Object Instance Monitoring

Lisa Arnold[1]

[1]*Institute of Databases and Information Systems, Ulm University, Germany*

Abstract

The monitoring of business processes represents a pivotal component of contemporary management practices to ensure the efficiency and effectiveness of company processes. Object-centric business process monitoring is a concept that involves the continuous observation and analysis of object instances to identify deviations from defined standards and initiate prompt measures for optimisation. The utilisation of specific technologies, such as *Object Instance Monitoring*, empowers companies to acquire valuable insights that facilitate the enhancement of performance and the augmentation of customer satisfaction. The ability to identify risks at an early stage, optimise the use of resources, and increase the company's agility are key to long-term success in an increasingly dynamic business world. The objective of this paper is twofold. Firstly, it seeks to define process metrics (i.e., the global status and duration of instances). Secondly, it discusses visualisation techniques for object instance monitoring. The purpose of these two objectives is to facilitate resource and risk management.

Keywords

process monitoring, business processes, object-centric, object instance monitoring, resource management

1. Introduction

Object instance monitoring (OIM) is an essential aspect of process management. It facilitates the real-time tracking and analysis of individual object instances, enabling companies to make informed decisions. An object instance (e.g., Instance *Application1*) is defined as a specific execution of a business object (e.g., Object *Application*). This is characterised by various connected states with business attributes and decisions (i.e., lifecycle processes). The primary objectives of monitoring are to ensure process quality, identify bottlenecks and increase efficiency. The implementation of individual process monitoring, for instance, through the utilisation of dashboards, empowers organisations to respond expeditiously to deviations and proactively implement measures [1].

Traditional business processes are defined as a series of activities and order constraints. These systems offer the user a process-centric perspective. The monitoring of such traditional business processes has the potential to provide Business Activity Monitoring (BAM) [2, 3]. However, it is not possible to provide information regarding the specific execution of individual activity instances. Consequently, the execution of these activities occurs within a black box, which is inaccessible to the user [4].

The `PHILharmonicFlows` framework is a runtime engine that facilitates the execution of object-centric business processes. Furthermore, a monitoring engine that is integrated within the runtime engine is provided. The fundamental premise of this paper is to define the concept of object instance monitoring, which will be integrated within the monitoring engine. To facilitate the monitoring of object-centric business processes, each instance status (e.g., running or terminated) that may exist during runtime must be defined and delimited. In addition to the monitoring of status, the duration of each instance constitutes a fundamental element of OIM. The objective of this paper is to establish metrics to measure *On Time*, *On Risk*, and *Overdue* instances. In addition, the utilisation of visualisation techniques to illustrate OIM is discussed. In light of this, a range of visualisation techniques is hereby proposed, adapted to object-oriented processes and their respective instances.

The remainder of this paper is structured as follows. Section 2 provides a concise overview of the key principles and characteristics of object-centric business processes. In Section 3, an overview is provided of the potential statuses of instances during runtime. The concept and idea of instance

Vienna'25: 17th Central European Workshop on Services and their Composition (ZEUS), February 20–21, 2025, Vienna, Austria
lisa.arnold@uni-ulm.de (L. Arnold)
0000-0002-2358-2571 (L. Arnold)

duration monitoring is defined in Section 4. Section 5 provides a detailed exposition of the visualisation techniques that have been developed for object instance monitoring. Related work is discussed in Section 6. Section 7 concludes the paper.

2. Fundamentals of object-centric Business Process

In the object-centric process management paradigm PHILharmonicFlows, a business process is described in terms of interacting business objects that correspond to real-world entities. The interactions between the objects, as well as their relations, including their cardinalities, hierarchical structure, and semantic relations, are manifested in the **Relational Process Structure** (RPS) (cf. Fig. 1)[5]. During execution, business objects can create any number of object instances, provided that the constraints imposed by their cardinalities are respected. Furthermore, business attributes may be defined for each business object, specifying the business process. The RPS corresponding to the recruitment business process, along with several of its key business attributes, is depicted in Fig. 1.

