IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 3 March 2024, accepted 16 April 2024, date of publication 25 April 2024, date of current version 3 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3393831

RESEARCH ARTICLE

Simulation Based Resource Optimization Using a Decision Tree Clearing Function

CIHANGIR ERTABAN, (Member, IEEE), AND ERINÇ ALBEY
Özyeğin University, Çekmeköy, 34794 İstanbul, Turkey
Corresponding author: Cihangir Ertaban (cihangir.ertaban@ozu.edu.tr)

ABSTRACT This study presents a novel approach to resource allocation in software development teams working with Kanban. The simulation algorithm created in this study takes three types of resources, three types of work, resource capabilities, and a blocking mechanism different from the classic machine breakdown scenario. The data generated by the simulations are used to train a decision tree regression which is integrated into an optimization model as a clearing function. In numerical analysis, the research compares the decision tree clearing function to a straightforward two-step model that only takes the best of the simulation data and finds a resource allocation and a greedy heuristic algorithm which starts from an initial feasible solution and improves it step-by-step. Findings show that the developed decision tree clearing function model outperforms the other two benchmark models in mid and high amounts of data.

INDEX TERMS Agile software development, clearing functions, decision tree regression, optimization methods, simulation.

I. INTRODUCTION

The advent of agile ways of working [1] has revolutionized the world of software development, offering iterative production in short cycles, focused on the value delivered to the customer, via small self-organized teams [2]. Among popular agile frameworks, Kanban with its roots in lean manufacturing and project management, has shown significant efficacy in managing software development schedules [3].

Principles of Kanban, which include visualizing the workflow, limiting work-in-progress, managing flow, making process policies explicit, and improving continuously, show software development teams a solid path of effectiveness without disrupting them, since Kanban is an evolutionary approach rather than a revolutionary change [4]. On the other hand, applying these principles in software development teams is a complex flow to model due to the unpredictability and dependencies in the nature of the work. Currently, software development teams, especially those working with Kanban, lack comprehensive scientific analysis tools and decision-support systems to guide them in managing their workflow and resource allocation effectively. This gap in the literature and practice motivates the current study, which aims to provide an analytical overview and a decision support tool for optimizing the work and resources of software development teams working with Kanban.

The objective of this study is to devise a model that can consider multiple factors in the workflow, such as modifications in resource allocation and work-in-progress limits. For instance, this model will help teams understand how increasing the capacity of a station may affect the output, and how to decide a resource with the capability of processing a single type of work versus multiple types of work with different costs. At present, there are only limited analysis and decision-support tools that can accommodate such analysis.

Our approach, detailed in Fig. 1: Overview of the Study, involves a comparison of two benchmark models and a decision tree clearing function model. Given the complex nature of the Kanban system, we model the process in a simulation, incorporating elements like multiple work item types at multiple stations, including blocks, and a resource optimization model.

The contribution of this study lies in its novel approach of combining simulation, resource optimization, and a decision tree clearing function for resource allocation in software development teams working with Kanban. This approach

The associate editor coordinating the review of this manuscript and approving it for publication was Jesus Felez.

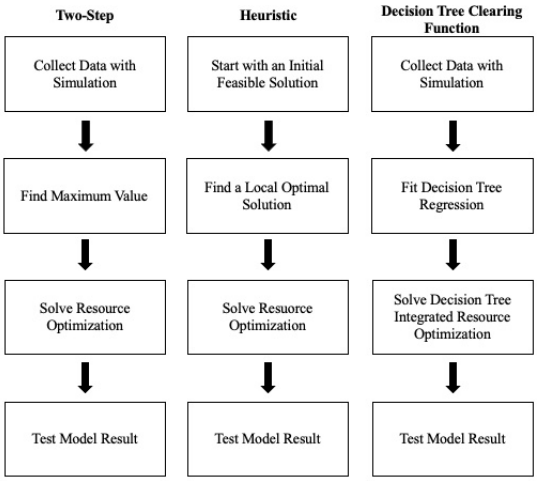

FIGURE 1. Overview of the study.

not only offers practical benefits to software teams but also provides a foundation for further academic exploration in this area.

The rest of this paper is organized as follows: Section II presents a literature review relating to queueing models, simulations in software development, and clearing functions. Section III describes the simulation model used in this study and explains the models. Section IV covers the numerical analysis and the comparison of the models. The final section concludes the paper and discusses potential avenues for further research.

II. LITERATURE REVIEW

This study is based on a holistic approach combining simulations to collect data regarding a software development team working with Kanban with resource optimization. The literature review starts with the queueing models, since the Kanban system can also be considered as a complex queueing system. Due to the complexity of finding closed form solutions to queueing systems, either some approximations are used or some tools such as simulations are utilized for modelling and solution generation. Therefore, in the second part of the literature review, simulations in software development field are investigated. Finally, the overview of the literature relevant to clearing functions, which helps to capture the relationship between characteristics and existing status of the manufacturing systems and desired outputs of the system such as cycle time or throughput, are presented.

A queueing model serves as a system where flow items needing a service are processed. Due to limited capacity, the items wait for their turn, and once serviced, they promptly exit the system [5]. Erlang, initiated one of the earliest significant researches on queueing for telephone lines [6]. Queueing models have since been examined as an applied probability theory subset. Over time, queueing models have evolved into a robust research field with various models including ''Birth-Death (M/M/1)'' processes, ''Multi-Server'' systems, ''Finite Capacity (M/M/1/K)'' queue, ''Multi-Server Finite Capacity (M/M/c/K)'' queue, ''Finite Source (M/M/c/M)'' systems, ''State-Dependent'' service, ''M/G/1'' queue, ''G/M/1'' queue, ''Queueing Networks'' among others [7], [8].

In the context of queueing models, the Kanban system in this study can be regarded as a multi-server priority queueing network with feedback. Research has generated queueing models for traditional and adaptive single stage Kanban systems [9], but introducing multiple stages, multiple work item types and the block concept in this study amplifies the complexity. The blocks or interventions typically stem from machine breakdowns, signifying a server halt [10], unlike this study where the block applies to the work item with the server maintaining its ability to work on another item. Consequently, the flow in this study is too intricate to be modelled as a queueing model, which propels the choice for simulation to understand the Kanban system dynamics in the software development team.

