Proceedings of the 53rd Hawaii International Conference on System Sciences | 2020

Establishing an Extendable Benchmarking Framework for E-Fulfillment

Magdalena A. K. Lang
RWTH Aachen University
magdalena.lang@rwth-aachen.de

Catherine Cleophas
Christian-Albrechts-Universität zu Kiel
cleophas@bwl.uni-kiel.de

Abstract

The growth in attended home deliveries motivates research in prescriptive analytics for e-fulfillment. Introducing new analytics solutions, for instance, for vehicle routing or revenue management, requires simulation-based benchmarking and analyses on relevant problem scenarios. Unfortunately, creating the required systems induces high overhead for analytics researchers. This paper introduces the simulation-based benchmarking framework SiLFul, which aims to support scientific rigor and practical relevance of research by reducing this overhead. It provides a toolbox of approaches, a modular and extendable architecture, and a comprehensive, application-related data model. Thereby, it facilitates controllable analyses and transparent and replicable research. Moreover, we propose a research process that leverages the framework for evaluating analytics and allows continuous development of the framework as a community effort.

1. Introduction

Attended home deliveries are both a driver and a symptom of the ongoing world-wide growth in e-commerce sales. For instance, when ordering groceries or fresh flowers online, customers have to attend the delivery. High customer expectations regarding narrow delivery time windows and punctual deliveries evoke high fulfillment cost for retailers [1]. This calls for dedicated planning through service analytics.

Prescriptive analytics aims to improve performance in terms of profit or service quality indicators by modelling complex planning problems and introducing optimal or heuristic solution approaches (see [2]). Fulfilling online orders (e-fulfillment) requires tactical and operational vehicle routing and revenue management in the face of multiple performance indicators as well as stochastic demand and logistic settings. The complex problem settings impose high standards on analytics, and recent research has responded by proposing diverse processes and algorithms (compare [3] for a typology).

In this contribution, we introduce the **Si**mulation **L**aboratory for e-**Ful**fillment (SiLFul), which allows researchers and business professionals to benchmark related approaches and analyze their behavior in alternative problem settings. Researchers need quantitative measures to evaluate new approaches. Businesses need to understand the impact on their performance in order to weigh effort with profit.

Field studies are not ideal for such analyses: Dynamic market environments make it difficult to isolate effects and real-world failures can induce high financial costs. Therefore, e-fulfillment research commonly relies on simulation studies (see, for instance, [4, 5]). Simulation modelling entails simplifying a problem description and deliberately deciding what aspects to keep and which to neglect [6]. This approach enables experiments with fully controllable parameters. As processes like online ordering of products allow automated logging of data, calibrating simulation models towards empirical validation is possible.

Designing rigorous simulation studies is challenging [7]. Implementing and testing simulation systems, validation, parameter setting, and ensuring replicability cause efforts beyond the primary objective of benchmarking and analyzing analytics solutions. These tasks require skills in conceptual modelling, systems engineering, and data management. Moreover, integrated and application-oriented research fields, such as e-fulfillment, call for embedding individual algorithms in the larger problem context for benchmarking. As a consequence, simulation studies become time-consuming or ineffective. SiLFul goes beyond implementing specific simulation models for specific solutions and experiments. Instead, it models problem context and alternative solution approaches and constitutes an extendable platform for researchers and practitioners.

URI: https://hdl.handle.net/10125/63934
978-0-9981331-3-3

By offering SiLFul as an open source platform, we complement the research knowledge base with a new benchmarking tool. SiLFul strives for scientific rigor and practical relevance of e-fulfillment analytics (as called upon in [8]). As required in design science research [9], this paper serves to communicate SiLFul and its benefits to relevant research and business communities. Moreover, a useful, up-to-date framework requires continuous development, preferably in a community effort. Therefore, we propose a prescriptive analytics research process, facilitating communication and collaboration between researchers and practitioners and resulting in both better analytics solutions and an improved benchmarking system. In the following, we first introduce challenges of e-fulfillment research before sketching the research process that motivated SiLFul's design. Afterwards, we provide an overview of SiLFul's features and architecture and shed light on different details of the system via three case studies.

2. Challenges of e-fulfillment research

E-fulfillment in general includes three major steps: order acceptance, order packing, and order delivery. As the research project that motivated the design of SiLFul focuses on integrating order acceptance and delivery, particularly in the context of attended home deliveries, we focus on these steps in the following.

When retailers offer attended deliveries, they accept orders during a dedicated order period, for instance, within seven days before the day of delivery. Usually, they offer a set of delivery options for customers to choose from, in terms of delivery time windows and delivery fees. Before the order period begins, the retailer can update demand forecasts and prepare controls on which delivery options to offer per order request. After the order period has ended, the retailer aims to efficiently deliver all accepted orders.

E-fulfillment can span tactical planning such as designing time slots [10] as well as operational planning. For instance, as accepted orders determine the delivery requirements, firms can shape the expected requirements by selectively offering delivery options depending on order request characteristics. On the one hand, firms can dynamically set delivery fees to influence choices of their customers and increase profit [3]. For example, [4] maximize the number of acceptable orders given a fixed vehicle fleet by nudging customers towards specific time slots via fees. [5] present an integrated dynamic pricing approach that considers both the effect of accepting an order on delivery routing and future acceptable value.

