

Universität Stuttgart

Fakultät Informatik, Elektrotechnik und Informationstechnik

**An Overview on Implicit Green
Business Process Patterns**

Alexander Nowak, Frank Leymann

Report 2013/05
August 07, 2013



**Institut für Architektur von
Anwendungssystemen**

Universitätsstr. 38
70569 Stuttgart
Germany

CR: H.4.1

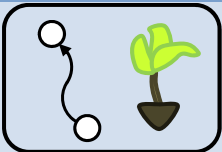
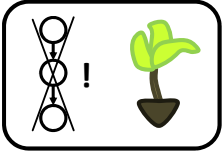
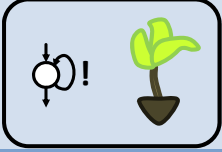
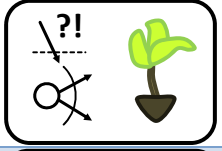
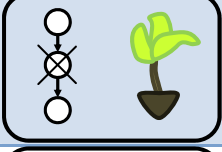
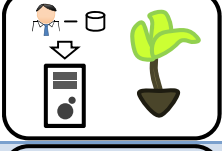
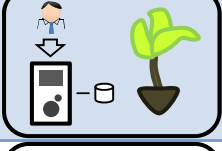
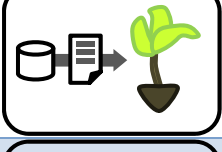
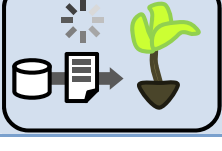
Abstract

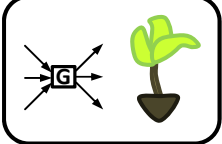

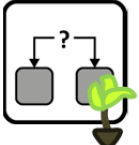
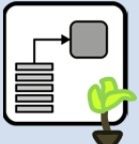

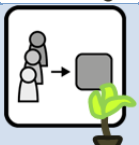
The environmental impact of doing business becomes an increasingly relevant aspect for organizations, not only because of legal requirements but also because more customers care about environmental influences. Most organizations, however, do not have comprehensive knowledge on how to restructure their business processes accordingly. In previous work we proposed green business process patterns that address the environmental impact explicitly as one possible solution. Beyond those patterns, we now provide environmentally relevant patterns derived from existing patterns of different domains. In this work, we present a set of such patterns related to automated business processes. The identified patterns support stakeholders when analyzing their business processes with respect to the environmental impact and, therefore, supporting green business process reengineering.

1. INTRODUCTION

This document provides a set of patterns that may be used to consider the environmental impact of business processes. The patterns are based on existing knowledge that has been captured in patterns from the domains of workflows, application architectures, and Cloud computing. The proposed list of patterns, a first attempt of building a pattern language, is not complete but serves as a starting point for further research. The following Table contains an overview of the patterns described in this document.

Table 1 Pattern Overview.

Pattern-Icon	Pattern Name	Pattern Intent
	Green Control Flow	Smart control flow constructs improve the efficiency of resource usage while executing a business process.
	Green Explicit Termination	Terminate sub-processes when there is nothing else to do for them.
	Green Multiple Instances With a Priori Runtime Knowledge	Handle the number of activities in a process model while the exact number of instances is not known during design time.
	Green External Choice	Provide several alternative control flow branches that are selected depending on the environment.
	Green Cancel Activity	Withdraw activity instances that are no longer relevant for the execution of the process instance.
	Green Client Session State	Optimize the number of server nodes by making them stateless.
	Green Server Session State	Hold session information on server side to reduce energy consumption in communication and transformation.
	Green Data Transfer Object	Optimize system calls to reduce load and energy consumption.
	Green Lazy Load	Avoid upfront loading of possibly unnecessary information, instead load data only when really needed.

	Green Gateway	Manage the access to external systems or resources in order to reduce total number of access instances.
	Green Public Cloud	Use external cloud resources to improve resource efficiency and reduce in-house power consumption.
	Green Loose Coupling	Reduce the dependencies between individual components to use varying resources for their runtime environment and execution.
	Green Batch Processing Component	Delay and bundle the processing of requests based on internal or external requirements and dependencies.
	Green Eventual Consistency	Distribute data among replicas while reducing the synchronization overhead to a minimum.
	Green Shared Component	Share application components between multiple tenants to improve resource efficiency.

2. Characteristics of Green Business Processes

Business processes are performed by invoking different activities defined in the business process model. Each of these activities is using resources that support and ensure the correct execution of that activity. Such resources can be raw-materials, a drill bit, a hard disk drive, electricity, or another process or service, for example. In (Nowak et al. 2011) we found that in order to cope with the holistic character of environmental aspects, we need to consider a wide variety of perspectives. Moreover, we can postulate that the impact of resources to the environmental impact of a business process is crucial. Here, we need to differentiate and consider two different aspects: (i) which resources are environmentally compatible and best fit a given work item or objective, and (ii) in which way should resources be used within a business process.

To identify the potential of reducing the environmental impact of a business process, we analyzed current patterns and specify characteristics that can be applied to new patterns. To cover the variety of perspectives, we decided to use the different perspectives of business processes, described by (Jablonski and Bussler 1996). Especially, we emphasize their influence on the environmental impact. The perspectives are described in the following.

Process perspective: The process perspective describes the control flow of a business processes, i.e. it defines the order in which activities are executed.

Usually, this perspective is represented by some graphical model. Depending on the defined control flow a business processes may lead to a different environmental outcome. Process models that contain many different roles, for example, may lead to a higher environmental impact due to the necessary context transmission. In other scenarios it might be useful to structure process models in such a way that the use of particular resources is somehow bundled. Different database queries, for example, may be fetched at once. This allows keeping resources in standby for a longer time which saves energy consumed by the database management system.

Data perspective: The data perspective describes the structure of the data objects used by a business processes as well as the way they are associated to the business process. The way of associating data elements to processes has significant influence on the total environmental impact. For example, consider an automated business processes that invokes an activity multiple times. If this activity is designed as stateless component, all information necessary for the execution must be provided to the activity each time it gets invoked. If you use a stateful component or some caching functionalities instead, information transfer will be reduced which may lead to a decrease in energy consumption. A similar scenario may be envisioned in industry where raw materials or semi-finished goods need to be transferred from one place to another.

Organization perspective: The organization perspective describes the roles and organizational units that are involved in a business process. That includes both internal and external roles. For a sustainable execution of processes a lot of changes in different roles or even external organizational units may be disadvantageous. In this case, the process context needs to be transferred each time the role changes or goods need to be transported to other sites, for example. Thus, when considering the environmental perspective of business processes, the proper selection of roles and of collaboration partners as well as their integration becomes even more important than in regular process design.

Resource perspective: The resource perspective is another important perspective as it covers which types of resources are used within business process. Resources may be humans, machines, raw-materials, and auxiliary materials like energy, for example. Depending on the chosen resources, different environmental outcomes may be achieved. In some IT scenarios it might be useful to use Thin-Clients or renewable energy. Another example is the use of centralized environments which allows utilizing economies of scale or the bundling of activity executions in order to prevent servers from changing their state too often.

Operation perspective: The operation perspective describes how the atomic elements of a business process are used. This may be represented, for example, by a scripting language defining how to invoke external applications. For environmental-aware business processes, this perspective is strongly correlated with the resource and data perspective. It defines the way activities are invoked or which kind of activities or additional services are executed within the business

process. The use of the Green Compensation pattern from (Nowak et al. 2011), for example, may be implemented as part of the invocation of an activity.

Integration perspective: The integration perspective describes how the different perspectives are joined together. In most scenarios this ends up in some hierarchy of processes and activities. In our use case this may be interesting as the relations between the different activities and processes are described explicitly. However, this perspective is more important for the analysis of existing structures as many relations are already covered by the single perspectives. Nevertheless, in some scenarios this perspective may help stakeholders to coordinate process executions especially with respect to the distribution and sharing of resources that are needed for process execution.

