Supporting Data Collection in Complex Scenarios with Dynamic Data Collection Processes

Gregor Grambow, Nicolas Mundbrod, Jens Kolb and Manfred Reichert

Institute of Databases and Information Systems
Ulm University, Germany
{gregor.grambow,nicolas.mundbrod,jens.kolb,manfred.reichert}@uni-ulm.de
http://www.uni-ulm.de/dbis

Abstract. Nowadays, companies have to report a large number of data sets (e.g., sustainability data) regarding their products to different legal authorities. However, in today's complex supply chains products are the outcome of the collaboration of many companies. To gather the needed data sets, companies have to employ cross-organizational and long-running data collection processes that imply great variability. To support such scenarios, we have designed a lightweight, automated approach for contextual process configuration. That approach can capture the contextual properties of the respective situations and, based on them, automatically configure a process instance accordingly, even without human involvement. Finally, we implemented our approach and started an industrial evaluation.

Key words: Process Configuration, Business Process Variability, Data Collection, Sustainability, Supply Chain

1 Introduction

In todays' industry many products are the result of the collaboration of various companies working together in complex supply chains. Cross-organizational communication in such areas can be quite challenging due to the fact that different companies have different information systems, data formats, and approaches to such communication. These days, state authorities, customers and public opinion demand sustainability compliance from companies, especially in the electronics and automotive sector. Therefore, companies must report certain sustainability indicators such as, their greenhouse gas (GHG) emissions or the amount of lead contained in their products. Such reports usually involve data from suppliers of the reporting company. Therefore, companies launch a sustainability data collection process along their supply chain. In turn, this might involve the suppliers of the suppliers, and so on. Figure 1 illustrates this scenario with three exemplified tiers of suppliers of a company. While having only two direct suppliers on tier one, the company also has eight indirect suppliers on the tiers two and three.

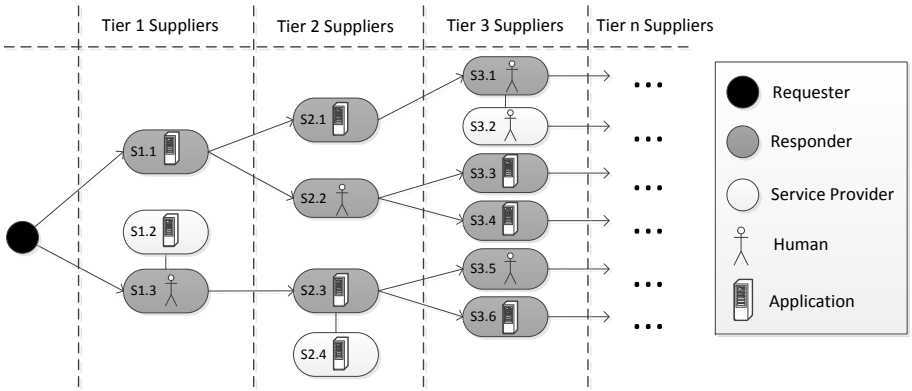

Fig. 1: Supply Chain Scenario

As sustainability data collection is a relatively new and complicated issue, service providers (e.g., for data validation or lab tests) are involved in such data collection as well. This fact is exemplified in Figure 1, where three service providers in different tiers are involved. Another property that makes these data collection processes even more complex and problematic is the heterogeneity in the supply chain: companies use different information systems, data formats, and overall approaches to sustainability data collection. Many of them even do not have any information system or approach in place for this and answer with low quality data or not at all. Therefore, no federated system or database could be applied to cope with such problems and each request involves an often long-running, manual, and error-prone data collection process. The following simplified scenario illustrates issues with the data collection process on a small scale.

Scenario: Sustainability Data Collection
An automotive company wants to collect sustainability data relating to the quantity of lead contained in a specific part. This concerns two of the company's suppliers. One of them has an IHS (In House Solution) in place, the other has no system and no dedicated responsible for sustainability. For the smaller company, a service provider is needed to validate the manually collected data in order to ensure that it complies with legal regulations. The IHS of the other company has its own data format that must be converted before it can be used. This simple scenario already shows how much complexity results even from simple requests and indicates how this can look like in bigger scenarios involving hundreds or thousands of companies with different systems and properties.

In the SustainHub[1] project, we develop a centralized information exchange platform that supports sustainability data collection along the entire supply chain. We have already thoroughly investigated the properties of such data collection in the automotive and electronics sectors and reported on the challenges and state-of-the-art regarding this topic [1]. This paper, proposes an approach that enables an inter-organizational data collection process. Thereby, the main focus is the capability of this process to automatically configure itself in alignment with the context of its concrete execution.