On the other hand, firms can control the set of time slots offered per request, as described in [11, 12]. By offering the most popular time slots only to those customers with the most valuable orders, retailers can increase the expected overall order value and avoid delivering orders of lesser value.

In the following, we elaborate on the two main reasons that motivate the introduction of a simulation laboratory for e-fulfillment research. On the one hand, current research considers individual challenges of e-fulfillment, but the effort to combine solutions to multiple requirements in an efficient, integrated fashion is still ongoing. On the other hand, current, isolated research practice calls for alternative, more effective means to evaluate approaches.

2.1. Complexity of e-fulfillment research

Several characteristics of e-fulfillment research call for complex prescriptive analytics and make simulation studies especially cumbersome. We point out exemplary advances in research but refer to reviews as in [12] for a comprehensive overview of the current state-of-the art.

Multifaceted, interdependent planning problems. E-fulfillment analytics tackles multiple, interfacing problems, which can concern tactical and operational planning levels, methods in different disciplines as vehicle routing or revenue management, and overall process sequences or individual process steps. For instance, an e-fulfillment process can be implemented either by iteratively processing multiple planning steps or by solving them in an integrated manner. An approach could first estimate the delivery capacity per time slot and delivery area via vehicle routing and subsequently control the assignment of capacity to orders [11]. Alternatively, it could predict the effect of promising a particular time slot to the next request on the set of future acceptable orders and on delivery schedules simultaneously, as proposed by [5]. Comparing iterative and integrated solution approaches requires modeling alternative processes.

Alternatively, research may focus on a particular step of the process, for instance, improving the vehicle routing algorithm used to estimate capacity before order acceptance. Nevertheless, evaluating effects of changes in individual steps requires embedding these steps in the overall planning process. A modular framework that allows a straightforward exchange of the routing component could be helpful for a vehicle routing expert without knowledge in demand management.

Online versus Offline Decision Making. By deciding which delivery options to offer per arriving order request, planners reserve judgement until the last minute. Thus, e-fulfillment planning entails potential for online decision-making. However, online

decisions require quick solutions, as customers expect instantaneous offers. Planners can prepare decisions offline before the start of the order period, predicting demand and delivery requirements (for instance, [5, 13]). This preparation speeds up decisions but reduces flexibility. When integrating online decisions in the overall process, a simulation-based analysis should evaluate the acceptability of online calculation times and the advantage gained through online decision making.

Optimization under Uncertainty. Examples of uncertainty in e-fulfillment include arrival times of customer requests, customer choice of time windows, and travel times. Optimization models that rely on strongly simplifying assumptions require a simulation system with stochastic modelling and analyses on multiple samples as well as worst-case analyses. Moreover, publishing not only model parameters but also simulated demand in a common data model can support the replicability of results.

Various Problem Scenarios. E-fulfillment settings can vary significantly, for instance, in problem size, time slot design, or the distribution of customer locations from multiple demand segments. Thus, a thorough benchmarking over alternative settings is required because generalizations are difficult. A simulation system needs to enable multiple settings as well as document them and the results transparently.

Multiple Performance Indicators. Attended deliveries involve diverse stakeholders and time horizons, justifying multiple performance indicators. For instance, the firm may aim to minimize the cost of deliveries on the short-term, but cannot neglect customer satisfaction as affected by the availability of preferred delivery options to ensure a good customer relationship on the long-term. Therefore, benchmarking may feature alternative criteria, both individually and in context of multi-objective optimization. The criteria can contribute valuable domain knowledge themselves.

In sum, these characteristics induce high complexity for evaluating new research contributions.

2.2. Current research practice

In current publications such as [5, 11, 12], e-fulfillment researchers, as well as analytics researchers in the majority of application domains, predominantly conduct computational studies on systems implemented for a specific set of experiments. Contributions describe their novel approaches and aggregated results of computational studies. When researchers evaluate a new approach, they mostly implement benchmarks from scratch. Alternatively, they might get access to program code and simulation data from other authors, requiring adjustments to fit in their own study. In particular, the exchange is hindered by heterogenous programming languages and a lack of consistent data models.

While sets of problem scenarios exist, such as the Solomon instances for routing [14], these are not domain specific and lack empirical validation. Similarly, existing benchmarking frameworks, for instance, to test the theoretical performance of meta-heuristics [15], do not directly translate to the e-fulfillment setting or are not sufficiently extendable.

E-fulfillment researchers depend on industry partners to provide empirical demand scenarios for validation, as for example in [5, 13]. However, limiting the re-use of such settings to their description in the paper and to individual interactions between authors is inefficient and error-prone. Furthermore, we cannot verify industry firms implementing methods from research, but instead observe independent research and development efforts. Papers are often too short to add comprehensive analysis on alternative settings. This precludes implications for other problem scenarios and reduces relevance for businesses, creating a gap between research and practice.

