UTASiMo: a simulation-based tool for task analysis

Anastasia Angelopoulou¹ and Konstantinos Mykoniatis²

Abstract
Research on task analysis and human performance has focused on the development of adequate tools, models, and methods to understand, analyze, and improve the relationship between humans and systems. As technology continues to advance and to change the nature of human work, techniques of analysis are changing to meet the new needs. This work attempts to fill the gaps in the current task analysis tools and describes the architecture and development of a simulation model, named UTASiMo. UTASiMo is a simulation tool that aims to enhance task analysis by automatically generating a multi-method simulation model for well-defined tasks based on a spreadsheet template filled in by the user. The generated model analyzes and simulates tasks performed by individual simulated agents representing human operators while accounting for the estimation of an operator's utilization and error prediction. This work highlights the design, development, and evaluation of the hybrid architecture (discrete event and agent based) of UTASiMo. The development of the system dynamics model, which is responsible for the human error assessment, is a work in progress and is excluded from the present paper. A real-world case study has been adapted to evaluate the hybrid architecture of UTASiMo. The same case study was modeled using the Micro Saint simulation tool. The results produced by UTASiMo were compared with the real-world data as well as with the results produced by Micro Saint for validation purposes. The comparisons indicate the validity of the UTASiMo-generated model and also that the hybrid architecture produces more variability in the results than using only one method. The comparisons also show promise that the tool will reduce the time and effort of the task analysis simulation.

Keywords
Agent based, discrete event, human operator variability, modeling, simulation, task analysis, UTASiMo

1. Introduction
The field of Human Factors and Ergonomics (HF/E) is currently going through changes due to the advances in technology that have resulted in more complex and demanding tasks. Various tools, such as Micro Saint and TaskArchitect, have been used and are still predominately used to aid the process of studying human performance, designing a system, or developing a training program. Task analysis is a common tool used by HF/E professionals to identify the factors that affect human performance and to understand the system requirements. While much effort for research and development has been applied upon task analysis, a combined theoretical and methodological progress is needed.¹

Current research efforts that focus on the development of adequate tools and methods of tasks analysis have revealed unresolved issues concerning the concepts and methods used. Task analysis is a static process, traditionally conducted using pen and paper, which means that the representation is difficult to change after it is drawn on the paper. When software tools are used to perform task analysis, representations can be modified easier and change dynamically. The simulation of task models is one way to create more dynamic representations.

Simulation allows analysts to step through a scenario, observe the postulated processes that take place and identify bottlenecks in a system design before the system is actually constructed. However, simulation studies can be complicated due to usage of simulation software varying from programming languages to unique software operations or instructions. In particular, the modeling process is the hardest and most time consuming part of simulation.

¹Institute for Simulation and Training, University of Central Florida, USA
²Industrial Engineering and Management Systems, University of Central Florida, USA

Corresponding author:
Anastasia Angelopoulou, Institute for Simulation and Training, University of Central Florida, Orlando, FL 32816 USA. Email: aangelopoulou@knights.ucf.edu
as it may require knowledge in both programming and simulation theory. Preferably, the modeling process and the analysis of the results should be done automatically, thereby reducing the required time and effort of the analysts and eliminating potential errors. Here, we present the implementation of a simulation-based task analysis tool capable of automating the modeling process and simulating the generated models to produce valid statistical results for task analysis.

1.1 A brief history of task analysis

Task analysis is a methodology that studies the activities that an operator (or a group of operators) needs to perform to accomplish a goal. The primary purpose of task analysis is the identification of the tasks the operator needs to perform, their decomposition into primary task steps, and the comparison of task demands to the operator’s capabilities in order to describe human performance in a particular task or scenario. Although task analysis is a time consuming and challenging process, it was chosen as a basis for the development of the simulation tool. Task analysis is effective for procedural tasks and for providing a concrete representation of the steps and logical relationships required to achieve the operator’s goals. Through task analysis, the resources and processes related to current tasks can be defined and the results of analysis can be used to make recommendations regarding changes to current procedures and/or new tasks.

