2015

Unknown Exception Handling Tool Using Humans as Agents

Mwaka Mahanga
University of North Florida

UNKNOWN EXCEPTION HANDLING TOOL

USING HUMANS AS AGENTS

by

Mwaka Mahanga

A thesis submitted to the
School of Computing
in partial fulfillment of the requirements for the degree of

Master of Science in Computing and Information Sciences

University of North Florida
School of Computing

April, 2015

The thesis "Unknown Exception Handling Tool Using Humans as Agents" submitted by Mwaka Mahanga in partial fulfillment of the requirements for the degree of Master of Science in Computing and Information Sciences has been

Approved by the thesis committee: Date

Dr. Karthikeyan Umapathy
Thesis Advisor and Committee Chairperson

Dr. Sherif Elfayoumy

Dr. Zornitza Genova Prodanoff

Accepted for the School of Computing:

Dr. Asai Asaithambi
Director of the School

Accepted for the College of Computing, Engineering, and Construction:

Dr. Mark Tumeo
Dean of the College

Accepted for the University of North Florida:

Dr. John Kantner
Dean of the Graduate School

ACKNOWLEDGEMENT

I would like to express my special appreciation and thanks to my advisor Dr. Karthikeyan Umapathy. I would like to thank you for encouraging me to continue with my research and not giving up on me. Your continued support, time and patience have been unmeasurable. I would also like thank my committee members, Dr. Sherif Elfayoumy and Dr. Zornitza Genova Prodanoff for the patience they had in this long process and for all the suggestions and comments. A special thanks to my family, my father Dr. Petero Kwizera, Mother Gladness Mtango, my siblings Baraka Senzira, Tumaini Munezero and Upendo Hawa. Last but not least would like to express my appreciation to my beloved wife Naomi Lupemba for the unwavering support and understanding during the many hours I dedicated to achieving this milestone in my life and career.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

In a typical workflow process, exceptions are the norm. Exceptions are defined as deviations from the normal sequence of activities and events. Exceptions can be divided into two broad categories: known exceptions (i.e., expected and predefined deviations) and unknown exceptions (i.e., unexpected and undefined deviations). Business Process Execution Language (BPEL) has become the de facto standard for executing business workflows with the use of web services. BPEL includes exception handling methods that are sufficient for known exception scenarios. Depending on the exception and the specifics of the exception handling tools, processes may either halt or move to completion. Instances of processes that are halted or left incomplete due to unhandled exceptions affect the performance of the workflow process, as they increase resource utilization and process completion time. However, designing efficient process handlers to avoid the issue of unhandled exceptions is not a simple task. This thesis provides a tool that handles unknown exceptions using provisions for exception handling with the involvement of human activities by using the BPEL4PEOPLE specification. BPEL4PEOPLE, an extension of BPEL, offers the ability to specify human activities within BPEL processes. The approach considered in this thesis involves humans in exception handling tools by providing an alternate sub process within a given business process. A prototype application has been developed implementing the tool that handles unknown exceptions. The prototype application monitors the progress of an automated workflow process and permits human

involvement to reroute the course of a workflow process when an unknown exception occurs. The utility of the prototype and the tool using the Scenario Walkthrough and Inspection Methods (SWIMs) are demonstrated. We demonstrate the utility of the tool through loan application process scenarios, and offer a walkthrough of the system by using examples of instances with examples of known and unknown exceptions, as well as a claims analysis of process instances results.

Chapter 1

INTRODUCTION

A workflow is composed of a series of tasks whereby each task is initiated after the completion of the previous task (Wil M.P. van der Aalst & Kumar, 2003); the process continues until the final task is completed. Workflow systems are focused on supporting a business process (Wil M.P. van der Aalst & Kumar, 2003). A typical workflow process involves tasks performed by people and applications. In a workflow process, a task is assigned to an individual or to an application, and after the completion of the task, the subsequent task is assigned to the next individual or application as specified in the process. The major concern with these workflow processes is that they tend to be coercive, isolationistic, and inflexible (Smith & Fingar, 2004).

Workflow management systems are software applications that can help manage these concerns, in that they provide the capability to define and manage the workflows between resources, such as people and applications (((Wil M.P. van der Aalst & Hee, 2002), p. 27). Workflow management systems offer provisions to define different workflows for different types of tasks or processes. At each stage of the workflow, the workflow management system ensures that the individuals or groups responsible for the next task are notified and receive the data they need to execute their stage of the process.

Workflow systems are also capable of automating redundant tasks and ensuring that uncompleted tasks are automatically followed up. Such automated business processes are pre-configured machine-to-machine processes that do not need human involvement. Many businesses rely on these systems to measure and analyze the execution of a process so that process improvements can be made. Business processes that are not automated generally are based on workflow technologies that require some portions of a task to be completed by a specific user and then passed on to the next one.

As with any business process, changes do occur constantly, and these workflow processes are subject to change. Workflow management systems can handle deviations from normal processes that have already been modeled into the workflow. Known exceptions that alter the normal path of a workflow can be controlled with the use of fault handlers. Fault handlers can catch exceptions that range from network issues to particular services not being available. In most cases, when an exception is caught by fault handlers, the process is terminated. Workflow management systems also support compensation handlers that have been modeled for specific scenarios based on fault handlers that can trigger different workflow paths designed to handle specific known exceptions (Akram, Meredith, & Allan, 2006).

Unknown exceptions, on the other hand, are known to be costly, as they are not designed within the business process and not handled by workflow management systems (Derbali, 2011). Any unexpected deviations to anticipated workflow events would alter the workflow process and affect subsequent tasks in the workflow. The entire workflow

process instance may need to be remodeled in order to handle unknown exceptions. There are a number of reasons that could lead to an unexpected exception or deviation in a workflow process, such as a system failure or user generated exceptions (Wil M. P. van der Aalst, 2001). Once an exception is not handled, a process execution is halted, and as a result, the next task is not completed. Companies have to invest in resources to restart the process instance or to begin a new one. Existing workflow methodologies provide the ability to access information about unknown exceptions that have occurred; however, they do not provide the capability to handle exceptions during runtime (Dickson K.W. Chiu, Li, & Karlapalem, 1999).

Unknown exceptions occur frequently in traditional workflow management systems, because not all workflow deviations in the process can be foreseen at build time (Song, Han, & Thiery, 2007). Human involvement is necessary to handle process instances that have been halted due to unknown exceptions (Casati et al., 1999). Most exception handling techniques are encapsulated into core business processes, but the runtime exception corrupts the activated process instance (Vojevodina & Kulvietis, 2004). A halted process can be difficult to track and resolve, thereby causing greater frustration and more time spent troubleshooting on the part of the end user. Thus, developing a tool that improves the performance of exception handling tools is an important research problem.

In our context, performance is determined by the number of process instances that result in meaningful conclusions; for example, a process that has 5 out of 5 meaningfully completed instances has a higher performance than one that has 3 instances out 5 completed

meaningfully. Instances of meaningful completion states include completed, reassigned, and terminated states. Instance states that are not meaningful include failed and in-progress states. Our thesis provides a meaningful completion for process instances with unknown exceptions using human involvement.

If a tool were able to provide the ability to include human agents in a running process instance, it could aid in handling unknown exceptions. The inclusion of human activities as part of the process could provide the flexibility needed to make changes in the workflow process instance. As unknown exceptions are critical in a workflow process, their effects might not be realized until a lot of damage has been done or time has been lost.

In this thesis, we have developed a tool for handling unknown exceptions by implementing a sub process to the core of the business process that allows for human intervention. We employ the use of BPEL4PEOPLE (BPEL4People, 2010), an extension to Business Process Execution Language (BPEL; (Jordan & Evdemon, 2007), and WS-Human Task (WS-HumanTask, 2012). BPEL4PEOPLE extends the capabilities of BPEL to provide an automated interaction between automated processes and tasks handled by people. WS-Human Task allows for interactions between humans and processes via web services. As a proof of concept, we implemented the tool using components of Intalio Tempo, an open source workflow framework, built as an extension to Intalio and deployed as services. BPEL4PEOPLE will be used to provide a human interface in which users can see a list of active tasks and have the option to complete these task instances.

Chapter 2

BACKGROUND AND LITERATURE REVIEW

2.1 Workflows

A workflow involves a set of coordinated activities within a business process. The procedural steps and rules associated with a process are controlled by a workflow engine, which determines how each step should be handled as well as the order of the steps' executions. Figure 1 is an example of a workflow process for online shopping (Fakhroutdinov, 2013). The basic example displayed in Figure 1 shows that the workflow process begins when a user searches for items online. Items found can then be viewed and added to the Shopping Cart. The cart can be canceled or completed to end the process by checking out.

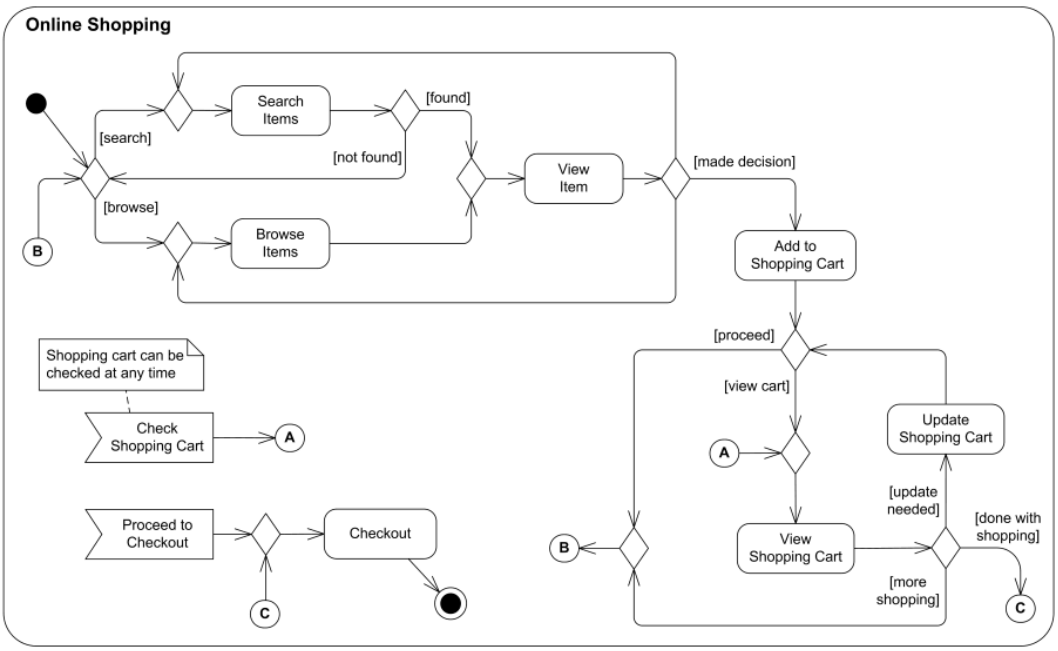

Figure 1. Online shopping workflow process

A number of studies have been conducted and have found that implementing workflow systems has resulted in a number of benefits, including reduced costs, faster production timeframes, improved organization of tasks, and greater consistency in the quality of products and services (Cain & Haque, 2008). Despite the above-mentioned advantages, workflow systems have still fallen short; they have proven to be expensive when it comes to the development, maintenance, and integration of different services. Most workflows are not open or flexible enough to work on different platforms or to interact with different applications (Hochmüller & Dobrovnik, 2005). There is an increasing need for the creation of workflow methodologies and implementations that are flexible and adaptable and that can be used with a variety of applications, regardless of the underlying system architecture.

2.1.1 Types of Workflows

There are a number of different varieties of workflows; these could be complex and involve a number of sequence events, or they could simply target a particular purpose or event. In the industrial world, these workflows are divided into three types: administrative, production, and ad hoc.

2.1.1.1 Administrative Workflows

Administrative workflows involve simple tasks, as opposed to sets of activity steps. With this type of workflow, the completion of the workflow process could be indicated by a notification sent to the intended recipient. Tasks are limited to basic activities that have been targeted for a particular task, such as a request for a product. These are not tracked within the workflow process, there is no series of events, and modifications occur with less frequency.

2.1.1.2 Production Workflows

Examples of production workflows include loan application, online shopping, insurance, and order processing workflows. These are mostly complex in nature and are comprised of a number of tasks that involve sending work from one task owner to the next. Production workflows occur on a large scale and are made up of activities that are performed within

the entire enterprise. These require frequent modifications based on fluctuating business needs.

2.1.1.3 Ad Hoc Workflows

Ad hoc workflows are normally unstructured and involve a number of entities. The basic principle is to have people work together and think together; these tasks can be completed by users from different departments or locations. Although they provide the collaborative aspect, since they are unstructured, these can often lead to confusion as to who is responsible for handling a specific task. Modifications on these workflows are also less frequent than production workflows.

2.1.2 Workflow Exceptions

Exceptions within a workflow process are situations in which a turn of events has occurred, and which has caused a change in the sequence of events (Wil M. P. van der Aalst, 2001). These exceptions could be predetermined and modeled within the process or could be unknown. These can be caused by system failures, faults, signal errors, or other unexpected scenarios. Modern systems have to be equipped with exception handling tools; as the need for flexibility arises, companies are more aware and willing to trade or work with companies in different locations.

2.1.3 Known Exceptions

Known exceptions are exceptions that have been modeled within a process (Wil M. P. van der Aalst, 2001). A specific set of events has been set to account for these exceptions. These events could range from the process being terminated to being forwarded to a different entity. Systems that provide these handling capabilities have to analyze their workflow processes, predetermine every single exception that could occur, and model the rerouting procedures for those particular exceptions. Figure 2 illustrates an online shopping example that shows a number of deviations within the process; these are the known exceptions. For example, at the beginning of the process, if items are not found, the user is redirected to the browsing stage where they could browse for more items. Figure 2 highlights the exceptions indicated in Figure 1.

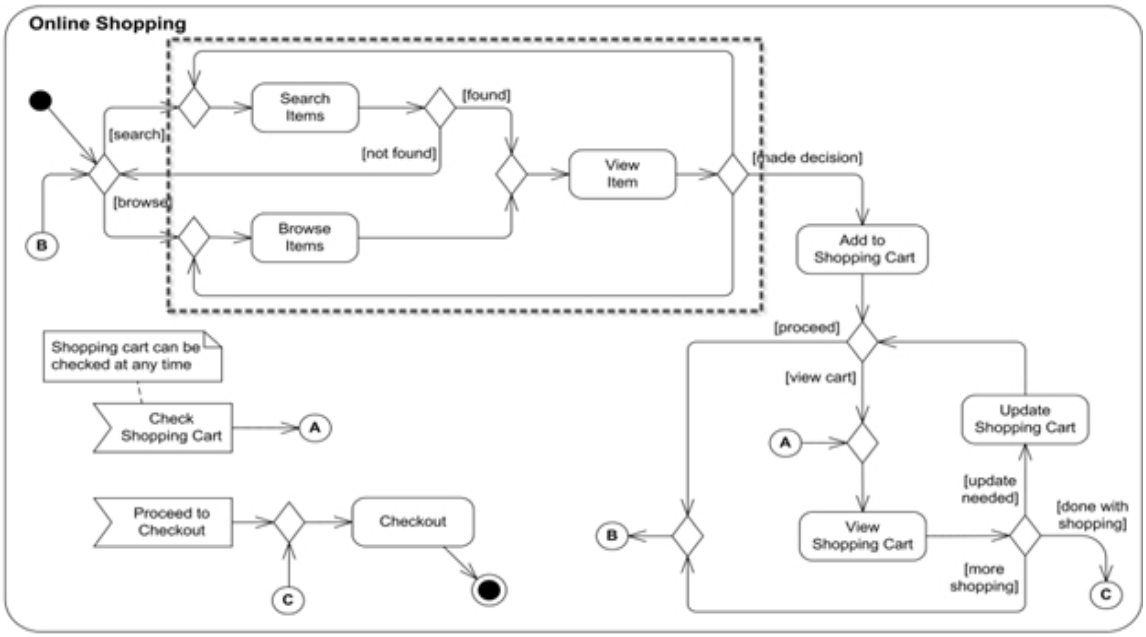

Figure 2. Online shopping with a known exception

2.1.4 Unknown Exceptions

Unknown exceptions occur unexpectedly; these are not modeled within the workflow process (Wil M. P. van der Aalst, 2001). This is a major issue within automated processes, automated processes fail to provide solutions when an unprogrammed event occurs. A human element is always introduced in these situations. When processes are halted, no synchronization can occur, and as a result, the process must be restarted again or must be left in an unknown state. Human involvement offers the added advantage of being able to think and provide solutions based on current situations. Consider the online shopping workflow, which could be halted for unknown reasons, thereby causing unexpected errors. For example, during the checkout step, users could, for unknown reasons, face an online purchasing restriction set by banks, in which case merchants could be flagged as suspicious, preventing the completion of purchase orders, unless these merchants are added to a safe list. Figure 3 highlights the step in which the above-mentioned unknown exception would occur in the online shopping example.