Future research could benefit from a reusable framework to avoid the currently isolated research practice and enhance practical relevance. The overhead to create such a system and to motivate relevant communities to use and continuously develop it sets a high barrier to introduce a framework for e-fulfillment. Nevertheless, we argue that the advantages are high enough to pay off our effort to instantiate it.

3. Pursuing a collaborative research methodology

We aim for an e-fulfillment research process that is supported by an extendable simulation-based benchmarking framework. The process outcome is twofold. First, we want to enable rigorous research towards new prescriptive analytics solutions and increase its practical relevance. Second, we want to establish a collaborative, continued development of the benchmarking framework to enhance and motivate future research. Here, we shortly review a nominal research process and the role SiLFul could play in it. Afterwards, we consider relevant factors that detail the expectations of such a framework and that, therefore, guided SiLFul's design process.

3.1. Prescriptive analytics research process

From a design science perspective, a prescriptive analytics contribution is an artifact that traverses several steps in a research process ([16], refer to Figure 1).

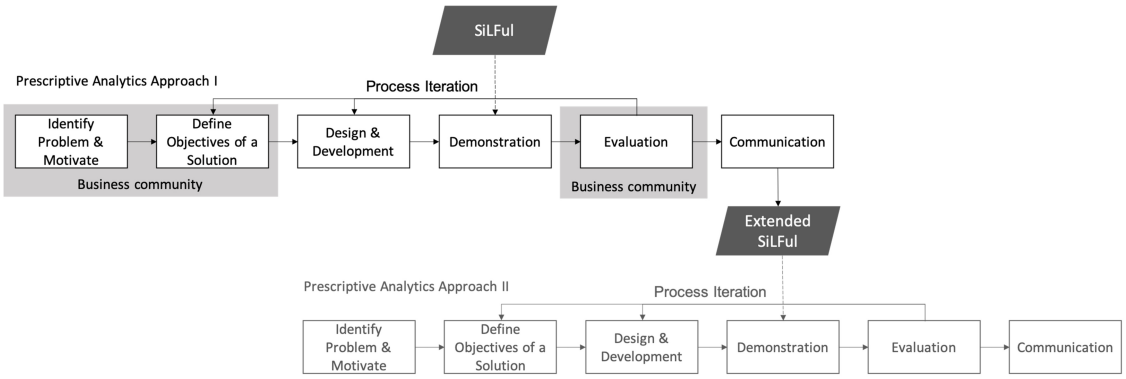

Figure 1. Prescriptive analytics research process for e-fulfillment (adapted from [16])

The first step is to identify and motivate a research problem. Afterwards, formulating the objectives is a requirement before implementing the solution approach in an artifact. For instance, an optimization approach in e-fulfillment could strive for high profitability but low runtimes. Design and development usually entail the detailed definition of a prescriptive model or solution approach. Demonstrating the effectiveness of an analytics solution requires an implementation that is applied to exemplary problem scenarios, usually in a simulation experiment. In an evaluation phase, the outcomes have to be compared to the research objectives in an extensive computational study. Finally, only the communication of the new artifact to relevant communities ensures to actually provide an impact [9]. Moreover, multiple process iterations as well as research entry points are possible.

Within such a process, SiLFul aims to provide an open-source environment for benchmarking and analyzing prescriptive analytics solutions. To that end, it constitutes an infrastructure for conducting simulation experiments in the demonstration and evaluation phases.

Publishing a version of SiLFul that is extended by a newly proposed approach (Extended SiLFul, cp. Figure 1) as part of the communication phase can ensure a constant collaborative improvement of the framework. Thus, the framework advances as benefit of prescriptive analytics research to in turn better support future research. Extended SiLFul can comprise new algorithms and standardized data dumps containing experimental settings for modeling problem scenarios or simulation output data for complete replicability.

Moreover, an easily accessible framework can lower the barrier for the relevant business community to adopt state-of-the-art approaches. Practitioners can use the system or extract relevant components to test them on their specific problem settings or benchmark own approaches. This can both validate research and raise new, practical relevant research questions.

3.2. Ensuring added value for researchers

To exploit the full potential of this process, the acceptance of the framework by the members of the relevant communities is crucial. SiLFul has to provide added value for researchers to justify the effort of learning to use and extend the system as well as the reduction in flexibility induced by a reusable framework. Here we assume that SiLFul's primary target users are analytics experts, who have expertise in programming and optimization.

The main idea of the framework is to provide researchers a head-start when conducting a simulation study, allowing to focus on designing analytics solutions rather than the required simulation environment. In particular, SiLFul reduces the overhead of data management and effective design of experiments. To that end, the framework should enable controllable, transparent, and replicable experiments.

Experiments should allow to quantify the effects of applying alternative approaches to a constant problem model, to multiple parameterizations of the same model, or to different models in order to support the different categories of value construction as listed by [17]. For instance, a framework should serve as an *analyzer*, predicting the outcome from applying an approach to a specific problem. For example, SiLFul should enable to predict the profit outcome from a heuristic that controls the set of offered time windows per order request for a specific demand constellation. Moreover, it should allow to analyze the drivers of that outcome.

Furthermore, a benchmarking framework should serve as *tester*, as researchers formulate and test hypotheses about the comparative performance of solutions. For example, researchers should be able to compare the effect of accepting order requests for

attended home deliveries on a first-come-first-serve basis or via sophisticated order acceptance controls.