The analysis of each of these business process perspectives indicates that every perspective has an individual potential to decrease the environmental impact of business processes. Therefore, we can assume that patterns that address these perspectives from an environmental point of view may also have a significant relevance in designing environmentally-aware business processes. However, their application is strongly dependent on the scenario in which they are used. In order to support stakeholders that focus on the sustainable (re-)design of their business processes, we want to provide a set of commonly known patterns that are transferred in the context of this paper. This means that we analyze existing patterns addressing at least one of the introduced perspectives and check their applicability to environmental aspects of business processes. For better usability of our pattern catalogue, we describe variants of those existing patterns so that they are ready to use in green business process reengineering scenarios. Please note that the integration perspective is influenced whenever any of the other perspectives are modified. Thus, we do not describe this perspective in each of our patterns but consider it in the results presented in Section 5.

3. Green Business Process Patterns

In this section we want to provide a set of patterns that are derived from existing patterns of other domains but put in the context of environmentally-aware improvement of business processes. To provide an easy and comprehensible usage of the patterns we describe all of them in the same structured format (see Section 3.1). The patterns have been selected based on the impact on the perspectives described in Section 2. Considering the probably most important aspect, namely the resources of a business process, we have selected patterns from the domains of workflows (Aalst et al. 2003), application architectures (Fowler 2003) in general, and Cloud architectures (Fehling et al. 2013) in particular. Please note that the selection of these patterns is based on our personal investigations related to green business process patterns, their corresponding influence factors, and therefore their ability to positively support these influence factors. We do not claim that these are all patterns related to that subject. However, we think they provide a good starting point for (re-)designing business processes.

3.1 Pattern Format and Language

In the following sections we want to extend our previously proposed pattern language described in (Nowak et al. 2011). When talking about a “language of patterns” we stick closely to the interpretation of Hanmer (Hanmer 2012) who describes a language as a set of patterns that are used to solve a problem of a particular domain. The patterns comprised in a language are structured in such a way that stakeholders can navigate through the complete set of patterns, selecting suitable patterns, and recognize the relations between the patterns in order to solve even bigger problems. Depending on different use case scenarios, a different set of suitable patterns may be selected. This aspect is also reflected by the abstract description of solutions which explicitly does not provide final and concrete solutions for a very specific use case.

To describe and document the patterns of this work we have decided to choose a simple and straightforward format. This format is geared to commonly known literature, like the work of Alexander (Alexander et al. 1977), Fowler (Fowler 2003), and Gamma et al. (Gamma et al. 2000). However, our format differs from the original formats of the observed patterns. This is done on purpose as we want to emphasize the new aspects, problems, and solutions in the domain of environmental improvement of business processes. Moreover, the uniform format eases the use of patterns significantly. Please note that we do not intend to describe the existing patterns one-to-one. In fact, we want to present variants that are directly related to the subject of this work.

For documentation, each pattern is defined by a name that is unique within our pattern language. The name helps to identify patterns and to navigate through the catalogue. Moreover, we did not use the original name of the patterns but added the prefix “green” that emphasizes the new application domain. Besides the name of a pattern we also provide the intent of that pattern which describes the purpose of the pattern in a single sentence. Next, we describe the context the pattern may be used in. The context is strongly related to the new domain and represents the new challenges arising. In this section, we also provide some information about the affected business process perspectives, introduced in Section 3.1. Subsequently, we provide an abstract solution for the problem. Like any other patterns, this solution is typically not “ready-to-use”. We still need to consider the concrete use case and the corresponding constraints to develop a suitable solution. Next, we provide information on the result after applying the pattern and one or more examples, also known as “known uses”, describing concrete application scenarios of the patterns. The last section points out the relations to other patterns either because they are usually used together, they might be considered to use them together, or because they do not fit together due to mutual constraints. Please note that we do not provide all possible relations in this work as we have limited the number of patterns to a basic set. Thus, there might much more patterns from the different domains that are able to support the patterns of this work in concrete use cases. The identification of these patterns is, however, subject to the user knowing his concrete application scenario.

In the pre-selection phase of our method we have analyzed various patterns of the aforementioned domains. All of the identified patterns do have the potential to improve the environmental impact of business processes. In the following, we provide a brief overview of the resulting patterns:

Workflow Patterns: *Green Control Flow, Green Explicit Termination, Green Multiple Instances With a Priori Runtime Knowledge, Green External Choice, Green Cancel Activity, Multiple Instances Without Synchronization, Multiple Instances With a Priori Design Time Knowledge, Cancel Case.*

Application Architecture Patterns: *Green Client Session State, Green Server Session State, Green Data Transfer Object, Green Lazy Load, Green Gateway, Green Query Object.*

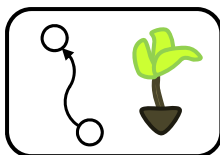
Cloud Patterns: *Green Public Cloud, Green Private Cloud, Green Elastic Infrastructure, Green Elastic Platform, Green Loose Coupling, Green Batch Processing Component, Green Data Access Component, Green Stateful Component, Green Stateless Component, Green Shared Component, Green Eventual Consistency, Green Elasticity Manager, Green Elastic Load Balancer, Green Elastic Queue, Green Elasticity Management Process, Green Standby Pooling Process.*

The following sections provide the results of the detailed analysis phases of our method. We have selected five patterns of each domain that are documented in detail using our proposed pattern format.

3.2 Workflow Patterns

Workflow patterns basically describe solutions on how to design business process in order to achieve certain business objectives. So far, these patterns did not consider the environmental impact as a design criterion. Besides the overview in Section 4.1 we have analyzed the following patterns in detail: Control Flow, Explicit Termination, Multiple Instances With a Priori Runtime Knowledge, External Choice, and Cancel Activity. For each of these patterns we present a corresponding variant that focuses on the improvement of the environmental impact. For further details on the original patterns see (Aalst et al. 2003).

Green Control Flow



Smart control flow constructs improve the efficiency of resource usage while executing a business process.

Context: Business processes are designed to compose multiple activities, i.e. working tasks, whose purpose is to achieve certain business objectives. To perform those activities corresponding implementation artifacts need to be specified. Such implementation artifacts can either be humans that perform a task manually or software that performs a task automatically. Based on the expected result, the activities are arranged in a specific order. In most cases the interactions between activities and resources are designed from a pure technical point of view. However, from an environmental point of view, the usage of those resources might be inefficient because there could be many different requests of a

resource, huge amounts of data needs to be transferred, and different organizational departments are involved.

When using this pattern the following perspectives are affected:

Process Perspective: The control flow, i.e. the ordering of the activities of a process, might be changed to achieve an improved execution.

Operation Perspective: Based on a changed control flow the way external operations are invoked may be changed, too.

Organization Perspective: A redesigned control flow might influence the communication between different organizational departments. Basically, the objective is to reduce the communication and management overhead.

Solution: The basic idea of this pattern is the restructuring of the execution steps of a business process while explicitly considering environmental objectives. Like applying conventional process optimization patterns, the order in which the different steps are performed will probably change. Based on existing design patterns, the following control flow design constructs may be used to reduce the total environmental impact of a business process:

Sequence: This construct is used to execute one activity after another, i.e. after the predecessor activity has completed. This may be used to keep resource utilization at a stable level.

Parallel Split: This construct is used to perform activities in parallel. This increases the resource utilization and may therefore lead to better duration and reduced resource allocation time.

Synchronization: This construct is used to merge one or more paths in a control flow, e.g. to switch from parallel to sequential execution. The results of each incoming path will be evaluated and processed.

Exclusive Choice: This construct is used to explicitly choose a certain path in the control flow. An example is the shipment of products which may be shipped as conventional or carbon free package. Therefore, the decision might be based on process data or user data, for example.

Simple Merge: This construct is used in a point in the workflow process where two or more alternative branches come together without synchronization. Depending on the concrete use case this might be an alternative to the synchronization construct.

Result: The result of the application of this pattern is a restructured business processes that considers both the functional aspects as well as an efficient and environmentally beneficial execution of the different process steps. This allows changing the execution order of activities based on the efficient usage of the underlying resources.

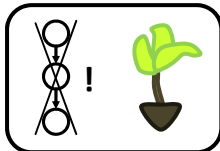
Example: On example for an exclusive choice is the “go green” option provided by DHL (Deutsche Post Ag 2013). Customers are able to choose whether they want to ship their packages in a conventional way or in a carbon-free way. When

choosing the latter one, DHL additionally charges the customers for investments in climate projects. Another example is the parallelization of backup tasks of a system. Due to the parallelization, the resource utilization will be increased to a certain level which improves the efficiency and releases the resources earlier so that they can be used by other tasks.