Task analysis has its roots in the work of Taylor and Gilbreth, which introduced techniques for identifying, measuring, and organizing task components of manual work. The tendency of complex systems toward failure due to human errors shifted attention to components of human performance, leading to the development of more methods of analysis. Currently, over 100 different task analysis methods have been described in the literature, each of them selected based on different criteria, such as the purpose and scope of the analysis, as well as task, cost, and time factors, among others. However, task analysis methods and tools are not flawless. To date, most of the developed methods and tools allow for representation of the static aspects of the tasks performed by expert human operators, neglecting aspects of the work environment, that is, physical layout, and dynamic aspects of the task. In addition, some of the developed task analysis tools that support simulation are not available at the current time, while others may be inconvenient to use due to costs and learning curves. For example, Micro Saint, which is a generic purpose discrete event (DE) simulation tool used for task and human performance analysis, requires elementary programming skills. Table 1 summarizes the differences and similarities between the proposed tool and the following tools: Work Models that Compute (WMC), the extension of the Operator Function Model (EOFM), Micro Saint, TaskArchitect, and the Man Machine Integrated Design and Analysis System (MIDAS). According to Table 1, there is no simulation software that (i) automates the development of task analysis simulation models from a spreadsheet without the need for programming skills, training, or modeling and simulation (M&S) expertise; (ii) allows for simulation of the dynamic aspects of the tasks, the heterogeneity of human operators, and the internal state of human operators, as well as visualization and animation of the physical layout of the environment, that is, locations of tasks and animated humans within a simulated environment; and (iii) combines and integrates the three M&S approaches, agent based (AB), DE, and system dynamics (SD) simulation, to generate task analysis models. Each of these simulation methods is responsible for different aspects of the task analysis simulation process. The present work highlights the design, development, and evaluation of the hybrid simulation architecture of UTASiMo. The SD model is a work in progress and is not included in the present paper.

<table>
<thead>
<tr>
<th>Software tool</th>
<th>No need for programming skills, training, or knowledge of simulation theory</th>
<th>Simulation capabilities</th>
<th>M&amp;S methods</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No need for programming skills, training, or knowledge of simulation theory</td>
<td>Supports simulation</td>
<td>Automated model construction</td>
</tr>
<tr>
<td>WMC</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Micro Saint</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TaskArchitect</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>EOFM</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>MIDAS</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>UTASiMo</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

1.2. Current work

This paper describes the implementation of a Universal Task Analysis Simulation Model, named UTASiMo, which was developed in AnyLogic. UTASiMo is a simulation-based tool that automatically generates a task analysis simulation model from a spreadsheet based on user inputs without the need for programming skills or M&S experience. UTASiMo allows the user to upload an image with the workplace layout as well as the destination address of a spreadsheet template (i.e., Excel spreadsheet), which includes tasks, task locations, and time estimations (Tables 4–6). At the moment, task locations are obtained manually, but future work will consist of reading the locations from a computer-aided design (CAD) file or a drawing of the physical layout. Once the user has provided the required information into the Excel spreadsheet template, UTASiMo processes the information provided by the user and generates a simulated scenario. More specifically, UTASiMo generates a simulation model that estimates task execution times and operator utilization (OU) considering time constraints, variability in human operators’ skills, and stochastic task times, as well as the location of human operators and tasks to be performed. Furthermore, UTASiMo allows the user to alter the spreadsheet information in order to perform and estimate alternative behaviors and task execution sequences. The key functionality of UTASiMo is its capability to simulate and generate task analysis models and to assist in decision making on changes to current well-defined procedures.