To guarantee the utility of our approach as well as its general applicability, we have started with collecting problems and requirements directly from the industry. This included telephone interviews with representatives from 15 European companies from the automotive and electronics sectors, a survey with 124 valid responses from companies of these sectors, and continuous communication with a smaller focus group to gather more precise information. Among the most valuable information gathered there was a set of core challenges for such a system: as most coordination for sustainability data exchange between companies is done manually, it can be problematic to find the right companies, departments, and persons to get data from as well as to determine, in which cases service providers must be involved (the first Data Collection Challenge - DCC1). Moreover, this is aggravated by the different systems and approaches different companies apply. Even if the right entity or person has been selected, it might still be difficult to access the data and to get it in a usable format (DCC2). Furthermore, the data requests rely on a myriad of contextual factors that are only managed implicitly (DCC3). Thus, a request is not reusable since an arbitrary number of variants may exist for it (DCC4). A system aiming at the support of such data collection must explicitly manage and store various data sets: the requests, their variants, all related context data, and data about the different companies and support manual and automated data collection.

The remainder of this paper is organized as follows: Section 2 shows our general approach for a process-driven data collection. Section 3 extends this approach with additional features regarding context and variability. Section 4 presents the implementation for our concept. This is followed by a discussion of a preliminary practical application in Section 5, a comprehensive discussion of related work in Section 6, and the conclusion.

2 Data Collection Governed by Processes

The basic idea behind our approach for supporting data collection in complex environments is governing the entire data collection procedure by explicitly specified processes. Furthermore, these processes are automatically enacted by a Process-Aware Information System (PAIS) integrated into the SustainHub platform. This way, the process of data collection for a specific issue as a sustainability indicator can be explicitly specified through a process type while process

[1] SustainHub (Project No.283130) is a collaborative project within the 7th Framework Programme of the European Commission (Topic ENV.2011.3.1.9-1, Eco-innovation).

instances derived from that type govern concrete data collections regarding that issue (cf. Figure 2).

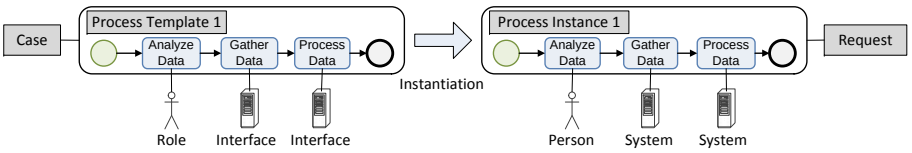

Fig. 2: Utilizing Processes for Data Requests

Activities in such a process represent the manual and automatic tasks to be executed as part of the data collection by different companies. This approach already covers a number of the elicited requirements. It enables a centralized and consistent request handling (cf. DCC1) and supports manual as well as automated data collection (cf. DCC2). One big advantage is the modularity of the processes. If a new external system shall be integrated, a new activity component can be developed while the overall data collection process does not need to be adapted. Finally, the realisation in a PAIS also enables the explicit specification of the data collection process (cf. DCC4). Through visual modeling, the creation and maintenance of such processes is facilitated.

However, the process-driven realization can only be the basis for comprehensive and consistent data collection support. To be able to satisfy the requirements regarding contextual influences, various types of important data and data request variants, we propose an extended process-driven approach for data collection as illustrated in Figure 3.

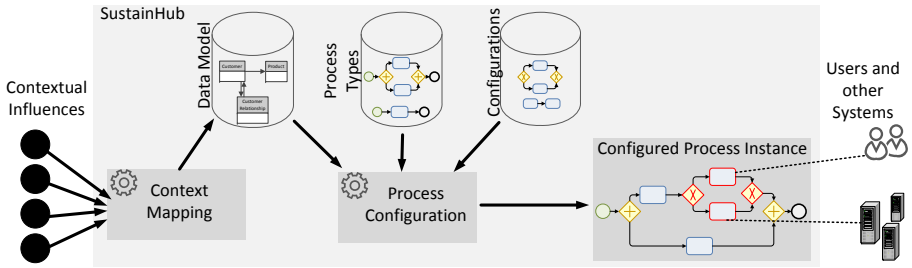

Fig. 3: SustainHub Configurable Data Collection Approach

To generate an awareness of contextual influences (e.g. the concrete approach to data collection in a company, cf. DCC3) and to make them usable for the data collection process, we define an explicit context mapping approach (as discussed in Section 3.1). This data is required for enabling the central component of our approach, i.e., the automatic and context-aware process configuration (as discussed in Section 3.2). That component uses pre-defined process types and

configuration options to automatically generate a process instance containing all necessary activities to match the properties of the current requests situation (cf. DCC4). As basis for this step, we include a comprehensive data model where contextual influences are stored (cf. DCC3) alongside different kinds of content-related data. This data model integrates process-related data with customer-related data as well as contextual information.

We now briefly introduce the different kinds of incorporated data by different sections of our data model. At first, such a system must manage data about its customers. Therefore, a customer data section comprises data about the companies, like organizational units or products. Another basic component of industrial production, which is important for topics like sustainability, are substances and (sustainability) indicators. As these are not specific for one company, they are integrated as part of a master data section. In addition, the data concretely exchanged between the companies is represented within a separate section (exchange data). To support this data exchange, in turn, the system must manage certain data relating to the exchange itself (cf. DCC1): For whom is the data accessible? What are the properties of the requests and responses? Such data is captured in a runtime data section in the data model. Finally, to be able to consistently manage the data request process, concepts for the process and its variants as well as for the contextual meta data influencing the process have been integrated with the other data. More detailed descriptions of these concepts and their utilization will follow in the succeeding sections.

