strictly necessary. In the remainder, we refer to AP-START and AP-END as the anchoring points enabling and disabling the Monitoring Infrastructure MI, respectively.

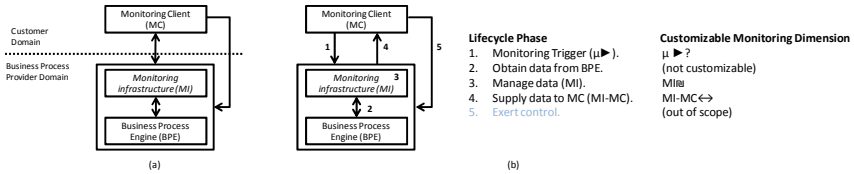

Fig. 1. (a) Monitoring Scenario; (b) Business Process Monitoring Lifecycle

Fig.1(b) refines the conceptual outline of a monitoring solution by showing the lifecycle of communications among the different elements of the architecture. We use the phases of such lifecycle to derive the next three dimensions of our framework. In particular, the first phase concerns commanding the acquisition of monitoring data. This can be done either by MC or by MI (Phase 1). For instance, MC may specify that MI has to acquire values of a specific MV periodically or may want to be allowed to command the acquisition pro-actively.

The second phase concerns MI obtaining the required data from BPE (Phase 2). Since this phase is internal to the business process provider domain and cannot be customized, we do not consider it in this paper. Then, MI may have to manage, e.g. store in batches, the monitoring data obtained by BPE (Phase 3) before supplying them to MC (Phase 4). Eventually, MC processes monitoring data and may decide to exert control on the monitored process executed by BPE (Phase 5). This last phase is out of scope in this paper.

Phases 1, 3, and 4 of the lifecycle in Fig. 1(b) are characterized by one monitoring dimension in our framework, since they involve aspects that are customizable by MC. Each dimension has a set of options. Options represent the solution space for the customization of process monitoring, that is, a customized monitoring infrastructure can be built once the customer has chosen one option for each possible monitoring

dimension. In the remainder of this section we present the monitoring dimensions and their possible options.

Monitoring Trigger (μ►). The monitoring trigger phase is characterized by one dimension that identifies the entity commanding the acquisition of values of MV (μ►**?**). The acquisition of a monitoring value can be triggered by MC or by MI. In the former case (μ►**?**=*MC-trg*), MC makes a request to the provider for commanding the acquisition of the value of MV. In the latter case (μ►**?**=*MI-trg*) MI acquire values of MV proactively. Note that in this case MC should be able to customize the way in which MI will acquire values, for instance periodically at a certain frequency, or when a threshold is exceeded, or on change. The third option (μ►**?**=*mix-trg*) fits cases in which acquisition is triggered by MI, e.g. periodically, but also the client MC wants to be allowed to request acquisition. Once acquisition is triggered, MI will obtain monitoring data from BPE. This phase of the monitoring lifecycle is internal to the business domain of the provider and, therefore, it cannot be customized by MC.

Manage Data (MI). For this lifecycle phase we define one monitoring dimension (MI₪), which captures MI's logic in managing the values obtained from BPE before supplying them to MC. New values obtained from BPE can rewrite old values captured for the same MV or the obtained values can be stored (persisted), e.g. to produce historical series of values of MV. When supplied to MC, values can be consumed, i.e. they will not be available in the future to MC, or they can just be read, remaining available also in the future. Thus, we identify four options for MI ₪, i.e. (i) *rewrite-consume*, (ii) *rewrite-read*, (iii) *persist-consume*, and (iv) *persist-read*.

Supply Data to MC (MI-MC). This phase is characterized by the dimension MI-MC↔ referring to the direction of the interaction between MI and MC. For MI-MC↔ we define the options *push* and *pull*. The option *push* models cases in which MI pushes monitoring values to MC, whereas *pull* models cases in which MI sends values of MV only after having received a request from MC. Generally, communication between MI and MC is a distributed systems communication issue and its customization may require the definition of additional dimensions, such as space and time decoupling option [9, 10]. However, since those aspects are not monitoring-specific, we do not further discuss them here.