To improve efficiency in benchmarking, a framework should provide an up-to-date pool of benchmarks, allowing to embed individual algorithms within the larger context of an e-fulfillment approach to predict the effect on relevant performance indicators. As a basic requirement, it should be easily extendable by new algorithms.

When SiLFul provides adequate data management, researchers can benefit from other researcher's publication of problem settings in a common data format, both to replicate research and to analyze the performance of a new approach. In this, users can especially benefit from an application-tailored simulation system with relevant entities.

Researchers can also profit from business communities having a simpler access to their approaches. Thereby, they achieve empirical validation of solutions and raise new, practically relevant research questions. Retailers should be motivated to conduct simulation studies with new solution approaches for their specific problem setting or to predict the impact of changes in business processes on the outcome of their applied solutions. Until now, they have to implement approaches according to short descriptions in papers or have to contact respective researchers, preventing to or inducing a high barrier to try the approaches. With a framework like SiLFul, they can immediately make use of the existing code base and initiate discussions with involved researchers about limitations or potential enhancements discovered in practice. When asking the right questions based on state-of-the-art research solutions, they can even nudge research to practically relevant research gaps.

3.3. Ensuring quality of development

SiLFul should be up-to-date in terms of benchmarks and support of potential new research fields of e-fulfillment to stay relevant in the long run. To this end, it requires community effort to further build and maintain its code base. This calls for code review processes and technical means to enable a collaborative and accountable development of the system. From this perspective, SiLFul is a design artifact itself, constantly undergoing development and evaluation phases.

SiLFul needs to rely on distributed programming and version control. Version control systems allow authorship tracking for accountability. Moreover, accountable development requires peer-review and testing, as highlighted with regard to open-source development in [18]. Therefore, systems can enforce multiple review stages before integrating new content into the main code base.

The idea of peer reviewing code is frequently regarded as admirable in theory but sometimes neglected in practice. Nevertheless, when e-fulfillment researchers use SiLFul's pool to benchmark their own approaches, they have a direct incentive to review the code of the benchmark. Moreover, similarly to peer-review of papers, reviewers should be assigned to validate and approve the code.

Extensions of the system environment like the data model or additional support functions are not in focus of a prescriptive analytics contribution. They require researchers that see the overall potential for the community and volunteer to contribute to the base functionality, potentially without immediate reward.

4. SiLFul: Design and technical details

We provide an overview of SiLFul's currently implemented features before introducing the system architecture and selected design details.

4.1. Overview of features

SiLFul offers features in five categories to address challenges of e-fulfillment research and provide value to researchers and other target groups in a prescriptive analytics research process.

First, SiLFul provides an *extendable pool of planning approaches*. It initially features a small pool of approaches, from simple rules to sophisticated algorithms for diverse planning steps. Additionally, it explicitly supports integrating new prescriptive analytics approaches into the framework. To this end, it features a modular design with appropriate interfaces and the required environment.

Second, SiLFul supports *flexible processes*. It allows comparing the outcome of individual prescriptive analytics solutions in isolation or in the context of processes including interdependent activities.

Third, SiLFul provides an *application-tailored environment*. The laboratory models domain specific concepts and relationships. For instance, relevant concepts are the customers and their orders, delivery time windows, vehicles, or routing schedules.

Fourth, SiLFul supports *stochastic modeling*. To represent the uncertain planning environment, SiLFul allows, for instance, modelling stochastic customer choice behavior and arrival distributions.

Lastly, SiLFul features a *holistic, domain-specific data model* that supports transparent and replicable experiments. It allows users to add problem scenarios and analyze diverse settings. This includes scenarios

derived from published research as well as scenarios based on empirical data. For instance, the model allows to integrate alternative models of time slot demand, such as preference lists or random utility models as the multinomial logit model in [5]. Moreover, SiLFul ensures transparent simulation results by saving data from intermediate steps in the database. This data can be aggregated in performance indicators via a subsequent result data analysis. Not only for transparency but also replicability of results, meta-data, such as input settings of experiments, are stored in the database and directly linked to respective outcomes of simulation runs. When the simulation includes, for instance, stochastic demand, the generated demand is persisted in the database. As a data model for prescriptive analytics, it additionally allows to save optimization models for reuse in subsequent analyses. Note that parameter settings and the resulting simulation data are stored in distinct database tables such that they can be easily published separately.

4.2. A modular system architecture

SiLFul enables benchmarking and analyzing prescriptive analytics via *experiments*. An experiment simulates a specific *process*, given specific *input settings*, and returning *simulation output data*. Output data can result from individual process steps and include, for instance, a set of orders or a routing schedule. To model the effect of stochastic models, an experiment with a constant input setting can include multiple, independent repetitions (*simulation runs*) of the same process with samples drawn from probability distributions.

To start an experiment, users have to define the input settings and a process. Input settings can include the specific set of delivery time slots, the number of vehicles available, and customer demand samples. The samples can either come from real operational data or can be artificially generated order requests. The process indicates the execution sequence of one or more activities, potentially including algorithms for prescriptive analytics. Processes can model the planning prior to an order period, order acceptance during the period, and delivery planning afterwards.