Relations to other patterns:

<i>Green Data Transfer Object:</i>	The change of the control flow does also determine when and in which order different data objects are needed and which information must be included.
<i>Green Gateway:</i>	It might be suitable to integrate a green gateway that allows to bundle requests in order to reduce communication and management overhead.
<i>Green Variant:</i>	This pattern may be used to introduce new control flow paths that build a new, green variant of the existing one. An Exclusive Choice may be used to decide which patch should be executed.
<i>Process Automation & Human Process Perf.:</i>	If the way of performing activities and the underlying resources should be changed, these patterns may provide some basics on how this can be done.
<i>Resource Change:</i>	The substitution of existing resources to better fit a new control flow may be beneficial for the total environmental impact.

Green Explicit Termination



Terminate sub-processes when there is nothing else to do for them.

Context: Complex business processes are usually not performed within a single hierarchical tier but consist of several different sub-processes. Such sub-processes are intended to encapsulate specific tasks that are created by experts and are reusable in different scenarios. In some cases, however, the state of a sub-process remains active although there is nothing more to do, for example, there are no more active activities within the sub-process and no more activities that can be activated. In such scenarios, the corresponding sub-process has to be terminated in order to release the resources for usage with other activities.

When using this pattern the following perspectives are affected:

<i>Process Perspective:</i>	Unless the process engine supports the termination of such sub-processes, the termination of a sub-process must be modeled explicitly.
<i>Resource Perspective:</i>	The bound resources may be release earlier which influences the way these resources are used. For example, the utilization of the resources will be decreased and, therefore, other or less resources may be used.

Solution: If there are no consistency checks done by process engines, the termination activity, i.e. the termination nodes, must be modeled explicitly within a sub-process. Based on various conditions, like the state of the containing activities, the continuity of the process instance has to be checked. These conditions can either be based on the process instance information or in some cases based on user decisions. If the conditions are fulfilled, the instance of the sub-process can be terminated. Note that in some cases the process engine covers that checks. In those cases, the explicit termination solely focuses on termination decisions that are made on purpose.

Result: The result of the application of this pattern is a restructured business processes (or sub-process) that is able to explicitly terminate based on various decision criteria. Therefore, the termination of a sub-process is not done at the end of the complete process but right at the time it is not needed any more. Based on this behavior the allocated resources may be released earlier or fewer resources are needed at all.

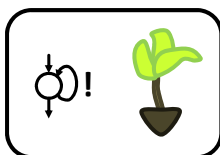
Example: In manufacturing or trading, the quality management is an important aspect. However, the quality is often assessed using control samples only. Thus, the sub-process of quality assessment is sometimes executed completely, while sometimes there are only a few minor checks that are performed. In the latter case, there is no need to wait for all activities to be executed. The sub-process can be canceled and marked as successfully completed as soon as the minor checks have been performed.

Relations to other Patterns:

Green Cancel Activity / Case: An explicit termination may also be combined with the Green Cancel Activity or Green Cancel Case patterns to improve termination conditions.

Green Lazy Load: It might be suitable to not load and provide all data at the time of process initialization but reload required data later on.

Green Multiple Instances With a Priori Runtime Knowledge



Handle the number of activities in a process model while the exact number of instances is not known during design time.

Context: In some business scenarios, an activity of a business process needs to be executed multiple times to complete a certain objective. During design time, however, the concrete number of activity executions is not known as it may be dependent on runtime information that has been produced by previous activities. The number of instances of a given activity for a given use case may also vary and depend on characteristics of the use case or the availability of resources. Moreover, the number of activity instances may also change during process execution based on certain criteria. Thus, a modeler is not able to model all

activities explicitly in advance and, therefore, does not know the necessary amount of resources.

When using this pattern the following perspectives are affected:

Process Perspective: The introduction of multiple instances influences the complete design of a process model as it must support the dynamic and varying number of activity instances.

Operation Perspective: Based on a changed control flow the way external operations are invoked may be changed, too. Moreover, this perspective may also provide criteria that determine the number of activity instances that are necessary.

Resource Perspective: Due to the dynamic amount of resources, the total amount may be changed.

Solution: Depending on the Workflow engine in use, choose suitable constructs that can define the number of instances before instantiating the corresponding activities. Due to the various implementations and workflow languages a concrete guideline for designing those constructs is very hard. An overview of possible implementations is presented in Aalst et al. (Aalst et al. 2003). Usually, a modeler would choose a design that instantiates only that number of activities that are commonly used in most cases, i.e. he would not instantiate the maximum number of possibly required activities. Moreover, in some cases it might be beneficial to introduce explicit cancel-conditions that allow to further reduce the total resource allocation. For example, in some cases the process may not need the result of all activity instances but only a certain number. As soon as this number is reached there is no need for further instances.

Result: Due to a smart design of business processes the number of activity instances may be reduced to the number of activities that are suitable in most common cases. Therefore, fewer resources need to be allocated in advance which allows using the difference for other activities or applications.

Example: An example for applying this pattern is derived from Aalst et al. (Aalst et al. 2003): In the review process for a paper submitted to a journal, the *review paper* task is executed several times depending on the content of the paper, the availability of referees and the credentials of the authors. The review process can only continue when all reviews have been returned.

Relations to other Patterns:

Green Control Flow: The design of multiple instances of an activity within a process has also effect on the control flow of the process. This pattern supports the restructuring of the control flow.

Green Explicit Termination: The explicit termination pattern may support the design of cancel-conditions.

Green Cancel Activity: This pattern may support the design of cancel-conditions and activities.

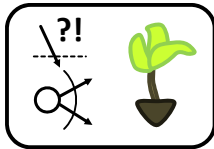
Green Lazy Load: Lazy Load supports the loading of data right at the time it will be needed. If there are activities that might not be

executed, the corresponding data must not be provided initially, for example.

Resource Change

Depending on the number of activity instances different resources may be put in place which may lead to better resource efficiency.

Green External Choice



Provide several alternative control flow branches that are selected depending on the environment.

Context: Business processes are usually not only sequences of activities. They contain different decision points, branches, and paths. To realize such constructs, workflow languages provide different possibilities like (1) AND constructs are able realize parallel splits where all outgoing edges are run in parallel, or (2) XOR constructs that are able to realize explicit decision points where a single outgoing edge is chosen based on the incoming edges, for example. Beside those explicit constructs that are modeled within a process model, the selection of a particular path or branch can also be transferred to the environment in which a process is running, i.e. external events influence the control flow of that process.

When using this pattern the following perspectives are affected:

Process Perspective: In order to react to external events the process model and the control flow need to support those events as well as the corresponding alternatives in the process model.

Data Perspective: In order to use external events it must be defined which information is required to come to a decision.

Resource Perspective: Based on the decision made different resources may be used for further processing.

Solution: The process model needs to be extended by introducing external choices at those positions where external information should be used to distinguish different process behavior. The required information must be defined and provided to the runtime environment of the process. For example, the total power consumption of a process may be retrieved and used for the decision about the resource to be used in the next process step. Depending on specific use cases the control flow of the process may be explicitly designed to address various business objectives.

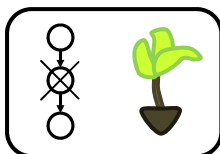
Result: Due to the integration of external information a process designer is able to explicitly consider information of the runtime as well as ecological environment. Like when using the green variant pattern, the execution path as well as the resources that are used to perform an activity can be defined during runtime. Moreover, the decision making based on environmental information is able to optimize the resource usage in general.

Example: One opportunity to choose between different paths of a process was described as Green Variant pattern in (Nowak et al. 2011)]. The example here was describing the choice for DHL customers to either ship their packages regularly or carbon-free. Using this pattern we can modify that example. As from now, the customer does not make the decision, but the process automatically checks information about the total carbon footprint at that time. Based on that information the process may decide which shipping option would be best to not violate defined business objectives for carbon emissions.