In previous work, we provided an overview of existing task analysis methods and tools and described the conceptual design of UTASiMo. More specifically, we used the Unified Modeling Language (UML) to describe the conceptual model and illustrate the different constructs and concepts of UTASiMo. In addition, we presented a case study for illustration purposes. Here, we describe the deployment of the hybrid DE–AB architecture of UTASiMo that is responsible for two main processes. The first process analyzes a task network comprised of task sequences and human operators are usually omitted or assumed to behave perfectly. A comprehensive consideration of human activity that includes a simultaneous representation of multiple users with varying levels of skills and abilities (i.e., from novice to expert human operators) is also missing. Therefore, a task analysis methodology, named Universal Task Analysis (UTA), was developed for filling these gaps and applying hybrid simulation to perform task analysis.

2. Theoretical foundation of UTASiMo

Current task analysis methods and tools lack a human operator model that indicates how operators process and execute tasks. Human operators are usually omitted or assumed to behave perfectly. A comprehensive consideration of human activity that includes a simultaneous representation of multiple users with varying levels of skills and abilities (i.e., from novice to expert human operators) is also missing. Therefore, a task analysis methodology, named Universal Task Analysis (UTA), was developed for filling these gaps and applying hybrid simulation to perform task analysis.
The UTA methodology considers basic concepts and elements of multiple task analysis methods, including Hierarchical Task Analysis (HTA), \textsuperscript{27,28} Timeline Analysis,\textsuperscript{29} and Spatial Operational Sequence Diagrams,\textsuperscript{30} to analyze tasks in various scenarios. UTA adopts the hierarchical decomposition approach of a task into subtasks and allows for simulation of task execution times. UTA also includes the estimation of the execution time of each task step and provides a graphical representation of a flow diagram linking the task steps in the order they are performed. In addition, UTA allows for defining human operator profiles based on their skills and abilities, as well as events that may happen during the task execution.

The UTA process is illustrated in Figure 1 using a UML diagram, which describes the user’s interaction with UTASiMo when performing simulation-based task analysis. The main steps of UTA are the following.

- Identification of the overall task under analysis.
- Decomposition of overall task into subtasks. The tasks and subtasks are decomposed into further subtasks until no further breakdown is necessary. Each subtask is performed by a single human operator.
- The data collection input process requires the user to fill in a spreadsheet template with the collected data. The input data include execution times for cognitive and physical tasks, task locations, human operator roles, and other environmental and human-related factors.
- Simulation model execution and output analysis. Once the subtasks have been fully defined and the data have been collected, the user uploads the spreadsheet on UTASiMo and runs the simulation. The UTASiMo algorithm automatically creates a model based on the spreadsheet inputs. The simulation model animates the system and the operator behavior and produces statistical results for execution times, utilization levels, and human error.

UTA considers the roles of agents, the types of tasks, and the work structure, among others. The following elements are included in the UTASiMo simulation model: (i) task characteristics; (ii) human operator (agent) characteristics; (iii) environmental conditions and events. These elements are used for determining task completion time and OU. OU is used as a measure for estimating human operator workload.

3. Hybrid architecture of UTASiMo

This section describes the hybrid DE and AB architecture of UTASiMo. Modeling an ordered sequence of well-defined events is one of the distinguishing features of DE, while modeling the heterogeneity of agents across a population is one of the distinguishing features of AB simulation compared to DE and SD. Integrating AB and DE simulation into the development of the task analysis model becomes essential in order to capture both the spatial and temporal task and human operator characteristics.

3.1. Discrete event module

Tasks are key components in the task analysis and the study of human performance. Although task-related research has broadly emerged in the field of HF/E, there is limited agreement on the understanding of a task and its characteristics in the literature.\textsuperscript{31,32} Different conceptual bases for defining the tasks exist in the literature.\textsuperscript{33–37} In this research, a task is defined as a set of discrete activities performed in a sequence to achieve an overall goal, which has a beginning and end. Activities are tasks that do not require further decomposition.

DE simulation allows for modeling a task as a series of discrete activities, such as decisions, perceptions, and/or physical activities. The activities are represented as nodes in a network that have dependencies (arcs) between them, while a task is a network of nodes. Task structure properties represented in the model are the number of subtasks, task complexity, required skills, priority, duration, and critical time.