3 Variability Aspects of Data Collection

This section deals with the necessary areas for automated process configuration: The mapping of contextual influences into the system to be used for configuration and the modeling of the latter.

3.1 Context Mapping

As stated in Section 1, a request regarding the same topic (in this case, a sustainability indicator) may have multiple variants influenced by a myriad of possible contextual factors (e.g. the number of involved parties or the data formats used). Hence, if one seeks to implement any kind of automated variant management, a consistent manageable way of dealing with these factors becomes crucial. However, the decisions on how to apply process configuration and variant management often cannot be mapped directly to certain facts existing in the environment of a system. Moreover, situations might occur, in which different contextual factors will lead to the same decision(s) according to variant management. For example, a company could integrate a special four-eyes-principle approval process for the release of data due to different reasons, e.g., if the data is intended for a specific customer group or relates to a specific law or regulation. Nevertheless, it would be cumbersome to enable automatic variant management

by creating a huge number of rules for each and every possible contextual factor. In the following, therefore, we propose a more generic mapping approach for making contextual factors usable for decisions regarding the data collection process.

In our approach, contextual factors are abstracted by introducing two separate concepts in a lightweight and easily configurable way: The *Context Factor* captures different possible contextual facts existing in the systems' environment. Opposed to this, the *Process Parameter* is used to model a stable set of parameters directly relevant to the process of data collection. Both concepts are connected by simple logical rules as illustrated on the left side of Figure 4. In this example, a simple mapping is shown. If a contact person is configured for a company (CF1), parameter 'Manual Data Collection' will be derived. If the company is connected via a tool connector (CF2), automatic data collection will be applied (P3). If the company misses a certain certification (CF3), an additional validation is needed (P2).

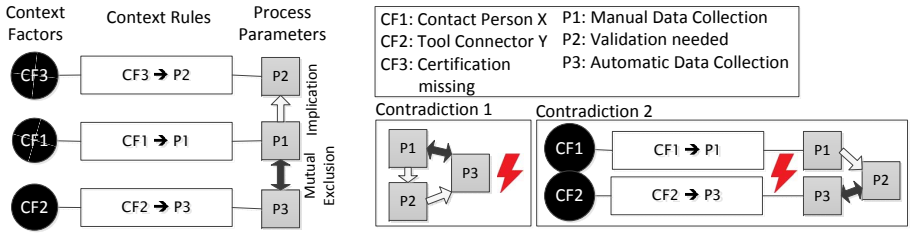

Fig. 4: Context Mapping

When exchanging data between companies, various situations might occur, in which different decisions regarding the process might have implications on each other. For example, it would make no sense to collect data both automatically and manually for the same indicator at the same time. To express that, we also include the two simple constraints 'implication' and 'mutual exclusion' for the parameters. For an example, we refer to Figure 4, where manual and automatic data collection are mutually exclusive.

Though we put emphasis on keeping the applied rules and constraints simple and maintainable, there still exist situations, in which these lead to contradictions. One case (Contradiction 1 in Figure 4) involves a contradiction only created by the constraints, where one activity requires and permits the occurrence of another activity at the same time. A second case (Contradiction 2 in Figure 4) occurs when combining certain rules with certain constraints, in which a contradicting set of parameters is produced. To avoid such situations, we integrate a set of simple correctness checks for constraints and rules.

3.2 Process Configuration

In this section, we will introduce our approach for process configuration. We have not only considered the aforementioned challenges, but also want to keep the approach as easy and lightweight as possible to enable users of the Sustain-Hub platform to configure and manage the approach. Furthermore, our findings include data about the actual activities of data collection as well as their relation to contextual data. Data collection often contains a set of basic activities that are part of each data collection process. Other activities appear mutually exclusive, e.g. manual or automatic data collection, and no standard activity can be determined here. In most cases, one or more context factors impose the application of a set of additional coherent activities rather than one single activity.

In the light of these facts, we opt for the following approach for automatic process configuration: For one case (e.g. a sustainability indicator) a process family is created. That process family contains a *base process* with all basic activities for that case. Additional activities, added to this base process, are encapsulated in *process fragments*. These are automatically added to the process on account of the parameters of the current situation represented in the system by the already introduced *process parameters* and *context factors*. Thus, we only rely on one single change pattern applicable to the processes, an insert operation. This operation has already been described in literature, for its formal semantics, see [2]. Thus our approach avoids problems with other operations as described in the context of other approaches like Provop [3].

To keep the approach lightweight and simple, we model both the base process and the fragments in a PAIS that will be integrated into our approach. Thus, we can rely on the abilities of the PAIS for modeling and enacting the processes as well as for checking their correctness.