Relations to other Patterns:

- Green Control Flow:* The design of external choices usually influences the design of the control flow of a process, i.e. the introduction of a new alternative path or the use of different resources that might be handled in a new way.
- Green Compensation:* Instead of creating new process path the external information may also be able to trigger certain compensation activities or processes.
- Green Cancel Activity:* When deciding for a specific path in the process model all activities of the other paths can be canceled.
- Green Variant:* The green variant of a process usually describes a modified process model or the process model contains a “green path”. The external information supports the selection of which of the different path should be used in which case.
- Green Lazy Load:* Data may only be provided for the most common paths of the process model.

Green Cancel Activity



Withdraw activity instances that are no longer relevant for the execution of the process instance.

Context: In some scenarios, especially when containing several branches, not all activities of a process model are relevant for the actual execution of a business process. Depending on specific decision, only a subset of activities is actually executed. However, when a new process instance is created from a process model, all activities of the process model are instantiated as well. The instantiation of activities that are not needed in a specific scenario, however, allocate resources anyhow. At that point in time where it is clear that some activities are not needed at all or not any more they can be canceled and resources may be released. Note, it is also possible to cancel running activities when it turns out that their result is not needed any more.

When using this pattern the following perspectives are affected:

- Process Perspective:* The control flow of the process model might be changed due to shadow or cancel activities that are put in place.
- Operation* The execution of an activity can directly be influenced by

Perspective: the use of cancel activities. This may also have impact on other activities or the way other activities are executed.

Resource Perspective: The execution of activities can be canceled in case the result is no longer required for the process instance. The allocated resources may be released and used for other tasks.

Solution: In a first step those sections of a business process need to be identified that (i) will eventually never be executed, or (ii) that will be executed due to several varying conditions. With respect to (ii) it might be possible that the expected results may not be required in each and every use case as information may be retrieved from multiple sources. The identified sections need to be restructured in a way that either explicit cancel activities are inserted or implicit functions provided by the workflow engine are used. However, the designer has to ensure that no information will be lost nor that activities are canceled by mistake and the further, i.e. the correct, control flow cannot be assured.

Result: Activities that are affected by cancelation criteria may be disabled as soon as it is assured that they are not needed any more. Even running activities may be canceled in case the result of this activity is no longer important for further processing. Latter one might occur when two independent information sources are requested but only one answer is required. The cancelation of those activities releases the allocated resources and makes them available for other tasks.

Example: Consider an online travel agency that provides a hotel booking service. Whenever a customer requests available hotels different hotel broker services are invoked. To avoid long waiting times for customers the result is supposed to show up in a maximum of 15 seconds. If a broker does not respond within this period of time, the activity and the invocation of the broker service will be canceled. The expected results of this service will not be considered in the result set. A similar case exists if a customer cancels the request for information explicitly. In this case the corresponding activity will be disabled and the allocated resources are released.

Relations to other Patterns:

Green Control Flow: The insertion of cancel activities usually influences the design of the control flow of a process.

Outsourcing: In order to use a dynamic infrastructure that is able to flexibly claim and release resources external services like Cloud environments might be a good choice.

Green External Choice: The cancelation of an activity may also be based on external information.

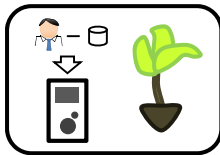
Green Multiple Instances Without Synchronization: In case there are only some of the requested results of importance the results may be merged using the Green Multiple Instances Without Synchronization Pattern.

<i>Green Cancel Case:</i>	This pattern extends the scope of the Green Cancel Activity pattern and covers multiple activities that are grouped together.
<i>Green Explicit Termination</i>	Terminates Sub-Processes when there is nothing else to do for them.
<i>Green Lazy Load</i>	Information that is used by activities that might not be executed are loaded not before the activity really needs it.

3.3 Application Architecture Patterns

Application Architecture patterns basically describe solutions on how to design applications in order to achieve certain business objectives. So far, these patterns did not consider the environmental impact as a design criterion. Besides the overview in Section 4.1 we have analyzed the following patterns in detail: Client Session State, Server Session State, Data Transfer Object, Lazy Load, and Gateway. For each of these patterns we present a corresponding variant that focuses on the improvement of the environmental impact. For further details on the original patterns see (Fowler 2003).

Green Client Session State



Optimize the number of server nodes by making them stateless.

Context: Operators of datacenters typically want to decrease the total number of resources as the operation of additional resources leads to additional costs. Thus, they try to implement a highly scalable infrastructure where resources can be started and stopped dynamically depending on the current workload. To use such kinds of clusters the application architecture running on that resources must support corresponding mechanisms to decouple customer requests and specific resources, i.e. avoid specific one-to-one relationships between a request and a specific resource.

When using this pattern the following perspectives are affected:

<i>Data Perspective:</i>	The data perspective describes how data is exchanged between the client (which might be a business process) and a resource (which might be a server) as well as which kind of information is provided by which party.
<i>Operation Perspective:</i>	This perspective describes which kind of activities or additional services are executed by the business process. This may change depending on the chosen application architecture.
<i>Resource Perspective:</i>	In this pattern the resource is set to be stateless, i.e. client information is not stored at server side. If a resource fails another resource takes the work on.

Solution: The basic idea of this pattern is to create a pool of resources that can be stopped or started dynamically and that is able to perform specific tasks interchangeably. In order to use such a pool of resources the application using those resources must support a so called stateless resource design. The client side of the application must be designed in a way that all information that is necessary to process the request, i.e. the session state, is available with the request. To implement such a solution methods like URL parameters, hidden fields, or cookies may be used. Further information is available, for example, at (Oracle 2013).

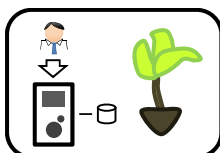
Result: If the session state is provided completely by the client the resources do not need any specific information on how to handle a request. Every resource that provides the functionality can be used to process a request. Such stateless servers can be scaled much more effective than stateful servers. Resources can be switched on or off depending on the concrete workload. This may reduce the total number of resources or makes those resources available for other tasks. However, we have to keep in mind that transferring the session state all the time from the client to the server will increase the total data transferred. The more data needs to be transferred the more energy will be consumed for the transfer and, in some cases, the performance may decrease.

Example: A typical example for session data stored in the client tier is online shopping. Customer identification information as well as items that are put into a shopping cart are usually stored in cookies at the customer's device. Thus, browsing of items in a catalog and resulting customer orders can be handled by multiple computing nodes.

Relations to other Patterns:

<i>Green Data Transfer Object:</i>	The Green Data Transfer Object may help to design a suitable data transfer object which can be used to provide the session state to the server.
<i>Resource Change:</i>	When using stateless server resources the type of resource may be changed due to the new requirements.
<i>In-&Outsourcing:</i>	The client session state enables a loose coupling between clients and servers. This architecture can be used to provide the resources at different locations and to achieve economies of scale in operating those resources.
<i>Green Server Session State:</i>	If the amount of data that needs to be transferred to the server exceeds a certain threshold it might be suitable to store at least some information on the server side.

Green Server Session State



Hold session information on server side to reduce energy consumption in communication and transformation.

Context: The energy consumption of server resources is one major aspect in reducing the environmental impact of an organization. However, the client infrastructure as well as the communication infrastructure is also an important aspect. The main challenge here is to reduce the communication between a client and the application without losing too much flexibility. Another issue is the use of appropriate client devices, especially when using, for example, Cloud applications. In those scenarios client devices are supposed to consume very low energy, provide quick boot time, and are able to continue working from the point of leaving. All necessary and resource intensive processing is done on server side.

When using this pattern the following perspectives are affected:

<i>Data Perspective:</i>	The data perspective describes how data is exchanged between the client (which might be a business process) and a resource (which might be a server) as well as which kind of information is provided by which party.
<i>Operation Perspective:</i>	This perspective describes which kind of activities or additional services are executed by the business process. This may change depending on the chosen application architecture.
<i>Resource Perspective:</i>	All relevant session information is stored on server resources. The application infrastructure must support this architecture style by providing appropriate resources.
<i>Organization Perspective:</i>	The use of this pattern affects the communication as well as the data objects that are transferred between different roles and organizational units.

Solution: A common approach to implement server session states is to provide the session state as well as all relevant client data on the server side, e.g. in memory or in a database. All data objects can be accessed from the server at any time. For concrete use cases it must be evaluated whether the state has to be persisted or not.