Software development is a complex procedure due to the unique features being developed and often entails working with new technologies or acquiring new skills. Therefore, it's challenging to model a software development team realistically using a queueing model. The alternative lies in using simulations to understand system dynamics, accommodate variations, and envisage potential scenarios in intricate environments [11], [12].

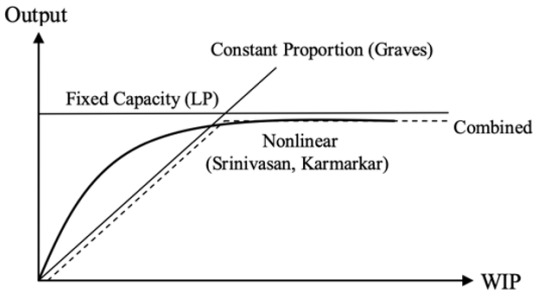

FIGURE 2. Examples of clearing functions [13].

Simulations have been previously used as an effective tool to model complex queueing systems in various fields. Pellegrini et al. combine simulations and a multi-objective optimization model for air traffic management [14], Hsu et al. develop a simulation based optimization approach for material handling planning [15], Osorio and Bierlaire apply a simulation based optimization model to complex stochastic situations in urban transportation. Moreover, examples of previous studies related to Kanban include Baradaran and Akhavan's combination of simulation and optimization of Kanban system parameters [16] and Weflen et al. [17] use of Kanban to estimate lead time using a probabilistic model. This study employs simulations to predict key aspects such as output, lead time, and efficiency to assist decision-making in resource optimization via clearing functions.

Clearing functions have been used extensively to capture the complex relationship between throughput, utilization,

workloads, and lead times in manufacturing systems [13], [18], [19]. As a clearing function, Graves defined a linear function [20] whereas Srinivasan et al. [19] and Karmarkar [13] proposed nonlinear functions, as seen in Fig. 2, Albey et al. consecutive studies use clearing functions to model complex manufacturing models and generate accurate and effective estimates [21], [22].

Baycik [23] proposes machine learning based approaches including decision tree regression and random forest to address integer programming in large size instances. In tree models, the quality of the nodes and the method of split are key. Lee and Jun use a tree model in a logistic regression tree [24] and propose a split model based on separability. Hsu [25] uses decision trees to predict cycle time in manufacturing and share that decision trees perform better than other machine learning algorithms.

Ertaban and Albey [26] use a simulation model to collect data about a team working with Kanban and and train machine learning models to find an optimum parameter set. This study extends Ertaban and Albey's [26] simulation basis and extends with a different perspective including an integrated clearing function. Moreover, the combined resource optimization model is compared with a two-step model and a greedy heuristic algorithm.

III. SIMULATION MODEL AND PROPOSED SOLUTION APPROACHES

This section presents the simulation models that constitute the backbone of the solution approaches (a proposed solution and two benchmark solution approaches).

Given its complexity, the Kanban system employed by the software development team in this study cannot be adequately represented by traditional queueing models. Thus, a simulation model is used to make realistic estimates about the flow of work. Based on the results of the simulation, three solution approaches are investigated. The first two are benchmark models and the third one is the proposed decision tree clearing function, which constitutes the main contribution of this paper. The first benchmark model is a two-step model, where the first step consists of the simulations and chooses the best result of the simulations. Then, the best result of the simulation is used as input for the second step, which is the multi-capability resource optimization. The second benchmark model is a greedy heuristic approach. The heuristic starts with an initial feasible solution. Then it goes on with increasing the resources in an incremental way until reaching a maximum number of iterations or a maximum amount of resources. The benchmark models are used to compare the results with the proposed decision tree clearing function. Both benchmark models come up with a local optimal simulation result which is the input for the multi-capability resource optimization model explained afterwards.

A. SIMULATION MODEL

The Kanban System subject to this study defined as multi-server priority queueing network with feedback before-is a combination of different queueing models including multi-server queueing networks, feedback models and priority queues. All these different queueing models are studied in separate studies with probabilistic models. Integration of two of such models have been also researched in variations, but combining all these models (and more) make the resulting probabilistic model highly complex. Even if such a model might be constructed, reaching a solution using that model might be intractable. For this reason, data collection from simulation and learning from the data is taken as the tractable path.

The simulation software is a key element in the construction of the model. After investigations with regards to speed, integration, analytical and automation capabilities, Python software language is chosen as the simulation basis. Simulations, visualizations and optimization models are carried out end-to-end through a Jupyter notebook based on an open-source discrete event simulation framework Simpy running on Python. In addition, many libraries including Numpy, Cplex, Matplotlib have been used in this study. The simulations are run on a Macbook Pro (2015) equipped with Intel Core i5 processor.

The simulation base is built on Ertaban and Albey's [26] previous work and has similar elements without prioritization of work item types. The process-based simulation starts with different types of work items being randomly created. Then, each work item is pulled and processed by resources at multiple stations. When the work in all the stations is complete, the work item is done. The simulation is run for 52 weeks. All parameters of the simulation are driven from data collected from a large Telco organization and further adaptations are done based on authors' experiences.

The simulation consists of three main stations representing analysis, development, and test. The stations are adapted from a typical software development value stream. In the analysis station, the requirements of business are captured and a solution is described. At the end of the analysis process, an analysis document is likely to be created. The process is carried out by the analysts/ business analysts. After the analysis station, the work is processed in development by the developers. During the development phase, all software development is carried out according to the requirements. At the end of development, the code is ready to be tested. Although some of the tests are automated in IT organizations, it is still a common practice to split the test and development phases. The role of tester carries out the functional tests. At this station, various functions and scenarios are tested and the errors are fixed. The goal is to finalize the code and make it ready to go live. After this station, the code is deployed, and the work item is done. Every work item must be processed in these three stations before being completed.

Software development teams conduct different types of work. The Kanban approach starts with understanding existing processes and optimizing through a series of steps. One of these steps is described as classification of work item types as part of mapping the value stream [4]. Some common types of work include requirements, features, improvement stories, bugs, maintenance, and refactoring. Moreover, some teams may classify the work items based on the application it is being developed, and some teams may classify based on the technology.

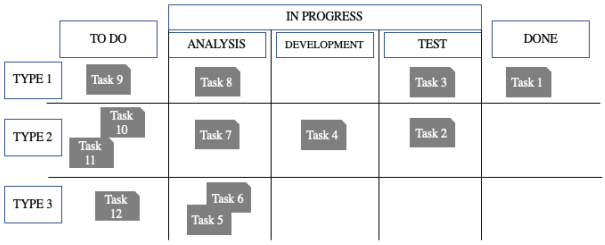

FIGURE 3. Kanban board with three work item types.