To enable the system to automatically extend the base process at the right points with the chosen fragments, we add the concept of the *extension point (EP)*. Both EPs and fragments have parameters the system can match to find the right EP for a fragment (see Figure 5 for an example with two EPs and three fragments with matching parameters). Regarding the connection of the EPs to the base processes, we have evaluated multiple options as, for example, connecting them directly to activities. Most options introduce limitations to the approach or impose a fair amount of additional complexity (see [3] for a detailed discussion). For these reasons we have selected an approach involving two *connection points* of an EP with a base process. These points are connected with nodes in the process as shown in Figure 5. Taking the nodes as connection points allows us to reference the nodes' id for the connection point because this id is stable and only changes in case of more complicated configuration actions [3]. If the base process contains nodes between the connection points of one EP, an insertion will be applied in parallel to these, otherwise sequentially. Furthermore, if more than one fragment shall be inserted at one EP, they will be inserted in parallel to each other.

The example from Figure 5 illustrates this approach refining the aforementioned scenario. It comprises four basic activities for configuring the data collec-

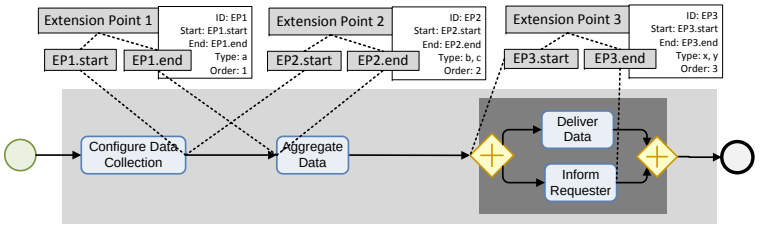

Fig. 5: Process Annotation

tion, aggregating the data, and delivering the collected data. To insert further activities for data collection and processing, three EPs are defined. These have different connection points to the nodes in the process. Figure 5 further shows properties for the EPs including the connection points and an EP type (e.g. 'review' or 'approval') that will be used to determine which extension(s) may be applied at that point during configuration. A particular EP may be applicable for multiple fragments, but multiple EPs for the same fragment are not possible as it would be ambiguous, for which point to apply it. This is automatically checked during modeling. Another property, called 'Order', governs in which order extensions will be applied. It will also be checked that the ordering is not ambiguous.

However, to correctly insert fragments into a base process other facts must be considered as well: First, it must be determined whether a fragment, inserted in a LOOP, shall be executed multiple times. Second, it must be defined how the activities of a fragment are exactly inserted into the base process. Therefore, we extend the process fragment with a number of parameters. The first parameter, 'Insert', governs, if the fragment shall be inserted directly into the base process or as sub-process. The latter could be considered, i.e., when the fragment comprises a bigger number of activities with a complicated structure. The second is the 'Type' that relates to the EP type and is used to match both of them. The third, 'Exec' governs, if a fragment might be executed multiple times.

Figure 6 illustrates process fragments and their parameters utilizing the scenario presented in Section 1. It contains five fragments: Fragments 1 and 2 comprise the activities for data collection of the two suppliers. They are integrated at the same position in parallel. Fragments 3 and 4 comprise the activities for data processing, integrated in parallel as well. Fragment 3 further demonstrates the insertion as sub-process if the base process shall not be bloated with many activities from large fragments. As both EP1 and EP2 are at the same position the 'Order' parameter comes into play governing the insertion of the data processing activities after the data collection activities. Finally, Fragment 5 contains an activity for demonstrating the insertion at the erroneously defined EP3. This would cause a violation to the regular nesting of the patterns (as XOR and AND) in the process called block structure. This is recommended for understandable modeling [4] and required by many PAIS for correct execution. As aforementioned, such definitions are prevented by automatic checks.

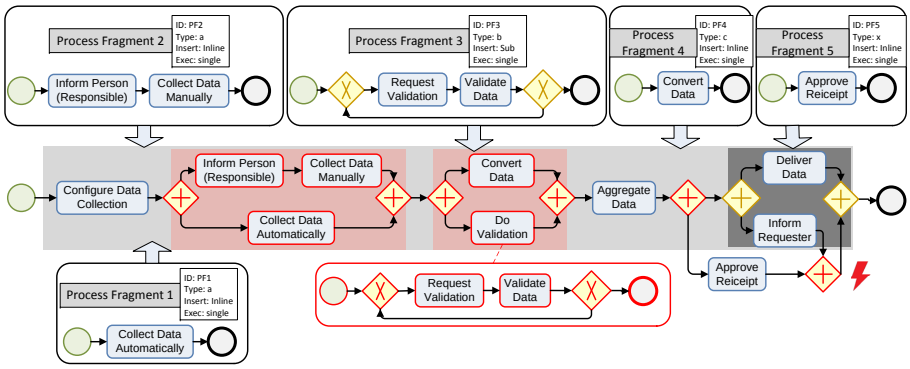

Fig. 6: Process Fragments Insertion

By relying on the capabilities of the PAIS, we keep the number of additional correctness checks small. However, connection points are not checked by the PAIS and could impose erroneous configurations. To keep correctness checks on them simple we rely on two things: The relation of two connection points of one EP and block-structured processes [4]. The first fact avoids the need to check all mutual connections of all connection points as two always belong together. The second one implies certain guarantees regarding the structuring of the process models. That way, we only have to check a small set of cases, as e.g., an erroneous definition of an extension point, as EP3 in Figure 5 that would cause a violation to the block structure when inserting a fragment as shown in Figure 6.