SiLFul is platform-independent and exclusively relies on open source software. In particular, it is written in the object-oriented programming language Java and makes use of the Spring framework for loading and persisting data. The current database management system (DBMS) is MySQL, which can be accessed by standard data analysis software, for instance R, for result analysis. We used automated testing by jUnit and code reviews to ensure the quality of the current system state.

Figure 2 depicts the two-tiered system architecture as a Unified Modeling Language (UML) package diagram. The data layer comprises the data services that retrieve and persist the entities from the database. Note that although the `mysql`-package contains MySQL-specific implementations to communicate with the DBMS, the `service`-package contains the interfaces used by the logic layer. This strict separation renders the DBMS exchangeable and allows developers to add, for instance, a NoSQL database. The logic layer can access the data layer to retrieve settings and input data and to persist the algorithms' outputs to the database. This renders algorithm performance transparent and replicable.

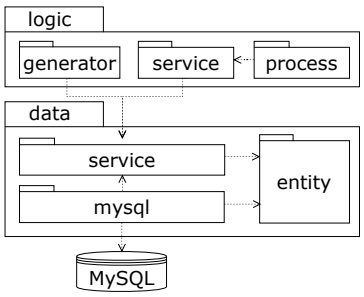

Figure 2. System architecture

The `generator`-package from the logic layer allows generating artificial data from demand models. The `service`-package contains algorithms and services to host them. These algorithm provider services prepare input and output and initiate the execution of an algorithm. Moreover, support services provide functionality useful to multiple algorithms. The processes in the `process`-package can combine and start sequences of services. The architecture does not include a presentation layer, as experiments progress in an automated fashion given complete input settings. Thus, user interaction is limited to defining initial settings via console commands, entries in the database, or configuration classes. Nevertheless, the architecture allows developers to add a presentation layer, without the need to alter any structure in the other layers.

4.3. Processes as flexible orchestration of services

To ensure flexible processes, the design considers processes as a sequence of activities, implemented through algorithms. The relevant concepts in the logic layer are processes, algorithm provider services, and algorithms. Figure 3 presents a UML sequence diagram that describes the interaction between these concepts, as

represented by interfaces.

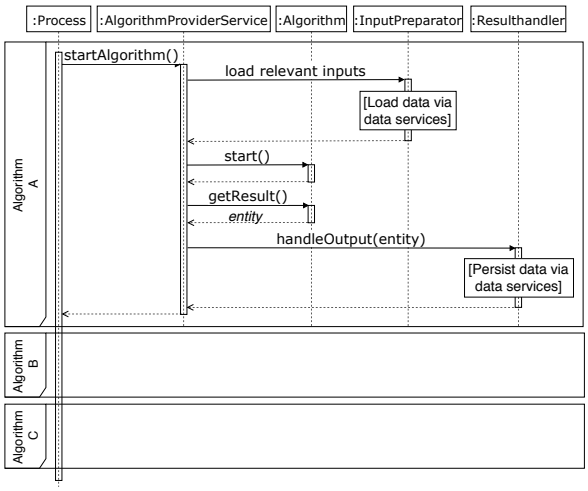

Figure 3. Process, service, and algorithm interaction

A process defines the sequence of execution. Services (`AlgorithmProviderService`-Interface) offer algorithm functionality to processes: The service initiates loading of all relevant inputs via an `InputPreparator`, invokes the execution of the algorithm, and, on completion, initiates persisting of the output via a `ResultHandler`. Afterwards, the process asks the next service to execute an algorithm.

An algorithm does not have to know details about the system environment but can concentrate on the analytics operations. When users want to add a new approach, they need to implement the algorithm interface. That is, developers have to implement three methods: one starting the execution - `start()` -, one retrieving the results - `getResult()` -, and one to request algorithm-specific parameters - `getParameters()`. In this frame, developers can flexibly implement any algorithm that transforms the input into output data, seamlessly integrating it as a module in the system.

Output from one algorithm can be the input for the next. Therefore, algorithms have to provide output in the form of one of the standard system entities. For instance, algorithms that implement the interface `AcceptanceAlgorithm` simulate the order period with request arrivals and acceptance decisions and return a set of accepted orders (`OrderSet`). The system provides one interface per possible output and can be extended by adding new output entities.

The framework can evaluate algorithms separately or within a process context. For instance, a user can compare the number of accepted orders for two different order acceptance algorithms A and B by defining two processes each containing a single algorithm and by analyzing the two `OrderSet` outputs. Alternatively, when interested in profit, the user can define two-step processes invoking algorithms A and B followed by the same vehicle routing algorithm, respectively. Thus, A and B can be compared based on the resulting schedules, weighing basket profit and delivery cost. Processes can be flexibly composed by a user while services and algorithms are designed for reuse. A larger process means less configuration of individual experiments but is not always possible, especially when combining planning and operational tasks.

4.4. Application-oriented standard entities

SiLFul supports reusable, domain-specific data entities. The data model represents experiments with input and output as illustrated via an Entity-Relationship-Model in Figure 4. The entity `Experiment` relates to all input settings of an experiment. Each `Run` is a repetition of an experiment with one to many outputs.