Task complexity is defined as the degree of difficulty to perform a task and it measures the task’s potential for being perceived as difficult.\textsuperscript{12–14,38–40} The user is called to define each task’s complexity based on a scale from 1 to 5, with 1 being the simplest task and 5 the most complex task. The task complexity is considered into the estimated task times that the user enters into the simulation. Thus, the complexity of a task will only be used in the error assessment as one of the major determinants of the
probability that human operators make errors while performing their tasks. Task duration is the estimated execution time for an average operator to perform a task. Task duration is assumed to be stochastic following a triangular probability distribution. The distribution is incorporated to account for the variability in execution times, since most of the times it is unlikely to perform a task in exactly the same time on two different occasions. The triangular distribution was selected for the following reasons: (i) it is the most commonly used distribution for modeling expert opinion; and (ii) it can be used as a rough model for the time required to perform a task when limited or no real-world data are available. At the moment, the tool supports input only with triangular distribution. Work is under progress in order to further increase the capabilities of the tool in order to support various types of distribution.

Each task requires a certain set of skills and experience that the human operator should have in order to complete it efficiently. A skill factor is assigned to each task to denote the level of performance required to accomplish the task. The match between the operator’s level of performance and the experience required by a task impacts the duration of the task and the probability of a failure during task execution. Table 2 describes the task structure in more detail.

### 3.2. Agent-based module

Task analysis effectively integrates the human element into the system design. One of the highly significant areas of interest in task analysis is the simultaneous representation of human operators with varying characteristics. By modeling human operators individually using AB simulation, the diversity that exists among them in their attributes and behaviors can be observed. UTASiMo models each agent independently from each other and does not currently support collaboration among multiple agents. Teamwork in task analysis is a topic that requires special considerations and will be investigated in the future.

A typical structure of an AB model consists of the following:

- a set of agents, which have attributes and behaviors;
- a set of agent rules defining agent behavior and/or interaction among agents; and
- the agents’ environment.

Work is under progress in order to further increase the capabilities of the tool in order to support various types of distribution.

Each task requires a certain set of skills and experience that the human operator should have in order to complete it efficiently. A skill factor is assigned to each task to denote the level of performance required to accomplish the task. The match between the operator’s level of performance and the experience required by a task impacts the duration of the task and the probability of a failure during task execution. Table 2 describes the task structure in more detail.

### Table 2. Task structure.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Definition</th>
<th>Model variablea</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task ID</td>
<td>Unique task identifier</td>
<td>tid (integer)</td>
</tr>
<tr>
<td>Task name</td>
<td>Name of the task</td>
<td>tname (string)</td>
</tr>
<tr>
<td>Task location</td>
<td>Node location in two-dimensional continuous space</td>
<td>tlocation (double, double)</td>
</tr>
<tr>
<td>Number of subtasks</td>
<td>Sequence of connected activities</td>
<td>tnumber (integer)</td>
</tr>
<tr>
<td>Task complexity</td>
<td>Degree of difficulty</td>
<td>complexity (double)</td>
</tr>
<tr>
<td>Skill factor</td>
<td>Level of performance required by each operator to perform the task</td>
<td>tskills (double)</td>
</tr>
<tr>
<td>Priority</td>
<td>Recommended order to perform the tasks</td>
<td>priority(integer)</td>
</tr>
<tr>
<td>Critical time</td>
<td>The time in which the task needs to be performed</td>
<td>tcrit (double)</td>
</tr>
<tr>
<td>Duration</td>
<td>The estimated time it takes for an average operator to perform the task</td>
<td>duration (double)</td>
</tr>
</tbody>
</table>

aThe first part of the model variable indicates the name of the variable in the model and the second part in the parenthesis declares the type of variable.
The skill factor is used to denote the level of performance attained by an operator working under customary conditions.