4 Implementation

This section elaborates on the concrete realization of the concepts presented in Section 3. It shows how the abstract context mapping and process configuration concepts can be transformed into a concrete implementation. At first, we discuss the classes we created to implement our approach. Thereafter, we elaborate on the components we apply to concretely conduct the process configurations.

To illustrate the relations of the concepts crucial for our concept, Figure 7 shows a simplified class diagram. The latter indicates a separation between the classes defining the configuration concepts (build-time) and classes managing their execution (runtime). To be able to reuse a process family easily in different contexts, we separate the concepts into a process family (class `ProcessFamily`) and a so-called context application (class `ContextApplication`). The process family comprises the concepts for representing the base process as well as the process fragments (classes `AdaptableProcessTemplate` and `AdaptableProcess-TemplateFragment`). These classes hold references to concrete process templates of the PAIS. In addition, the base process has an arbitrary number of extension points (class `ExtensionPoint`) that mark the points where process fragments may be inserted. In order to be able to determine, when and at which

extension point one of these fragments shall be inserted, we integrate an activation condition (class `Condition`). This condition implements an interface we use to unite all rules or expressions of our implementation (interface `Expression`). The activation condition depends on the set of process parameters connected to the process template (class `ProcessParameter`). As discussed, the process parameters may be associated with constraints. These are realized by the class `ProcessParamterConstraint`, which implements the `Expression` interface as well.

To be able to use a process family for a certain situation, we must map existing contextual factors into SustainHub. That way, they become usable for the process parameters. This is done by the context application (class `ContextApplication`) that encapsulates process parameters as well as context rules (class `ContextRule`). A context rule, in turn, refers to context factors and process parameters. The rule itself is implemented as a JavaScript expression (class `JavaScriptExpression`) that also implements the expression interface.

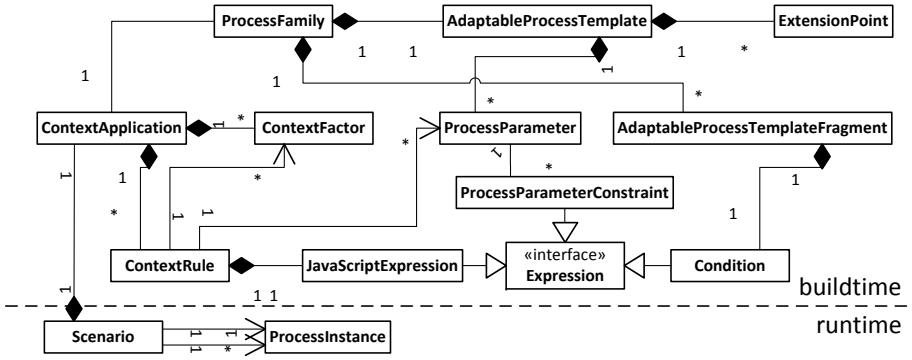

Fig. 7: Class Diagram

At runtime, we apply a so-called scenario (class `Scenario`) that models a current situation. It comprises all concepts of a process family and a context application via an included context application. Besides that, the scenario also refers to multiple process instances (class `ProcessInstance`), which represent the base process and potential sub-processes executed in the PAIS.

We proceed with a discussion of the concrete components and procedure at runtime. First of all, we have implemented the process configuration as an adaptation operation on the running process instances instead of configuring the process templates. Otherwise we would have created a high number of additional configured process templates for each possible configuration. Therefore, we have created an additional automatic adaptation component that interacts with the PAIS and the process instance. This is illustrated in Figure 8.

In the following, we introduce the different steps performed for the execution of a scenario. The first action is to start a process instance (cf. Figure 8 (1))