3 Monitoring Patterns for Monitoring Dimensions

In this section, we use the monitoring dimensions and present the design of internal data and control flows of the Monitoring Infrastructure (MI). This is required for the provider to offer mass-customization monitoring capabilities to its customers. Following common principles in the design of distributed systems [3], we first define the interfaces required between MI and MC, and MI and BPE, respectively, to support the monitoring dimensions. Then, we propose a modeling pattern for each option characterizing the monitoring dimensions. Modeling patterns specify the data and control flow of MI's internal implementation of the interfaces. Eventually, Section 4 presents an example in the online advertising industry to combine the patterns in a

monitoring infrastructure with certain desired monitoring properties chosen by customers.

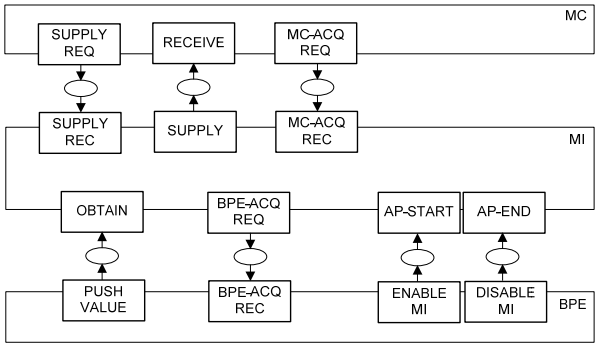

Fig. 2. Interfaces between MI, MC, and BPE

For business process modeling purposes, we use Colored Petri Nets (CPNs) [10-11]. CPNs have been chosen because they have a graphical representation, they have mature and freely available tool support, e.g. CPN Tools, and they have a precise semantic that can be translated to or directly implemented in other languages, e.g. Event-driven Process Chains [12] or YAWL [6], used by commercial and non-commercial workflow engines. Readers unfamiliar with CPNs' notation are referred to [11]. We use two token colors in our patterns, i.e. UNIT and MV. UNIT represents the default color of black tokens, and it is used to model the control flow in our modeling patterns. Tokens of color MV represent a single value of the monitoring variable MV. The precise definition of the color MV, e.g. possible values and unit of measure, is part of the definition of the monitoring context and we do not further discuss it in this paper.

Fig. 2 shows the interfaces between the elements of the monitoring solution MI, MC, and BPE. Note that, although aspects related to the interaction between MC and BPE are internal to the provider domain, and therefore not subject to possible customization, considering the interfaces between MI and BPE allows separation of concerns between business process execution and monitoring and, therefore, for more robust patterns for customizable process monitoring infrastructures.

MI requires one interface to receive acquisition triggers from MC (MC-ACQ-REC) and supply data to MC (SUPPLY). Optionally, MI may require also an interface to receive supply requests from MC (SUPPLY REQ). This interface is necessary only when MC decides to pull monitoring data from MI. The interface between MI and BPE is constituted by a generic interface for receiving monitoring values (OBTAIN), by the two anchoring points, and by interface BPE-ACQ-REQ allowing MI to command acquisition of values of MV by BPE. Note that Fig. 2 does not show the color of tokens in places connecting interfaces. As shown later, the color of such places is determined by the modeling pattern realizing the MI's internal implementation of the corresponding interface(s).

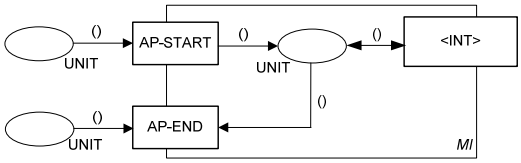

Fig. 3. Pattern for anchoring point implementation

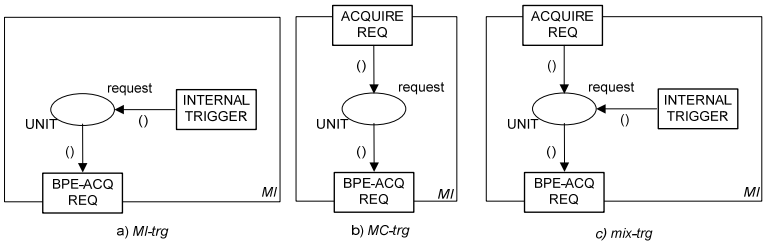

Fig. 4. Monitoring patterns for μ►**?** dimension options

Concerning the context, MI is required to expose interfaces to define the anchoring points of monitoring to the business process executed by BPE. The tokens required to fire the transitions AP-START and AP-END are produced by BPE when the process execution reaches the point enabling and disabling the monitoring, respectively. Fig.3 shows the pattern for the implementation within MI of the anchoring point business logic. Specifically, the generic interface <INT>, which can be any of the ones defined in Fig. 2, becomes enabled after the firing of the AP-START transition and is no longer enabled after the AP-END transition has fired.