Regarding the other three characteristics, an agent has a spatial property as well as internal behavior modeled by a statechart. The operators can change states based on their task execution strategy: how to choose the next task position and when to stop. Thus, an agent’s internal state consists of three components to control its actions (Figure 2): (i) the ability to “Perceive” the current state of environment; (ii) the ability to “Adapt” and update its own representation of the environment; and (iii) the ability to “Act” based on its updated internal state and its current perception. The agent’s perception is related to information that the agent receives (“receiveEvent”). The event includes information regarding the following: (i) the agent’s physical location; (ii) if there are still tasks to execute; and (iii) the locations of the tasks. A detailed list of the agent structure is presented in Table 3.

The internal state is modeled as a statechart with three states as follows.

- **The state of Perception**, which is an event-handler triggered every millisecond by “receiveEvent” to check the environment for any event that is either user-defined (i.e., alarm, stairs) or can be retrieved from the model (i.e., number of tasks to be executed, next task location, etc.).
- **The state of Adaptation**, which includes a list of actions linked with their corresponding events. Each occurred event may be linked to one or more actions. The agent selects the appropriate action based on the triggering event and the information gathered from the environment. In the case of user-defined events, the action is selected based on equal probabilities. For example, the agent checks the environment and receives information for tasks that need to be executed. The agent then checks the list with the assigned tasks and moves to either the next in the list if there is a task priority/prerequisite or to the closest one.
- **The state of Action**, when the execution of a task or other event-triggered action takes place.

The initialization state is activated once, when the simulation starts and directs the agent to execute the first task.

3.2.2. Human operator utilization estimation as a workload measure. Workload is one of the factors that may lead to human error and failure to perform a task correctly when it is too high (overload) or too low (underload).
A variety of subjective and analytic workload techniques exist, including uni- and multi-dimensional rating scales, NASA-TLX, questionnaires, interviews, and timeline analysis. The concept of utilization as a workload measure has been used in numerous studies.\textsuperscript{38–40}

Here, the OU is used as a proxy for measuring a human operator’s workload. Utilization refers to the percentage of time that the human operator is busy, given a time period. The OU is given by the following expression:

\[
OU = \frac{\text{Time Busy}}{\text{Total Time}} \times 100
\]

where

\[
\text{Time Busy} = \text{skill factor}_{\text{agent}} \times \text{duration}_{\text{task}} \times \text{Task Frequency}
\]

The resulting percentage is used to categorize utilization according to the following rates: Low (\(< 60\%\)), Medium (60–75\%), High (75–90\%), and Extreme (\(> 90\%\)).\textsuperscript{44} Prior human performance models and empirical studies suggest that utilization levels of Low and Extreme are undesirable.\textsuperscript{38–40}

4. Case study

For the purpose of evaluating the UTA framework and the UTASiMo interface, a real-world example was adapted and modified\textsuperscript{45} to match the spreadsheet template the user needs to fill in. The example concerns the need to provide back-up emergency cooling in a plant in the event of a switch room fire. Plant operators need to complete the task within 2 hours of the initial failure of the primary means of cooling provided by the plant. The UTA framework starts with a hierarchical decomposition, similar to the Hierarchical Task Analysis (HTA) method. The overall task “Provide Emergency Cooling” is divided into 11 subtasks. Figure 3 illustrates the flow of the overall task execution. Then, the user fills in a task spreadsheet template, as depicted in Table 4. The user also provides information for the agent and events, as described in Tables 5 and 6. The critical time for the overall task “Provide Emergency Cooling” is 120 min. Each subtask has a specified location and a mean execution time, and it may be composed of both physical and cognitive components. The cognitive task components are incorporated into the event spreadsheet. The execution times provided in the adapted example are rounded up to the nearest 5 min to account for possible problems that may occur during the task execution.\textsuperscript{45} The layout and coordinates are not to scale.