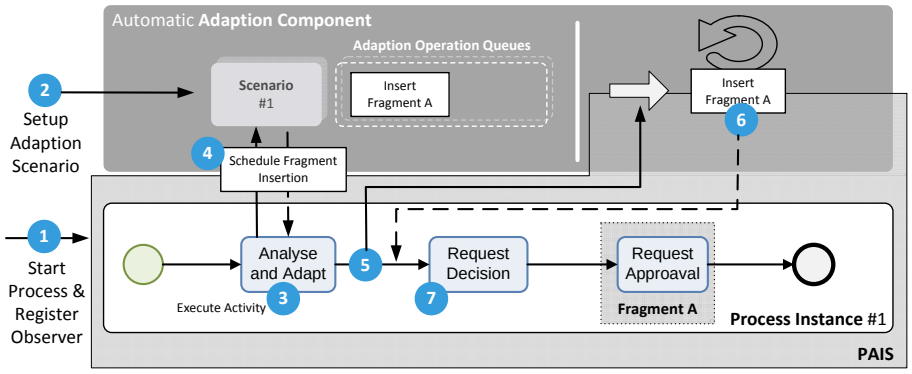

Fig. 8: Configuration Procedure

that corresponds to a base process of a process family. This action also registers the automatic adaptation component as observer on this process instance so the former can interact with the latter. This is necessary due to a specific property of adaptable PAIS: For adapting a running process instance, instance execution must be temporarily suspended. This can only be done when no activity is active. Being registered as observer, the automatic adaptation component may apply adaptations directly when the instance gets suspended. The first action of the automatic adaptation component is to setup the scenario for the current process instance (2). After that, the first activity of the process instance gets executed (3). For every base process, this activity is a so-called 'analyse and adapt' activity we apply to gather context information from both the user and the environment. This data is then stored as context factors and passed to the automatic adaptation component (4). The latter then starts a scenario engine that determines the adaptation actions from the context factors. After that it schedules a suspension of the process instance, which is applied right after the termination of the 'analyse and adapt' activity.

When the 'anaylse and adapt' activity is finished, the process instance is suspended and the automatic adaptation component is triggered (5). The latter then uses the API of the PAIS to apply the scheduled adaptations (6). Following this, the instance is reactivated and proceeds with its execution (7). This approach bears the advantage that the running process instance can be adapted at any position after the 'analyse and adapt' activity, while the user is not impeded from the adaptation.

Having explained our adaptation approach abstractly, we now go into detail about the interaction between the components. This is illustrated as a sequence diagram (cf. Figure 9).

After being activated by the PAIS, the 'analyse and adapt' activity determines the context factors values. Then, it calls the automatic adaptation component via a REST interface passing the id of the process instance and the context values to it. The adaptation component uses the ID to determine the right sce-

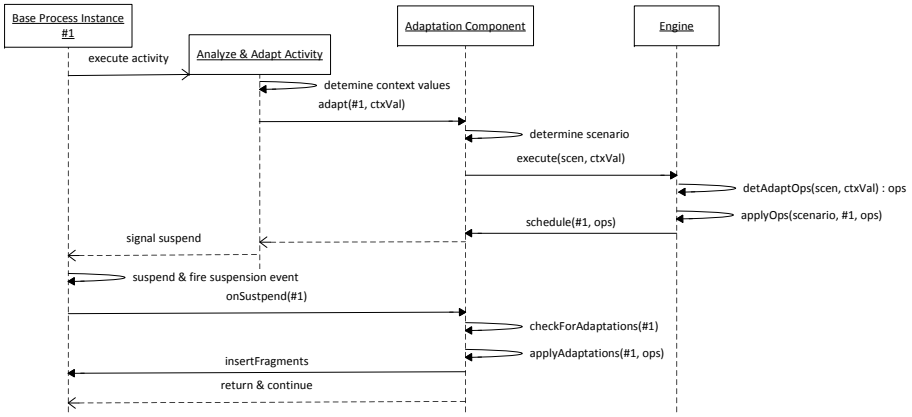

Fig. 9: Sequence Diagram

nario and calls the engine to execute the scenario with the received context values. In turn, the engine uses this data to determine the concrete adaptation operations. In particular, it runs the context rules to obtain the process parameters and the activation conditions to determine which fragments are to be inserted for the current situation. After that, the engine applies these operations meaning it signals the adaptation component and the activity to signal the suspension of the process instance.

When the 'analyse and adapt' activity finishes, the instance gets suspended automatically and the adaptation component gets triggered to apply the adaptations. The latter then checks, which operations are scheduled and applies them to the instance. Finally, the adaptation component returns control to the process instance that proceeds with its execution.

5 Preliminary Practical Application

We already implemented a set of indicator use cases and, based on that, we have had a first feedback loop with our industry partners in the SustainHub project. As our partner companies could confirm the practical applicability of our approach, we will now continue to implement a bigger set of indicators for further evaluations. In the following, we show one use case dealing with energy consumption that is rather simple. It involves three context factors based on the following questions: Can the responder distinguish between own and bought energy, between consumption categories, and energy sources? The context factors are mapped to one process parameter. In Figure 10, we show this mapping, the base process and an example fragment modeled in the PAIS we used for our implementation (AristaFlow [5]), and the SustainHub web-GUI while executing this process. In addition to this, we have recently started to implement a use case from the educational domain: the management of theses at a university. Our approach shows promise to also suit this domain well.