In this study, based on the initial data from the Telco company, three types of work are considered:

Type 1: Front-end software development of the user interface and connection to the backend

Type 2: Backend development of the work including servers, database and preparation of the connection with the front-end.

Type 3: Data management.

Fig. 3 depicts three types of work in three swim lanes. Each type of work must go through respective analysis, development and test stations.

Processing different work item types requires different skills. The members of the team may have only one capability to process one type of work or multiple capabilities. The variations in capabilities add the next level of complexity to the model. At each of the analysis, development, and test stations all three types of work need to be conducted. To conduct this type of work, the resource at this station needs to have the capabilities to carry out the work. Each resource may have one of the 7 capabilities showing its ability to conduct the types of work. These abilities are shown in Table 1.

TABLE 1. Work item type - capability matrix.

Type - Capability	C1	C2	C3	C4	C5	C6	C7
Type 1	1	0	0	1	1	0	1
Type 2	0	1	0	1	0	1	1
Type 3	0	0	1	0	1	1	1

The resource capabilities in Table 1 include single capability resources (Capability types 1, 2, 3), double capability resources (Capability types 4, 5 and 6) and a triple capability resource (Capability type 7). Modelling multi-capability resources (double and triple capabilities) requires one resource to handle more than one type of work item at the same station. This resource ends up splitting its time into different types of work. To translate this structure into the simulation, a day-based resource capacity structure is used. The simulation is implemented in a way that each resource may work on a day or not, corresponding to 50% or 100% of capacity.

As an example, capability type 4 resources may handle both type 1 and type 2 work. The resource may distribute the time into types 1 and 2 with a 50% split.

Limiting WIP is one of the key principles of Kanban [4]. When a WIP limit is applied, the resources may not pull additional work when the number of work being processed is equal to the WIP limit. A new work may be pulled only if one work is processed. It is possible to apply a WIP limit to all in process work or separately to each station. In this study, a WIP limit is applied to all work items in process. Application of this principle to the simulation requires using a "hold" function which keeps the work items unless the work items in the process are less than the WIP limits.

Due to the natural complexity of software development in especially large organizations, the work being carried out might be blocked due to a reason outside the team. The block might be waiting for some information, waiting for integration, or approval from another individual or a department. In such cases, the teams working in Kanban put the work item into a block status and pull the next work. When the block is resolved, the team pulls it back at the first chance without sending it to the end of the queue.

In this study, a work item may be blocked at any of the analysis, development, and test stations with a probability. The collected data shows that each of these stations' block probabilities are independent from each other, which means that being blocked at one of these stations does not correlate to a block at any other station. In the simulation software that is being used, the block probability, time of processing before the block, block time and time of processing after the block are all parameters derived from the original data.

The parameters of the simulation are based on the data collected from a large international telecommunication company. The data is based on the work conducted at analysis, development, and test stations including blocked, non-blocked work items and their processing and waiting times of one team.

B. TWO-STEP MODEL

The two-step model (see Fig. 4) is a straightforward model. It starts with collecting data from simulations. The first step includes running simulations including all combinations of the 9 different types of resources and the WIP limit parameters from the initial value to the upper limit. In this study, the initial value for the resources is 0.5 person increasing with an increment size of 0.5 to 2 person, and the WIP limit starting at 3, with an increment size of 6 until 21. In total, this consists of $(4^9 * 4 = 1,048,732)$ simulations.

The highest value of these simulations is picked at the end of the first step.

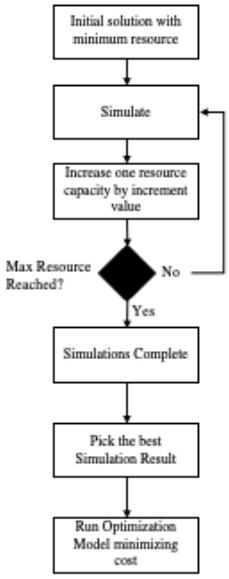

FIGURE 4. Diagram of the two-step model.

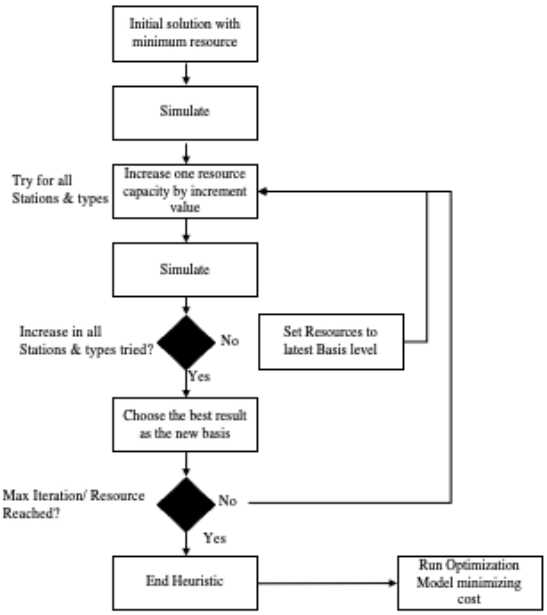

FIGURE 5. Diagram of the heuristic model.

The second step includes taking the resources of the highest value simulation of the first step and using this resource allocation as input for the multi-capability resource optimization. At the end of the multi-capability resource optimization, a final resource set that satisfies the requirements of the first step is found.

C. GREEDY HEURISTIC MODEL

In the greedy heuristic model (which will be referred to as simply the heuristic model from this point on), the simulations start from a minimum feasible resource allocation, which is 0.5 at each station for each work item type. This accumulates to a total of 1.5 person analysis resources, 1.5 person development resources and 1.5 person test resources. Then the simulation increases every resource one by one, tries different WIP limit parameters and finds the best value of the iteration. At the end of the iteration, the base point is improved to the next base point and the next iteration begins. In this study, a maximum resource amount of 1.5 at each station is taken as the upper limit. The algorithm stops if any resource goes beyond the limit and takes the maximum value as the final value as shown in Fig. 5.

After the iterations hit the upper limit and the simulation phase ends, the final result is taken into the multi-capability resource optimization model to find the best allocation of resources.