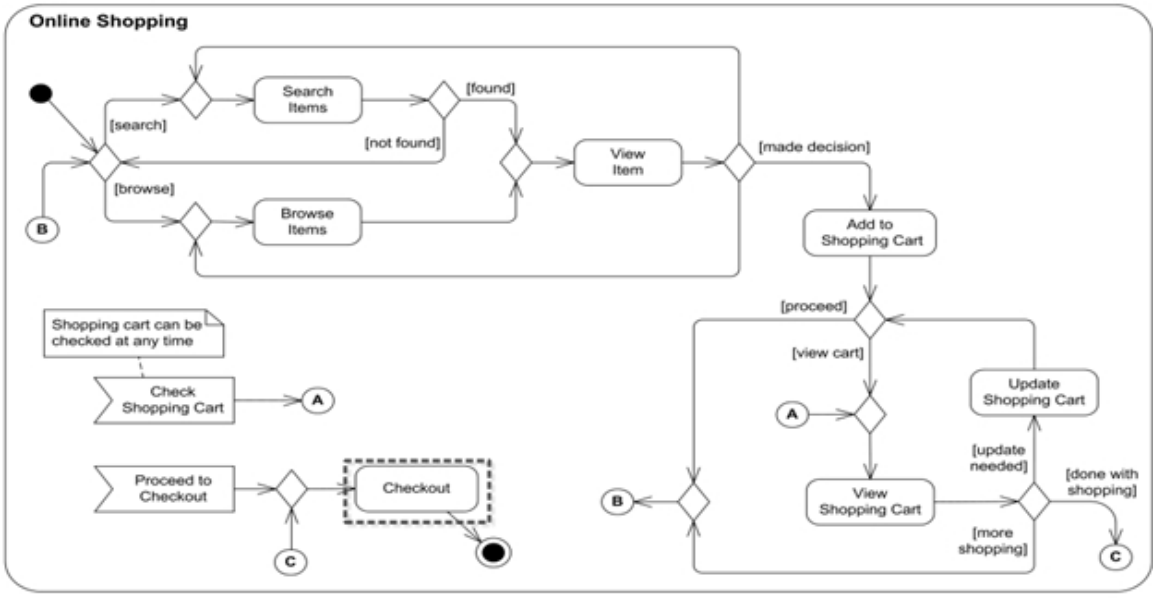

Figure 3. Online shopping with an unknown exception

2.2 Workflow Management Systems

Workflow management systems help businesses achieve high returns by allowing companies to configure their workflows according to their business processes. The goal of these systems is to coordinate the activities within a business process. A number of workflow management systems exist, including automated workflow systems. Furthermore, these systems are able to completely define the order of execution through a designed logic. Workflow management systems are composed of the following components (Shi et al., 1998):

- Workflow Model

  A series of activities and steps are programmed within a process. The workflow model, also known as the process definition, defines these activities and steps

of a process, which include workflow as well as the starting and ending points of that process. Rules are modeled within a given process to provide workflow logic.

- Activity

  Activities are the steps within a workflow. These are predefined based on the workflow model and logic. Different activities include starting events, intermediate events, and decision points.

- Process Instance

  A process instance begins when a workflow process is executed. These instances are controlled via the workflow model and the set of activity steps that control the workflow. A number of process instances can be running at the same time. Some workflow management systems provide tools to track the process instances within a workflow process.

2.2.1 Advantages of Workflow Management Systems

Workflow management systems provide a number of financial benefits. With workflow management systems, companies are able to achieve a competitive advantage, as these systems provide them with the ability to change business processes to meet the current market needs. Each company could achieve similar results using a different set of business tasks—some performed by applications, and some performed by humans. These systems provide the necessary tools to monitor processes and some systems support simulations.

With the use of monitoring systems, companies are able to reduce business process inefficiencies.

2.2.2 Disadvantages of Workflow Management Systems

Although workflow management systems reduce costs to companies, they are still regarded as high maintenance systems, as business processes continue to become complex, involving deadlines, schedules, and various constraints. Workflow management systems require trained professionals that are aware of the underlying design of the workflow, to modify existing processes as needed. End users are not generally knowledgeable about the underlying processes. Most workflow technologies have been in use for a long time and involve a number of processes; large companies that use these workflows are resistant to change processes that have been working and generating results. Security is also an issue, since in these workflow systems, specific users are given access to specific areas of an application.

The introduction of automated workflows has resulted in companies modifying sub processes within automated processes. As a result, certain sections of the workflow are manually executed, while others are automatically executed. These two methods are executed separately. If a process is halted or incomplete, there is no definitive method to track a process instance to complete it. Consequently, one or more active process instances may inadvertently be created for the same task. Process instances with unknown

exceptions require human intervention to resolve the issue. However, workflow management systems do not provide adequate tools for humans to intervene in a running process instance (Patnaik, 2011).

2.3 Web Services

Web services are designed to support interoperable machine-to-machine communication over the web (Booth et al., 2004). Web services facilitate and provide a standardized method of applications interoperability over the Internet. Open standards, such as Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL), and Universal Description Discovery and Integration (UDDI), are used to support these services by providing an interface between underlying networking protocols and applications. Web services have become popular within organizations because they allow communication between different systems, without knowing the underlying details of each system (i.e. hardware components, operating systems, lower layer network protocols being supported, etc.).

Web services communicate with other services using XML formatted messages, and consequently, there is no need for Graphical User Interfaces (GUIs) for machine-to-machine communication. There are two major classes of web services, REST-Compliant and SOAP. REST web services are "stateless" services, in which each URL represents an object and content is retrieved using HTTP operations, such as GET. SOAP is mainly used

for exchanging messages between enterprise applications (Potti et al., 2012). SOAP uses XML for its message format, and messages are sent via envelopes that define the message content (Potti, et al., 2012).

Web services can be employed to access applications that are used to execute tasks within a workflow process. Services can be used to access applications that span across domains regardless of platform and language. The addition of web services to a workflow helps to improve its flexibility. Web services can be invoked externally, and there is no dependence on the underlying technologies. Once invoked, the task results are then passed on to the next step of the workflow process, which continues running based on the generated results.

A service within a workflow can be easily reused, as it is built with standard interface descriptions. Thus, the same service can be used for different workflow management systems. Programming languages such as WS-BPEL provide definitions to specify the interactions between the multiple services involved within a workflow process. WS-BPEL is supported by workflow engines that describe the semantics, service interactions, and service invocations of the process.

2.4 WS-BPEL

Web Services Business Process Execution Language (WS-BPEL) is an OASIS (Organization for the Advancement of Structured Information Standards) standard

executable language designed to specify the coordination of multiple services for executing various tasks within a workflow (Jordan & Evdemon, 2007). It is built on top of XML and other web service standards and supports both executable processes that allow business interactions and abstract business processes that are not executable. Within XML, it is defined as a process element that encompasses other elements, such as the partner link, which specifies roles and messages communicated between different connections. The relevant logic sequences are defined under the activities section, which includes sequences that are used to execute different activities and operations (e.g., receive, reply, flow, switch, while, invoke, and assign), and which supports the use of XPath functions. Compensation is also supported within BPEL.

Business protocols are defined within BPEL as sequences of activities that specify the interfaces exposed by each partner. The definition of business protocols is a necessity within business processes, as different companies working to fulfill different orders need a defined protocol to follow. Process definitions within BPEL are used to define the protocols needed by both the business and customer. This makes it easier for different parties to interact together via web services. External services can be invoked as web service calls within a BPEL process. It supports both abstract processes that handle only protocol relevant data and executable processes that are used during runtime. Companies like Oracle and SAP use BPEL on a large scale, since it is platform independent and it provides the automation of processes as services (Louridas, 2008).

2.4.1 Fault Handling

During the execution of a business process, fault handling can be used to alter its course, depending upon the fault. BPEL specifications provide fault handling provisions via its Fault Handlers element. Figure 4 shows the syntax for handling faults. In this figure, fault handlers is the root element in which faults to be handled are specified using the catch and catch-all elements. A catch element can be used to specify a particular fault condition and actions to be taken. For every known exception, a separate catch element should be defined. The catch-all option is designed to catch any fault that is not caught by a specific catch handler or any unexpected faults that are not defined within the catch elements.

```
<faultHandlers>?
<!-- Note: There must be at least one faultHandler -->
   <catch faultName="QName"?
          faultVariable="BPELVariableName"?
          ( faultMessageType="QName" |
            faultElement="QName" )? >*
       activity
    </catch>
    <catchAll>?
       activity
    </catchAll>
</faultHandlers>
```

Figure 4. Fault handler syntax

2.4.2 Shortcomings of BPEL

BPEL processes are based on the concept of system to system communication. Although BPEL provides a standardized way to model the interactions between processes that are bound by interactions between services, it does not take into account the involvement of human activities in the process. This leads to a huge gap for many real-world business processes, as they involve human interactions along with service interactions. To fill this gap, BPEL4People extends the orchestration of web services that BPEL offers to include the orchestration of role-based human activities as well.

When business processes involve human activities, web service support for workflows becomes a major hurdle. These processes involve predefined assignment tools in which tasks are automatically routed to the next role, regardless of any changes that may arise, such as changes to organizational structures. When changes occur in these processes, they involve the redesign of the entire business task assignment. To address these limitations, BPEL4People and WS-HumanTask were introduced to execute business processes with the involvement of human tasks (Holanda et al., 2010). With these specifications, work items can be assigned to humans, as opposed to web services.

2.5 BPEL4People

BPEL4People is comprised of the WS-BPEL extension for people and WS-HumanTask. The WS-BPEL extension for people supports the role-based interactions of people, provides a means to assign users to generic human roles, and reassigns tasks to owners (BPEL4People, 2010). BPEL4People introduces an element that addresses human interactions within a business process. It is developed as an extension of BPEL, thus providing the ease of compatibility with BPEL features as well as the development of further extensions.

WS-HumanTask defines standalone human tasks as services "implemented" by people. (WS-HumanTask, 2012). WS-HumanTask facilitates the integration of human interactions in service-oriented applications. These tasks involve a number of roles, including task initiators, stakeholders, owners, and administrators. A typical human task scenario would include a task initiator beginning a process instance that is sent to the owners, who, depending on the task, can complete and forward it or wait for the correct stakeholder to take action on the task. Task administrators may be able to observe the task progress and possibly ensure that tasks are assigned to potential task owners.

The involvement of human task invocation with interoperable services increases flexibility in workflow management systems. A workflow management system involving BPEL4People and WS-HumanTask would provide a better tracking tool composed of

interrelated tasks that are both human and machine generated, as opposed to the tasks being completely separate from the workflow tool.

2.5.1 Supporting Scenarios

As an extension to BPEL, BPEL4People is used to support human activities; these activities are defined using scenarios. The scenarios are used to formalize the interaction between people, which can be between two or more people. The following scenarios are supported by BPEL4People:

- Four-Eyes Principle

  The four-eyes principle is a scenario in which a decision is made by two or more people independent of one another. This is also known as the "separation of duties principle" by which, in extreme cases, users are not allowed to know the other members involved in the decision-making process.

- Escalations

  Escalations occur whenever a task is not completed within the allotted time. This Scenario Could occur for a number of reasons, such as the person not being available; in this scenario, a notification is sent to escalation recipients, who are required to define actions that would bring the process back on track.

- Nominations

Nominations refer to the actual assignment of tasks. For example, a supervisor could decide to assign a task to different individuals based on their availability or expertise. Or a nomination could be for a specific use within a large number of users.

- Chained Execution

  Chained execution is a process that requires a user to perform a sequence of inter-related steps. The person performing the task is also the process initiator. These steps may be determined during the actual execution of these tasks.

2.5.2 Interactions Between a Process and Task

The interaction within a business process is one of the main element in a BPEL process. Figure 5 shows different ways interactions can be achieved within BPEL (BPEL4People, 2010). Human tasks within a process are divided into inline, standalone, and remote tasks. In the first model, the human task is specified inline within the people activity as a part of the BPEL process and limited to that activity. The difference in the second model is that task specification is not limited by the activity; thus, it can be shared with multiple activities. This is an advantage in terms of reusability.

In the third model, the task from the BPEL activity is external but located within the same environment. In this case, tasks can be initiated using BPEL process engines. In the fourth model, the task is from a different environment, which is accessed via a web service call. WS-HumanTask is used to communicate between the process and task; it supports human

task state changes. The difference between the fourth and the fifth models is the invocation method; remote services can be either a human activity via its BPEL4People extension for people, or automatically invoked via a BPEL activity.

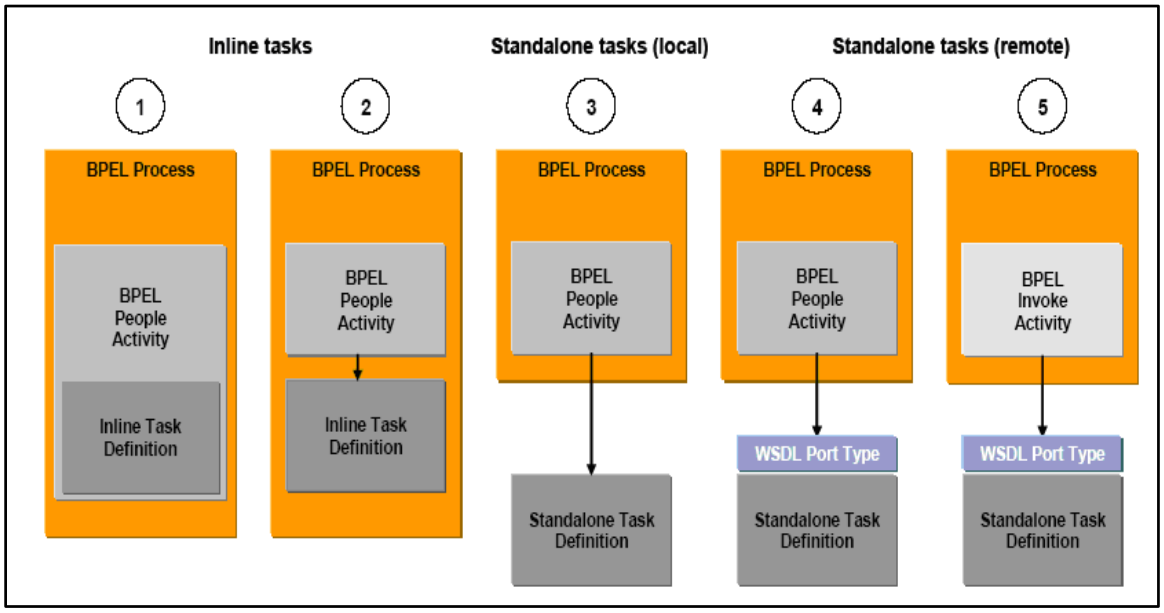

Figure 5. Interactions between process and task

2.6 Motivations for Handling Unknown Exceptions Using BPEL4People

Automated business processes have helped companies save money and time, but they still come with challenges. Although systems today have been equipped with more "smart" technologies, such as fully automated warehouses where robots can pull inventory on demand, automated call systems, and fraud detection systems that allow production companies to run fully automated business processes, these systems are still limited in cases in which exceptions or deviations occur in the normal process.

In fully automated processes, when a business exception occurs and there are no predetermined activities that can be executed to remedy the exception, workflow systems fail to sync without any real humans to offer a solution within the process. The ability to have a human resolve the unforeseen exceptions that occur within a process instance can greatly impact the completion of business processes, in that it provides the flexibility needed to modify a business process instance. Requirements within a process are mostly dependent on the circumstance of that particular instance, and processes need to take into consideration, changes that may arise unexpectedly. BPEL4People facilitates the integration of human roles into WS-BPEL automated business processes (BPEL4People, 2010), which allows business to effectively address the possibility of unexpected changes. With BPEL4People, humans can be involved in the execution of certain tasks in a process. A workflow process that involves BPEL4People can automate processes performed by people through a central system that would assign different tasks, validate results, and send the results for approval.