The remainder of this section presents the monitoring patterns for the remaining monitoring dimensions. Each pattern models one option of one monitoring dimension identified in Section 2.

Monitoring Trigger (μ►). The patterns modeling the options of dimension μ►**?** implement the ACQUIRE-REC and BPE-ACQ-REQ interfaces of MI.

The patterns corresponding to the three values *MI-trg*, *MC-trg*, and *mix-trg* are shown in Fig.4. In Fig.4(a), the transition INTERNAL TRIGGER captures MI's business logic to trigger acquisition, e.g. periodically or on change, which can be customized by MC. In Fig.4(b) the acquisition is commanded by MC, by firing the transition ACQUIRE REQ, whereas Fig.4(c) shows the mix case, i.e. MC or MI can both trigger an acquisition request.

Supply data to MC (MI-MC). The patterns for the *push* and *pull* options of the dimension MI-MC↔ are shown in Fig. 5(a) and Fig. 5(b), respectively. The *push* option requires MI to only expose the SUPPLY interface. The transition SUPPLY TRIGGER fires accordingly to the logic chosen by the customer to receive monitoring values. The customer, for instance, may require monitoring values

periodically or only when values exceed a certain threshold. For the *pull* option, the SUPPLY interface can fire, i.e. supply a monitoring value, only after a request is received. Note that that this implemented by limiting the control exerted by the anchoring point to the SUPPLY-REQ interface.

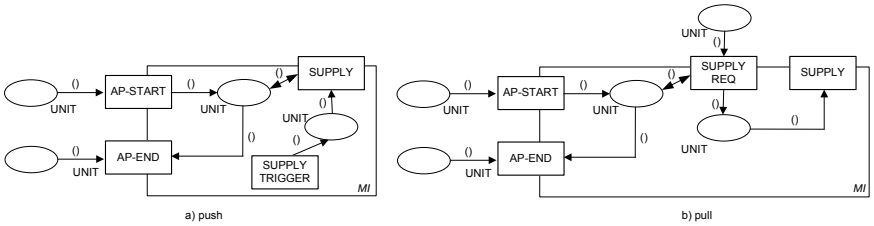

Fig. 5. Monitoring patterns for the MI-MC↔ dimension options

Manage data (MI). The patterns for the MI₪ dimension implement the connection between the interfaces SUPPLY and OBTAIN exposed by MI. Fig. 6 shows the patterns for the four options.

Note that the place bpe stores values of MV ready to be obtained by MI, whereas the place mc stores the value or set of values supplied to the monitoring client MC. Also, note that the color L_MV represents a list of tokens of color MV.

In the *rewrite*- options, the new value of MV v_new always replaces the old value v_old. In the *persist*- options, the values of MV are stored by MI in a list lv (the list in this case represents a generic data structure); a new monitoring value v_new is simply inserted into lv.

In the *-consume* options, the SUPPLY transition, when fired, replaces the current monitoring value stored by MI with the default, whereas in the *-read* options the monitoring value is put back in the place mi_db after being read and can be read again in the future by MC. Note that the default monitoring value is the empty list [] for the *persist*- options and the token with value "dft" of color MV for the *rewrite*- options.

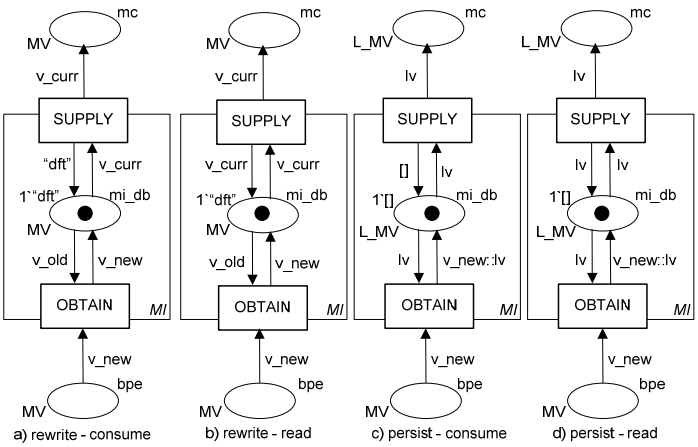

Fig. 6. Patterns for MI℠ dimension options

4 Combining Patterns to Design Monitoring Infrastructures

To demonstrate the value of the framework, we provide an example for its application in an on-line advertising scenario. The example shows how the framework can be applied for the design of a customizable monitoring solution and the advantages that it offers to the designers of the monitoring infrastructure at the provider side.