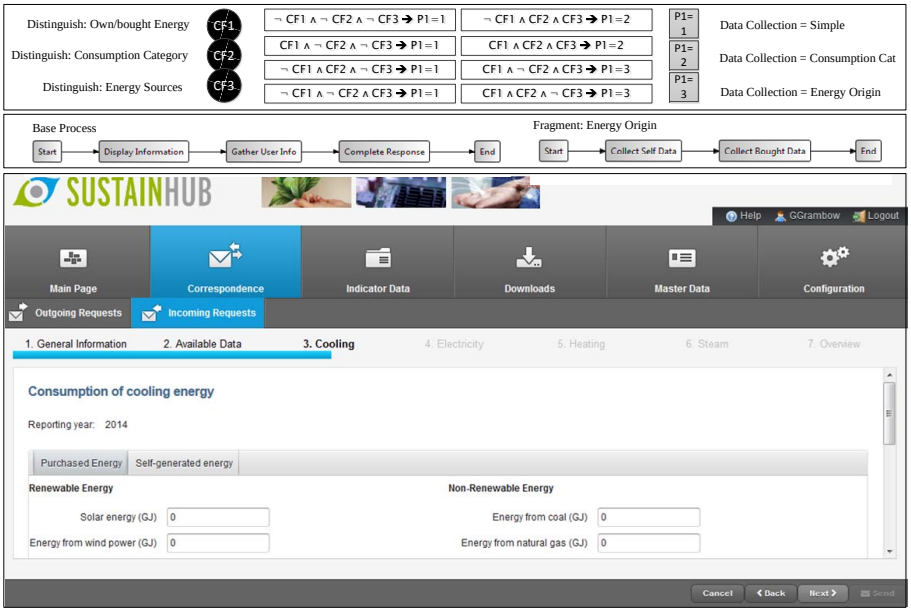

Fig. 10: Application Use Case

6 Related Work

The the area of sustainable supply chain communication is relatively new. Despite this fact, a set of approaches exist in this area (e.g., [6] or [7]). However, instead of proposing technical solutions, they focus on analyzing the importance of corporate sustainability reporting or evaluating sustainability indicators. Therefore, we will focus on the technical aspects in the rest of this section.

Regarding the topic of process configuration, various approaches exist. Most of them focus on the modeling of configuration aspects of processes. One example is C-EPC [8], which enables behavior-based configurations by integrating configurable elements into a process model. Another approach with the same focus is ADOM [9]. It allows for the specification of constraints and guidelines on a process model to support variability modeling. Such approaches focus on the extension of available process modeling notations and produce maximized models containing all possible activities. Such models might be difficult to comprehend and maintain.

In [10], a comprehensive meta-model for process variability is proposed. It incorporates a set of different perspectives, as e.g., a functional as well as a resource perspective. Therefore, it allows specifying a rich set of process information as well as necessary configurations and changes to it. Compared to our approach, this meta model is rather complicated and heavyweight and therefore difficult to understand. Moreover, it lacks adequate facilities for context modeling.

The approach presented in [11] features a meta model for process fragments as well as also a definition for actions on these fragments (e.g., composition). However, it strongly focuses on this topic and neglects the modeling of contextual facts that trigger such operations. Another meta model is proposed by [12]. Its intend is to extend process models to incorporate aspects that have been neglected so far, i.e., the data and resource perspectives. The authors therefore introduce role-task and object-task associations for EPCs. This, however, provides no additional support for easier automated or contextual configurations.

Besides the approaches just discussed, which focus on the configurability topic in general, there exist other approaches focusing on specific aspects like correctness or recovery. For example, [13] provides a concept for enabling recovery strategies for workflows incorporating dynamically inserted fragments. This is achieved by introducing transactional behaviour for the fragments, which enables two different recovery strategies: a forward recovery and a backward recovery strategy. The former repairs a faulted process fragment by executing additional process logic. This is achieved by inserting a new fragment into the faulted one that is capable of repairing it. The backward recovery strategy, in turn, is applied if the problem is too severe to repair the fragment. Therefore, it is compensated or removed to enable another way of achieving the business goal of the process. The approach presented in [14] aims at presenting correctness for process configuration. In particular, it enables verification of configurable process models intended for execution. This verification is executed at design time and imposes no additional constraints on the model. The authors have further implemented their approach concretely for the C-YAWL language. Both of these approaches offer solutions for specific aspects of process configuration. None of them, however, provides means to support automated and contextual configuration abilities as out approach does.

All modeling approaches share the same drawbacks: First, they strongly focus on the modeling of process configuration aspects and neglect its execution. Second, process configuration must be manually applied by humans, which might be complicated and time-consuming. Two approaches, which take the automatic composition of processes out of a set of fragments into account are presented in [15] and [16]. The former addresses the issue of incorporating new run-time knowledge in pervasive computing. It employs pervasive process fragments and allows modeling incomplete and contextual knowledge. To be able to reach the process goal and include dynamic contextual knowledge, all relevant data is encoded as AI planning problem. In [16], however, an approach for dynamically composing fragments at run-time is presented. It is applied it to a logistics scenario and includes an explicit variation model as addition to a base process model and process fragments. At run-time a solver creates a solution using the specified models and context data implying the insertion of fragments matching the specified situation.

Both approaches focus strongly on the automatic selection of potentially concurring fragments. In contrast, our approach targets user support as well: we emphasize keeping the model simple and apply correctness checks for the user-

modeled concepts. An approach, more closely related to ours, is Provop [3]. It allows storing a base process and corresponding pre-configured configurations. As opposed to Provop, SustainHub provides a framework for completely automatic, context-aware configuration of processes without need for any human interaction. Furthermore, Provop is more fine-grained, complicated, and heavyweight.

Another approach having many similarities to ours is Corepro [17]. It enables modeling and automated generation of large process structures. Furthermore, it comprises features for dynamic runtime adaptation as well as exception handling. However, it has one major drawback: context data utilized to generate the processes is limited to product data. Processes get generated solely relating to the product for whose production they intended. Further reading regarding other configuration approaches can be found in [18] and [19] as well as our predecessor paper for SustainHub [1].