BPEL4People supports the role-based interaction of people (BPEL4People, 2010), whereby users can be assigned specific roles, and ensures that only one person takes ownership of a task. WS-HumanTask, as a web service standard that works with BPEL4People, is used to track the task list and task assignments to people; moreover, WS-HumanTask, provides the details of what happens to a task when it is handled by a user, which allows for greater control over what happens to the process. A number of current business processes do not have sufficient tracking capabilities; in other words, once a

process begins, they become ad-hoc processes, and there is absolutely no control over what happens to the process. Unexpected exceptions within these processes are not handled.

BPEL4People and WS-Human Task can be used together to introduce a human element into an existing process instance, which allows for human intervention when an unexpected exception occurs in a process. Specifically, when an unexpected exception occurs, that process instance can be assigned to a human using capabilities provided by BPEL4People and WS-HumanTask. Necessary information about the process instance can be provided to the human via graphical user interfaces. The human can then take appropriate action to resolve the exception. Figure 6 provides an overview of the approach on handling unknown exceptions using BPEL4People.

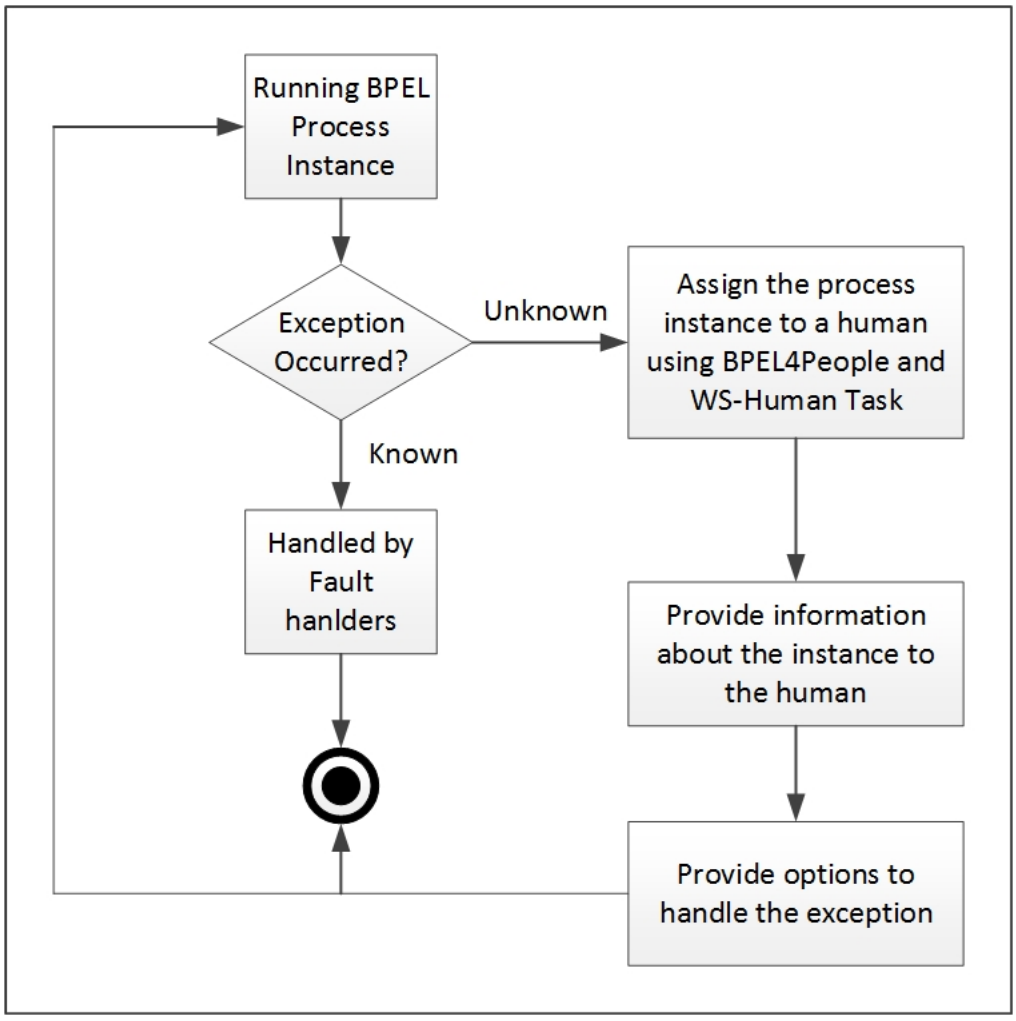

Figure 6. Overview of the approach to handle unknown exceptions

2.7 Intalio

With the need for business process management increasing in organizations, the need for BPM tools is also increasing. Open source products have become popular, as users of these products are able to expand the different functionalities. There are two major open source BPM tools widely used today: JBoss JBPM (jBPM, 2013) and Intalio BPMS (Intalio-BPMS, 2013). Table 1 shows a comparison of these two open source tools (Nie, Seppälä,

& Hafrén, 2010). The results presented using a scale: 1 (does not meet expectations), 2 (meets expectations), and 3 (exceeds expectations) (Nie, et al., 2010). Intalio BPMS is the only open source tool that supports BPEL4PEOPLE, which is an essential aspect of this thesis. Therefore, we decided to use Intalio BPMS to develop and implement the tool for handling unknown exceptions.

Criteria	JBoss JBPM	Intalio BPM
Process Modeling	2	3
Workspace	2	2
Process Administration	2	2
Business Rules Management	2	2
Business Activity Monitoring	1	1
Process Engine	2	3
Process Repository	1	1
Resource Management	1	2
Connectivity	3	2
BPEL4PEOPLE SUPPORT	NO	YES

Table 1. Comparison of JBoss and Intalio BPM tools

Intalio is an open source business process management tool that is designed to support BPEL4People transactions through the use of Intalio Tempo Components, as shown in Figure 7. A User Business Process (UBP) component is used to make web service calls to create and complete tasks. When a task is created, a web service call is made to the Form Dispatcher Service (FDS), which acts as a proxy by routing the request to the Task Management Process (TMP), a BPEL process that manages the task lifecycle; through TMP, a task is created by calling the Task Manager Service (TMS) (Gerber, 2006).

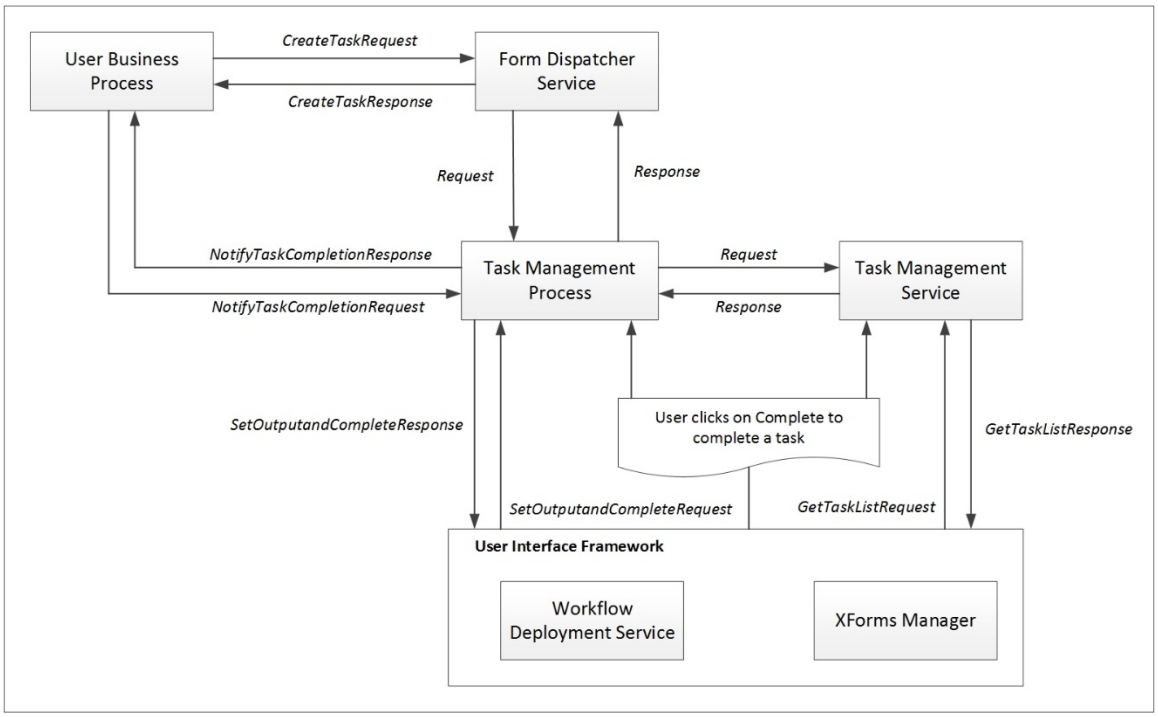

Figure 7. Tempo component interactions in Intalio

The TMP is a BPEL process that manages the task lifecycle once a request has been routed to the TMP. The User Interface Framework (UI-FW) makes a call to the Task Management Service whenever a user logs in or refreshes the task list in order to update it. XForms in the application is managed via the XForm Manager (XFM), while the Workflow Deployment Service (WDS) is responsible for storing and providing access to these forms (Gerber, 2006). The Intalio Tempo workflow framework offers flexibility due to the different components mentioned above, which can be replaced or modified as needed. The workflow logic and persistence layers can be accessed through a web service interface.

2.8 Review of Prior Works on Handling Unknown Exceptions

The METEOR workflow management model uses a knowledge-based approach to managing exceptions by enabling survivability in workflow management systems (Cardoso et al., 2001). A case-based logic is used to improve workflow exception handling capabilities. Humans intervene in this process when exception handling techniques are not established within the workflow. This model uses a 4-tier architecture in which the handling of the exceptions is based on a collection of compensation schemes. Different scenarios are explored and added to this knowledge-based tool. Similar patterns are reviewed based on exception similarity. The 4-tier architecture involves different modules for handling exceptions within the process, such as the adaptation module that uses case-based reasoning algorithms to identify points of exceptions within a process. This prior research is mainly targeted to create systems that could handle exceptions through adaptation; it involves human interaction as part of the adaptation process to rework compensation schemes identified by the system. Although this is a good solution as it lets the system learn through scenarios, it raises issues when requirements or needs change. The system would have to learn and adapt to new changes or may find solutions based on previous knowledge.

Mourão and Antunes proposed a novel architectural framework to effectively handle unknown exceptions in workflow management systems (Mourão & Antunes, 2007). They developed a framework based on the Open Symphony open source platform that provides basic workflow functionality. They relied on Java Short Message Service (SMS) for

communication, in which users would be notified by messages with details of any exceptions. Users could then edit exception details, or they could choose to collaborate with users involved in the process itself to handle those exception details. Errors were handled by following a fixed five-step process that guided users with regard to handling exceptions. Our tool accounts for any unexpected errors that cause systems to fail and provides options to change the process instance to its final state.

Chiu, Li, and Karlapalem developed a system to handle unknown exceptions through the use of a human intervention manager based on a specific algorithm that allows users to choose new paths (Dickson K. W. Chiu, Li, & Karlapalem, 2001). However, our system is more flexible, and human actions are invoked via web service calls that are separate from the workflow process composition.

Han, Thiery, and Song provided an analysis of exceptions on medical workflows and demonstrated the importance of research on exception handling (Han, Thiery, & Song, 2006). Their analysis determined that managing exceptions is essential because it provides a number of different exception categories. The paper gives an overview of exception handling methods but does not provide a viable tool to handle those exceptions. It aids in understanding the various types of exceptions that could occur and actions that should be taken to correct them.

Chapter 3

RESEARCH OBJECTIVE AND METHODOLOGY

3.1 Research Objective

Several factors can cause exceptions within a business process, particularly unknown exceptions for which no handling tools are specified, and which would halt the process or keep it in an unknown status. In many cases, the solution relies on restarting a new process instance or restarting the process, which would result in a number of processes running at the same time. In this thesis, we developed a tool to improve the handling of unknown exceptions within a workflow system by employing BPEL4People and WS-HumanTask specifications along with a module that interacts with Intalio and provides a necessary interface for human interactions.

The objective in this thesis was to develop a tool to handle unknown exceptions. The tool will provide a user interface to handle unknown exceptions utilizing a sub process modeled within the process. This exception handling will be designed by implementing a web service, which will allow humans to intervene in process instances when unexpected exceptions occur. Humans will thus be able to reroute processes.

3.2 Research Methodology

For this thesis, we employed the design science research methodology (Hevner et al., 2004), which provides a set of guidelines to conduct research on the creation and evaluation of information systems artifacts intended to solve an organizational problem. The design science research methodology, widely accepted within the information systems research community, includes the following phases: problem identification and motivation, objectives of a solution, design and development, demonstration and evaluation, and communication (Peffers et al., 2007). Table 2 highlights relevant thesis chapter sections characterized by each phase of the design science research methodology.

In applying the design science research methodology, for the first phase, we identified the issue of the handling of unknown exceptions within workflow processes. In identifying solution phase, we explored alternatives to solve the problem. We decided to develop a tool that uses BPEL4People and WS-HumanTask specifications to handle unknown exceptions. In the design and development phase, we developed the unknown exception handling tool with the use of Intalio and its Tempo sources. The tool is developed with a user interface that provides options for a task owner to take action on unhandled tasks. In the demonstration and evaluation phase, we evaluated the tool using different scenarios to demonstrate its ability to handle unknown exceptions.

As a part of the evaluation, different process scenarios were executed with and without the tool to demonstrate the effectiveness of the added capabilities within a business process. For the communication phase, which involves the presentation of the designed artifact and the publication of the results of the research, we will demonstrate the utility of the tool during the thesis defense and by describing the contributions of the research in the thesis document.

Design Science Research Phases	Phase Description	Relevant Chapter Sections
Problem Identification and Motivation	The focus of this phase is on problem identification and on the justification of the value of the solution.	In Chapter 1, we provide details of the specific problem addressed and the motivation and value of the research.
Identifying a Solution	This phase gives perspectives on how to solve the identified problem.	In Section 4.1, we provide a conceptual overview that identifies our solution based on functionalities currently available.
Design and Development	A design artifact is developed to solve the problem.	In Section 4.2, we provide an architecture that shows the framework of our design and a conceptual view of our solution.
Demonstration and Evaluation	The designed solution is evaluated to prove that the tool solves the identified problem.	In Chapter 5, we use scenario-based evaluation to demonstrate the utility of the tool. We demonstrate how the tool can be used to handle unknown exceptions and bring process instances to meaningful states.
Communication	This solution is displayed to relevant audiences.	The problem is identified, the solution is developed, and the evaluation results are then communicated through the thesis document and through presentations.

Table 2. Design Science Research Methodology Phases

Chapter 4

EXCEPTION HANDLING TOOL

4.1 Overview of the Tool

In this section, we provide an overview of how BPEL4People and WS-HumanTask is used to provide options for handling unknown exceptions. As fault handling is a complex element in a business process, we also introduce a fault handling tool that uses BPEL4People scenarios as a formal method to incorporate human interaction into a business process when an exception occurs. BPEL4People supports the involvement of people in activities, escalations, and notifications. Human Interactions within a business process can be modeled using the following scenarios, which are supported within BPEL4People: four eyes principle, nomination, escalation, and chained execution.

Our solution tool will implement an escalation scenario that would allow processes with unhandled exceptions to be escalated to the task owner. A task owner can perform administrative actions on the task instances. These task owners would then have the option to complete these instances. For this thesis, the focus is on the escalation scenario, since it involves the reassigning of tasks to different users. Our tool would allow task owners access to these task instances and would invoke and impose appropriate actions on these tasks. We did not use the other scenarios in our solution,

as our focus was on including human roles as secondary users within running process instances. Escalations were the best choice, therefore, as they involve triggering actions on tasks that have not been executed within a set deadline.