Most inputs and outputs are sets (`EntitySet`) with elements (`Entity`). For instance, an `OrderRequestSet` refers to all order requests for a specific order period. Likewise, `OrderSet` groups all orders from the same order period. A set models information that is common to its entities, for instance, it relates orders to a `Run`.

The data model distinguishes `OrderRequest` and `Customer` entities, as the same customer can launch an order request multiple times over multiple order periods of one experiment. A constant customer attribute is its location and its demand segment. Each order request has an arrival time and a basket value. An `Order` relates to a specific `OrderRequest` and specifies whether the order was accepted and, if it was, for which time window. Different acceptance algorithms can produce different orders given the same order requests. With the same `OrderRequestSet` as inputs, algorithms can be directly compared. Similarly, an `OrderSet` can be input for benchmarking different routing algorithms.

The organization into sets and entities is not limited to demand data. For instance, a `TimeWindowSet` describes a group of `TimeWindows`, each defined by a start time and a length. Different sets reflect different time window designs. For instance, an experiment may feature a set of non-overlapping one-hour time windows, whereas another may include two-hour time slots with consecutive slots overlapping by 30 minutes. Multiple `AlternativeSets` can refer to the same `TimeWindowSet`. An `Alternative` constitutes an actual offer to a customer. For instance, implementing the revenue management concept of flexible products described in [20] would mean to not fix the order

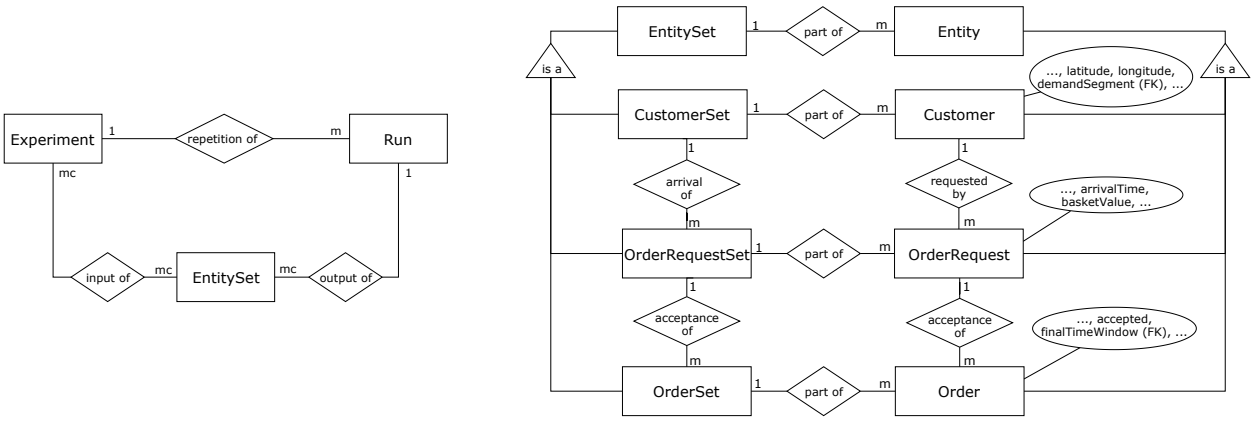

Figure 4. Data model– standard entities (Entity-Relationship-Model, modified Chen Notation [19])

promise to a single time slot, but to promise to deliver at one in a set of time slots.

SiLFul does not restrict developers to the given entities. Nevertheless, when extending the database model, developers have to add the respective loading and persisting operations in the data tier.

5. Case study 1: Simple demonstration

In the following sections, we introduce three case studies to provide first insights into SiLFul's capabilities to support benchmarking experiments. The studies use intergrated benchmarks and additionally extend the system by new concepts. Thus, all case studies feature functionalities of SiLFul that are available for future simulation experiments.

The first case study features a hypothetical e-grocer who wants to understand the effect of e-fulfillment analytics in their specific business scenario, starting by analyzing the impact of a simple change in order acceptance decisions. The practitioner wants to use SiLFul to avoid effects on daily operations and to analyze the impact on a potentially growing business while at the same time not having to implement a whole simulation system from scratch. The e-grocer has already estimated a demand model from operational data, for instance, a multinomial logit model of customers' choice of time slots (refer to [13]).

Until now, the e-grocer offers all feasible time slots to all customers free of charge. They check for feasible deliveries by "online routing" via a simple and fast algorithm (for instance, as in [13]). This computes tentative routes that contain the currently accepted orders and the new request. The retailer wants to determine the effect of updating, that is recalculating, the tentative routing schedules during the order period. On the one hand, updating can increase the number of acceptable orders but on the other hand, it effects the time customers have to wait for an offer.

SiLFul already includes an algorithm that builds new routing schedules over the order horizon, combining a cheapest insertion heuristic and greedy randomized adaptive search similar to [4]. Therefore, the e-grocer only has to add their own online routing approach and their specific problem scenario, generate simulated demand data, and configure the experiments to run. For instance, the e-grocer implements the `RoutingAlgorithm` interface to add their order acceptance with online routing to SiLFul. As a result, the `RoutingSchedule` obtained during a simulated order horizon is saved to SiLFul's database.