Result: If the complete session state is stored on a server, clients have to provide less information. This has impact to the total power consumption of computing resources in various ways: (1) the communication volume between client and server can be decreased. (2) The number of resources on the client side may be decreased if the server supports the persisting of the client state. In this scenario, the context may be loaded from a server and be provided to the client. This may also be interesting when using thin clients. There is no need for them to remain on all time instead they can request their current state from the server at any time. Thus, those devices may even be turned off completely. This enables to switch resources on and off depending on the current workload. (3) The persisted information is usually stored in a generic data format. Depending on the concrete use case the data can be serialized into different other formats, without using complex transformation activities.

Example: In a hospital a doctor or a nurse typically insert and retrieve patient data at very low frequency. Between those interactions the idle time for those

devices may be very long. Moreover, the information must be available at different devices that are, for example, in different sections of the hospital. When storing the session information at server side, the information is accessible from all devices. If there is some idle time, thin clients on the user side may even be turned off.

Relations to other Patterns:

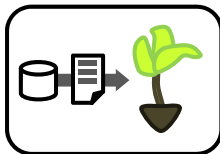
Green Client Session State: Depending on the use concrete scenario it might also be suitable to store partial or complete client data on the client side.

Resource Change: In some scenarios the used resources may be replaced by resources that better support the storage and processing of client information. The Resource Change pattern may help to choose the appropriate resources that positively influence the environmental impact.

Green Query Object, Green Data Access Component: These patterns may support the efficient retrieving of data it persisted on an external data store.

Green Stateful Component: This pattern may be used to introduce a component that is able to manage the data that is necessary to perform requests.

Green Data Transfer Object



Optimize system calls to reduce load and energy consumption.

Context: Each invoke of an application generates a certain load on a system and, therefore, increases the total power consumption. A communication channel must be established and data needs to be (de-) serialized. However, not only the invocation of an application or system determines the power consumption but also the amount of relevant data that needs to be provisioned. Consequently, a common challenge is to make each invocation more efficient or at least reduce the total number of invocations, respectively.

When using this pattern the following perspectives are affected:

Data Perspective: The data perspective will be affected as it describes how a data object looks like and how the communication between different systems is designed.

Operation Perspective: The operation perspective describes how external operations are invoked. When changing the data objects or even complete data sources the operation perspective needs to be changed accordingly.

Solution: To reduce the communication overhead, as well as the number of invocations, different information may be bundled into one single request.

Information that is part of such a request may be used by single activities of a business process or even by the complete process. For concrete implementations, the requests to a system are encapsulated into a so called data transfer object that is designed to appropriately cover all the information demand of an application or process. The data transfer object should be designed in a way that it contains all information that is most likely be used by the requestor. However, it may also contain information that is not needed or information that is missing and must be reloaded.

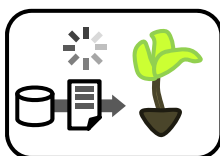
Result: The bundling of information reduces the total number of service calls, although more information may be transferred per request. However, the reduction of the number of requests makes the communication more efficient. Only one communication Channel needs to be established and the payload needs to be serialized only once. The bundling also allows servers or applications to remain in stand-by for a longer time because the requests are less frequent. This again reduces the total amount of power consumption.

Example: PostgreSQL (PostgreSQL 2013) from version 9.2 implements methods that reduce the number of calls to the database. This consequently reduces the number of wakeups and therefore reduces the amount of time where no productive work is performed.

Relations to other Patterns:

<i>Green Client</i>	The state of an application may also be described as Green
<i>Session State:</i>	Data Transfer Object.
<i>Green Gateway:</i>	It might be suitable to integrate a green gateway that allows to further bundle or cache requests in order to reduce communication and management overhead.
<i>Green Variant / Green Control Flow:</i>	Based on changes in the provisioning of information adaptations of the control flow may be necessary. This modification may also be represented as Green Variant. Moreover, different information may be requested on client side.
<i>Green Query Object:</i>	The Green Data Transfer Object may also contain optimized Green Query Objects that further improve the provisioning of requested information.
<i>Green Lazy Load</i>	This pattern might be suitable if not all information may be relevant at initialization time. Data that is rarely used will be reloaded on demand.

Green Lazy Load



Avoid upfront loading of possibly unnecessary information, instead load data only when really needed.

Context: During its execution a business process usually consumes different data from various sources. Especially in complex business processes not all data is

needed upfront. Depending on user inputs or different paths in the process model certain information is only needed in a specific number of process instances. Some information, for example, may only be used in one of hundred process instances. The challenge is to identify and provide only that information that is most likely used by all instances of the business process.

When using this pattern the following perspectives are affected:

Operation Perspective: The operation perspective describes which kind of activities or additional services are executed within the business process. This may be influenced depending on the information which has to be provided at a specific time in process execution.

Data Perspective: Data objects used at process instantiation time need to be designed to cover not all but the most likely used information. Other objects need to cover the remaining data.

Control Flow Perspective: The control flow may be modified in order to cover the reload of data that is not part of the initial data object.

Solution: In a first step the process needs to be analyzed in order to identify the information that is most likely used by all process instances. Subsequently, the requests for gathering the data need to be adapted. Initially, only data that is relevant for all process instances will be loaded and provided to the process. All other data elements will be tagged with a marker that indicates that the data is not available yet but might be reloaded on demand. Note that the complete structure of all information elements will be visible to the business process but the real data is only referenced.

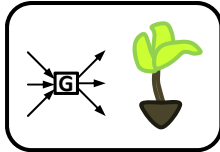
Result: The resulting processes and applications are designed in a way that communication effort is reduced. This affects both communication overhead and the processing of data that is used by the business process. Data elements that are tagged with a marker are not loaded initially and reference the corresponding endpoint where this information can be retrieved. The reduced communication consequently reduces query and transport efforts and therefore is able to reduce the total power consumption of a process and its corresponding applications.

Example: A common example is the customer web interface of an online sales company. Users are able to view their completed orders as well as checking the state of current and open orders. Usually, customers do not view all of their previous orders but only the latest ones. Therefore, there is typically no need to load all relevant order information from all previous orders but only the latest or open ones. However, the user will be able to browse a list of all of his orders. Whenever he wants to view an order the corresponding data will be reloaded. This behavior reduces the amount of required resources or makes them available for other tasks.

Relations to other Patterns:

Green Data Transfer Object: The Data Transfer Object may be used to define the data that will be provided at different points in time. It can be used, for example, to bundle data requests from different process instances.

Green Query Object: Using Green Query Objects may enhance the retrieving of data at any point in time.

Green Gateway

Manage the access to external systems or resources in order to reduce total number of access instances.

Context: In modern service-oriented architectures business process and applications are typically composed out of a set of multiple services. In order to invoke the functionality that is provided by such services, different interfaces need to be implemented. Data needs to be prepared in order to align with those interfaces, i.e. each time an external service will be invoked a collection of data as well as their serialization is required.

When using this pattern the following perspectives are affected:

Operation Perspective: The operation perspective describes which kind of activities or additional services are executed within the business process. This may be influenced depending on the information which has to be provided at a specific time in process execution.

Resource Perspective: The managed execution of activity allows preventing servers from changing their state too often due to the controlled access instances. This characteristic may also influence the type of resource that is used to handle corresponding requests.

Solution: To manage the access to external process, applications, and services a Green Gateway may be introduced. This component manages all requests to specified resources, i.e. it encapsulates the original requests and invokes the target resources based on different rules like time schedules, number of total requests, etc. Basically, the gateway may cover two different aspects: (1) it is able to encapsulate multiple or difficult to understand APIs of resources, and (2) the requests may be managed based on business, technical, and environmental objectives. A gateway may also introduce caching functionality that makes multiple reloads of information needless.

Result: As processes and applications must be changed in a way that all requests for particular resources are sent to the gateway that acts as an intermediary. Depending on the concrete scenario this gateway is able to manage all incoming requests based on the demands of the process or the application. The bundling and execution in a batch-like fashion is only one example. This behavior leads to

a decreased number of resource invocations and may therefore let the resources remain in stand-by for longer time. As a side effect, the provided interfaces may be much easier to understand and use.

Example: Consider a sales company that has a contract with a global logistics company that ships their products. They pick up parcels three times a day. In order to automate prepay functionalities the parcel information will be transferred to the logistics company before the parcels are picked up. Using this pattern an intermediary gateway may be able to collect all parcel information and transfer this information at once right before picking them up. This would reduce the number of communications as well as the number of resources that must be provided for sending the parcel information.