Escalation scenarios are handled by task owners that have been specified to handle process instances once escalations have been triggered. One example of an escalation trigger is an incomplete task, and one of the actions supported in handling escalations is the reassignment of tasks to task owners. Escalations can be triggered based on deadlines as explained in the following scenario. A deadline element is used to specify deadlines within a task definition. Its start deadline defines the time by which a process must reach an in-progress status. Its completion deadline is measured from the time a process starts or is created to when it reaches one of the final states (Completed, Failed, Error). If the task has not been performed by the time the completion deadline is reached, then an escalation is triggered.

The central function of our tool is to use this escalation scenario and WS-HumanTask operations to allow a task owner to intervene and handle unknown exceptions. For the purpose of understanding our tool, let us consider a generic business process with a known exception. A known exception is handled using fault handlers specified within a BPEL process. In Figure 8, this is depicted by the innermost fault handler, wherein checkpoint gateway activity throws a fault, which is subsequently handled by a catch handler. Within the catch handler, Task A would specify how the exception should be resolved. When an unknown exception occurs, it would halt the process instance, as

there are no fault handlers capable of resolving the exception. In order to handle these exceptions, we would need to add a catch-all fault handler for the entire BPEL process. This catch-all will catch any exceptions that have not been handled by any of the fault handlers specified within the BPEL process. In Figure 8, the outermost fault handler depicts the Catch-All element used to catch all undefined faults.

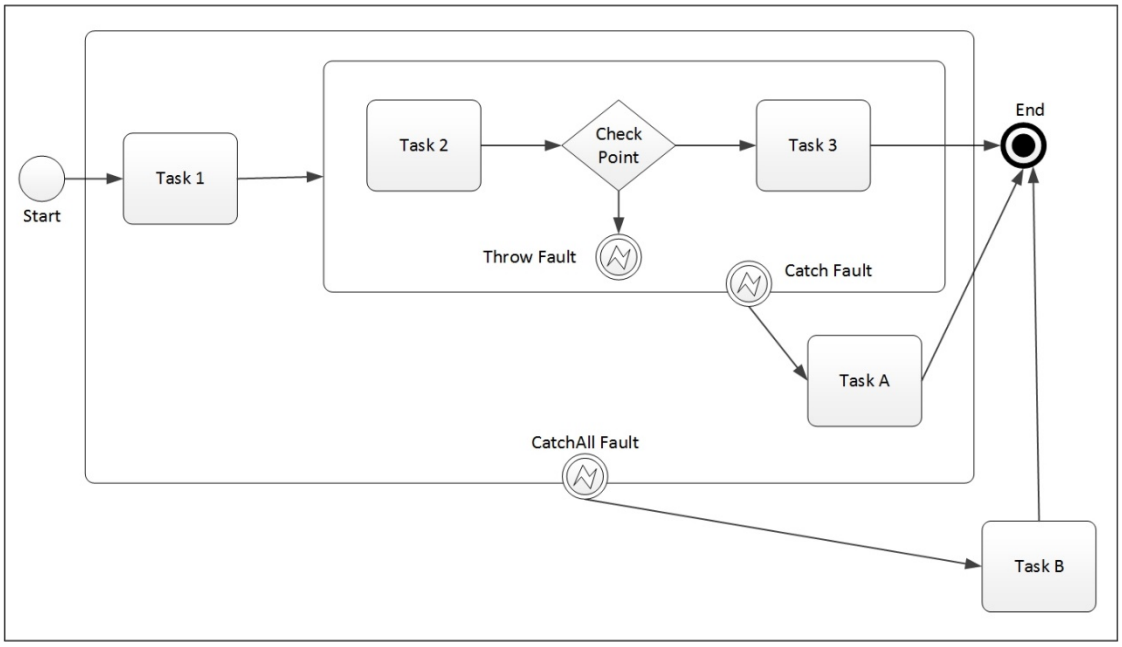

Figure 8. Framework overview

Within Task B, we utilize the BPEL4PEOPLE escalation element. Within the escalation element, we specify a completion deadline and a business administration user role to designate a human to act on the process instance. This escalation action allows us to associate a process instance with an authorized task owner. The task owner will be provided with following three options to handle the unknown exception: "complete," "reassign," and "terminate." These options are developed based on a set of WS-

HumanTask operations that pertain to actions that can be executed by people in an intermittent process context. Table 3 provides a description of the options.

Option	Description
Complete	Completes a task; the process is set to complete.
Reassign	Reassign a task to a different task owner.
Terminate	Task instance processing is stopped.

Table 3. Exception Handling Options

4.2 Implementation of the Exception Handling Tool

We developed our tool using Intalio BPM, henceforth referred to as Intalio. Intalio provides a web service that handles task workflows. Intalio uses a WS-BPEL 2.0 standard-compliant BPM process engine, which can be deployed on an Apache Axis2 web service server. The process engine provides a platform to manage business processes, and Axis2 provides a web service platform to run service applications. Intalio also provides other services, such as a task management service, which manages the overall lifecycle of tasks including the initiation and completion of tasks, a process management service for the management of processes, and token services, which are used for the authentication of users within Intalio. Figure 9 provides an overview of our tool's components and the Intalio services used for the handling of unknown exceptions.

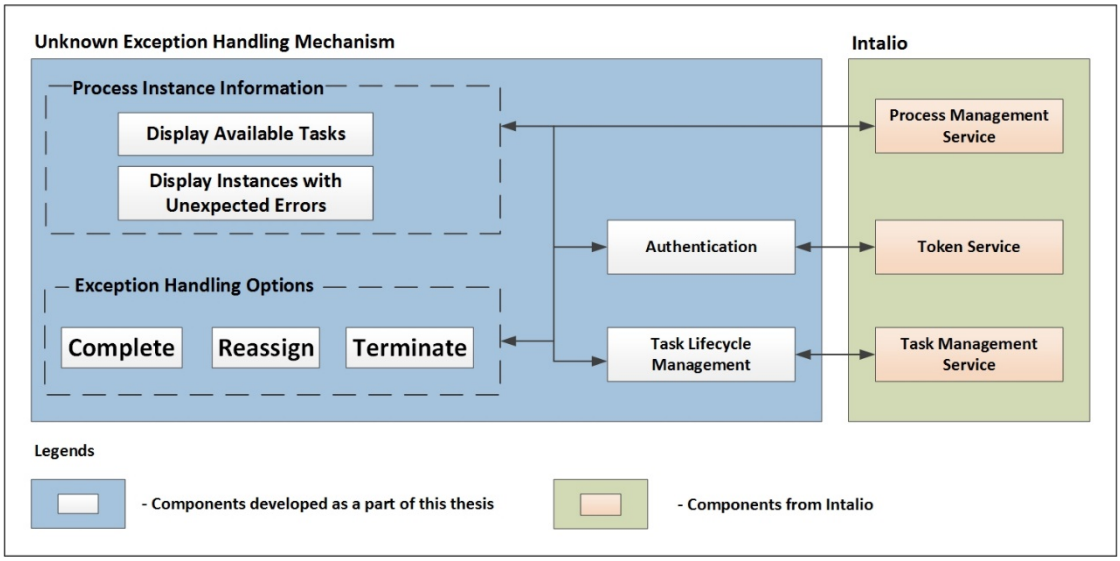

Figure 9. Framework detail overview

4.2.1 Steps for Handling Unknown Exceptions Using the Tool

When a BPEL4PEOPLE user with a task owner's role logs into the tool for handling unknown exceptions, the user's login information is authenticated using the token service, and a user token is assigned to that user's role. After successful authentication, the tool displays the process instance information options. Available tasks can be displayed by making a request to Intalio's task management service via a task query operation. This operation is invoked as a result of a catch-all event within a business process. The process instance and the task metadata are then displayed to the user. Metadata information will be obtained using the process management service and will be used to identify the specific instance being handled.

Depending on the process instance and the exception context, the user can make use of the available set of exception handling options to resolve the exception. The exception handling options are operationalized by invoking the appropriate task management service operations. These operations are completed by approval or denial, by reassigning and then completing by approval or denial, and by terminating the process instance. All of these actions produce a meaningful end result for the user.

4.2.2 Process Modeling Using Intalio BPM

Intalio BPM offers the Intalio Designer tool, which allows users to model business process instances using Xform technology. With this format, a user is able to create a functional business process by starting with a business process diagram. These business process diagrams allow for the addition of pools, which contain lanes that represent the separation of activities, such as between system activities and human activities. A business process designer can then add process events to these pools, including start, intermediary, stop, and error events.

Creating a business process that involves humans as process initiators involves using an Xform that specifies a human task event as the initiator. Process tasks that require human intervention are grouped into non-executable pools. By contrast, executable pools are pools with tasks that the business process engine would automatically handle.

For example, system events are grouped into executable pools. For this thesis, all of the above-named process events were used.

Chapter 5

EVALUATION

5.1 Evaluation Objective

As per the design science research methodology, it is necessary to evaluate the proposed Exception Handling Tool to demonstrate its utility. The objective of the evaluation is to show how the proposed tool aids in handling unknown exceptions. The proposed tool relies on human interactions within business process instances to handle unknown exceptions. We exhibit the utility of the proposed tool by demonstrating that it can handle process instances with unknown exceptions and bring them to meaningful states (e.g., completed, terminated, and reassigned).

The proposed tool accounts for the following statuses within business processes, any of which result after a process instance is initiated: "in progress," "failed," "terminated," and "completed." A changed process instance status can occur due to a number of factors within a business process, such as an unhandled exception, a known exception, or a completed process instance. The evaluation of the tool will be based on its ability to change the course of events of a process instance containing an unhandled exception. The utility will be tested in different scenarios to initiate changes in the status of different business process instances. The business process instance status would then

be changed by users who are able to act on a process instance that develops an unknown error. We aim to demonstrate the utility of the tool by achieving the above-stated objective.

5.2 Scenario-Based Evaluation

A scenario-based evaluation approach can be used to assess the capability of a software implementation. Capability is defined "as the ability to achieve an effect" (Looker et al., 2008). With regard to the Exception Handling Tool, the capability is, therefore, the ability for human users to handle unknown exceptions. Capability of software implementation is evaluated using scenarios that describe situations likely to occur during operation.

Scenarios are generic descriptions of interactions between users and the software system in the context of daily activity (Looker, et al., 2008). With regard to the exception handling tool, a scenario would describe the interactions between activities within a process. A typical scenario could be characterized with the following: "initial assumption," "normal," "what could go wrong," "other activities," and "system state on completion." In a normal process, exception handlers would be defined to catch errors within the process. If any other error occurs within the process that is not defined, it could then cause unexpected events. The process could halt, fail, or remain in an ad-hoc status; therefore, the final state within the process would become undetermined.

In order to operationalize the scenario-based evaluation, we have adapted a modified version of the Scenario Walkthrough and Inspection Methods (SWIMs) approach to evaluation (Haynes, Purao, & Skattebo, 2009). SWIMs was originally proposed to evaluate collaborative systems by using scenarios and user reflections. We have modified SWIMs to fit the context of this evaluation while ensuring that the goals of key stages of SWIMs are achieved. The key adaptation that we have incorporated is that instead of using real user interactions and experiences to conduct an inspection of the system, we relied on self-reflection to inspect the system.

The SWIMs evaluation method has the following four key stages (Haynes, et al., 2009): (1) building a priori contextual appreciation, (2) eliciting current and envisioned scenarios, (3) validating these scenarios, and (4) analyzing the outcomes of these scenarios in context with claims analysis.

5.2.1 SWIMs: Stage 1

The goal of the first stage of SWIMs is to establish an a priori understanding of the contextual information relevant to the articulated components in the scenarios used to inspect the system. The original SWIMs focuses on users, user roles, processes, and system capabilities. Unlike the original SWIMs, we did not focus on users or on user roles; rather, we placed emphasis on business processes. In the first stage of this thesis, we established a priori understanding by creating a generic automated loan process. We

used the generic automated loan process to establish necessary components to articulate scenarios that would be used to evaluate the tool proposed in this thesis. The generic automated loan process is described in section 5.2.5.

5.2.2 SWIMs: Stage 2

The goal of the second stage of SWIMs is to produce scenarios that could be used to inspect systems capabilities and to identify barriers to system usability. While the focus of the original SWIMs was on using user experiences to develop the scenarios, we used the generic loan process established in the first stage. In the second stage of this study, we developed three scenarios based on the generic loan process. The first scenario was a normal automated business process, in which a loan was submitted for approval and could either automatically be approved or be automatically denied. The objective of the first scenario was to establish a baseline process that successfully completed without any exceptions.

The second scenario extended the first scenario by adding an exception handler within the business process. The exception handler was modeled to catch specific exceptions. The objective of the second scenario was to demonstrate what would happen to process instances with unknown exceptions. In cases in which an error might have occurred that had not been modeled within the exception handler tool, the error would not have been caught and the process would end in an ad-hoc state.

The third scenario extended the second scenario by adding the exception handling tool and tool implementation developed in this study. This scenario aimed to prove that with the addition of this tool, an unhandled exception could be brought to a meaningful status. As task owners, humans are the experts in the subject matter who can complete, terminate, or reassign process instances, enabling unhandled exceptions. Scenarios A, B, and C are described in sections 5.3 through 5.5.

5.2.3 SWIMs: Stage 3

The goal of the third stage of SWIMs is to produce a Scenario Catalog that comprehensively captured both the characterization of system capabilities and users' understanding of the problem solved by the system. While the original SWIMs uses focus groups to present the scenario catalog and to conduct walkthroughs of system usage to validate system capabilities, in this study we created the scenario catalog by creating several instances for each scenario that characterized the system capability and exceptions associated with the loan process. Instead of utilizing a focus group, we utilized a system walkthrough to demonstrate system capability in each identified scenario. Scenarios and the system walkthrough for each scenario process instance are described in sections 5.3 through 5.5.

5.2.4 SWIMs: Stage 4

The goal of the fourth stage of SWIMs is to capture both positive and negative claims with respect to system support for the scenarios identified in the scenario catalog. Claims reflect the results of interactions between users and system features as a consequence of scenarios. As per SWIMs, this stage is about aggregating and analyzing claims of system support for given scenarios. SWIMs categorizes contributions of the system support identified by claims analysis as follows: (a) measurable, (b) tangible, (c) intangible, and (d) unrealized. *Measurable contributions* represent user perceptions of an organizational contribution that is concrete and that can be measured quantitatively: for example, dollars saved per month or number of tasks completed. *Tangible contributions* reflect users' perceptions of a specific system benefit that cannot be easily measured, but which can be easily identified and agreed upon, such as support for better decision-making by providing additional information or by automating selected tasks to improve the overall efficiency of the process. *Intangible contributions* refer to users' perceptions of a benefit that is not tied to organizational goals, such as access to archived documents or incomplete processes. *Unrealized contributions* refer to potential contributions that have not been realized by the system, such as the lack of system support for an important organizational task. In this thesis, we will perform a claims analysis for the scenario involving intangible contributions (Scenario C) with regard to the support provided by the tool, and further discuss the four categories of contributions. Discussions on claims analysis and contributions are provided in section 5.6.

5.2.5 A typical Automated Loan Approval Process

A typical automated loan approval process (shown in Figure 11) involves the following steps, which are modeled within the process:

1. The loan applicant (user) provides loan amount information.
2. The risk associated with the loan is assessed based on the loan amount and salary.
3. Based on the risk assessment results, the loan is either approved or routed for further processing.
4. The loan is then either accepted or rejected.