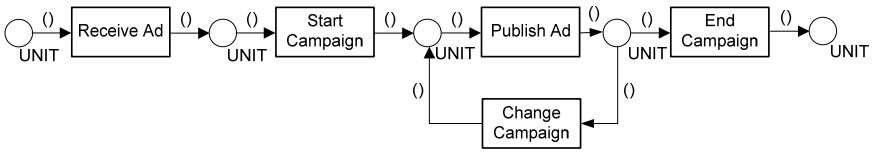

Fig. 7. Excerpt from the on-line advertising process

The advertising provider (e.g. a newspaper) offers advertising space to companies (customers). In a simplified scenario (see Fig. 7), the customer sends the advertisement to the provider and when the time agreed to start the campaign comes, the provider starts the advertising campaign by publishing the ad in its newspaper. In a basic scenario, the ad is shown each time a reader loads the newspaper page, while in a more complex case, the most relevant and highest priced ad is shown in a specific advertising spot (e.g. in Google AdWords). When the budget of the customer is depleted and/or the number of agreed appearances of the ad is reached, the campaign ends. The customer may ask to change its campaign if it observes that the ad is not reaching the target audience, the campaign has little impact, etc. This business case is described in greater detail in [28].

In a traditional advertising setting, the provider offers a fixed set of tools to all customers for monitoring their campaigns. For example, the customers can monitor the IP addresses of the readers seeing their ad, the number of shows of their ad, number of clicks on their ad, the spots where the ad has been published (banner, column, pop-up, in-text), etc. Typically, customers have to access their account to view this information. This monitoring advertising scheme does not address the individual preferences of the customers. Customers may be interested in receiving the monitoring information directly instead of having to query it from the provider; they may be interested in obtaining the information in an aggregated format at the end of the campaign or be constantly updated to be able to adapt their campaign; they may be interested, motivated by a cheaper price, in obtaining only some of the monitoring information instead of monitoring all possible variables, etc. Next, we describe how our framework can be applied to set up a mass-customizable monitoring infrastructure. For brevity, we focus on the construction of monitoring infrastructure for only one monitoring variable, i.e., the number of clicks on an ad, which is the most straightforward indicator of the ad success and profitability over time.

To apply the framework for the mass-customization of a MV, the provider has to consider the possible options from the monitoring dimensions presented in Section 2 to offer to its customers (see Table 1). The MV in this case is the current value of number of clicks on the ad of the customer. We hypothesize that the provider (newspaper), through an advertising engine, keeps track continuously of this value. The customer wants to monitor MV along the campaign. Therefore, the anchoring point enabling and disabling monitoring are the start and the end of the campaign, respectively. Note that, in principles, customers may choose a different anchoring point, for instance enabling monitoring only after the campaign has been changed a first time.

Table 1: Possible monitoring dimension values for the IP address MV

Monitoring Variable	Number of clicks on the customer's ad (cumulative)
Anchoring Point	"Start Campaign" (enabling); "End Campaign" (disabling)
Monitoring Trigger	μ►?: *MC-trg, MI-trg*
Manage Data	MI₪: *persist-read, persist-consume*
Supply data to MC	MI-MC↔: *push, pull*

A customer may like to be pushed monitoring information or to pull it (thus, MI-MC↔=*push* or MI-MC↔=*pull*). The customer may prefer to monitor the MV only at a specific point in time that cannot be revealed (e.g., it is not known) to the provider (hence, μ►?=*MC-trg*) but may also like to delegate the acquisition of the monitoring values to the provider at a pre-agreed time, periodically or when the number of clicks exceeds a certain threshold (hence, μ►?=*MI-trg*). Typically, customers would prefer to have all the monitored values stored by the provider, so that these can be queried any time later on and used to analyze the number of clicks trend over time (hence, MI₪=*persist-read*). Of course, if storage space is crucial for the provider, it may offer some incentives (e.g., financial) to the customers to choose also MI₪=*persist-consume*.