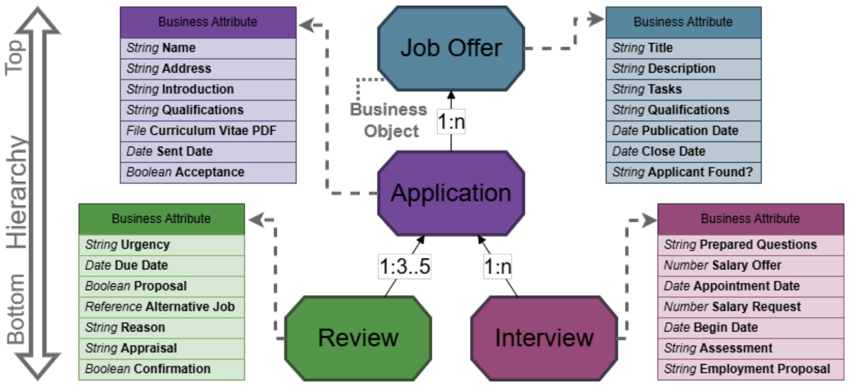


Figure 1: RPS of the recruitment business process with a number of the objects' business attributes.

The runtime behaviour of these business objects is defined in terms of **object lifecycles** (lifecycle for short) [6]. The lifecycle of the object *Job Offer* is depicted in Fig. 2. In general, a lifecycle comprises *states*, with one start state (*Preparation*) and at least one end state (*Position Filled* and *Position Vacant*), as well as any arbitrary number of intermediate states (*Published* and *Closed*). The runtime behaviour of each object instance is defined by its own lifecycle instance. The start state of an instance is automatically assigned as *activated* when the instance is created. Moreover, it is essential to note that during the execution of an instance, it is only permissible for one state of the lifecycle to be marked as *activated* at any given moment. It is consequently evident that parallel execution within a single lifecycle process instance is not a possibility. However, executing multiple instances in parallel is indeed feasible. To facilitate user interaction at runtime for each state, an automatically generated *form sheet* is created from the lifecycle structure. In particular, the states of a lifecycle define the form sheets and their steps, which in turn design the input fields. These are constructed from the object attributes. The result is a data-driven business process. Furthermore, the lifecycle may encompass backwards transitions to previous states, allowing for the reading, verification, or adjustment of previously entered data. However, it should be noted that by default, there are no backwards transitions, as a modeller must explicitly set these. Moreover, the intention behind backwards transitions is not to establish loops, wherein a new instance is generated. Instead, the previous form is displayed once more, accompanied by the input variables that the end user has previously entered.

A **coordination process** (cf. Fig. 3) controls the interactions between the lifecycles of multiple objects and defines the sequence of states between multiple lifecycle states. A coordination step is defined as a reference to the lifecycle state of an object. Moreover, a coordination process is generally represented by a graph, in which the vertices correspond to the coordination steps and the edges correspond to the coordination transitions. The coordination process graph is defined as a directed, acyclic and connected

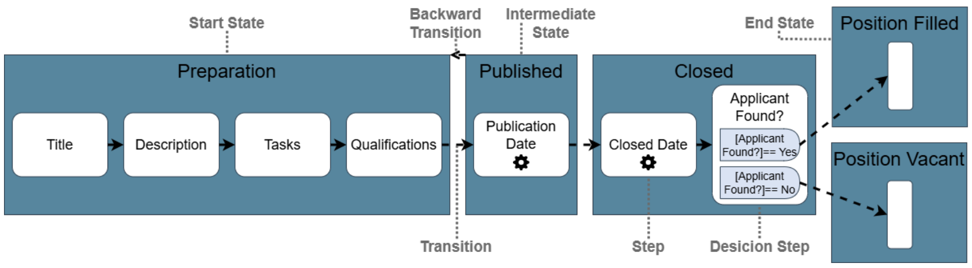


Figure 2: Lifecycle of the object *Job Offer*.