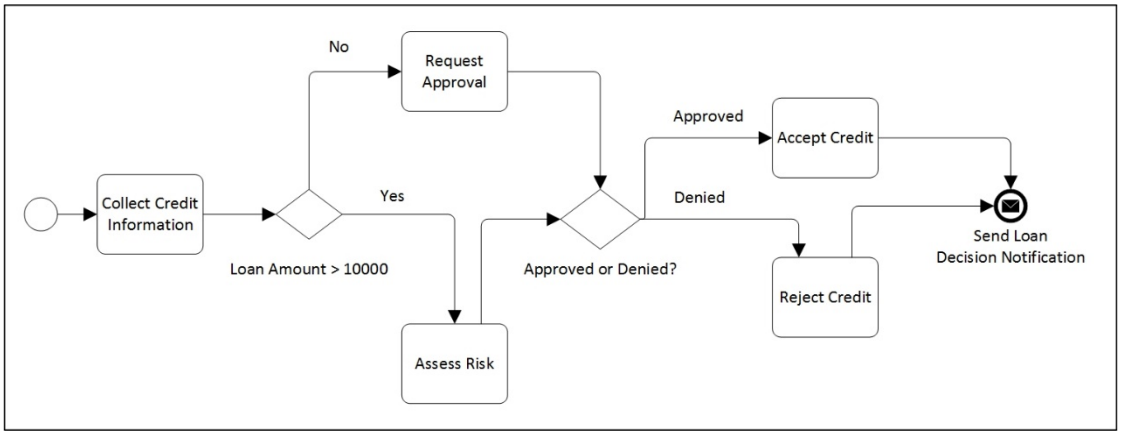

Figure 11. Loan approval process specification example in BPMN

Using the loan approval process as an example of a motivating process, we have modeled three scenarios for our evaluation purposes, based on the following factors:

- If the loan amount were less than or equal to $10,000, then the Loan Approval Service would be triggered.

- If the loan amount were greater than $10,000, the Loan Assessment Service would be triggered.

Risks associated with the loan are processed by the Loan Approval and Loan Assessment services. The Loan Approval Service would approve the loan as long as the requested loan amount were equal to or less than the salary. The Loan Assessment Service would approve the loan as long as the requested loan amount were equal to or less than 10 times the salary. It should be noted that both services will only process loan amounts greater than zero. A known exception for this process would be the value of zero for the salary or loan amount. An unknown exception would be a negative value for the salary or loan amount.

The three scenarios implemented were named Scenarios A, B, and C. Scenario A was a generic automated business process, and more functionality was added to Scenarios B and C. The scenarios below were created as an Intalio business project named "Loan Approval." The following steps were carried out to implement the process scenarios in Intalio BPM.

5.2.5.1 Steps Followed to Create Process Scenarios in Intalio BPM

The loan approval process was created using the Intalio Designer tool described in section 4.2.2. The processes were created as an Intalio Business Project called "Loan Approval" with the following steps:

1. A business diagram was created and named "Loan Approval."
2. Two pools were added named "Client" and "Process," visual elements used to distinguish responsibilities for the sub-processes of a business process. The *client pool* was set to a non-executable state and would be the pool to initiate the process; no BPEL code would be generated. The *process pool* was set to an executable state, and BPEL code would be generated, as actions would be carried out automatically.
3. The process instance would begin with the "Apply Loan Task" in the client pool.
4. A start event was then added in the process pool; this event would actually begin the process.
5. Once the process was started, it would go through a BPM Gateway, which, depending on the amount of the loan, either the loan would be routed to the Loan Approval Service or the Loan Assessment Service.
6. Based on the results of these services, a loan would either be Approved or Denied, and a notification would be sent to the applicant.

5.3 Scenario A: Automated Process Without the Tool

Scenario A represents a normal automated business process wherein a loan is submitted for approval and can either be automatically approved or denied. The objective of Scenario A was to demonstrate a baseline implementation of the typical loan process in the Intalio BPM (following the steps described in section 5.2.5.1), and was created to demonstrate how a process instance without any exceptions would behave and to determine if it would produce the intended result. Scenario A was also used to demonstrate how one can identify process instances that successfully complete without any exceptions in Intalio BPM.

Details of the process steps for Scenario A are given below:

1. The process begins with an applicant submitting a loan for approval.
2. If the loan amount is greater than $10,000, the loan is routed to the Loan Assessment Service.
3. If the loan amount is less than or equal to $10,000, the loan is automatically routed to the Loan Approval Service.
4. The user receives either loan Approved or Denied notification based on the results of the Loan Approval and Loan Assessment services.

The Scenario A process was first run with input for the loan amount of $10,000 and the salary of $10,000. The Intalio BPM showed the Process Instance Status as Completed. As per the scenario process steps (Figure 12), since the loan amount was $10,000, the

Loan Approval Service was triggered and the loan was approved, because the requested loan amount was equal to the salary. Thus, the first run of Scenario A successfully completed without any exceptions. The step-by-step actions performed in the Intalio BPM to conduct Scenario A with $10,000 inputted for the loan amount is shown in Figures 13 through 19.

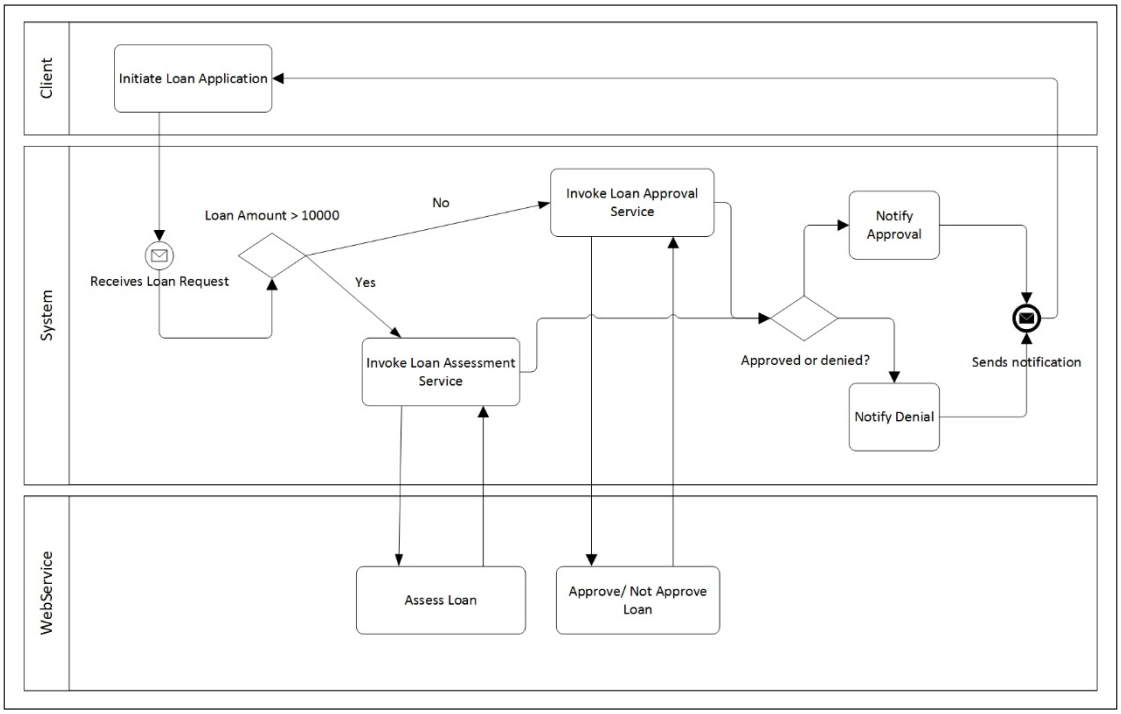

Figure 12. BPMN diagram for Scenario A

The Scenario A process was conducted for a second time with a loan amount of $20,000 and a salary of $10,000. Consequently, the Intalio BPM returned a Process Instance Status of Completed. Since the loan amount was greater than $10,000, the Loan Assessment Service was triggered and the loan was approved, because the ratio of the loan amount to the salary was less than 10.

The Scenario A process was conducted for a third time with an inputted loan amount of $110,000 and a salary of $10,000. The Intalio BPM subsequently led to a Declined Process Instance Status. As the loan amount was greater than $10,000, the Loan Assessment Service was triggered and the loan was denied, because the ratio of the loan amount to the salary was greater than 10. Table 4 provides a summary of the process instance results for Scenario A.

Scenario A Instances	Loan Amount	Salary Amount	Action	Final Status
First Instance	10,000	10,000	Approved	Completed
Second Instance	20,000	10,000	Approved	Completed
Third Instance	110,000	10,000	Denied	Completed

Table 4. Scenario A process instances results

5.3.1 Running a Process Instance in Intalio

We provide below a step-by-step description of the actions that were taken to conduct the first instance of Scenario A.

Step 1: The user logs in to the Intalio User interface, as shown in Figure 13.

Figure 13. Intalio login user interface

Step 2: After authenticating the user, Intalio displays a list of processes available in the BPM, as shown in Figure 14. The Scenario A Loan Approval process is initiated by clicking on the "InitiateLoanApp-init request process ScenarioA/ScenarioA with form InitiateLoanApp.xform" link from the list of processes displayed.

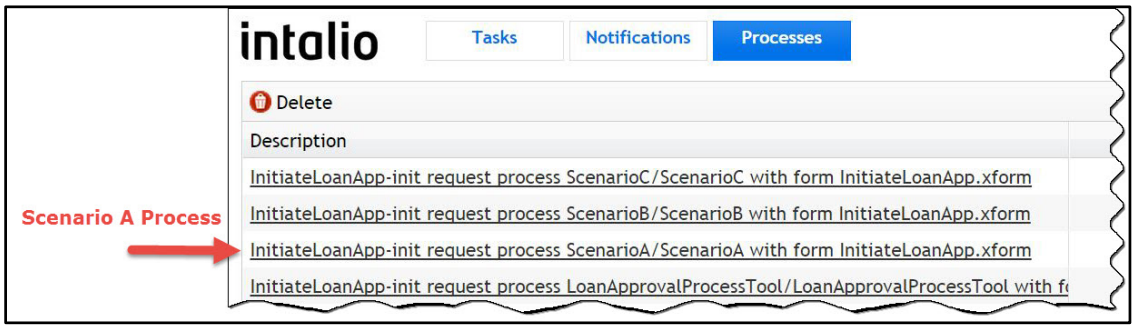

Figure 14. Intalio user interface process list

Step 3: Once the process is initiated, Intalio displays an input form for a loan request, as shown in Figure 15, in which a user can then input a name, salary, and loan amount.

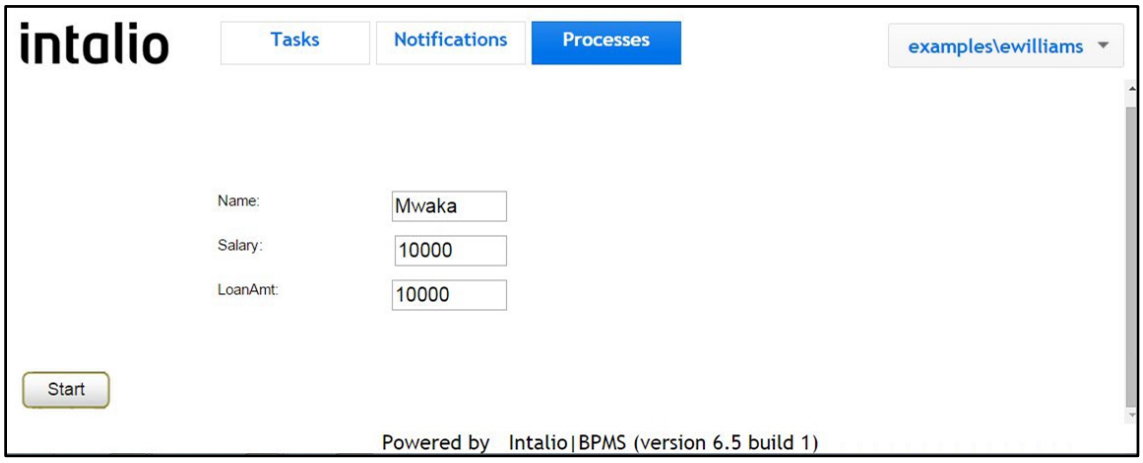

Figure 15. Intalio Scenario A loan input form

Step 4: As per the business process flow, a loan approval notice is sent to the user, and the first process instance is completed.

Figure 16. Scenario A first instance successful completion interface

Steps 1 and 2 described above also apply to the second instance of Scenario A with an input of 20,000. However, because the loan amount requested is 20,000 (above the baseline of 10,000), the Loan Assessment Service is invoked in step 3, as per the business process flow. As the ratio between the requested loan amount (20,000) and the inputted salary (10,000) is 2, the Loan Assessment Service approves the loan application. As designed within the business process, a loan approval notice is sent to the user and the process instance is completed. For this instance, the user is notified of the loan approval using an interface similar to Figure 16.

Steps 1 and 2 described above apply as well to the third instance of Scenario A with an input of 110,000, as shown in Figure 17. However, in step 3, as the loan amount requested is 110,000, the Loan Assessment Service is invoked, as per the business flow. As the ratio between the requested loan amount (110,000) and the input salary (10,000) is 11 (above the allowed ratio of 10 for approval), the Loan Assessment Service denies the loan application. As designed within the business process, a loan denial notice is provided to the user, and the process instance is completed. For the third instance, the user is notified of the loan denial using an interface similar to Figure 18.

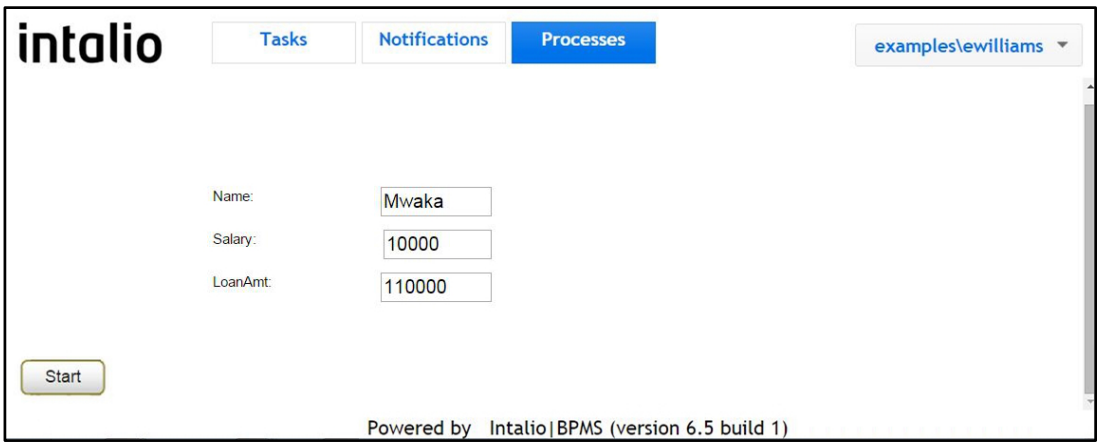

Figure 17. Intalio Scenario A user inputting 110,000 as loan amount

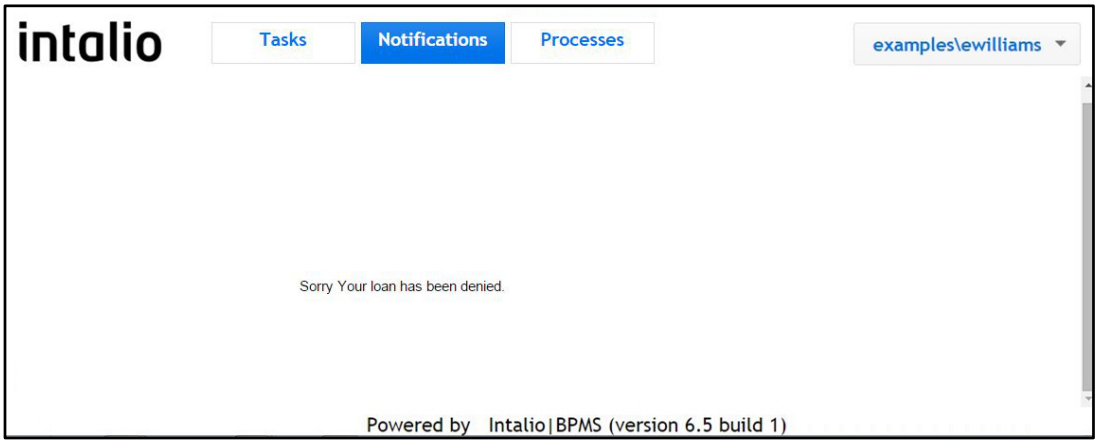

Figure 18. Scenario A third instance loan denial interface

In the Intalio BPM notification interface (this is the same screen wherein the user is notified of the loan request results, as shown in the Figures 16 and 18), at the bottom of the screen, users can see a list of process instances that have been successfully completed. Users can verify whether a process instance has successfully completed, regardless of loan process results. Figure 19 depicts the three instances of Scenario A, indicating that all three instances were successfully completed.