Table 2: Selected monitoring dimension options by Customers A and B

	Customer A	*Customer B*
Monitoring Trigger	μ►?: *MI-trg*	μ►?: *MC-trg*
Manage Data	MI₪: *persist-read*	MI₪: *persist-consume*
Supply Data to MC	MI-MC↔: *pull*	MI-MC↔: *push*

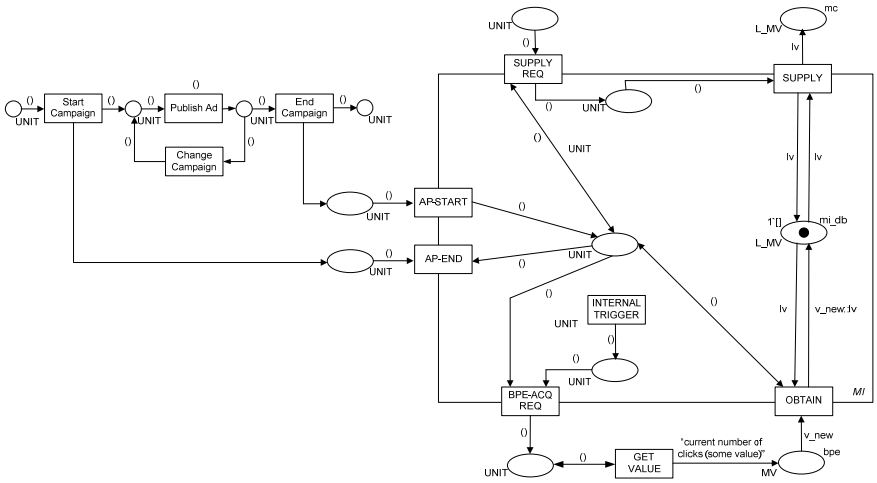

Fig. 8. Customized business process monitoring for Customer A

Thus, having the framework as a guiding tool in the set of possible monitoring styles, the provider has straightforwardly defined all possible monitoring styles for the number of clicks MV. Each customer interested in monitoring information on the cumulative number of clicks is presented with a set of possible options. Table 2 presents two possible sets of choices for customers A and B. Customer A delegates the acquisition of monitoring values to the provider at the beginning of its campaign, e.g. every six hours, it wants to be able to pull monitoring information when needed, and it also requires that the information is kept after it reads it to be able, for instance, to show the trend of this MV over time as soon as the information is pulled from MI. Thus, each time Customer A pulls monitoring information it will get a list of sampled (one every six hours) values of the number of clicks on its ad since the publication of the ad. Customer B wants to be able to specify when the values of number of clicks have to be acquired (μ►?=*MC-trg*) and to automatically receive the monitoring information, for instance as soon as this is acquired by the provider or at the end of each day (MI-MC↔: *push*; where the trigger for the push is the availability of a new value for number of clicks MV or the end of the day). Customer B also does not require the provider to locally store the acquired values (MI₪: *persist-consume*). This

is a reasonable assumption, for instance, when data are directly pushed to MC as soon as they are acquired by the provider.

As different customers may choose different values in each dimension (as demonstrated in Table 2), the provider has to be prepared to support each possible combination. Having the patterns corresponding for each monitoring dimension values pre-defined in the framework, the provider can now directly apply them for these specific MV and AP. In Fig. 8, we demonstrate this, by showing the customized MI for Customer A. Note that Fig. 8 includes also a possible characterization for the internal implementation of BPE. The customized MI to satisfy the requirements of Customer B can be similarly derived by combing the requirements specified in Table 2 (see Fig. 9).

To summarize, the framework has given the set of possible values to the provider to identify all possible options valid for this specific MV (i.e. the *solution space*). Again using the framework, the provider can directly apply pre-defined patterns to ensure support for these monitoring styles whenever they are selected by a specific customer (i.e. achieving a *robust process* supporting customization).