D. MULTI-CAPABILITY RESOURCE OPTIMIZATION MODEL

Both benchmark models find a simulation result which is the best solution on the horizon of the approach. This simulation result includes various capacity usages of resources at stations and work item types. As an example, the model starts with 50% employees at each station at each type. On the other hand, practically, the software development team consists of full-time equivalent team members. So, it is necessary to find a resource allocation solution which accommodates the simulation result in both benchmark models. This solution is found by the multi-capability resource optimization model. The mathematical model is explained below:

Sets:

$i = workitemtypes, i \in I = \{1, 2, 3\}$

$j = stations, j \in J = \{1 : Analysis, 2 : Development, 3 : Test\}$

$k = capabilities, k \in K = \{1, 2, 3, 4, 5, 6, 7\}$

Parameters:

t_{ik} = 1 if capability k is able to handle work item type i, 0 otherwise

c_{jk} = cost of each resource at station j with capability k

Decision Variables:

s_{ij} = resource capacity input used on work item type i at station j

x_{jk} = # of resource of capability k at station j

y_{ijk} = amount of resource capacity from capability k on work item type i at station j

Constraints:

$$y_{ijk} = t_{ik} M \quad \forall i, j, k \tag{1}$$

$$s_{ij} \leqq \sum_{i} y_{ijk} \ \forall i, j \tag{2}$$

$$\sum_{i} y_{ijk} = x_{jk} \quad \forall j, k \tag{3}$$

The first constraint guarantees that capacities of resources will only be used based on defined capabilities. The second constraint assigns more capacity than the input coming from the simulation and the third constraint turns the capacities into full-time resources at specific capabilities at each station.

TABLE 2. c_{jk}; cost of each resource capability.

	Analyst	Developer	Tester
Capability 1	1000	2000	1000
Capability 2	1000	2000	1000
Capability 3	1000	2000	1000
Capability 4	1500	2500	1500
Capability 5	1500	2500	1500
Capability 6	1500	2500	1500
Capability 7	2500	3500	2500

Objective Function:

$$Min \sum_{jk} x_{jk} c_{jk} \tag{4}$$

Objective function is a minimization of cost searching for the minimum cost level that meets the capacity requirement.

The first parameter t_{ik} representing the capability of resource and work item relationship is shown in Table 1 and the cost of each resource (in hypothetical units) according to their capability is shown in Table 2. It may be observed that multi-capability resources are more expensive than the ones that can process one single type of work.

The last set of parameters is the output of the work item type. For the base simulation parameter set, the first work item type is 50 units, whereas the second work item type is 100 and the third type is 60 units. The output is the generated value of the team.

IV. DECISION TREE CLEARING FUNCTION

The decision tree clearing function model begins with a series of comprehensive simulations, akin to the two-step model. A distinctive feature of this model is its integration of decision tree analysis into the multi-capability optimization framework, enhancing predictive accuracy and resource allocation. This approach provides the opportunity to combine the simulation results with the optimization model rather than separating the two. Thus, simulation results which have a lower value than the best solution, but are at the same time cost effective are considered as well. Such simulation results are almost disregarded in the benchmark models. Fig. 6 represents the flow logic of the proposed decision tree clearing function.

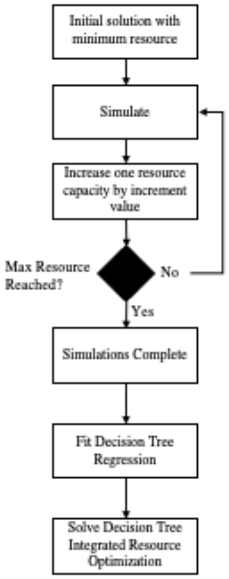

FIGURE 6. Diagram of the decision tree clearing function model.

A. DECISION TREE REGRESSION

The first part of the decision tree clearing function is the decision tree regression. A decision tree regression is fit to the comprehensive simulation results. The parameters x in this regression are the resources and the WIP, whereas the goal, y, is the output.

Fig. 7 shows a high level overview of the decision tree with a base parameter set of values with lighter colored nodes on the left with lower final results and darker nodes on the right with higher results. Fig. 8 shows the final three nodes of the decision tree later in the numerical analysis.

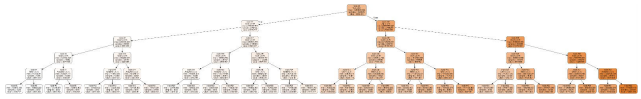

FIGURE 7. Decision tree regression fit on the simulation results max depth: 5.

B. 1.6 DECISION TREE INTEGRATED OVERALL MATHEMATICAL MODEL

The overall mathematical model of the decision tree clearing function is as follows:

Sets:

i = work item types

j = stations

k = capabilities

f = feature

m = leaf node

Parameters:

t_{ik} = 1 if capability k can handle work item type i, 0 otherwise

c_{jk} = cost of each resource at station j with capability k

$mean_m$ = mean value of node m showing output

u_{fm} = upper bound of the decision variables representing feature f at node m

l_{fm} = lower bound of the decision variables representing feature f at node m

$output$ = output value

Decision Variables:

x_{jk} = number of resource of capability k at station j

y_{ijk} = amount of resource capacity from capability k on work item type i at station j

z_m = 1 if node m is active, 0 otherwise

r_f = value of the decision variable representing feature f

Constraints:

$$y_{ijk} \leqq t_{ik} M \quad \forall i, j, k \tag{5}$$

$$\sum_i y_{ijk} = x_{jk} \quad \forall j, k \tag{6}$$

$$\sum_m z_m = 1 \tag{7}$$

$$output = \sum_m mean_m z_m \tag{8}$$

$$r_f = \sum_{ijk} y_{ijk} \quad \forall f \tag{9}$$

$$r_f \leqq \sum_m u_{fm} z_m \quad \forall f \tag{10}$$

$$r_f \sum_m l_{fm} z_m \quad \forall f \tag{11}$$

$$z_m \in \{0, 1\} \tag{12}$$

This model integrates the decision tree clearing into the multi-capability resource optimization model. Thus, the first two constraints come from the multi-capability resource optimization model ensuring capacity usage based on capabilities (5) and capacity usage conversion to full-time equivalent resources (6). The following constraints (7) force only one decision node to be active and (8) describe the output as the mean value of the active node. The constraint (9) is linking the capacity usage y_{ijk} to the respective value of the decision tree r_f and the following two constraints are upper (10) and lower (11) bounds. The last constraint declares that decision variable for the nodes can only be 0 or 1.