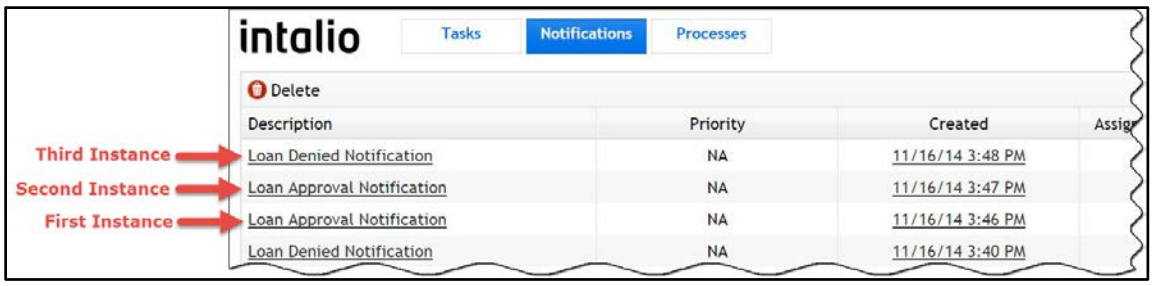

Figure 19. Intalio Scenario A instance completion list

5.4 Scenario B: Automated Process with Exception Handling

Scenario B extends Scenario A by adding an exception handler into the loan approval business process. The objective of Scenario B is to demonstrate what happens to process instances with known exceptions and to instances with unknown exceptions. Known exceptions modeled within the process would be handled by the exception handler, whereas, exceptions that are not modeled within the process would not be caught by the exception handler; thus, instances with unknown exceptions would end in an ad-hoc state. In Scenario B, entering a value of zero for the salary or loan amount would be considered a known exception. The exception handler is modeled within the Scenario B process to catch the above specified exception. The unknown exception for Scenario B would be a negative value for the salary or loan amount which, according to the model, would not be caught by any of the exception handlers.

The process steps for Scenario B are outlined below:

1. The process begins with an applicant submitting a loan for approval.

2. If the loan amount were greater than 10,000, the loan would be routed to the Loan Assessment Service.
3. If loan amount were less than or equal to 10,000, the loan would automatically be routed to the Loan Approval Service.
4. If a known error (modeled to be caught by the handler) were to occur within the Loan Approval or Loan Assessment services, then the user would be notified of the error and would be asked to reapply. If an unknown error were to occur, the process would be left in an ad-hoc status (unknown status), and the user would not receive any notifications.
5. The user would receive either a loan approved, denied, or error notification, or no response at all, based on the results of the Loan Approval and Loan Assessment services.

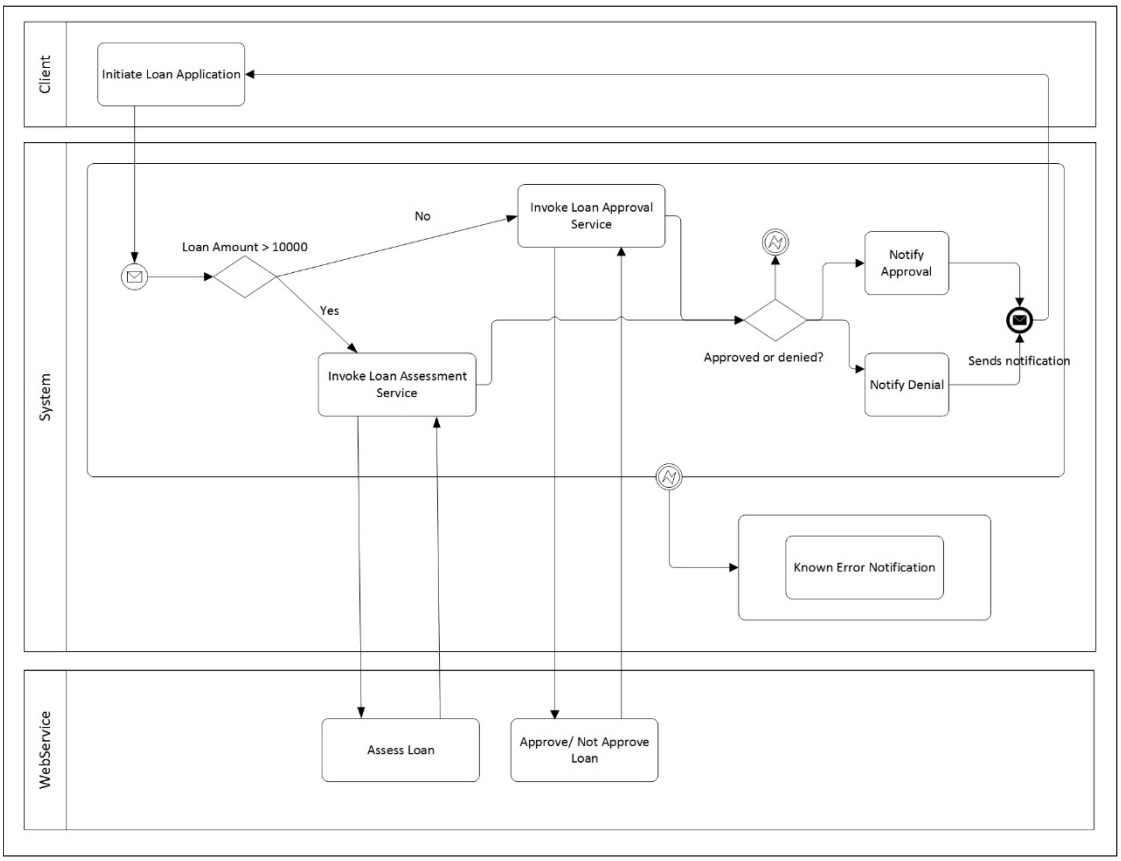

Figure 20. BPMN diagram for Scenario B

In order to demonstrate what would happen to business process instances with known and unknown exceptions, the Scenario B process shown in Figure 20 was implemented in Intalio BPM following the steps described in section 5.2.5.1. The Scenario B process was first conducted with an inputted loan amount of 10,000 to ensure that the process instance completed successfully. Scenario B was conducted a second time with an input loan amount of 110,000 to ensure that the instance would be completed with the expected result of the loan being denied. The Scenario B instances were conducted following the steps described in section 5.3.1. In step 2, the Scenario B Loan Approval

process was initiated by clicking on the "InitiateLoanApp-init request process ScenarioB/ScenarioB with form InitiateLoanApp.xform" link, as shown in Figure 21.

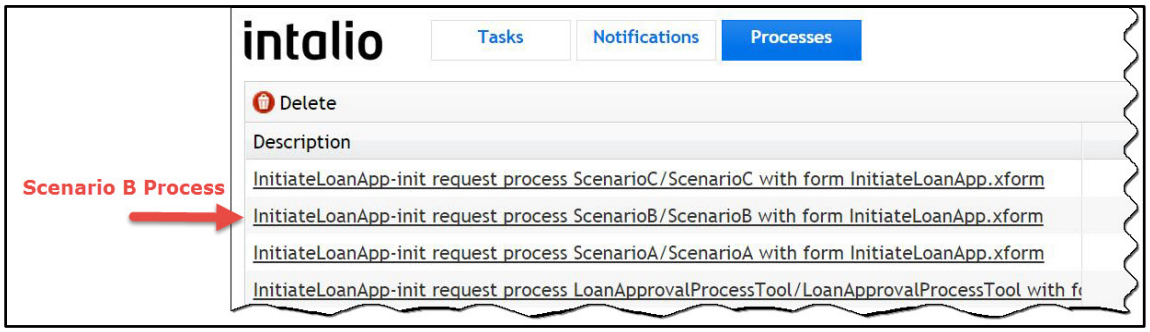

Figure 21. Intalio process list for Scenario B

For the third instance of Scenario B, a zero value was inputted for the loan amount, as shown in Figure 22.

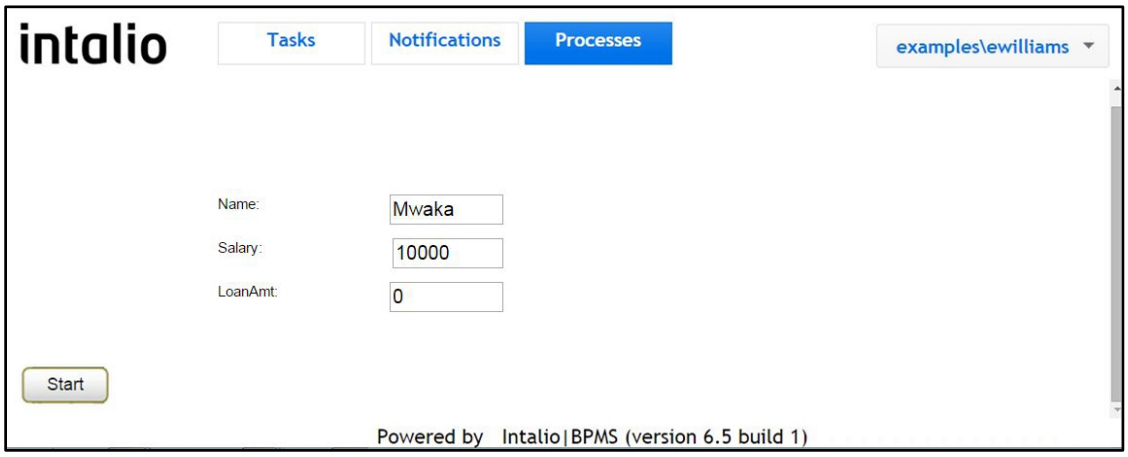

Figure 22. Intalio Scenario B user inputting zero for the loan amount

Since the loan amount was less than 10,000, the Loan Approval Service was triggered, and it found a known exception for the zero value loan amount, an error which Intalio displays for the user, as shown in Figure 23.

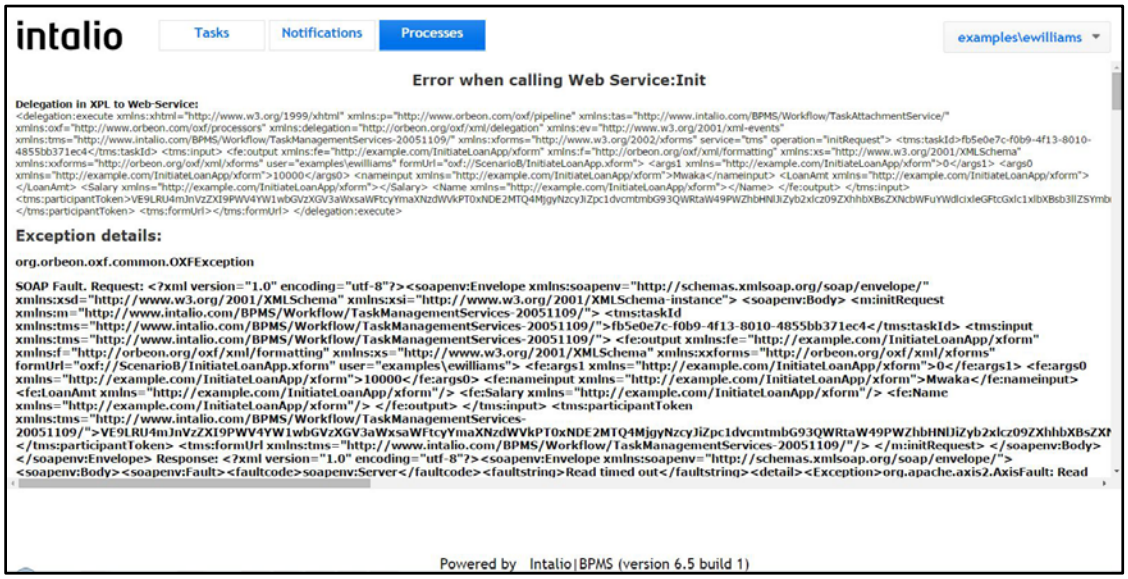

Figure 23. Intalio displays error in scenario B third instance

As this is a known exception, it was caught by the exception handler. As the error was caught by the exception handler, Intalio brought the process to completion by notifying the user, via the notifications interface to reapply for the loan (Figure 24).

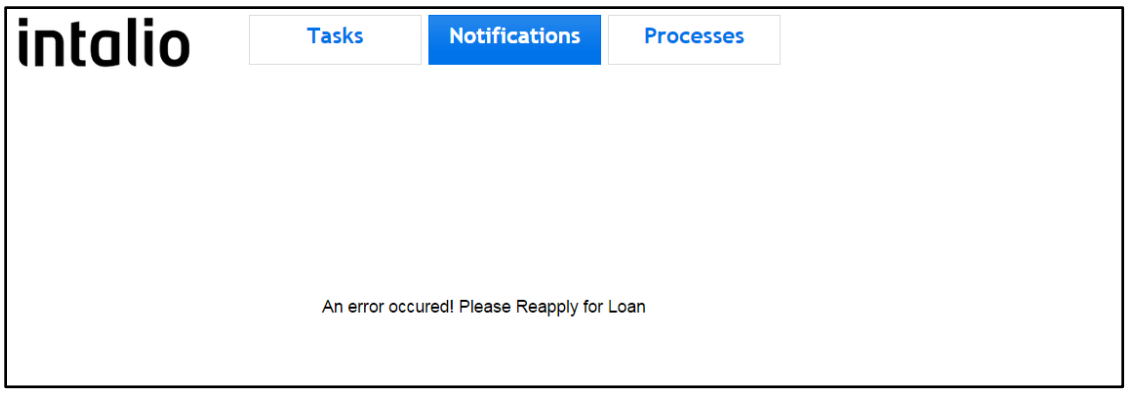

Figure 24. Intalio displays Scenario B third instance error message

Thus, the third instance of Scenario B reached a completed state. Figure 25 shows the notification indicating that the instance was successfully completed despite having an exception.

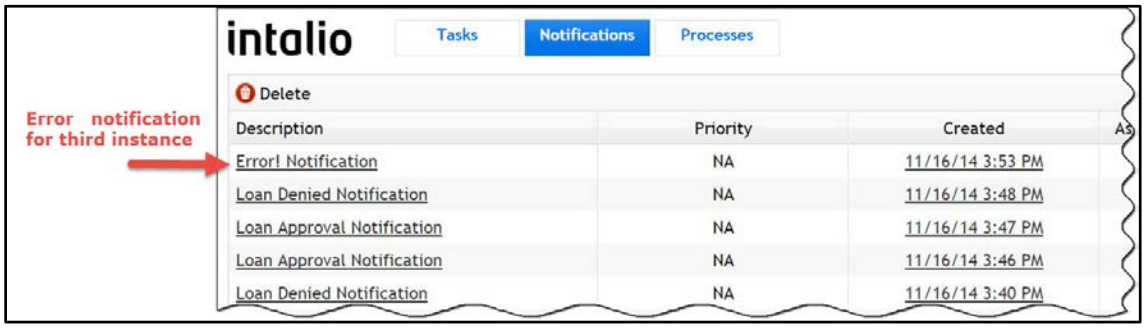

Figure 25. Scenario B third instance error notification

A fourth instance of Scenario B was created by inputting -15,000 for the loan amount, as shown in Figure 26. Since the loan amount was less than 10,000, the Loan Approval Service was triggered and an unknown exception was found, as the requested loan amount was less than zero. This error was then displayed by Intalio for the user, as shown in Figure 27. As this is an unknown exception, it was not caught by the exception

handler. Therefore, the exception was not handled by the process, and the instance acquired an ad hoc status, due to an impeded process flow caused by the unknown exception. Consequently, the instance did not achieve completion, and Intalio did not provide any notification to the user.