Relations to other Patterns:

<i>Green Data Transfer Object:</i>	The Data Transfer Object may be used to define the data that will be provided at different points in time. It can be used, for example, to bundle data requests from different process instances.
<i>Green Control Flow:</i>	If activities of a business process invoke Green Gateways they may interact with their underlying resources in a different way as before.
<i>Outsourcing:</i>	Based on the decoupling introduced by the Green Gateway resources may be outsourced in order to achieve economies of scale and therefore reduce the total number of resources.
<i>Green Batch Processing Component:</i>	A Green Batch Processing Component may be part of a Green Gateway.
<i>Green Loose Coupling:</i>	The Green Loose Coupling pattern supports the introduction of a broker component that is able to decouple different application components.

3.4 Cloud Computing Patterns

Cloud Computing Patterns basically describe solutions on how to design Cloud applications and components in order to achieve certain business objectives. The Cloud Computing patterns presented in (Fehling et al. 2013) consider all deployment models, service models, and the corresponding management. So far, these patterns did not consider the environmental impact as a design criterion. Besides the overview in Section 4.1 we have analyzed the following patterns in detail: Public Cloud, Loose Coupling, Batch Processing Component, Eventual Consistency, and Shared Component. For each of these patterns we present a corresponding variant that focuses on the improvement of the environmental impact. Please note that some of the patterns are related to the Application Architecture Patterns, however, focusing on the properties of Cloud Computing. For further details on the original patterns see (Fehling et al. 2013).

Green Public Cloud



Use external cloud resources to improve resource efficiency and reduce in-house power consumption.

Context: Organizations that run their own datacenters always have to deal with the provisioning and operation of a suitable amount of resources. To avoid bottlenecks and, therefore, loose customers they need to provide as much resources as necessary to handle eventually occurring peak loads. However, the resources are often not utilized on a very high level but need to be held available. The challenge of providing and operating the best number of resources has been taken up by specialized Cloud providers. Those providers offer pools of resources that are operated in a standardized manner, achieve economies of scale, and are offered to customers based on their demand. Customers may also use public Cloud offerings in a hybrid fashion. Hybrid Clouds combine the usage of a private and a public Cloud, and community Clouds describe a Cloud environment that is shared between different trusted parties.

When using this pattern the following perspectives are affected:

Data Perspective: When introducing external services the data objects that are transferred between the different sites may be redesigned or transformation components need to be introduced. Moreover, the secure communication of data needs to be ensured.

Organization Perspective: The usage of a Public Cloud introduces a new partner, i.e. organizational unit, to business processes and their applications. Especially cross-boundary communication must be defined and implemented properly.

Resource Perspective: This perspective is affected as new resources are used for performing the tasks of a business process.

Operation Perspective: The operation perspective describes which kind of activities or additional services are executed within the business process. The way of usage for new, external resources may therefore change.

Solution: The first step towards the use public Cloud offerings is the identification of application components and hardware resources that may be suitable to run in a Cloud environment. This analysis step needs to cover both functional as well as non-functional aspects. Components that are scalable, i.e. components that can be distributed to multiple various nodes, are typically good choices for running in Cloud environments. If they do not need a specific context, resources that are hosting those components can be turned on and off dynamically based on their current workload. Moreover, for migrating the identified components suitable Cloud service models need to be identified. For application components that, for example, use a common application server stack may use either Infrastructure as a Service or Platform as a Service offerings.

Result: Instead of hosting all required resource within a private datacenter application components are hosted at external Cloud providers. The resources that are hosting those components are accessed via network, provided in a self-service manner, and are accounted in a pay per use fashion. The advantage of such a solution is twofold: (i) a Cloud provider operates a pool of resources that are used by different customers simultaneously. This leads to better utilization and allows realizing economies of scale. (ii) Customers can use resources based on their current demand. Using common standards enables the provisioning of new resources within minutes. Moreover, the complete management, like installing updates, is in responsibility of the provider. Consequently, not only the amount of resources may be reduced but also capital and operational expenditures may be reduced.

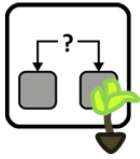
Example: There are already many different Cloud offerings available to customers. One example is the usage of comprehensive Customer Relationship Management (CRM) solutions that are provided by companies like Salesforce (Salesforce 2013). Another example is the use of software development and test environments in public Cloud environments. Infrastructure or platform services enable the provisioning of complete development environments within minutes instead of weeks or months. Moreover, those resources can be turned on and off dynamically depending on the actual requirements. Thus, resources may be allocated only for the time of specific projects.

Relations to other Patterns:

<i>Resource Change:</i>	The resource change pattern provides more abstract information on how to exchange resources that are used by a business process.
<i>Green Private Cloud:</i>	The Private Cloud pattern describes the use of Cloud technologies within the own datacenter.
<i>Green Explicit Termination:</i>	This pattern may be used to explicitly terminate resources that are not needed any more. Especially in Cloud environments, allocated resources may be released.
<i>Green Client Session State:</i>	Using this pattern supports the decoupling of customer requests and applications that perform the requests. This enables better scalability functionalities.
<i>Green Stateless Component:</i>	Using this pattern supports the decoupling of customer requests and applications that perform the requests. This enables better scalability functionalities.
<i>Green Elastic Infrastructure:</i>	The Green Elastic Infrastructure pattern may be used to design Public Cloud infrastructures that are able to scale elastically.
<i>Green Eventual Consistency:</i>	Designing and using an eventual consistent environment further improves the communication and integration efforts when using scalable and elastic resources.
<i>Green Load Balancer:</i>	This pattern may be used to manage the current workload of a resource and consequently switch resources on or off.
<i>Outsourcing:</i>	This pattern describes the outsourcing of resources in

general.

Green Loose Coupling



Reduce the dependencies between individual components to use varying resources for their runtime environment and execution.

Context: In order to use the full potential of Cloud environments the loose coupling of application components is tried to be maximized. Such architectures enable the exchange of the used resources and providers as well as the exchange of application components. However, an upcoming challenge in such scenarios is the integration of all of these components. Guidelines and patterns like the Green Loose Coupling help stakeholders to design the components in such a way that they can be used in those scenarios.

When using this pattern the following perspectives are affected:

<i>Data Perspective:</i>	The use of loosely coupled components needs well defined data objects that are used for communication between the different components.
<i>Resource Perspective:</i>	This perspective is affected as new resources may be used for performing the tasks of a business process.
<i>Operation Perspective:</i>	The operation perspective describes which kind of activities or additional services are executed within the business process. The way of usage for new, external resources may change.

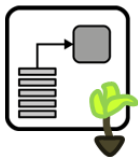
Solution: One solution to tackle this challenge is to introduce a broker component that encapsulates the communication layer between the different components. This component can be invoked from different platforms and is aware about location, protocol and the corresponding format of the data. Using such an architecture design enables a separation of concerns and makes components standalone units that are able to interact with other components.

Result: The use of a broker enables and provides a high degree of decoupling of different components of an application. This allows to dynamically selecting the resources and their location. The flexible selection of resources also enables the temporary use of certain resources based on the current workload, e.g. provided by Cloud environments. The dynamic selection in combination with scalable components further decreases the total amount of resources. On the other hand, a new resource needs to be provided in order to host the broker component. However, the environmental impact of this component is usually compensated by the target-oriented use of all other components that communicate via the broker.

Example: A common example of loose coupling is a service oriented architecture (SOA). Web Services, for example, are typically designed as self-contained components. If multiple Web services are used to achieve a business objective they usually communicate with a broker that may be implemented as a business process. Web Services may even be stateless and therefore very flexible for use in loosely coupled environments.