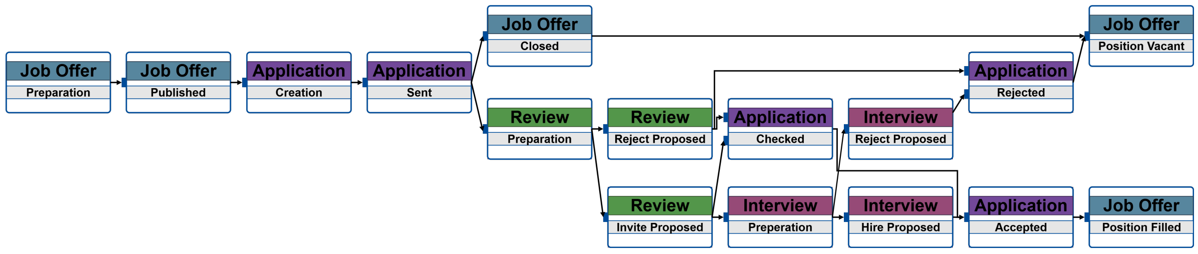


Figure 3: The coordination process of the coordinating business object *Job Offer*.

graph. This implies that it does not permit backwards transitions or loops to preceding coordination steps [7]. Conversely, the absence of such mechanisms can lead to cyclic dependencies, which, in turn, can result in deadlocks. Consequently, the occurrence of cyclic dependencies may result in deadlocks, thereby introducing an inherent risk to the process [7].

3. Status of an Instance

This section provides an overview of the various statuses that can be assigned to an instance.

Running Instance: A running instance is defined as an instance with an activated state that is not the end state. In Fig. 4, the start state *Preparation* is the active state. Data collection is done using forms. The form can be automatically generated from the process elements. Typically, in activity-centric processes, forms must be designed manually for each activity. However, since `PHILharmonicFlows` is data-centric, form generation is built in. The operational semantics control the dynamic aspects of the form, such as the next value that is required. The active state of the process (cf. Fig. 4) gives rise to the current form sheet *Preparation*, incorporating the designated input fields, which include *Title*, *Description*, *Category*, *Tasks*, and *Qualifications*. Input fields designated *Title* and *Description* are filled in with data, whereas a value for *Category* is required, indicated by the marking *Enabled* on the *Category* step (cf. Fig. 4). The input fields designated *Tasks* and *Qualifications* have been marked as *ready*. Nevertheless, the operational semantics have been designed to allow for flexibility. It is not obligatory for a user to complete the form in the correct order. However, they are at liberty to deviate from this by filling in the *Qualifications* field first and the *Tasks* field afterwards. This provides the end user with a certain degree of flexibility. Provided that a lifecycle instance is active (not marking the end state as *active*), it is considered to be running.

Terminated Instance: Terminated instances are defined as instances that have reached one of their end states (end state marked as *activated*). As demonstrated in Fig. 2 and 4, the end states comprise precisely one step. This step is devoid of any reference to a business attribute and is consequently designated as an *empty step*. It is important to note that an end state does not generate a form sheet or input fields. The primary purpose of end states is to enable the monitoring or collection of terminated instances without the need to delete or eliminate these instances. Multiple end states can characterise a lifecycle. The various potential end states of each instance (e.g., the state of a position being either vacant or filled) can be leveraged to demonstrate their different outcomes. Generally, these instances