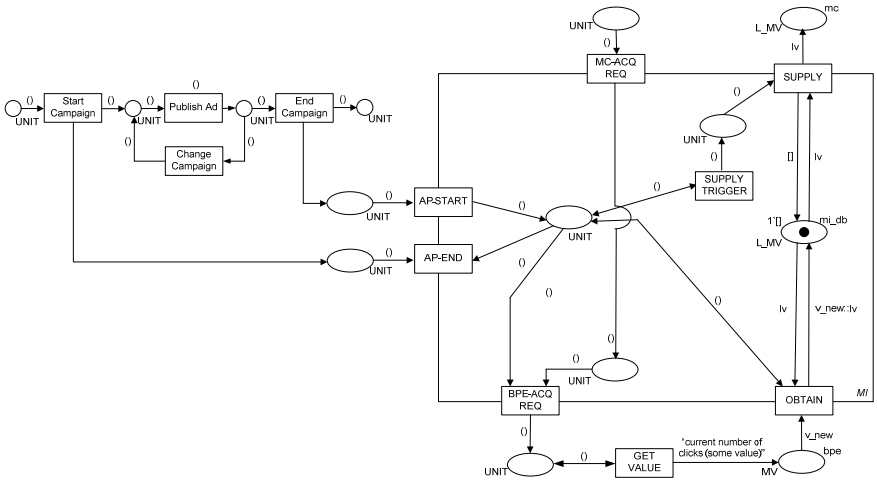

Fig. 9. Customized business process monitoring for Customer B

5 Related Work

Process customization is a paramount activity in the implementation of complex enterprise systems, such as ERP [25]. Traditional process customization in enterprise systems is a design time concern, which aims at designing standard processes across the implementing organization. While process standardization across the enterprise promotes uniformity and interoperability [25], it is also seen as a constraining institutional factor for large companies with diverse business units [26], limiting the flexibility of the company. In this paper, we take a much more dynamic perspective

on process customization, allowing each single client to specify his or her own monitoring requirements at runtime and supporting the derivation of a monitoring infrastructure to support those.

As remarked in the Introduction, the literature on business process customization has not extensively considered monitoring as an aspect that can be customized. As an example, the work in [21] proposes an approach to add control flow options, such as adding, skipping, or redoing an activity, to an original process model, but with no explicit mention of monitoring options. In a different perspective, QoS-based Web service selection can be seen as a form of customization, since the building blocks of a business process are selected at run-time to satisfy the QoS requirements expressed by the user [27]. However, also in this case, we did not found approaches considering monitoring as a customizable aspect.

Business process customization is usually opposed to *configuration*, which aims at designing reference process models capturing the behavior of a set of process variants serving the same business goal [22-24]. We argue that the business process configuration approach does not suit mass-customization, since capturing all possible process variants in a single process model leads to very complex models that are not easily understandable by the process customers. Note that a reference configurable process in our example should contain all possible combinations of monitoring options values for each possible combination of MV and AP in the original process. In a nutshell, configuration remains a design-time concern of process designers [23], whereas customization should be performed directly by the customer right before the process enactment. In our review of related work, we were not able to find approaches to process configuration explicitly capturing monitoring options.

To compile a list of possible behaviors of the monitoring infrastructure and its communication with the business process and the client application, we analyzed the literature on traditional software program monitoring, Web service-based monitoring and business process monitoring.

A survey on software programs and software requirements monitoring can be found in [8]. From this survey, we take the notion of monitoring points. Monitoring points define the anchors of the monitoring program to the monitored program. Similarly, in our model we define *anchor points* for monitoring options to the monitored process.

The survey in [8] is used by [5] to classify approaches to Web service-based process monitoring. In particular, [1] considers the modality to notify monitoring information as classification criterion. Monitoring information, usually captured by an instrumentation of the Web service container, can be either pushed to the monitor or pulled by it. Web service monitoring usually takes an event source-listener approach [9], where the instrumented Web service container is the source that pushes monitoring related event to the monitor (listener) [15]. When monitoring information is pushed, the work in [5] also considers the multiplicity and frequency with which monitoring information is made available to the monitor. Still in the context of Web service monitoring, the model in [16] considers the concept of monitoring socket, i.e. a generic component which is responsible for the generation of monitoring data, which can then be pushed to or pulled by the monitor.

6 Conclusions

In this paper, we present a conceptual framework for the modeling of mass-customized business process monitoring infrastructures. The framework identifies a set of orthogonal dimensions and options along which the customer monitoring requirements may vary. For each option, we provide patterns that model the process and data aspect of the monitoring infrastructure. The customized process monitoring infrastructure is then modeled through the combination of patterns from the dimensions. We illustrate the applicability of the framework and discuss its value using an online advertising example process.

The embedding of our framework in a wider context raises several issues that must be paid closer attention. Firstly, our framework is of descriptive nature and does not specify concrete steps and guidelines that need to be followed for its usage. A method that describes its application would improve its value as a design and analysis tool. Secondly, exercising control over a business process is the logical continuation of monitoring activities. Thus, the value of the framework can be further extended by incorporating it in a generic framework for monitoring and control. This is the focus of our ongoing work [18].