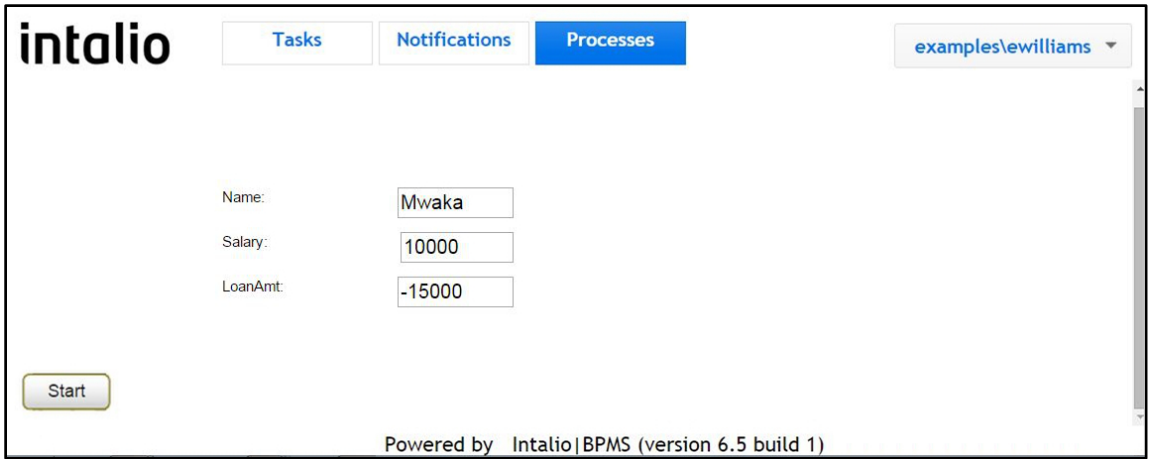

Figure 26. Intalio Scenario B user inputting -15,000 as loan amount

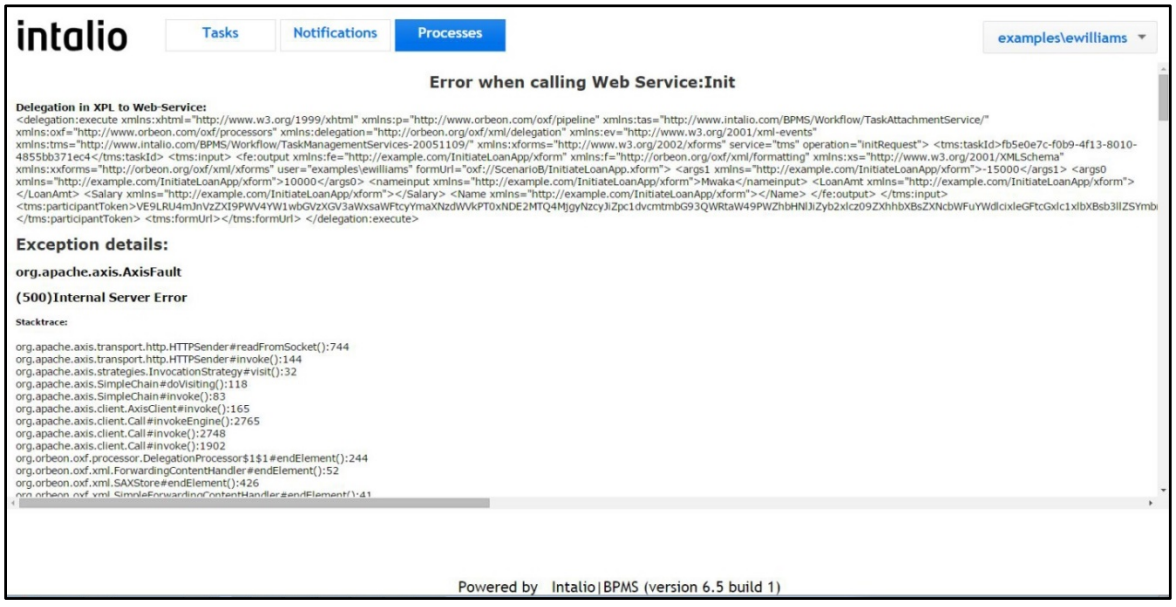

Figure 27. Intalio displays error in Scenario B fourth instance

With the process instance in an ad hoc-state, there is no conclusive information about the process status. The process instance would thus appear with either an in-progress status or in other cases, a fail status. Table 5 displays a summary of the results for Scenario B instances.

Scenario B Instances	Loan Amount	Salary Amount	Action	Final Status
First Instance	10,000	10,000	Approved	Completed
Second Instance	110,000	10,000	Denied	Completed
Third Instance	0	10,000	User is notified to reapply	Completed
Fourth Instance	-15,000	10,000	Unknown Status	Unknown

Table 5. Scenario B process instances results

One of our goals for the exception handling mechanism is that it will be able to redirect these cases to users or process owners that would be able to lead these instances to a final meaningful end. The results of these instances would be determined by the user (i.e. completed, reassign or terminated). Scenario C adds an exception handling functionality to the process.

5.5 Scenario C: Automated Process with Exception Handling Tool

Scenario C extends Scenario B by adding an Exception Handling Tool into the loan approval business process. The objective of Scenario C was to demonstrate how the unknown exception handler tool could be used to manage process instances with

unknown exceptions. Similar to Scenario B, known exceptions modeled within the process would be handled by the exception handler. However, in Scenario C, we created another catch-all exception handler to catch unknown exceptions. When an unknown exception is caught, the unknown exception handler is invoked. Thus, in Scenario C, unlike Scenario B, process instances with unknown exceptions would reach a meaningful state instead of ending up in an ad-hoc state. The unknown exception handler provides the user the following options to manage a process instance with an unknown exception: complete, reassign, and terminate. The complete option would mean that a loan associated with the process instance were either approved or denied. The reassign option would allow the user to reassign the process instance to a different user for additional processing. The terminate option would allow the user to terminate the instance regardless of the loan process results. Table 6 provides a summarized list of actions available to the user.

The process steps for Scenario C are outlined below:

1. The process begins with an applicant submitting a loan for approval.
2. If the loan amount were greater than 10,000, the loan would be routed to the Loan Assessment Service.
3. If the loan amount were less than or equal to 10,000, the loan would automatically be routed to the Loan Approval Service.
4. If a known error (modeled to be caught by handler) were to occur within the Loan Approval or Loan Assessment services, then the user would be notified of the error with a request to reapply. If an unknown error were to occur (not modeled within

the handler), the process would be transferred to the Exception Handling Tool, which offers the user the following options: complete, reassign, and terminate.

5. The user receives either a loan approved, loan denied, or error notification, based on the results of the Loan Approval and Loan Assessment services. All process instances would reach meaningful states, regardless of the loan process results.

Option	Description	Final Status
Complete	Complete task; process is set to complete by either approving or denying the loan.	Task is completed.
Reassign	Reassign task to a different task owner.	Task is reassigned to a different task owner.
Terminate	Task instance processing is stopped.	Task is terminated.

Table 6. User actions to manage process instances with unknown exception

In order to demonstrate how process instances with unknown exceptions are handled by the Exception Handling Tool, as seen in Figure 28, Scenario C was implemented in Intalio BPM, following the steps described in section 5.2.5.1. The Scenario C process was first conducted with an input of 10,000 for the loan amount to ensure that the instance would complete successfully. Scenario C was conducted a second time with an input of 110,000 for the loan amount to ensure that the instance would complete with the expected result of the loan being denied. The first three instances of Scenario C were conducted following the steps described in section 5.4 for Scenario B. In step 2 of Scenario C, the loan approval process was initiated by clicking on the "InitiateLoanApp-init request process ScenarioC/ScenarioC with form InitiateLoanApp.xform" link (see Figure 29). Table 7 provides a summary of the results for the Scenario C instances.

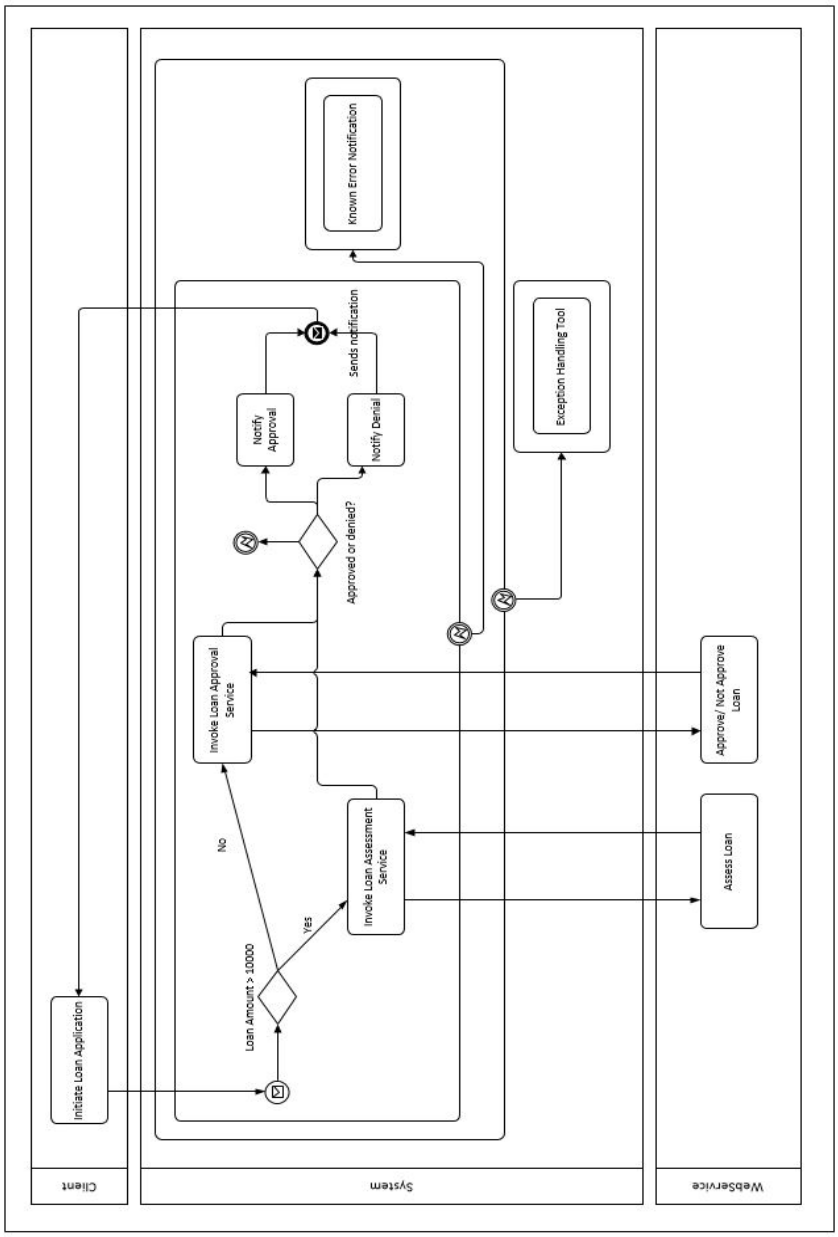

Figure 28. BPMN diagram for Scenario C

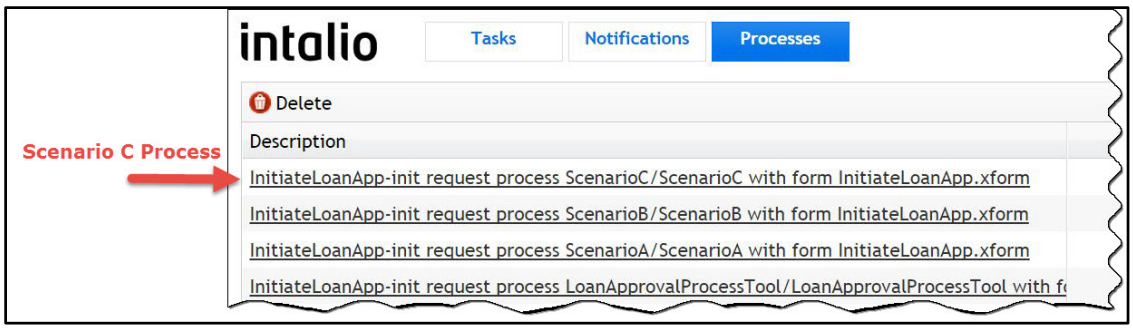

Figure 29. Intalio process instances list for Scenario C

Scenario C Instances	Loan Amount	Salary Amount	Action	Final Status
First Instance	10,000	10,000	Approved	Completed
Second Instance	110,000	10,000	Denied	Completed
Third Instance	0	10,000	User is notified to reapply.	Completed
Fourth Instance	-15,000	10,000	User chooses Complete option and chooses to approve the loan.	Completed
Fifth Instance	-15,000	10,000	User chooses Complete option and chooses to deny the loan.	Completed
Sixth Instance	-15,000	10,000	User chooses Reassign option	Completed
Seventh Instance	-15,000	10,000	User chooses Terminate option.	Terminated

Table 7. Scenario C process instances results

The fourth instance of Scenario C was conducted by inputting -15,000 for the loan amount (see Figure 30). Since the loan amount is less than 10,000, the Loan Approval Service was triggered and an unknown exception was found, as the requested loan amount was less than zero; this error was then displayed by Intalio for the user, as shown in Figure 27. As this is an unknown exception, it was caught by the catch-all exception handler, which transfers the process instance to the unknown exception handler tool. Even though the instance was transferred to this exception handler tool, the current

status of the instance was "In Progress." A user with admin privileges could obtain the meta-data about an instance status as well as the instance identifier information from the Intalio Instance Detail interface (shown in Figure 31). As can be seen in Figure 31, the instance ID for the fourth instance of Scenario C is 286, and its status is "In Progress."

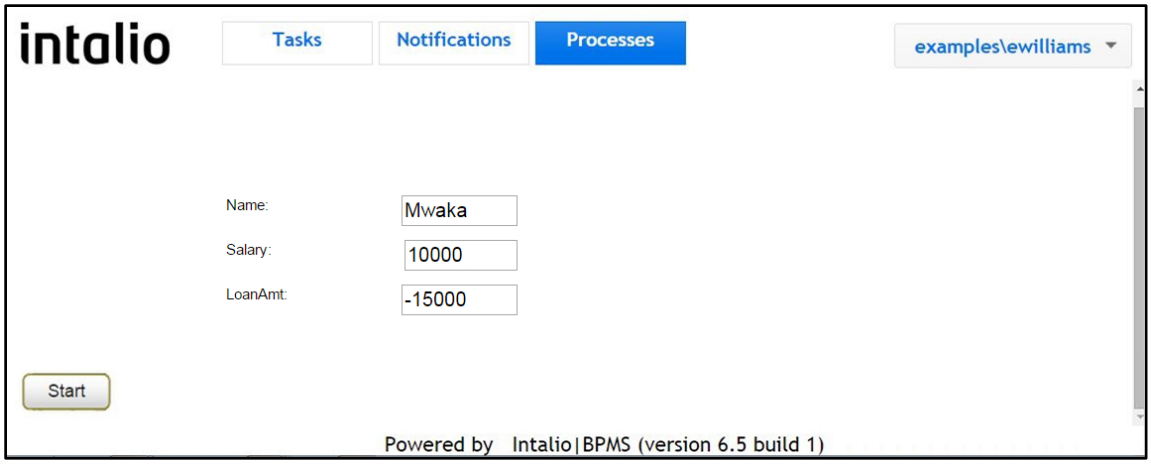

Figure 30. Intalio Scenario C user inputting -15,000 for the loan amount

The Intalio Instance Details interface (seen in Figure 31) can be accessed from the Intalio Processes list interface (see Figure 32), which is displayed immediately after logging in as the admin (note that the admin login interface is the same as in Figure 13).

In order to manage the instances labeled with an "In Progress" status, the user logs into the unknown exception handler tool, as shown in Figure 33. The exception handler tool displays a list of process instances with the "In Progress" status, as shown in Figure 34. The user can then take appropriate actions for a selected instance.