Objective Function:

$$Max\ output - \sum_{jk} x_{jk} c_{jk} \tag{13}$$

Objective function is a maximization of profit calculated by output minus cost. This suggests a balance between more resources for maximal output and less resources for minimal cost.

V. NUMERICAL ANALYSIS

The numerical analysis consists of the results of a base parameter set for the benchmark models and the decision tree clearing function followed by a sensitivity analysis. The base parameter set results include the total value, costs calculated by the resource optimization, and the final value. The sensitivity analysis shows the changes of the final value in reflection with the parameters of resource costs, value per work item type and sample size.

A. BASE RESULTS

Base results consist of the two-step, heuristic, and decision tree clearing function methods using the same parameter set. This base parameter set includes the values and the costs of resources described above in the parameters.

As seen in Table 3, decision tree clearing function model with a depth of five outperforms the benchmark models due to its capability of taking the costs into account and not only seeking for the highest output value. The two-step model finds a final value of -2390 which is the worst result. In the following sections, execution steps and performance of each method are explained in detail, and then a sensitivity analysis is conducted with further parameters.

TABLE 3. Comparison of results.

	Benchmark Models		Decision Tree Clearing Function	
	Two-Step	Heuristic	Max Depth: 5	Max Depth: 10
Final Value	-7500	-8110	-5420	-4670
Output Value	7500	7390	4080	4830
Cost	15,000	15,500	9500	9500

TABLE 4. Resource setting of the best result of the simulations.

	A1	A2	A3	D1	D2	D3	T1	T2	T3
Resource	1	1	1	1	1.5	1.5	0.5	1.5	1

1) TWO-STEP MODEL

The two-step model starts with the simulation phase consisting of incremental simulation of resources from 0.5 to 2 at each of the 9 resource types and 4 different WIP limits(3, 9, 15, 21), which accumulates to $4^9x4 = 1{,}048{,}576$ simulations. The model finds the highest output with a maximum resource limit of 10. This value is 13,610 and found at 10 resources and a WIP value of 21. The resource setting of this maximum output is shown in Table 4.

Then, the method proceeds with the second step: multi-capability resource optimization. This step takes the resource setting of the highest output simulation as an input and allocates the minimum cost resource capability distribution. The multi-capability resource allocation finds a result of 3 analysts (capabilities 1, 3, 6), 4 developers (capabilities 1, 2, and two times 3), and 4 testers (capabilities 2, two times 3, and 4) with a total cost of 16,000. Since the total output is 13,610, the final result is -2390 for the two-step model.

2) GREEDY HEURISTIC MODEL

Table 5 shows the incremental steps of the heuristic model. The first column shows the total number of resources at the base level of each step, the output value. Then, the best improvement column depicts the specific resource type (Analyst, developer, or tester and the type of this resource from 1 to 7) that generates the maximum increase compared to the base value. Increasing this resource would lead to the base value of the next step. The final column in the table shows the WIP limit at the best improvement found.

The heuristic model ran for 11 steps and finished at the 12th. Each step consists of trying out an increment of 0.5 at each of the 9 resource types and 4 different WIP limits(3, 9, 15, 21) at each of these resource increments. Thus, each step consisting of 36 simulations accumulates to 396 simulations for the whole model. The heuristic hits the maximum resource limit of 10 finding the highest output value of 12,390. At this moment, the resource setting is shown in Table 6:

This resource setting is the input for the multi-capability resource optimization model which finds a final solution with more than 10 resources, which is the maximum resource cap.

TABLE 5. Steps of the heuristic model.

Step	Total no of Resources	Output Value at Base	Best Improvement	WIP Limit
1	4.5	2320	Dev. T. 3	15
2	5	3730	Dev. T. 1	21
3	5.5	4020	Test T. 2	15
4	6	4210	Dev. T. 2	9
5	6.5	4550	Dev. T. 1	15
6	7	4570	Analyst T. 3	15
8	7.5	5070	Test T. 3	15
9	8	5480	Test T. 1	21
9	8.5	5600	Analyst T. 1	21
10	9	5980	Dev. T. 3	21
11	9.5	6680	Test T. 3	21
12	10	7390	N/A	N/A

TABLE 6. Resource setting of the final result of heuristic.

	A1	A2	A3	D1	D2	D3	T1	T2	T3
Resource	1	0.5	1	1.5	1	1.5	1	1	1.5

For this reason, the result before the last step is taken into account (Row 11 in Table 5), and found a result of 3 analysts (capabilities 1, 2, 3), 3 developers (capabilities 2, 3, 5), and 4 testers (capabilities 1, 2, and two times 3). The feasible optimization result of the heuristic model provides an output value of 12,320 and a cost of resources of 13,500 which accumulate for -1180.

3) DECISION TREE CLEARING FUNCTION

The decision tree clearing function is a holistic approach combining simulations and resource optimization. It uses the data generated by the simulation set to train the decision tree regression and integrates it into the resource optimization model using upper and lower bounds.

In this study, the decision tree regressor function is used from the sklearn [27] package in Python. For the base results, a regression with a maximum depth of 5 is chosen. The R2 value of the regression is 0.8383 which can be accepted as a strong result. Based on the decision tree with a maximum depth of 5, 320 upper and 320 lower bounds are found. The bounds are used as input parameters for the integrated optimization model.

The integrated optimization model finds an optimum solution of 1554,13 using the decision tree regression with a maximum depth of 5. A snapshot, showing the final levels

TABLE 7. Heuristic model - resource allocation.

	Analysis	Development	Test
Capability 1	1	1	1
Capability 2	1	1	1
Capability 3	1	1	2
Capability 4	0	0	0
Capability 5	0	1	0
Capability 6	0	0	0
Capability 7	0	0	0
Total Resources	3	4	4
Total Cost	15.500		

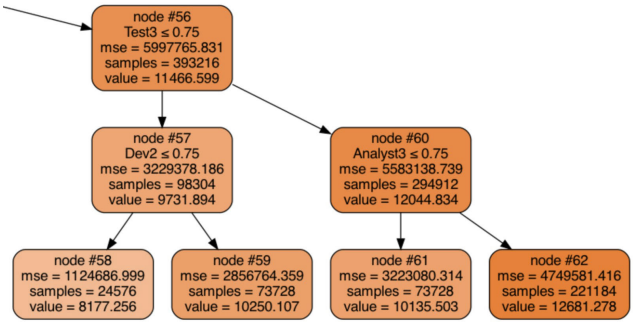

FIGURE 8. Decision tree regression final 3 nodes at max depth: 5.

of the decision tree that leads to the maximum output for the instance under consideration is shown in Fig. 8.