Finally, the user uploads the spreadsheet on UTASiMo and runs the simulation. The model is automatically created in a very short time based on the spreadsheet inputs. The generated model simulates and animates the system and the operator behavior and produces the results. We simulated the same scenario with UTASiMo and Micro Saint and compared the results. The scenario describes the expected behavior of the human operator in the plant after the initiating the event of a switch room fire. In both cases, we assumed that the physical system and equipment perform optimally without failures or malfunctions during the event of the switch room fire. In addition, the human operator is assumed to be of average experience and assigned with a skill factor of 1.0 and initial walking speed of 90 m/min.\textsuperscript{46,47}

4.1. Scenario simulated with UTASiMo

In this scenario simulated with UTASiMo, no human error or plant failure occurs following the initiating event of the fire. The parameters assumed as the input data of UTASiMo for this case are those of Tables 4–6. The spreadsheet and layout image are uploaded to UTASiMo and the simulation starts. Figure 4 depicts the generated work environment and the animation view of the UTASiMo model. The task locations are represented by yellow–red dots, while human operators are represented by agent icons that move across the task locations to perform the tasks. Agents change colors based on their state, that is, black for walking and blue for executing a task. The lines indicate the flow of the operators performing the tasks.

Various techniques were used for verification and validation.\textsuperscript{48} Like in other simulation software, the user can either run the simulation model with a fixed seed that provides deterministic results or with a random seed. When
using a random seed, every simulation run is unique and the results are stochastic. Firstly, the model was successfully tested for one operator in order to verify the total task execution time. We then verified and validated the model by observing the animation of the simulation output. Validation included comparison of the simulated system behavior with the behavior of the system as provided in the obtained case study.

Quantitative measures, such as the total task time, were examined for validity. We run the simulation model using the random seed for a number of replications to produce the confidence intervals and the total time histogram. The number of replications was calculated at a 95% confidence interval, based on the following expression:

$$n = \frac{n_0^2}{h^2}$$

where $h_0$ is the half-width from the “initial” number of $n_0$ replications and $h$ is the desired level of precision. A pilot run was executed at a 95% confidence interval for an initial number of $n_0$ replications and the reported half-width $h_0$ for the mean total time was used in expression (3) to estimate the number of replications so that the mean time will be within the desired half-width bound $\pm h$ of the true total time. The recommended number of replications is 122. The simulation results after 122 replications are illustrated in Figure 5 and Table 7.

The simulation outputs show that the corresponding actions are executed according to the regular procedure. Please note that the total simulated time is close enough to

Table 4. Task spreadsheet template.

<table>
<thead>
<tr>
<th>Task ID</th>
<th>Task name</th>
<th>Parent task</th>
<th>Initial location</th>
<th>Next location</th>
<th>Mean time (min)</th>
<th>OpID</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Provide emergency cooling</td>
<td>-</td>
<td>(120, 320)</td>
<td>(120, 320)</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Detect alarm</td>
<td>0</td>
<td>(150, 350)</td>
<td>(280, 380)</td>
<td>0.01</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Take readings</td>
<td>1</td>
<td>(280, 380)</td>
<td>(120, 191)</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Collect hoses</td>
<td>2</td>
<td>(120, 191)</td>
<td>(610, 220)</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Carry hoses</td>
<td>3</td>
<td>(610, 220)</td>
<td>(340, 151)</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Collect pump</td>
<td>2</td>
<td>(340, 151)</td>
<td>(610, 220)</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>Take pump</td>
<td>5</td>
<td>(610, 220)</td>
<td>(680, 151)</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>Connect hose to pump</td>
<td>6</td>
<td>(680, 151)</td>
<td>(610, 220)</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>Connect to Cooling</td>
<td>7</td>
<td>(610, 220)</td>
<td>(580, 151)</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>Start generator</td>
<td>8</td>
<td>(580, 151)</td>
<td>(581, 152)</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>Start pump</td>
<td>9</td>
<td>(581, 152)</td>
<td>(580, 151)</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>Monitor</td>
<td>10</td>
<td>(580, 151)</td>
<td>(580, 151)</td>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>

min: minutes; OpID: Operator ID.