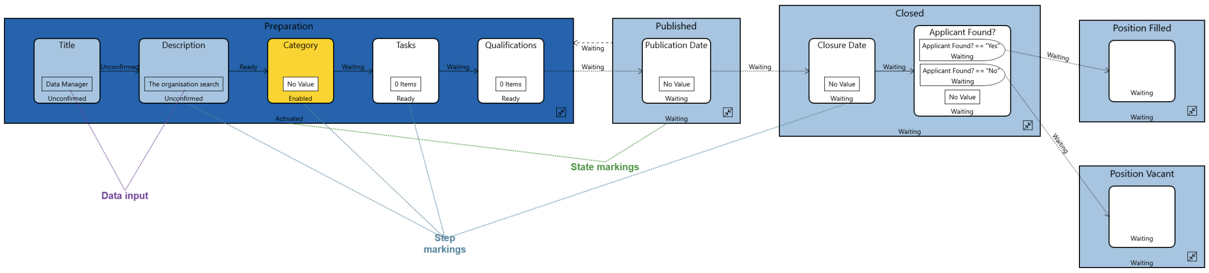
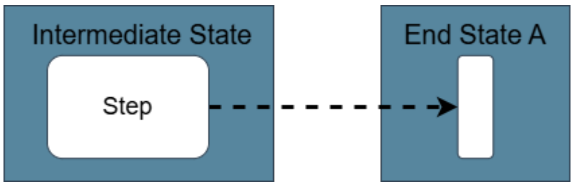
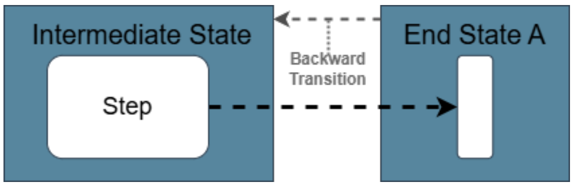


Figure 4: Screenshot from runtime engine of the Lifecycle *Job Offer* with markings.



(a) No outgoing Backwards transition from an end state.



(b) Backwards transition from an end state to a previous state.

Figure 5: Difference between completely terminated instances and temporarily terminated instances.

do not necessitate further work. End states can exhibit backwards transitions to one of their previous states. It is incumbent upon the modeller to define such transitions explicitly. In the event of such a backwards transition, a terminated instance may be reactivated. This enables the distinction between instances that have been *fully* and *non-fully* terminated.

Fully Terminated Instance: A fully terminated instance is defined as an instance that has reached its end state, and there is no backwards transition to reactivate the instance again. Fig. 5a shows an intermediate state with one step and a transition to the empty end state. An instance that can be fully terminated is the default way of modelling a lifecycle process.

Non-Fully Terminated Instance: A non-fully terminated instance is defined as an instance that has reached its end state, and there is at least one backwards transition to a previous state modelled to reactivate the instance. Fig. 5b shows the same constellation as in Fig. 5a with an additional backwards transition. This case is used when an instance that has reached an end state and is normally terminated can be considered again. Considering the recruitment process used in Section 2 as a running example, one use case is the rejection of applications because the vacancy is filled (cf. Fig. 2). However, the candidate who has been offered the job rejects it. In this case, the rejected application is reconsidered and the instance to find the most suitable candidate is reactivated. Furthermore, a lifecycle can have two types of end states: one with the possibility of reactivating an instance and another with no reactivation option. This eliminates the need for complex special-case modelling.

Expected Instance: Expected instances are defined as those which are yet to be created, but whose existence is already anticipated. The purpose of these instances is to facilitate resource management calculations and planning.

- **Minimum Cardinality:** Within the paradigm of the data model, the cardinality between two objects that are related can be defined. By default, a $1 : n$ cardinality is established, and the $n : m$ relation's placeholders n and m can be defined by the modeller as follows: minimum (e.g., $4..m$), maximum (e.g., $1..8$), or range (e.g., $3..5$). In the event of a minimum being specified, it can be used to calculate the expected instances. To illustrate this, consider the recruitment process illustrated in Fig. 1. This process stipulates that for each application, between three and five instances of reviews must be created.
- **Event Log:** Utilising an event log facilitates the calculation of the expected number of instances by leveraging the average number of instances generated by completed business process instances. However, it should be noted that this variation can be subject to inaccuracy. Nevertheless, it can

assist in approximating the anticipated expense. For example, the number of applications can be estimated during the recruitment process. Consequently, the anticipated quantity of application instances can be calculated, thereby facilitating the estimation of the number of review instances.