The output value found is 11,054.28 and the cost is 9500. At this node (see Fig. 8 right bottom node) there are 221,184 samples, which is an indicator of the need for further analysis of the decision tree clearing function. The multi-capability resource allocation finds a result of 2 analysts (capabilities 3, 4), 2 developers (capabilities 3, 4), and 2 testers (capabilities 3, 4) with a total cost of 9500.

Since the output value is an aggregated value at the best node, it does not reflect an exact simulation result. Thus, the resource allocation is simulated again and the output value is found as 8500 at a WIP value of 21.

The final value is found as -1000 which is better than the benchmark models. The power of decision tree is shown in this case, which finds an optimum value at a total resource of 6 (See table 8) whereas the benchmark models find their best values at 10 resources. The reason for this is the decision tree clearing function methods' ability to integrate the cost optimization instead of using two separate steps.

TABLE 8. Resource setting of the final result of decision tree clearing function model with max depth of 5.

	A1	A2	A3	D1	D2	D3	T1	T2	T3
Resource	1	0.5	0.5	0.5	0.5	1	0.5	0.5	1

B. SENSITIVITY ANALYSIS

The base results are merely objective and provide a biased view due to the chosen set of parameters. Also, in the study, the full simulation set of over one million data points is considered, whereas in practical cases, only a sample data set is accessible. For this reason, a sensitivity analysis is conducted. The sensitivity analysis covers three dimensions; the parameter set of values and costs, sample size, and decision tree depth. For the parameter set, two sets of values and two sets of costs are used. The values and cost sets are shown in Table 9.

Three different sets of values and two sets of resource costs end up in six value and cost sets.

The second dimension of the sensitivity analysis is the sample size. In this study, 1,048,576 ($4^9 * 4$) simulations represent all combinations. Instead of trying four different levels of resource capacity (0.5, 1, 1.5, 2) at each work item

TABLE 9. Sensitivity analysis value and cost sets.

Value and Cost Sets	Values Type 1, 2, 3	Resource Costs A1,A2,A3, D1, D2, D3, T1, T2, T3
Set 1	50, 100, 60	1000, 1500, 2500, 2000, 2500, 3500, 1000, 1500, 2500
Set 2	50, 100, 60	1000, 1300, 1500, 1500, 1750, 2000, 1000, 1300, 1500
Set 3	150, 200, 160	1000, 1500, 2500, 2000, 2500, 3500, 1000, 1500, 2500
Set 4	150, 200, 160	1000, 1300, 1500, 1500, 1750, 2000, 1000, 1300, 1500

TABLE 10. Sensitivity analysis.

Value and Cost	Sample Size	Results		
		Heuristic	Two-Step	Decision Tree (5)
Set 1	1000	-1180	-3370	-1516
	10,000		-3460	-1000
	Full		-2390	-1000
Set 2	1000	-920	-940	234
	10,000		-930	650
	Full		10	650

type at each station, trying five would end up in 7,812,500 ($4^9 * 4$) simulations, whereas adding one more work item type would need 67,108,864 ($4^{12} * 4$) runs.

The numbers can easily grow beyond computation limits. In such cases, since a full set of simulations is not available, the most favorable solution is to use sample data to conduct estimations. For this reason, data size is one of the dimensions of sensitivity analysis. In addition to the full set of simulations, two different sample sizes of 1000 and 10,000 are selected. For each of these sample sizes, five data samples are picked randomly. The results of the sensitivity analysis are presented in Table 10.

The results show that the decision tree clearing function outperforms the other methods in these value and cost combinations. It is seen that the heuristic method is very competitive at smaller sample sizes since it is not dependent on the sample size. Increased sample size improves the performance of the two-step method since the maximum value of the sample drives the method. Also, decision tree clearing function improves with an increase in sample size up to a level. Value and cost set one is totally cost dominated, in which the results are negative. On the other hand, value and cost set two is cost dominated for the heuristic method and the two-step method except for the full size. For this value and cost set, the decision tree method is capable of finding positive results. To test the methods under value dominated -profit- scenarios, a higher value set is tested with the same cost sets.

The results of the profit scenarios are shown in Table 11.

The profit scenarios add up to the findings regarding the heuristic method. It performs competitively with the two-step model with smaller samples. The main highlight of the profitable scenarios is the decision tree clearing function. In these scenarios, the decision tree with depth

TABLE 11. Sensitivity analysis - profit scenarios.

Value and Cost	Sample Size	Results		
		Heuristic	Two-Step	Decision Tree (5)
Set 3	1000	16,490	14,442	12,054
	10,000		16,088	12,500
	Full		19,240	12,500
Set 4	1000	16,070	16,692	12,852
	10,000		18,278	14,150
	Full		21,440	14,150

TABLE 12. Profit scenarios - additional decision tree depths.

Value and Cost	Sample Size	Results		
		Decision Tree (5)	Decision Tree (10)	Decision Tree (15)
Set 3	1000	12,054	11,338	11,474
	10,000	12,500	13,116	15,624
	Full	12,500	13,190	15,380
Set 4	1000	12,852	14,178	13,960
	10,000	14,150	15,406	17,676
	Full	14,150	15,390	17,380

of five performs worse. The last takeaway is the two-step method. Just like the cost dominated parameters, two-step method improves with larger sample sizes, and reaches to the highest results at full size. The profit scenarios show that the decision tree clearing function with depth of 5 is incapable of competing with the others. So, further analysis was conducted on these scenarios with increased depth of decision tree. In addition to 5, decision tree method with depths of 10 and 15 are conducted and shown together with previous results in Table 12.

Increased level of depth shows that the decision tree method is capable of finding better results with more depth. In both value and cost sets, decision tree results increase for the sample sizes of 10,000 and full size. For only value and cost set 3, decision tree at the depth of 10 and 15 cannot find a higher result than the depth of five, which indicates it is an exceptional case due to random samples.