The data generation module allows to generate multiple sets of order requests (`OrderRequestSet`), each providing the demand for one delivery period. The retailer has to configure an experiment by defining the demand setting, the arrival process, the length of the order horizon, and the number of runs. Respective data structures are part of SiLFul's data model. Moreover, order request arrival rates are required. The business is currently rather small, with an arrival rate (λ) of 0.2 per time step in a discrete time, homogeneous arrival process of 500 time steps per delivery period (100 expected requests). Nevertheless, they expect significant growth of business in the next year and want to evaluate $\lambda = 0.3$ and $\lambda = 0.4$. The retailer generates 100 sets, that is 100 runs, to account for stochastic effects in the further study. SiLFul draws sample demand from the respective stochastic models, for instance, the multinomial logit model.

The e-grocer has to set up two benchmarking experiments, one featuring the current online routing approach (OR) and one updating schedules (OR-update). The retailer sets parameter `no_routing_candidates` of OR-update to 3 to

build 3 routing schedules in parallel. Parameters allow to slightly vary approaches to analyze the effects of individual design decisions and to find best settings.

The e-grocer uses R to analyze simulation results. Table 1 provides the two performance indicators number of accepted orders (#) and the average waiting time (awt) as a ratio between the outcomes of OR-update and OR. The results suggest that employing OR-update increases the acceptable orders by 0.4% to 4.1%, depending on the experimental parameters λ and number of vehicles. The higher the ratio of expected demand to delivery capacity, the higher the potential gain.

Nevertheless, the e-grocer has to consider this advantage in the light of the increase in computational time, which clearly correlates with the increase in accepted orders. This is mainly due to the increased number of potential insertion positions in routing schedules enabled by a larger fleet. When evaluating these results, the e-grocer can consider compensating increase in runtime via more powerful hardware. Alternatively, they can check SiLFul's pool of benchmarks for more efficient online feasibility checks or, if not sufficiently available, trigger research in this direction by informing SiLFul's user community.

Table 1. Number of orders (#) and average waiting time (awt)

Vehicles	$\lambda = 0.2$		$\lambda = 0.3$		$\lambda = 0.4$	
	#	awt	#	awt	#	awt
2	1.020	4.548	1.029	8.142	1.041	9.119
3	1.011	6.359	1.018	13.243	1.026	21.085
4	1.004	7.217	1.010	18.001	1.014	30.234

6. Case study 2: Anticipative dynamic slotting

In a previous research effort [13], we used SiLFul to benchmark alternative dynamic slotting approaches. These approaches control the set of time slots offered to an order request in order to maximize overall accepted profit for a delivery period [3]. Thus, compared to the retailer in the previous study, they do not offer all feasible time slots but deliberately withhold resources. In this, they weigh a current request's basket value with its fit into delivery schedules and the loss in future acceptable value. This optimization problem is stochastic, complex, and decisions have to be made within milliseconds. Therefore, approaches prepare anticipative information in advance of the order horizon to support decisions on customer arrival.

The dynamic slotting approaches consist of multiple individual algorithms that we added to the benchmarking tool. For instance, we implemented a team orienteering approach, which takes as input a set of order requests and returns a routing schedule that accommodates the most profitable combination of requests from this set. Moreover, we introduced an offline value function approximation approach, which takes multiple sets of order requests as input and trains a linear model to predict the future acceptable value in the order horizon.

The modular design supports future research on individual components of the approaches. It allows straightforwardly exchanging algorithms and benchmarking them as embedded in the overall process. We strongly encourage, for instance, routing experts working on new team orienteering approaches to analyze their effect in a dynamic slotting setting.

The computational study includes analyses on multiple problem scenarios, both artificial and obtained from real data from a German e-grocer. The parameter settings are now persisted in the respective database tables, ready for distribution as database dump along with the software and its data base model.

7. Case study 3: Multi-criteria dynamic slotting

E-fulfillment research usually concentrates on maximizing the short-term profitability of delivery services. Nevertheless, targeted offers of popular time slots to allegedly low value order requests can positively affect how a customer reflects on their experience with the service. They may make further, more profitable orders or recommend the service to members of their social network. Therefore, we introduced the concept of multi-criteria optimization to dynamic slotting [21]. A weighted sum approach allowed to compare different weighted combinations of the criteria profit and social impact of accepted orders. To this end, SiLFul's evaluation component allows flexible adding of objectives and weights to obtain an aggregated value.

A preparation phase in advance of the order horizon includes a vehicle routing on forecasted order requests to obtain capacities per time slot and delivery area as well as a dynamic programming procedure. The latter results in a data structure that accommodates the expected value of being in any state of the order period. The respective object is persisted in the database for transparency and for use in order acceptance over multiple simulated order horizons. To this end, we added a new result type and extend the data model. We persisted the serialized structure as a JSON-string. As mentioned before, we did not design SiLFul as a complete and static system but as an extendable platform, which can be adjusted to new requirements.