Finally, we are looking at a possible implementation of our framework. Process configuration requires extensions of currently available workflow management systems to enable injecting behavior into existing process specifications. The efficient execution of customized processes may rely on the cloud computing paradigm, in which the resources required by each customer (or by a class of similar customers) for their customized monitoring can be bundled and provisioned as a service.

References

1. zur Muehlen, M.: Workflow-Based Process Controlling. Springer-Verlag, Berlin (2005)
2. Curtis, G., Cobham, D.: Business Information Systems: Analysis, Design and Practice 6th ed. Financial Times/ Prentice Hall (2008)
3. Wieringa, R.: Design Methods for Reactive Systems: Yourdon, Statemate, and the UML. Morgan Kaufmann Publishers (2003)
5. Baresi, L., Guinea, S., Nano, O., Spanoudakis, G.: Comprehensive Monitoring of BPEL Processes. IEEE Internet Computing, 50-57 (2010)
6. ter Hofstede, A. M., van der Aalst, W. M. P., Adams, M., Russell, N.: Modern Business Process Automation: YAWL and its Support Environment. Springer (2010)
7. IBM Corporation: WebSphere Business Monitor. Available at: http://www-01.ibm.com/software/integration/wbimonitor/
8. Delgado, N., Gates, A. Q., Roach, S.: A Taxonomy and Catalog of Runtime Software-Fault Monitoring Tools. IEEE Transactions on Software Engineering, 30(12), 859-872 (2004)
9. Eugster, P. T., Felber, P. A., Guerraoui, R., Kermarrec, A.-M.: The many faces of Publish/Subscribe. ACM Computing Surveys, 35(2), 114-131 (2003)
10. Aldred, L. et al.: Dimensions of Coupling in Middleware, Concurrency, and Computation. Practice and Experience, 21, 2233-2269 (2009)

11. Jensen, K., Kristensen, L. M.: Coloured Petri Nets. Springer (2009)
12. Scheer, A. W.: ARIS Business Process Modeling. Springer-Verlag (2000)
15. Spanoudakis, G., Mahbub, K.: Non Intrusive Monitoring of Service Based Systems. International Journal of Cooperative Information Systems, 15(3), 325-358 (2006)
16. Sadiq, S., Governatori, G., Namiri, K.: Modeling control Objectives for Business Process Compliance. In : Proc. 5th BPM Conference, pp.149-164 (2007)
18. Angelov, S., Vonk, J., Grefen, P., Vidyasankar, K.: Enhancing Business Collaborations with Client-Oriented Process Control. International Journal of Cooperative Information Systems, 20(1), 1-37 (2011)
19. Salvador, F., de Holan, P.M., Piller, F.: Cracking the Code of Mass Customization. SLOAN Management Review, 50, 71-78 (2009).
20. Pine, J.B.: Mass Customization - The New Frontier in Business Competition. Harvard Business School Press: Cambridge, MA (1993).
21. Hallerbach, A., Bauer, T., Reichert, M.: Capturing variability in business process models: the Provop approach. J Softw Maint Evol-R, 22, 519-546 (2010).
22. Lapouchnian, A., Yu, J., Mylopoulos, J.: Requirements-Driven Design and Configuration Management of Business Processes. In Proc. 5th BPM Conference, pp. 246-261 (2007).
23. Gottschalk, F., van der Aalst, W.M.P., Jansen-Vullers, M.H., La Rosa, M.: Configurable Workflow Models. International Journal of Cooperative Information Systems, 17(2), 177-221 (2008).
24. La Rosa, M., Dumas, M., ter Hofstede, A., Mendling, J., Gottshcalk, F.: Beyond Control-Flow: Extending Business Process Configuration to Roles and Objects. In ER 2008, 199-215 (2008).
25. Jacobs, F. R., Whybark, C.: Why ERP? Irwin/McGraw Hill, New York, 2000.
26. Gattiker, T.F. and Goodhue, D.L.: What Happens After ERP Implementation: Understanding the Impact of Interdependence and Differentiation on Plant-Level Outcomes. MIS Quarterly, 29, 559-585 (2005)
27. Gmach, D., Krompass, S. Scholz, A. Wimmer, M., Kemper, A.: Adaptive Quality of Service Management for Enterprise Services, ACM Trans. Web, 2(1), Article 8 (2008)
28. Angelov, S., Grefen, P.; Supporting the Diversity of B2B e-Contracting Processes; International Journal of Electronic Commerce, 12 (4); pp. 39-70.