- **Machine Learning:** The utilisation of supervised machine learning (i.e., neural networks [8] or random forest), in conjunction with a comprehensive event log, has been demonstrated to enhance the precision of the mean calculation of the method above (i.e., event log). This approach has been shown to yield more accurate predictions, as it incorporates additional factors beyond the number of applications. These factors considered are numerous. These include the nature of the job offer, the qualifications deemed necessary, and economic as well as environmental factors.

Deleted Instances: The `PHILharmonicFlows` framework facilitates the direct deletion of running instances by end users, contingent upon the possession of the requisite permissions. For instance, an applicant has the option to withdraw their application by deleting it. An employee can not remove an application without the necessary permissions.

4. Instance Duration Monitoring

The maximum processing time (i.e., the time span in which the instance must be completed) and the proceeding time (i.e., the time taken to complete the work of an instance) can be used to monitor risk management. For each lifecycle, a modeller (or process owner) can specify the amount of time for an instance, as well as each of its states, to be completed. For example, after an application has been received, a reviewer has two weeks to complete the review. Additionally, the maximum processing duration of one lifecycle state (i.e., one form sheet) can be defined. For example, if a candidate is successful in the assessments, they will be invited for an interview (i.e., state *Preparation Interview*). In this case, a recruiter has five days to send the candidate an interview date. Each running instance can be categorised into one of the following modes:

On Risk: This status is defined for instances or states where the proceeding time is nearly over. The determination of the point at which an instance or state is deemed to be at risk is made by a key performance indicator (KPI). By default, the value *On Risk* is set to 80% of the maximum processing time. Consequently, when 80% of the defined proceeding time has elapsed, an instance or state is flagged as being at risk. If an instance or state reaches a risk status, an alert is to be dispatched to the employees responsible for editing the instance or state. Furthermore, in the event of a reported absence (e.g., illness, holiday), the alert is to be forwarded to a colleague. However, this is only possible if the employee has designated a colleague in the system.

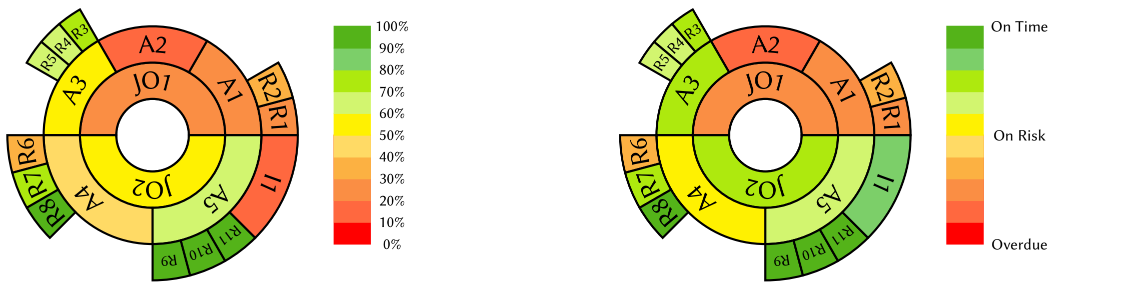
On Time: The definition applies to instances or states in which the proceeding time is on time, i.e., the proceeding time falls within the first 80% of the maximum processing time span. It is possible to adapt the KPI of the value *On Risk*. Consequently, the temporal span within which an instance or state is considered to be on time is also subject to adaptation.

Overdue: This is defined for instances or states where the defined processing time has elapsed. In such a scenario, an escalation message will be transmitted to a different employee. The identification of the employee concerned is specified within the runtime engine. Typically, the escalation message is sent to a colleague the first time, or to a manager if the deadline is missed more often.