Table 5. Agent spreadsheet template.

<table>
<thead>
<tr>
<th>Agent ID</th>
<th>Agent name</th>
<th>Agent skill</th>
<th>Initial location</th>
<th>Initial speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Plant Operator</td>
<td>1.0</td>
<td>(150, 350)</td>
<td>Default = 90 m/min</td>
</tr>
</tbody>
</table>

Table 6. Event spreadsheet template.

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Event name</th>
<th>Task ID</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alarm</td>
<td>1</td>
<td>Move to Task with ID = 2</td>
</tr>
<tr>
<td>2</td>
<td>Hoses heavy</td>
<td>4</td>
<td>Agent slows down Speed = 60 m/min</td>
</tr>
</tbody>
</table>

Figure 4. Automatically created model with layout image when one human operator performs the tasks. The layout is not created to scale. (Color online only.)
the critical time (120 min) and it may also exceed the 2-hour limit (see the histogram in Figure 5). Moreover, the utilization of the human operator ranges in high levels. Table 7 illustrates that the observed value for the total time is outside the simulated confidence interval. Although the confidence interval validation method is reassuringly quantitative, and as a validation method it is reliable, there nevertheless remain aspects of the method that call for subjective judgment by the analyst, model developer, model user, or accrediting authority. In the present case, the simulated results are close to the actual observed measurements of the task and the observed values are within an acceptable tolerance of the confidence intervals’ endpoints. Thus, the model is considered to be valid. Overall, a simulation model is usually a simplification and approximation of the real system. “There is no such thing as absolute model validity, nor is it even desired” (p. 24).

4.2. Scenario simulated with Micro Saint

The same scenario was then simulated with Micro Saint to compare the simulation results and validate the model. The Micro Saint software allows model construction using a series of modeling components and the Microsoft C# programming language. Therefore, more time and effort is required by the user in order to construct the model compared to the UTASiMo automated model construction.

More specifically, to develop the model, the task analysis data (Table 4) was entered into Micro Saint as a series of individual tasks, as illustrated in Figure 6. Then the interrelationships amongst the tasks were defined in the form of a task network. The interrelationships included the task sequence and which tasks were caused by events in the scenario. The attributes of each task included the task name, the operator that performs the task, task location, and task duration.

In contrast to UTASiMo, the Micro Saint software does not provide a built-in estimator for utilization. Thus, we implemented a function “CALC_OU” in C#, which is called in each task’s beginning effect field and calculates the current utilization for the operator performing the task being executed.

Finally, we ran the simulation model for 122 replications using the same input data and assumption as in the case of the UTASiMo model. We then obtained the results

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Measure name</th>
<th>Observed value</th>
<th>Simulated average</th>
<th>Simulated standard deviation</th>
<th>Simulated confidence interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTASiMo</td>
<td>Total Time</td>
<td>110</td>
<td>115.579</td>
<td>5.051</td>
<td>[114.683, 116.475]</td>
</tr>
<tr>
<td>UTASiMo</td>
<td>OU</td>
<td>&gt; 75%</td>
<td>76.306</td>
<td>4.511</td>
<td>[75.506, 77.106]</td>
</tr>
</tbody>
</table>

*Total time is in minutes.

bOU = Operator Utilization level. Observed value was inferred from case study description.45

Figure 5. Simulation results from UTASiMo for 122 replications with one operator.

Figure 6. Simulation model created with Micro Saint software.
from Micro Saint and compared them with the UTASiMo output.