Relations to other Patterns:

<i>Resource Change & Outsourcing:</i>	The resource change and outsourcing patterns provide more abstract information on how to exchange resources that are used by a business process.
<i>Green Public / Private Cloud:</i>	The Green Private Cloud and Green Public Cloud pattern describes the general use of Cloud technologies within own and external datacenters.
<i>Green Batch Processing:</i>	The asynchronous access to loosely coupled components can be further managed in order to increase power savings.
<i>Green Client Session State & Green Server Session State:</i>	Those patterns specify where context information is stored. The chosen location, i.e. on client or server side, influences the way of communication between the different components.
<i>Green Stateless Component, Green Elasticity Manager, Green Elastic Load Balancer, Green Elastic Queue, Green Gateway:</i>	These patterns may be used in order to specify and improve the communication between several loosely coupled components.
<i>Green Data Access, Green Query Object, Green Data Transfer Object & Green Lazy Load:</i>	Each use case has different needs with respect to data that needs to be transferred. The patterns of this category may help to design the required data objects as well as the time of providing them to a business processes and the application components.
<i>Green Control Flow & Green External Choice</i>	Due to the use of different components the way they interact with a business process may change, too.
<i>Green Variant</i>	Loosely coupled application components may be used to provide a green variant of an existing process, without the need of changing the original infrastructure or process.

Green Batch Processing Component

Delay and bundle the processing of requests based on internal or external requirements and dependencies.

Context: To provide scalable and flexible applications that can be performed on different resources the processing functionality is usually distributed between different application components. These components are often invoked by asynchronous communication facilities, perform a certain task and deliver a corresponding result. The communication, therefore, is typically managed by a broker that encapsulates the communication requirements of each node. The challenge for reducing the energy consumption of the total system is to improve the efficiency per request, i.e. invoke each associated component in a way that minimal power will be consumed.

When using this pattern the following perspectives are affected:

Resource Perspective: Depending on the actual workload of a component various resources may be used for performing the workload. Thus, the tasks that are performed by a business process may be performed on various resources, respectively.

Operation Perspective: The operation perspective describes which kind of activities or additional services are executed within the business process. Due to the modification of the invocation of external services the process may change as well.

Solution: The invocation of application components can be improved with respect to different properties that belong to a request. One possible solution is to introduce a broker component, similar to a Green Gateway, that executes incoming requests based on a set of predefined rules, like resource availability, costs, current power consumption, and time of day. Especially when executing many small requests a bundling of requests may provide better utilization for resources as they can be dimensioned based on the exact workload. Requests from any of the components must be asynchronous and can be emitted at any time. The broker component collects them and forwards them based on the predefined rules. For example, based on the number of queued requests a proper resource for efficiently executing them may be selected.

Result: After introducing a broker component the other components will send all of their requests to the broker, i.e. they will no longer communicate directly with the target resource. As the number of requests to be processed is now known in advance, suitable resources may be selected for their execution. Depending on the concrete scenario, the allocation of resources may be triggered each time a new batch has to be executed. During times of idle those resources can be released or put into stand-by. If Cloud resources are used, for example, they can be allocated only for performing the defined set of requests.

Example: Consider a sales company that has a contract with a global logistics company that ships their products. They pick up parcels three times a day. The batch processing component is able to collect all parcel information and transfers this information at once right before picking them up. This reduces both the number of communications between the two parties as well as the utilization of resources that are responsible to transfer the information.

Relations to other Patterns:

Resource Change: The Green Batch Processing Component may allocate different resources on demand. The Resource Change pattern provides abstract information on how to exchange resources that are used by a business process.

Green Public Cloud & Green Private Cloud: The Private Cloud and Public Cloud patterns describe the use of Cloud technologies within the own and external datacenters, respectively.

Green Elastic Infrastructure: The Green Elastic Infrastructure pattern may be used to design infrastructures that are able to scale elastically.

<i>Green Elasticity Manager:</i>	The Green Elasticity Manager pattern may support elasticity functionalities.
<i>Green Elastic Queue:</i>	Elastic Queues may be used for storage of incoming requests.
<i>Green Loose Coupling:</i>	The Loose Coupling pattern supports the design of asynchronous communication between different components and may therefore be used to realize batch processing.
<i>Green Gateway:</i>	A Green Gateway may operate as a broker that further optimizes the access to resources.

Green Eventual Consistency



Distribute data among replicas while reducing the synchronization overhead to a minimum.

Context: Organizations need to persistently store all kind of relevant data to prevent damage or loss of data. Due to reliability reasons data is often replicated among different locations that are connected via some kind of network. In order to keep the data up to date different synchronization mechanisms need to be implemented. Each modification of data at one location needs to be synchronized with all other locations. The replication of data leads to significant communication overhead and increases the utilization of all storage nodes even with only one single write operation. The challenge for reducing the environmental impact is to minimize that synchronization overhead while ensuring proper quality of services to customers.

When using this pattern the following perspectives are affected:

Process Perspective: If a process instance may deal with eventual consistent data the control flow of the corresponding process model may be changed to deal with this issue. For example, further checks need to be integrated.

Data Perspective: The use of eventual consistent data may force organizations to redesign their data objects as additional information may be necessary. Processes must be aware of the fact that data might not be the latest one.

Resource Perspective: The activities of a business process that deal with eventual consistent data may use different resources for storing the data. Unlike with strict consistency, the storage nodes can be coupled much more loosely and the number of nodes can be adapted according to the current demand.

Solution: Basically, the communication between different nodes used for data replicas will be designed simple. Write operations will not be replicated immediately but based on different aspects like time of day and resource utilization. The overall synchronization overhead as well as communication and

processing efforts will be reduced because less read and write operations are performed on every node simultaneously. The replication may, for example, take place if the utilization of a node is below a certain threshold.

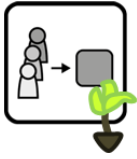
Result: The application of this pattern is able to reduce total overhead for replicating data between different locations by reducing the concurrent read and write operations. The effect of such an architecture is twofold: (1) in strict consistency scenarios applications request data from every replica node in order to ensure to process the latest data. When using eventual consistency, an application requests data from fewer nodes knowing that data might not be the latest data available. Consequently, the utilization of each of the nodes will be reduced and nodes with less capacity may be used. (2) The network communication between the different nodes will be reduced when data is not replicated immediately. Thus, (1) and (2) lead to the optimization of both the reduction of resources and their efficient usage.

Example: A commonly known example of eventual consistency is Facebook (Facebook 2013). Users are using different resource zones where information is not replicated instantly. However, this leads to the fact that information of users is not available to all Facebook users at the same time. Another example is Amazon SimpleDB (Amazon Web Services 2013). It allows customers to setup eventual consistent storage clusters. Applications that read data from that cluster request data from fewer nodes accepting that the read data might not be the latest data. Moreover, it also supports conditional writes of data.

Relations to other Patterns:

<i>Green Public Cloud & Green Private Cloud:</i>	The Private Cloud and Public Cloud patterns describe the use of Cloud technologies within the own and external datacenters, respectively. The patterns provide solutions to realize scalable resources.
<i>Green Data Access Component:</i>	The use of a Green Data Access Component may act as a multiplier as it manages the queries to databases.
<i>Green Data Transfer Object & Green Query Object:</i>	The use of those patterns may be able to enhance the communication between the data replica nodes.
<i>Green External Choice:</i>	If a process model does use eventual consistent data a Green External Choice may be used to reflect this issue within the business process. For example, if data might be eventually consistent, a specific branch within the process will be chosen.
<i>All other Green Cloud Patterns:</i>	All other Cloud Patterns might be suitable as well as they support elastic and scalable resources. This, however, depends on the concrete use case.

Green Shared Component



Share application components between multiple tenants to improve resource efficiency.

Context: New technologies like Cloud computing enable a more efficient usage of existing computing resources. Applications that cover specific business tasks are often provided not only to a single customer but a group of customers that use the application in a similar way. Such customers can be both internal and external users of an application. From a resource point of view the customers, so called tenants, share parts or a complete application stack including the underlying hardware resources. The sharing of resources achieves better resource utilization as well as economies of scale. However, the challenge is to define components that can be accessed by multiple customers and are able to handle data from those customers.

When using this pattern the following perspectives are affected:

<i>Organization Perspective:</i>	The change of an application used by a business process usually results in a change of the communication within the organizational boundaries, e.g. new departments are involved. If an application is provided by a third party a new external partner role has to be introduced.
<i>Operation Perspective:</i>	The operation perspective describes how the atomic elements of a business process are used. If there is another application used by one of the activities, a different invocation method may be necessary.
<i>Resource Perspective:</i>	When using a new, multi-tenant aware type of application to perform the activities of a business process new types of resources have to be used. Depending on the type of application as well as the target customers, those resources may be hosted internally in the own datacenter or externally by third parties.