5. Visualisation

In the context of object-centric business processes, the `PHILHarmonicFlows` monitoring framework has been developed to facilitate the monitoring of instances created within its runtime engine. The monitoring framework enables users to define bespoke monitoring components and charts, thereby empowering them to track and analyse process metrics meticulously. The primary objective of the monitoring framework is to facilitate a collaborative process that enables users to construct their own monitoring framework through a straightforward click-and-build interface. This will allow users to create bespoke monitoring dashboards in accordance with their respective permissions, specific task area and individual interests. The monitoring framework provides a range of options for data visualisation, including pie charts, doughnut charts, and bar charts. These visualisation charts can be employed to represent the instance duration (i.e., *On Time*, *On Risk*, or *Overdue*) or status (i.e., running, terminated, or expected) of a single instance. Furthermore, the framework provides options for data visualisation at the level of a collection of instances (e.g., all application instances related to the same job offer), all instances of the same object type, or the lifecycle state of an instance.

The use of rudimentary diagrams (e.g., bar charts) is optimal for monitoring individual instances (i.e., single review instances) or instances of the same type (i.e., all review instances). The utilisation of sophisticated diagram types is imperative for the comprehensive monitoring of business processes. The sunburst chart [9] is a prime example of a sophisticated diagram. It is particularly well-suited to the visualisation of hierarchically dependent data structures. The adaptation of sunburst charts for the illustration of our recruitment process example is demonstrated in Fig. 6. The sunburst chart [10] was applied to the progress indicator (cf. Fig. 6a) calculated with a one-dimensional *Kalman Filter* [11] and the risk management (cf. Fig. 6b) of all running instances.



(a) Current progress (from 0% to 100%). (b) Current status (*On Time*, *On Risk*, or *Overdue*).
Figure 6: Sunburst charts illustrate the runtime behaviour of the recruitment business process (JO := *Job Offer*, A := *Application*, R := *Review*, I := *Interview*) [10].

Heat maps [1] are another method for gaining an overall view of the business process. The coordination process is visualised through the utilisation of a heat map. Two variants of the coordination process are distinguished: the standard coordination process (cf. Fig. 3), which displays solely the coordination steps that have interactions with other objects, and the extended coordination process, in which all absent coordination steps (i.e., coordination steps without interactions between different objects) are automatically incorporated. The employment of a heat map facilitates the representation of the number of active instances for each coordination step. In this representation, the colour red is used to denote a low number of active instances. In contrast, green is used to indicate a high number of active instances in a coordination step. The heat map has the potential to optimise resource management by helping to identify personal bottlenecks more efficiently and to take prompt countermeasures.

6. Related Work

The field of Business Activity Monitoring (BAM) has already been established as a subject of research, with a considerable number of academic papers [2, 3, 12]. Furthermore, BAM has been incorporated into commercial tools, such as `Bizagi` [13]. The `BAM` tool from `Bizagi` is an analytical instrument that enables the graphical representation of information about the status of ongoing cases. BAM comprises three constituent elements. Primarily, *Process BAM* is responsible for the analysis of the present status (i.e., *On Time*, *On Risk*, or *Overdue*) of all ongoing processes. Secondly, the function of *Activity BAM* is to analyse the current state (i.e., *On Time*, *On Risk*, or *Overdue*) of ongoing activities. Finally, the function of the *Resources Monitor* is to analyse the current workload (i.e., *On Time*, *On Risk*, or *Overdue*) and performance of end users and work teams[14]. Furthermore, `Camunda` employs a heat map to facilitate the monitoring and optimisation of business processes, visualising the duration or the most frequent path of these processes [1].