C. FINDINGS

The value and cost parameters end up in negative results in the first two sets, and positive results in the latest two sets. The first observation by clustering the results in positive and negative is that the two-step model with a full data set finds the highest results in profit scenarios. The reason is, by its nature, the first step of the two-step model is finding the highest single simulation result of the model. This is a straightforward global maximum point. From this level, the model applies resource optimization to find the best fitting resource set. In case the model finds a resource allocation satisfying the highest value resource simulation, then the problem is solved. Only in the case that the resource optimization cannot find a satisfying result, another lower simulation result can be tried. The resource optimization model is flexible, and especially in positive cases, its contribution to the positive results is relatively small.

TABLE 13. Best performing method based on scenario types.

	Amount of Data		
	Small	Mid	Large
Profit Scenarios	Heuristic	Two-step/ Decision tree	Two-step
Cost Scenarios	Heuristic/ Decision Tree	Decision tree	Decision tree

The other benchmark model, which is the heuristic, starts with an initial minimal data set and goes only until a local optimal point before conducting resource optimization. When compared to the global maximum value of the two-step model, it is only a matter of an unlikely coincidence that the heuristic model would find the global maximum. On the other hand, the power of the decision tree clearing function comes from integrating the costs of the resources and the generated value together into the optimization model. This strength comes forward in cases where resource costs balance the values, and the final results are close to zero, or negative.

The second observation is based on the heuristic method. The heuristic method starts with an initial feasible solution which has minimum adequate resources and increases the resources based on the highest improvement. The method stops at a given limit of a maximum number of resources. Unlike the other two models, the heuristic model does not need any sample data. Hence, the method itself uses simulations to find the highest improvement at each node. In a case where the heuristic model ends at 11 steps as in the base scenario above, 396 simulations are conducted, where the minimum number of base data for the other two steps is 1000. Thus, the total number of simulations and necessary computation time is much smaller for the heuristic method. The results of the heuristic method are very competitive with limited data (sample size of 1000), and even for a relatively small amount of data (sample size of 10,000) the heuristic model can still be competitive. The heuristic model falls behind with an increased amount of data.

The decision tree clearing function provides the best results in all of the cost dominated sets. In these sets, the highest value is achieved through minimum cost. In most cases, the decision tree clearing function with the depth of five ends at a resource distribution of six resources in total. This is a conservative, low-cost solution.

Another significant pattern is that the decision tree clearing function results are improving in line with decision tree depth in profit scenarios. In both profit scenarios at sample sizes of 1000, and 10,000, the decision tree clearing function with the depth of 15 performs better than with the depth of 10. So, the decision tree gets better and better. Theoretically, if the decision tree goes to a depth that is enough to include only one result in a node, it should reach or pass all other methods, since it should find the exact best solution. But this case is not practically possible due to computation time at large sample cases and is not the intent of using the decision tree clearing function.

Table 13 summarizes the best performing methods based on scenario types and amount of data. Taking the summary and the results into consideration, as a conclusion of the numerical analysis, it is possible to come up with three key findings:

1. Two-step model performs best in profit scenarios. In both profit and cost dominated scenarios, the performance of the two-step model increases with the amount of data.

2. Heuristic model is competitive with limited data and computational time. When the information level increases, heuristic starts to be irrelevant.

3. Decision tree clearing function performs best in cost dominated scenarios. Also, when there is reasonably good and high amount of data, decision tree performs well. As seen in profit cases, the method's performance increases with higher level of depth.

VI. CONCLUSION

In conclusion, this study introduces a novel decision tree clearing function that effectively balances output and cost considerations. This study is one of the pioneering works encompassing extensive simulations of a Kanban-driven IT team and integrating the simulation results with a multi-capability resource optimization model.

This approach addresses complex elements of software development such as including multiple work item types, resources with varying capabilities, work-in-progress limits, and blocked work items. The proposed decision tree clearing function exhibits superior performance against benchmark models under the studied parameters.

However, software development is a dynamic process filled with uncertainties. Despite our model's performance, potential enhancements exist. Future enhancements might include integrating lead and cycle times, enhancing work efficiency, and improving resource utilization strategies. We could further optimize the model by incorporating lead and cycle times, work efficiency, resource utilization, prioritization, and multi-capability resource allocation. Furthermore, considering the dynamic and uncertain nature of software development, integrating stochastic models or robust optimization approaches could be beneficial.

Our proposed model does require extensive computation and simulation, a recognized limitation. Future research could aim to streamline the computational process or devise faster solution methods. Despite the mentioned limitations, this work presents a significant step forward, offering a comprehensive tool for IT leaders to optimize their decision-making and team management process in a challenging, fast-paced environment.

APPENDIX
PARAMETERS OF THE SIMULATION MODEL

Parameters of the simulation model are collected from a telco company. Table 14 explains processing times and block times as input parameters for the simulation.

TABLE 14. Processing and block times as parameters.

Station	Expression
Analysis no Block	LOGN(26.3, 67.6)
Analysis to Block	LOGN(32, 70.4)
Analysis after Block	53*BETA(0.438, 0.438)
Analysis Block	2 + EXPO(36.3)
Development no Block	WEIB(34.1, 0.988)
Development to Block	WEIB(31.2, 1.06)
Development after Block	LOGN(28.9, 59.1)
Development Block	2 + LOGN(24.3, 46.9)
Test no Block	EXPO(26.6)
Test to Block	0.999 + EXPO(16.1)
Test after Block	1 + EXPO(15.6)
Test Block	1 + WEIB(23.9, 0.703)

In addition to these parameters, the block percentages are based on the same data provided in Table 15.

TABLE 15. Block probabilities.

Station	Block Probability
Analysis	18.45 %
Development	35.42 %
Test	19.59 %

ACKNOWLEDGMENT

For the purpose of editing and grammar enhancement, OpenAi GPT-4 is used in this article.

REFERENCES

[1] K. Beck et al., "Manifesto for agile software development," 2001. [Online]. Available: https://agilemanifesto.org/

[2] S. Denning, "Why agile is eating the world," vol. 2, 2018. [Online]. Available: https://www.forbes.com/sites/stevedenning/2018/01/02/why-agile-is-eating-the-world%E2%80%8B%E2%80%8B/

[3] H. Lei, F. Ganjeizadeh, P. K. Jayachandran, and P. Ozcan, "A statistical analysis of the effects of scrum and Kanban on software development projects," *Robot. Comput.-Integr. Manuf.*, vol. 43, pp. 59–67, Feb. 2017.

[4] D. J. Anderson, *Kanban: Successful Evolutionary Change for Your Technology Business*. Washington, DC, USA: Blue Hole Press, 2010.