7 Conclusion

In this paper, we have introduced a lightweight approach for automatic and contextual process configuration as required in complex scenarios. We have investigated concrete issues relating to sustainability data collection in supply chains. Our approach centralizes data and process management uniting many different factors in one data model and supporting the entire data collection procedure based on process templates executable in a PAIS. Moreover, we enable this approach to apply automated process configurations conforming to different situations by applying a simple model allowing for mapping contextual factors to parameters for the configuration. However, our approach is not only theoretical but is applicable in a real information system to support supply chain communication. Therefore, we have shown specifics of the implementation of our concepts. In future work, we plan to continue the evaluation of our work with our industrial partners and also in the educational domain. Further, we plan to extend our approach to cover further aspects regarding runtime variability, automated monitoring, and automated data quality management.

Acknowledgement

The project SustainHub (Project No.283130) is sponsored by the EU in the 7th Framework Programme of the European Commission (Topic ENV.2011.3.1.9-1, Eco-innovation).

References

1. Grambow, G., Mundbrod, N., Steller, V., Reichert, M.: Challenges of applying adaptive processes to enable variability in sustainability data collection. In: 3rd Int'l Symposium on Data-Driven Process Discovery and Analysis. (2013) 74–88

2. Rinderle-Ma, S., Reichert, M., Weber, B.: On the formal semantics of change patterns in process-aware information systems. In: Proc. 27th Int'l Conf on Concept Modeling (ER'08). Number 5231 in LNCS, Springer (October 2008) 279–293
3. Hallerbach, A., Bauer, T., Reichert, M.: Configuration and management of process variants. In: Int'l Handbook on Business Process Management I. Springer (2010) 237–255
4. Mendling, J., Reijers, H.A., van der Aalst, W.M.: Seven process modeling guidelines (7pmg). Information and Software Technology **52**(2) (2010) 127–136
5. Dadam, P., Reichert, M.: The adept project: A decade of research and development for robust and flexible process support - challenges and achievements. Computer Science - Research and Development **23**(2) (2009) 81–97
6. Pagell, M., Wu, Z.: Building a more complete theory of sustainable supply chain management using case studies of 10 exemplars. Journal of Supply Chain Management **45**(2) (2009) 37–56
7. Singh, R.K., Murty, H.R., Gupta, S.K., Dikshit, A.K.: An overview of sustainability assessment methodologies. Ecological indicators **9**(2) (2009) 189–212
8. Rosemann, M., van der Aalst, W.M.P.: A configurable reference modelling language. Information Systems **32**(1) (2005) 1–23
9. Reinhartz-Berger, I., Soffer, P., Sturm, A.: Extending the adaptability of reference models. IEEE Trans on Syst, Man, and Cyber, Part A **40**(5) (2010) 1045–1056
10. Saidani, O., Nurcan, S.: Business process modeling: A multi-perspective approach integrating variability. In: BPMDS 2014. (2014) 169–183
11. Eberle, H., Leymann, F., Schleicher, D., Schumm, D., Unger, T.: Process Fragment Composition Operations. In: Proceedings of APSCC 2010, IEEE Xplore (December 2010) 1–7
12. La Rosa, M., Dumas, M., ter Hofstede, A.H., Mendling, J., Gottschalk, F.: Beyond control-flow: Extending business process configuration to roles and objects. In: Conceptual Modeling-ER 2008. Springer (2008) 199–215
13. Eberle, H., Leymann, F., Unger, T.: Transactional process fragments - recovery strategies for flexible workflows with process fragments. In: APSCC 2010. (2010) 250 – 257
14. Van der Aalst, W., Lohmann, N., La Rosa, M., Xu, J.: Correctness ensuring process configuration: An approach based on partner synthesis. In: Business Process Management. Springer (2010) 95–111
15. Sirbu, A., Marconi, A., Pistore, M., Eberle, H., Leymann, F., Unger, T.: Dynamic composition of pervasive process fragments. In: ICWS 2011. (July 2011) 73–80
16. Murguzur, A., De Carlos, X., Trujillo, S., Sagardui, G.: Dynamic composition of pervasive process fragments. In: CAiSE 2014. (2014) 241–255
17. Müller, D., Reichert, M., Herbst, J.: A new paradigm for the enactment and dynamic adaptation of data-driven process structures. In: 20th Int'l Conf. on Advanced Information Systems Engineering (CAiSE'08). Number 5074 in LNCS, Springer (June 2008) 48–63
18. Torres, V., Zugal, S., Weber, B., Reichert, M., Ayora, C., Pelechano, V.: A qualitative comparison of approaches supporting business process variability. In: BPM'12 Workshops. LNBIP, Springer (September 2012)
19. Ayora, C., Torres, V., Weber, B., Reichert, M., Pelechano, V.: Vivace: A Framework for the Systematic Evaluation of Variability Support in Process-Aware Information Systems. Information and Software Technology (May 2014)