8. Conclusion and next steps

We introduced SiLFul, an extendable, Java-based framework for controllable, transparent, and replicable simulation studies to benchmark prescriptive analytics in e-fulfillment. SiLFul provides an environment for researchers to focus on designing solutions, without the overhead of building a simulation system from scratch.

In the long run, such a tool can only be fully effective when continuously developed in a joint community effort. On the one hand, the pool of benchmarks should be up-to-date to stay relevant. On the other hand, the system's modular design should be leveraged to extend its focus on further e-fulfillment areas, such as order packing and inventory management. We are aware that achieving the full potential of a collaborative research practice requires significant change in habits and motivation of a whole research community. Moreover, problems like the validation of such a system in terms of code review still require appropriate guidelines. Nevertheless, we are sure that SiLFul can provide advantages in terms of rigorous and relevant research. The community will have to find a way to leverage that.

In a next step, we plan to publish SiLFul's source code as well as problem scenarios from our case studies. Future tasks include monitoring its usage and verifying extensions from the community, aiming to adjust the system to increase acceptance. Moreover, a user interface can increase ease of use and attractiveness for industry users. Furthermore, after gaining experience in e-fulfillment, future research could design such a framework for other prescriptive analytics research areas, such as health care and public transport.

Acknowledgement: This research was supported by a grant from the German Research Foundation (DFG, Grant No. CL605/2-1).

References

[1] M. Winkenbach and M. Janjevic, *Classification of Last-Mile Delivery Models for e-Commerce Distribution: A Global Perspective*, ch. 11, pp. 209–229. John Wiley & Sons, Ltd, 2018.

[2] A. Fink, N. Kliewer, D. Mattfeld, L. Mönch, F. Rothlauf, G. Schryen, L. Suhl, and S. Voß, "Model-based decision support in manufacturing and service networks," *Business & Information Systems Engineering*, vol. 6, no. 1, pp. 17–24, 2014.

[3] N. Agatz, A. M. Campbell, M. Fleischmann, J. Van Nunen, and M. Savelsbergh, "Revenue management opportunities for internet retailers," *Journal of Revenue and Pricing Management*, vol. 12, no. 2, pp. 128–138, 2013.

[4] A. M. Campbell and M. Savelsbergh, "Incentive schemes for attended home delivery services," *Transportation science*, vol. 40, no. 3, pp. 327–341, 2006.

[5] X. Yang and A. K. Strauss, "An approximate dynamic programming approach to attended home delivery management," *European Journal of Operational Research*, vol. 263, no. 3, pp. 935–945, 2017.

[6] S. Robinson, "A tutorial on conceptual modeling for simulation," in *Proceedings of the 2015 Winter Simulation Conference*, pp. 1820–1834, IEEE Press, 2015.

[7] B. L. Nelson, "'Some tactical problems in digital simulation'for the next 10 years," *Journal of Simulation*, vol. 10, no. 1, pp. 2–11, 2016.

[8] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[9] S. Gregor and A. R. Hevner, "Positioning and presenting design science research for maximum impact," *MIS Quarterly*, vol. 37, no. 2, pp. 337–355, 2013.

[10] N. Agatz, A. Campbell, M. Fleischmann, and M. Savelsbergh, "Time slot management in attended home delivery," *Transportation Science*, vol. 45, no. 3, pp. 435–449, 2011.

[11] C. Cleophas and J. F. Ehmke, "When are deliveries profitable?," *Business & Information Systems Engineering*, vol. 6, no. 3, pp. 153–163, 2014.

[12] J. Mackert, "Choice-based dynamic time slot management in attended home delivery," *Computers & Industrial Engineering*, vol. 129, pp. 333–345, 2019.

[13] M. A. K. Lang, C. Cleophas, and J. F. Ehmke, "Anticipative dynamic slotting for attended home deliveries," *Available at SSRN: https://ssrn.com/abstract=3354537*, 2019.

[14] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations Research*, vol. 35, no. 2, pp. 254–265, 1987.

[15] J. J. Durillo and A. J. Nebro, "jMetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, no. 10, pp. 760–771, 2011.

[16] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007.

[17] M. Zhang and G. G. Gable, "Rethinking the value of simulation methods in the information systems research field : a call for reconstructing contribution for a broader audience," in *Proceedings of the International Conference on Information Systems (ICIS 2014)*, (Auckland, New Zealand), 2014.

[18] P. Rigby, B. Cleary, F. Painchaud, M.-A. Storey, and D. German, "Contemporary peer review in action: Lessons from open source development," *IEEE Software*, vol. 29, no. 6, pp. 56–61, 2012.

[19] P. P.-S. Chen, "The entity-relationship model—toward a unified view of data," *ACM Transactions on Database Systems (TODS)*, vol. 1, no. 1, pp. 9–36, 1976.

[20] G. Gallego and R. Phillips, "Revenue management of flexible products," *Manufacturing & Service Operations Management*, vol. 6, no. 4, pp. 321–337, 2004.

[21] M. A. K. Lang, C. Cleophas, and J. F. Ehmke, "Multi-criteria time window allocation for attended home deliveries," *Available at SSRN: https://ssrn.com/abstract=3000322*, 2019.