[5] J. Howl, "Queueing systems," *Automatica*, vol. 3, nos. 3–4, pp. 231–244, 1966.

[6] A. K. Erlang, "Sandsynlighedsregning Og telefonsamtaler," *Nyt tidsskrift for Matematik*, vol. 20, pp. 33–39, Oct. 1909.

[7] W. J. Stewart, *Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling*. Princeton, NJ, USA: Princeton Univ. Press, 2009.

[8] G. Franzl, "Queueing models for multi-service network," Ph.D. dissertation, Inst. Telecommun., Tech. Univ. Vienna, Vieanna, Austria, 2015.

[9] V. Tardif and L. Maaseidvaag, "An adaptive approach to controlling Kanban systems," *Eur. J. Oper. Res.*, vol. 132, no. 2, pp. 411–424, Jul. 2001.

[10] D. P. Gaver, "A waiting line with interrupted service, including priorities," *J. Roy. Stat. Soc. Ser. B, Stat. Methodol.*, vol. 24, no. 1, pp. 73–90, Jan. 1962.

[11] Y. Pan, M. Zhou, and Z. Chen, "Simulation-based optimization for resource allocation at TPL systems," *Int. J. Ind. Eng., Theory, Appl., Pract.*, vol. 19, no. 2, pp. 1–12, 2012.

[12] J. A. García-García, J. G. Enríquez, M. Ruiz, C. Arévalo, and A. Jiménez-Ramírez, "Software process simulation modeling: Systematic literature review," *Comput. Standards Inter.*, vol. 70, Jun. 2020, Art. no. 103425.

[13] U. S. Karmarkar, "Capacity loading and release planning with work-in-progress (WIP) and leadtimes," *J. Manuf. Oper. Manage.*, vol. 2, nos. 105–123, p. 37, 1989.

[14] A. Pellegrini, P. D. Sanzo, B. Bevilacqua, G. Duca, D. Pascarella, R. Palumbo, J. J. Ramos, M. À. Piera, and G. Gigante, "Simulation-based evolutionary optimization of air traffic management," *IEEE Access*, vol. 8, pp. 161551–161570, 2020.

[15] H.-P. Hsu, C.-C. Chou, and C.-N. Wang, "Heuristic/Metaheuristic-based simulation optimization approaches for integrated scheduling of yard crane, yard truck, and quay crane considering import and export containers," *IEEE Access*, vol. 10, pp. 64650–64670, 2022.

[16] V. Baradaran and B. Akhavan, "Determining number of withdrawal Kanban using bi-level optimization and simulation approaches," *Int. J. Ind. Eng.*, vol. 26, no. 2, pp. 1–22, Mar. 2019.

[17] E. Weflen, C. A. MacKenzie, and I. V. Rivero, "An influence diagram approach to automating lead time estimation in agile Kanban project management," *Expert Syst. Appl.*, vol. 187, Jan. 2022, Art. no. 115866.

[18] J. Pahl, S. Voß, and D. L. Woodruff, "Production planning with load dependent lead times," *4OR*, vol. 3, no. 4, pp. 257–302, Dec. 2005.

[19] A. Srinivasan, M. Carey, and T. E. Morton, *Resource Pricing and Aggregate Scheduling in Manufacturing Systems*. Pittsburgh, PA, USA: Carnegie Mellon University, 1988.

[20] S. C. Graves, "A tactical planning model for a job shop," *Operations Res.*, vol. 34, no. 4, pp. 522–533, Aug. 1986.

[21] E. Albey, Ü. Bilge, and R. Uzsoy, "An exploratory study of disaggregated clearing functions for production systems with multiple products," *Int. J. Prod. Res.*, vol. 52, no. 18, pp. 5301–5322, Sep. 2014.

[22] E. Albey, Ü. Bilge, and R. Uzsoy, "Multi-dimensional clearing functions for aggregate capacity modelling in multi-stage production systems," *Int. J. Prod. Res.*, vol. 55, no. 14, pp. 4164–4179, Jul. 2017.

[23] N. O. Baycik, "Machine learning based approaches to solve the maximum flow network interdiction problem," *Comput. Ind. Eng.*, vol. 167, May 2022, Art. no. 107873.

[24] S. Lee and C.-H. Jun, "A novel split selection of a logistic regression tree for the classification of data with heterogeneous subgroups," *Int. J. Ind. Eng., Theory, Appl. Pract.*, vol. 30, no. 2, pp. 1–16, 2023.

[25] C.-H. Hsu, "Optimal decision tree for cycle time prediction and allowance determination," *IEEE Access*, vol. 9, pp. 41334–41343, 2021.

[26] C. Ertaban and E. Albey, "Optimization of workflow and output in Kanban teams through machine learning," *Manuscript Submitted Publication*, vol. 11, no. 1, Dec. 2024, Art. no. 2179123.

[27] F. Pedregosa, S. Varoquaux, A. Gramfort, V. Michel, and B. Thirion, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Dec. 2011.

CIHANGIR ERTABAN (Member, IEEE) received the B.S. degree in management engineering from Istanbul Technical University, in 2007, and the M.S. degree in industrial engineering from Boğaziçi University, İstanbul, in 2011. In 2017, he started the Ph.D. studies with Özyeğin University, İstanbul.

He worked in the banking, tech, and consulting industries as a Business Analyst, the Project Manager, a Product Owner, and an Agile Coach. He is currently a Professional Coach certified by the International Coaching Federation. He is the Senior Manager of Global Consulting Company, with a focus on agile ways of working, including Kanban. His research interests include team performance, agile in education, managing agile teams, and Kanban. He published and spoke at multiple conferences, including XP, Agile Prague, and Forward 3.0.

ERINÇ ALBEY received the bachelor's, master's, and Ph.D. degrees from the Department of Industrial Engineering, Boğaziçi University.

After completing the Ph.D. degree, in 2012, he was a Postdoctoral Researcher with North Carolina State University, where he also taught courses in operations research and production systems. He is currently an Assistant Professor and the Director of the OzUBEX Digital Transformation Center, Özyeğin University. During his academic career, he has worked on projects in collaboration with companies, such as SAP, Intel, and Novartis. In addition to his academic work, he has managed the analytics team at a firm specializing in customer analytics and developed various predictive models. His areas of research interests include optimization, planning and scheduling, predictive modeling, and applications of mathematical modeling and simulation in flexible production systems.