Solution: To provide shared components organizations have to create an applications stack that minimizes the number of components that are provided to a single tenant. Shared components can be achieved on different layers of an application. Tenants can share resources on infrastructure level (especially in virtualized environments), share Middleware environments, or even share complete applications that are able to separate between the different tenants. An important aspect, however, is to ensure privacy between the different tenants of a system, i.e. data from one tenant cannot be manipulated by any other tenant.

Result: The distribution of customers to fewer physical resources enables a more efficient utilization of those resources. Unlike applications that are hosted on single physical resources and are used by only one tenant, a shared component is able to scale depending on the current workload. As a result of applying this pattern the total number of resources may be reduced and the energy efficiency of the remaining resources can be improved.

Example: Commonly known examples for shared components are webmail providers like Gmail (Google 2013). Their web client is hosted on a shared application stack that is used by multiple different customers. Other examples are Infrastructure-as-a-Service offerings. Various customers share infrastructure and network components in order to consume computing resources.

Relations to other Patterns:

Green Public Cloud, Green Private Cloud, Green Elastic Infrastructure, Green Elastic Platform, Green Stateless Component, Green Stateful Component, Green Elasticity Manager, Green Elastic Queue, Green Loose Coupling, Green Gateway: All of these patterns may support the development of a flexible and scalable infrastructure that allows hosting different, independent tenants on the different levels of an application stack.

The Green Client Session State pattern supports the loose coupling of the application as the client explicitly brings in his own state.

Due to the use of tenant identifiers on shared components the data objects for requests and responses need to be adapted accordingly. The Green Data Transfer Object helps to create good data transfer objects.

In some cases the control flow of business processes needs to be modified in order to realize authorizing technologies that are required by the tenant-aware applications.

4. Results and Discussion

Our resulting set of patterns shows that utilizing systematic analysis methods involving certain domain knowledge as well as existing patterns may lead to a bunch of new patterns that are related to new domains and objectives. Like with any other patterns, however, it is important to note that we cannot derive completely generic solutions that can be used in each and every application scenario. Depending on concrete requirements, like the degree of business change, stakeholders need to traverse the pattern language and chose the patterns that are appropriate for their scenario.

Table 2 Green Business Process patterns and corresponding business process perspectives.

	Process Perspective	Data Perspective	Organization Perspective	Resource Perspective	Operation Perspective	Integration Perspective
Green Control Flow	X		X		X	X
Green Explicit Termination	X			X		X
Green Multiple Instances With a Priori Runtime Knowledge	X			X	X	X
Green External Choice	X	X		X		X
Green Cancel Activity	X			X	X	X
Green Client Session State		X		X	X	X
Green Server Session State		X	X	X	X	X
Green Data Transfer Object		X			X	X
Green Lazy Load	X	X			X	X
Green Gateway				X	X	X
Green Public Cloud		X	X	X	X	X
Green Loose Coupling		X		X	X	X
Green Batch Processing Component				X	X	X
Green Eventual Consistency		X		X		X
Green Shared Component			X	X	X	X

To provide guidance for identifying patterns that can be applied to an organization's business processes we suggest using the introduced process perspectives as a first indicator for the selection. Depending on which perspective is able to be changed within an organization, different patterns may be selected. An overview of these relations is shown in Table 2. Here, each "X" indicates that a pattern is related to the corresponding process perspective. Moreover, to better navigate through our extended pattern language, consisting of the patterns presented in (Nowak et al. 2011) and in this work, we provide an overview of the mutual relations between the different patterns in Table 3. This eases the traversal of the pattern language, i.e. the navigation through the patterns, and helps to find a set of suitable patterns for a concrete use case. Please note that within the detailed pattern analysis and the corresponding documentation of the patterns we have also described relations to patterns from the pre-selection phase. Due to lack of space these relations are not part of Table 3. Again, each "X" indicates that a pattern is related to the corresponding other pattern.

Table 3 Relations between the proposed patterns of multiple domains.

	Green Control Flow	Green Explicit Termination	Green Multiple Instances	Green External Choice	Green Cancel Activity	Green Client Session State	Green Server Session State	Green Data Transfer Object	Green Lazy Load	Green Gateway	Green Public Cloud	Green Loose Coupling	Green Batch Processing Component	Green Eventual Consistency	Green Shared Component	Green Compensation	Green Variant	Resource Change	Green Feature	Common Process Improvement	Process Automation	Human Process Performance	Insourcing	Outsourcing
Green Control Flow																								
Green Explicit Termination																								
Green Multiple Instances	X	X																						
Green External Choice	X																							
Green Cancel Activity	X	X	X	X																				
Green Client Session State																								
Green Server Session State						X																		
Green Data Transfer Object	X					X																		
Green Lazy Load		X	X	X	X			X																
Green Gateway	X						X	X																
Green Public Cloud		X				X																		
Green Loose Coupling	X			X	X	X	X	X	X	X	X													
Green Batch Processing Component									X	X	X													
Green Eventual Consistency				X						X	X	X												
Green Shared Component	X					X	X	X	X	X			X											
Green Compensation																								
Green Variant	X			X			X					X				X								
Resource Change	X		X			X	X			X	X	X												
Green Feature								X	X	X							X							
Common Process Improvement		X	X	X	X																			
Process Automation	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
Human Process Performance	X																			X	X			
Insourcing						X	X									X								
Outsourcing					X	X			X	X	X				X	X	X	X	X	X	X	X	X	X

5. Conclusion and Future Work

In order to advance the acceptance and usage of green business process management, suitable solutions and guidelines must be available. The introduction of explicit green business process patterns in our previous work (Nowak et al. 2011) was a suitable starting point. Unfortunately, only a few of today’s optimization projects do consider such explicit solutions. However, they do apply a lot of solutions that provide implicit opportunities to improve the environmental impact of business process, applications, and infrastructures. Consequently, we developed a method that guides stakeholders through the process of identifying suitable solutions, i.e. patterns, which properly fit to their domain of interest. Moreover, we used that method to identify a set of patterns from the domains of Workflow Management, Application Architectures, and

Cloud Computing Architectures that are applicable for the improvement of the environmental impact of organizations and their business processes.

In our future work we want to use the proposed method to identify some more key characteristics that help to identify more patterns from the introduced as well as new domains. Based on the identified patterns we want to extend our pattern language for green business processes. Moreover, we want to improve the decision support for stakeholders by introducing reasonable dependencies between patterns. This allows correlating the structure, use cases, and characteristics that are common for a specific type of pattern.

REFERENCES

Nowak, A., Leymann, F., Schleicher, D., Schumm, D. and Wagner, S. 2011. Green Business Process Patterns. In Proc. of PLoP 2011. ACM.

Jablonski, S. and Bussler, C. 1996. Workflow Management: Modeling Concepts, Architecture and Implementation. Cengage Learning EMEA.

Van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B. and Barros, A.P. 2003. Workflow Patterns. Distributed and Parallel Databases, 14(3), 5--51.

Fowler, M. 2003. Patterns of Enterprise Application Architecture. Addison-Wesley.

Fehling, C., Leymann, F., Retter, R., Schupeck, W. and Arbitter P. 2013. Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications. Springer, Berlin-Heidelberg.

Hanmer, R. 2012. Pattern Mining Patterns. In Proc. of PLoP 2012. ACM.

Alexander, C., Ishikawa, S. and Silverstein, M. 1977. A Pattern Language: Towns, Buildings, Construction. Oxford University Press, USA.

Gamma, E., Helm, R., Ralph E.J. and Vlissides, J. 2000. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

Deutsche Post AG 2013. Green Logistics. http://www.dhl.com/en/logistics/freight_transportation/go_green.html.

Oracle 2013. Session state in the client tier. <http://www.oracle.com/technetwork/java/session-state-140543.html>.

PostgreSQL 2013. PostgreSQL 9.2 release news. <http://www.postgresql.org/about/news/1415/>.

Salesforce 2013. Online CRM System. <http://www.salesforce.com/>.

Facebook 2013. <http://www.facebook.com/>.

Amazon Web Services 2013. Amazon SimpleDB. <http://aws.amazon.com/simpledb/>.

Google 2013. Gmail. <http://mail.google.com>.