7. Conclusions

The objective of this paper is to examine the application of object instance monitoring (OIM) in the context of object-centric business processes. The paper establishes and delineates a comprehensive categorisation of the statuses (e.g., *running* or *terminated*) that an instance may achieve during the execution of a business process. Furthermore, it establishes a duration concept for running instances, and specifies metrics *On Time*, *On Risk*, and *Overdue*. Finally, the utilisation of visualisation techniques (i.e., rudimentary diagrams, sunburst charts, and heat maps) adapted to object-centric business processes is explained, with a focus on illustrating OIM constituents. Further work is currently underway to develop personalised dashboards, the functionality of which will enable each user to create their own bespoke dashboard by selecting a combination of monitoring functions and visualisation types from a curated list.

Acknowledgments

This work is part of the ProcMape project, funded by the KMU Innovativ Program of the Federal Ministry of Education and Research, Germany (F.No. 01IS23045B).

Declaration on Generative AI

During the preparation of this work, the author used Grammarly in order to: Grammar and spelling check. After using this tool, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] E. Amann, C. Corea, C. Drodt, P. Delfmann, A dashboard creator suite for simultaneous predictive process monitoring., in: International Conference on Business Process Management Demo (BPM'22), 2022, pp. 152–156.

[2] J.-P. Friedenstab, C. Janiesch, M. Matzner, O. Muller, Extending bpmn for business activity monitoring, in: 45th Hawaii International Conference on System Sciences, IEEE, 2012, pp. 4158–4167.

[3] W. Schmidt, Business activity monitoring (bam), in: Business Intelligence and Performance Management: Theory, systems and industrial applications, Springer, 2013, pp. 229–242.

[4] V. Künzle, M. Reichert, Philharmonicflows: towards a framework for object-aware process management, Journal of Software Maintenance and Evolution: Research and Practice 23 (2011) 205–244.

[5] S. Steinau, K. Andrews, M. Reichert, The relational process structure, in: Int. Conf. on Advanced Information Systems Engineering (CAiSE'18), Springer, 2018, pp. 53–67.

[6] S. Steinau, K. Andrews, M. Reichert, Executing lifecycle processes in object-aware process management, in: Int. Symp. on Data-Driven Process Discovery and Analysis (SIMPDA'17), Springer, 2017, pp. 25–44.

[7] S. Steinau, K. Andrews, M. Reichert, Coordinating large distributed relational process structures, Software and Systems Modeling 20 (2021) 1403–1435.

[8] H. Liu, M. Xu, Z. Yu, V. Corvinelli, C. Zuzarte, Cardinality estimation using neural networks, in: Proceedings of the 25th Annual International Conference on Computer Science and Software Engineering, 2015, pp. 53–59.

[9] F. Le Guen, Sunburst Chart, https://www.excel-exercise.com/sunburst-chart/, 2022. [Online; accessed 14-February-2025].

[10] L. Arnold, M. Breitmayer, M. Reichert, Monitoring object-centric business processes: an empirical study, in: International Conference on Research Challenges in Information Science (RCIS'2023), Springer, 2023, pp. 327–342.

[11] L. Arnold, M. Breitmayer, M. Reichert, A one-dimensional kalman filter for real-time progress prediction in object lifecycle processes, in: 2021 IEEE 25th International Enterprise Distributed Object Computing Workshop (EDOCW'21), IEEE, 2021, pp. 176–185.

[12] H. Kim, Y.-H. Lee, H. Yim, N. W. Cho, Design and implementation of a personalized business activity monitoring system, in: Human-Computer Interaction. HCI Applications and Services: 12th International Conference on Human-Computer Interaction (HCI'07), Springer, 2007, pp. 581–590.

[13] F. M. Nafie, M. A. Eltahir, Real-time monitoring and analyzing business process performance, nternational Journal of Engineering And Science Vol. 6 (2016) 31–35.

[14] Bizagi-Germany-GmbH, BAM Documentation, https://help.bizagi.com/platform/en/index.html?bam.htm, 2023. [Online; accessed 14-February-2025].