We set-up two-sample \( t \)-tests to determine if there is statistical difference between the Micro Saint and the UTASiMo simulation outputs. The level of significance is \( \alpha = 0.05 \). More specifically, we set up the hypotheses for the comparison of the mean OU and total time obtained by UTASiMo and Micro Saint:

\[
\begin{align*}
H_0: & \text{ There is no difference between the mean utilization obtained by Micro Saint and the mean utilization obtained by UTASiMo (} \mu_{\text{OU,MS}} - \mu_{\text{OU,UTASiMo}} = 0). \\
H_1: & \text{ There is a difference between the mean utilization obtained by Micro Saint and the mean utilization obtained by UTASiMo (} \mu_{\text{OU,MS}} - \mu_{\text{OU,UTASiMo}} \neq 0). \\
\end{align*}
\]

\[
\begin{align*}
H_0: & \text{ There is no difference between the mean time per task obtained by Micro Saint and the mean time per task obtained by UTASiMo (} \mu_{\text{TT,UTASiMo}} - \mu_{\text{TT,MS}} = 0). \\
H_1: & \text{ There is a difference between the mean time per task obtained by Micro Saint and the mean time per task obtained by UTASiMo (} \mu_{\text{TT,UTASiMo}} - \mu_{\text{TT,MS}} \neq 0). \\
\end{align*}
\]

Table 8 summarizes the results of the two-sample \( t \)-tests and the performance measures comparison between the Micro Saint and UTASiMo simulation outputs in terms of total time and OU estimation. Since the \( p \)-value for all tests is greater than the level of significance, we cannot reject the null hypothesis, which states that there is no significant difference between the mean utilization obtained by Micro Saint and the mean utilization obtained by UTASiMo. Since the null hypothesis cannot be rejected, this indicates a lack of evidence against the null hypothesis. The \( p \)-value quantifies the confidence about this decision and the UTASiMo-generated model was considered valid for the acceptable range of accuracy under the given set of the experimental conditions.\(^{50}\) However, two types of errors need to be taken into consideration when using this validation method: Type I error (or model builder’s risk) and Type II error (or model user’s risk). Type I error is the incorrect rejection of a true null hypothesis and the probability of making it is \( \alpha = 0.05 \), while a Type II error is incorrectly retaining a false null hypothesis and the probability of making a Type II error is \( \beta \approx 0.016 \).

The comparison also revealed that building simulation models with UTASiMo requires less time and effort, since the process is automated. In addition, UTASiMo provides a built-in OU estimator, in contrast to Micro Saint. The latter requires programming and modeling in order to define a function that calculates OU.

Although both Micro Saint and UTASiMo models produce similar outputs, they presented different variation in their model outputs. The level of variance in the Micro Saint model was significantly lower compared to the UTASiMo variance. The reason for these differences in variance is probably due to the different simulation approaches inherited in each tool. A queuing discipline is inherited in Micro Saint DE process-based models where the system is more organized. On the other hand, UTASiMo hybrid DE–AB models are based on object-oriented organizational structures that are more related to the real world. Furthermore, UTASiMo hybrid simulation architecture allows for capturing additional features of complex human behavior, but these features may further increase the output variance.

### 5. Conclusion

UTASiMo is a multi-method simulation tool that automatically generates a task analysis simulation model based on a spreadsheet template. UTASiMo incorporated the integration of DE and AB methods with a focus on the importance of task-, human-, and environment-related characteristics. This work identified a number of developmental benefits and limitations in current task analysis tools to enhance UTASiMo design when task analysis occurs within time-critical, event-driven environments.

One of the major unique benefits of UTASiMo is the capability of generating task analysis simulation models without the need for programming skills, software experience, or knowledge of simulation theory. This allows for easier and faster simulation of task analysis models. UTASiMo was successfully tested using real-world data.

Currently, each subtask is performed by a single human operator, without collaboration among agents. In the
future, we plan to incorporate a teamwork module that will allow for advanced communication and collaboration among agents, as well as parallel task execution for a single agent.

**Funding**

This research was awarded with a training award with title “I-Corps: UTASiMo” under UCF’s prime award from the National Science Foundation (NSF), award number CNS-1347356 entitled “University of Central Florida I-Corps Sites: Enhancing Technology Commercialization to Develop a World-Class Innovation Ecosystem”.

**References**