Figure 31. Intalio Instance Details page for the Scenario C fourth instance

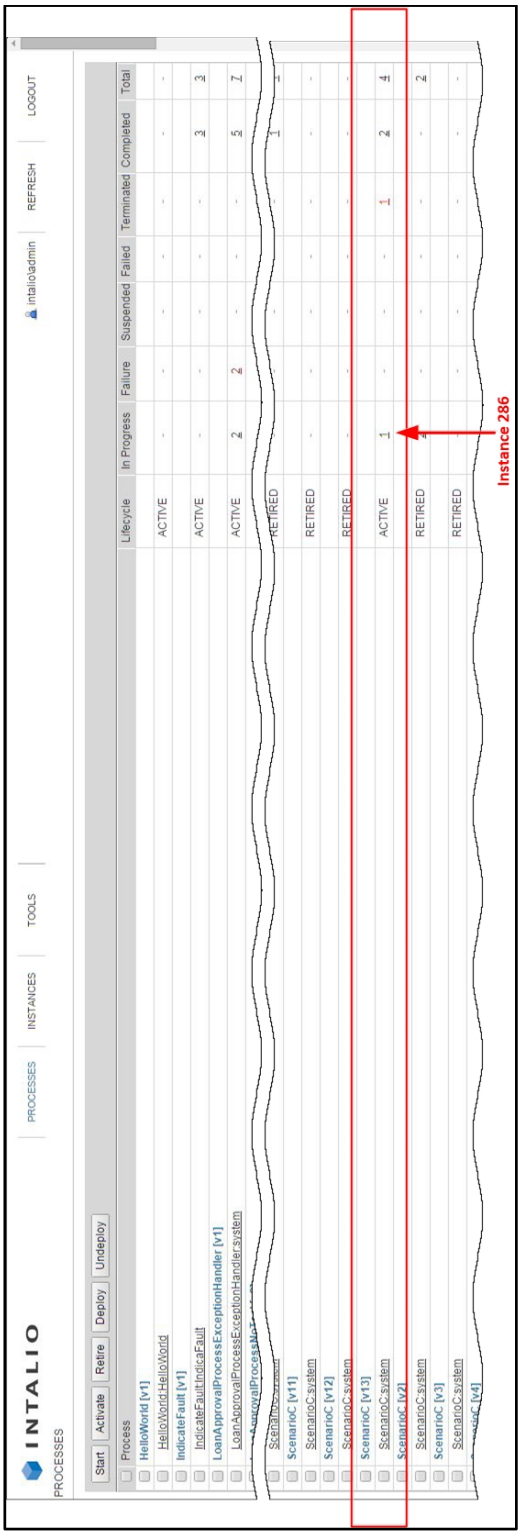

Figure 32. Intalio process list interface

After the fourth process instance of Scenario C had achieved "In Progress" status, a short series of steps were taken to complete the process instance with a loan approval. The exception handler tool displays a user interface to input the loan amount and select from the "approve" or "not approved" options, as shown Figure 35. For the fourth instance, 10,000 was inputted, the Approve option was selected, and the complete button (available under the Actions column) was clicked. The exception handler tool then set the loan amount's value to 10,000 and changed the process instance's status to "Complete." Note that a user with admin privileges could verify the process instance status in the Intalio Instance details page; Figure 36 shows that the status for instance 286 was Completed.

Figure 33. Unknown exception handler tool login interface

Figure 34. List of instances with In Progress status

Figure 35. Handling unknown exception using Complete and Approve actions

Figure 36. Instance Details interface for fourth instance of Scenario C completed

The fifth instance of Scenario C was conducted by inputting -15,000 for the loan amount, as shown in Figure 30. Similar to the fourth instance, Intalio displayed an error (as was shown in Figure 27), due to the occurrence of an unknown exception, as the requested loan amount was less than zero. Figure 37 displays the fifth process instance details, which are listed on the Intalio Instance Details page (which can be accessed by a user with admin privileges); it can be noted that the instance ID for the fifth instance of Scenario C is 288 and that the current status is "In Progress."

Once in progress, the fifth instance was then completed through a short series of steps, with the loan ultimately not approved. The unknown exception handler tool was accessed, and the Complete option, available under the actions column, was then selected. As shown in Figure 38, a loan amount of 10,000 was inputted, the Not Approve option was selected, and the Complete button was clicked. Consequently, the exception handler tool set the process instance's status to a completed state; Figure 39 shows the completed status of the fifth instance.

Figure 37. Instance Details interface for the Scenario C fifth instance

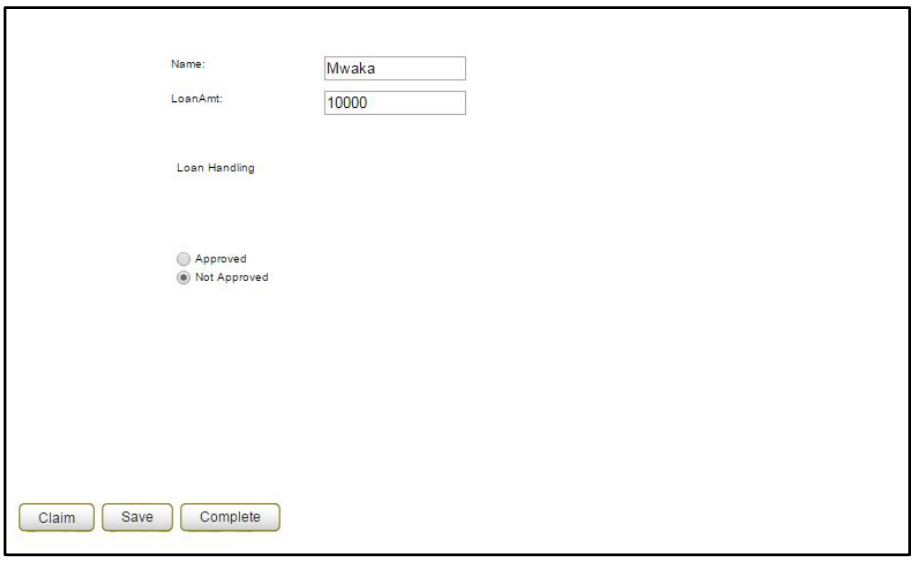

Figure 38. Handling unknown exception using Complete and Not Approved actions

Figure 39. Instance details for completed fifth instance

The sixth instance of Scenario C was conducted in a similar fashion to the fourth instance by inputting -15,000 for the loan amount, after which the Intalio displayed an error due to an unknown exception, as the requested loan amount was less than zero. Figure 40 shows the Intalio Instance Detail page for the sixth instance, which has the ID of 292 and its status as "In Progress."

The sixth instance was completed by reassigning the loan to a different user (manager), for the loan to be either approved or not approved. The unknown exception handler tool

was accessed, wherein the reassign options could be found under the reassign column, as shown in Figure 41. From Figure 41, it can be noted that the ID of the user who owned the sixth instance was 'ewilliams,'; ewilliams then chose to reassign the instance to Manager by selecting the "To Manager" option. Once the Instance Details were assigned to Manager, they were removed from the user screen, as shown in Figure 42. The Manager could log in and see that the sixth instance (ID 292) was assigned to the user, as shown in Figure 43. The tool then displayed on a user interface for the manager to input the loan amount and select from the Approve or Not Approve options, as shown in Figure 35.

For the sixth instance, the manager entered 10,000 for the loan amount, selected from the Approve and Not Approve options, and then clicked the Complete button. The exception handler tool then set the loan amount value to 10,000 and changed the process instance's status to a completed state. Figure 44 shows the completed status for Instance 292 on the Intalio Instance Details page.

Figure 40. Instance Details page for Scenario C sixth instance

Figure 41. Reassign option available for user

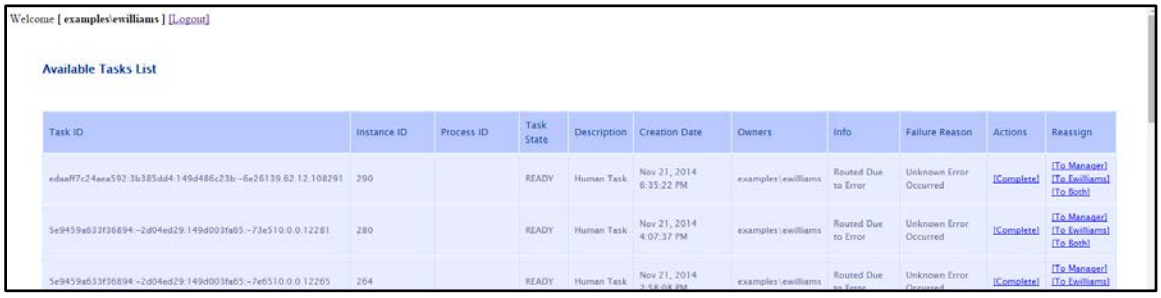

Figure 42. List of instances without Instance 292

Figure 43. List of instances assigned to Manager

Figure 44. Instance 292 completed

The seventh instance of Scenario C was conducted in a similar fashion to the fourth instance by inputting -15,000 for the loan amount, after which the Intalio displayed an error due to an unknown exception, as the requested loan amount was less than zero. Figure 45 shows the Intalio Instance Detail page for the seventh instance, indicating its ID of 294 and its status as "In Progress."

Once the seventh instance of Scenario C was in progress, a brief series of steps were taken to terminate the instance. The unknown exception handler tool was accessed, from which the list of instances that could be terminated could be found listed under the "Terminate" column; by selecting the Terminate option available in the row for Instance 294 (the seventh instance of Scenario C), and Instance 294 was thus terminated. Figure 46 features the Intalio Instance details page, indicating that the seventh instance—Instance 294—was terminated.

Figure 45. Instance Details page for Scenario C seventh instance

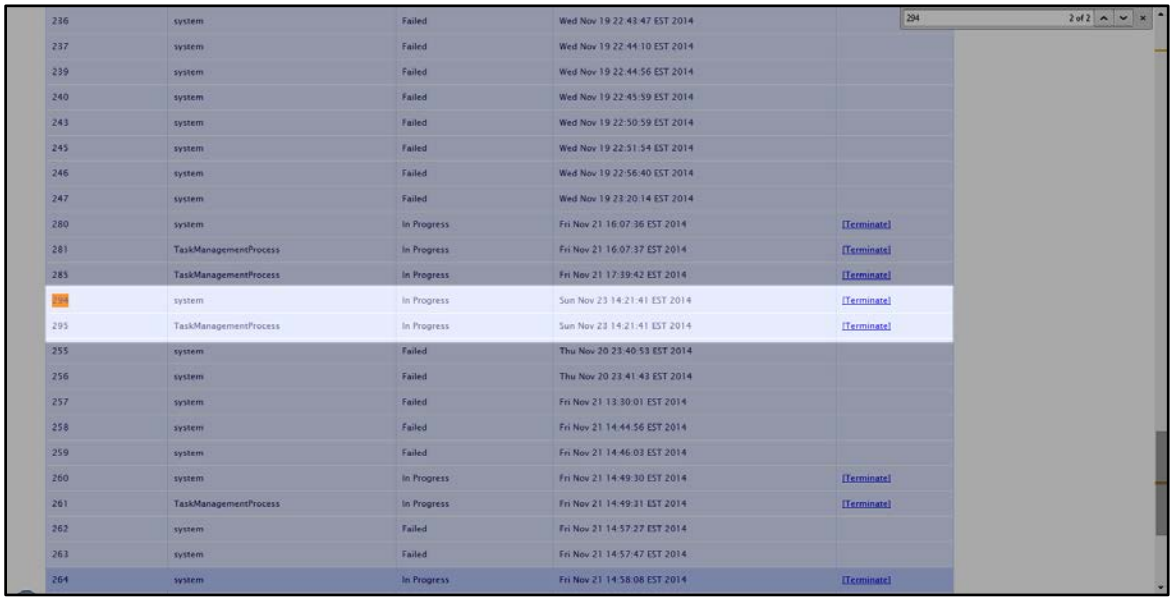

Figure 46. Terminate option

Figure 47. Terminated instance

5.6 Applying Claims Analysis to Assess Tool Contributions

In this section, we apply a claims analysis to Scenario C. Claims can be generated by an evaluator by simply scanning or questioning a scenario for its obvious effects on the scenario context and goals, in order to identify issues and possible problems. Claims analysis is used to identify positive and negative consequences of the design features of a system for a given scenario. Claims analysis allows an evaluator to explicitly identify trade-off scenario outcomes, the benefits of the system's support, and the adverse risks. An evaluator can identify appropriate actions needed to improve the support provided by the system to overcome the adverse consequences of the scenario.

From our walkthrough of Scenario C, we have identified several claims. We have also categorized each of the claims into four contribution categories suggested by the SWIMs approach. The first claim identified from Scenario C is the ability to catch unknown exceptions. We achieved this claim by using a catch-all exception handler to catch unknown exceptions. We placed the entire loan application process inside the catch-all exception handler. This allowed for the known exceptions to be handled by the predefined exception handler; if an unknown exception occurs in the process, those instances are directed to the unknown exception handler tool. We classify this first claim of being able to catch unknown exceptions as a tangible contribution, as the user can easily perceive the benefits of knowing about the occurrence of unknown exceptions.

The second claim identified from Scenario C is the ability to review process instances with unknown exceptions. The unknown exception handler tool provides a list of process instances that have unknown exceptions and that need a user's attention. The user then obtains the process instance ID information and further investigates the process instance so as to choose an appropriate action to handle the instance. We classify this second claim of being able to review process instances with unknown exceptions as an intangible contribution, as the benefit of reviewing instance information is not directly associated with an organization's goals of processing loan applications.

The third claim identified from Scenario C is the ability to handle process instances with unknown exceptions. This claim is the main objective of this evaluation and this thesis. The unknown exception handler tool allows users to manage instances with unknown exceptions by using one of the following meaningful states: Complete, Reassign, and Terminate. In the walkthrough, we used the fourth and fifth instances to demonstrate user actions that brought instances to completed statuses. Then we used the sixth instance to demonstrate user actions that reassigned the instance to another user, who could choose to complete or terminate the instance. Finally, we used the seventh instance to demonstrate user actions that terminated an instance. We classify this third claim of the ability to handle process instances with unknown exceptions as a measurable contribution, as the benefit of handling instances with unknown exceptions and bringing them to meaningful statuses is beneficial for process efficiency.

The fourth claim identified from Scenario C is the ability to complete the instances with unknown exceptions by reentering the loan amount value. The unknown exception handler tool allows the user to utilize contextual information to decide whether to complete the instance by approving or by denying the loan. This feature was demonstrated in the fourth and fifth instances. We classify this fourth claim of the ability to complete process instances by either approving or denying loans as a tangible contribution, as completing process instances is beneficial to achieving organizational goals, though it is not measureable.

The fifth claim identified from Scenario C is that the exception handler tool currently does not provide an option to send the instance back to the applicant so that the applicant can reenter loan and salary inputs. We classify this fifth claim of the ability to allow for applicant interaction for the handling of instances with unknown exceptions as an unrealized contribution, as the exception handler tool does not provide support for the identified capability. Table 8 provides a summary of claims analysis.

In summary, we have achieved our evaluation's objective to demonstrate how the unknown exception handler tool handles process instances with unknown exceptions. We achieved the objective with the use of a loan application process, the creation of three scenarios, and a walkthrough of the use of the system for a variety of instances, including instances with the occurrences of none, known, and unknown exceptions. With this claims analysis, we have demonstrated the utility of the exceptions handler.

Claim	Description	Category	Remarks
First Claim	Ability to catch unknown exceptions	Tangible	User can perceive benefits of knowing occurrence of unknown exceptions
Second Claim	Review process instances with unknown exceptions	Intangible	The benefit of reviewing instance information is not directly associated with organization goals of processing loan applications
Third Claim	Handle process instances with unknown exceptions	Measurable	The benefit of handling instance with exception and bringing them to meaningful status helps in improving process efficiency
Fourth Claim	Complete instances with unknown exceptions	Tangible	The unknown exception handler tool allows the user to utilize contextual information to decide on whether to complete the instance by approving or denying the loan.
Fifth Claim	Tool does not provide an option to send the instance back to applicant so that the applicant can reenter loan and salary inputs	Unrealized	The mechanism does not provide support for the identified capability

Table 8. Claims analysis summary

Chapter 6

DISCUSSIONS

6.1 Contributions

With this thesis, we make two major contributions. First, we make a conceptual contribution with the development of a framework for handling unknown exceptions. The conceptual framework was developed utilizing BPEL4People and WS-HumanTask functionalities to provide a meaningful way for people to resolve unexpected issues within a process instance. Second, we make a practical contribution by having developed the Exception Handling Tool as an extension of Intalio. The tool developed and discussed in this thesis provides a means to expand the involvement of task owners to act on unhandled exceptions that occur within a process instance. With this exceptions handler, task owners would have an option to extend the lifecycle of tasks. Our tool can potentially reduce costs and cycle time by eliminating unhandled errors and by improving quality through a reduction in the number of running task instances within a process.

6.2 Limitations

While our study led to satisfactory results, the study itself did have some limitations. We did not do a performance-based evaluation, as it would have taken an infinite amount of time before the next process change for the scenarios with unknown errors, if there were no interventions on the process instance. The main focus of this study was the end result of a process instance. The time taken to achieve the end result could be varied by external user factors within a process instance. For example, users could have initiated a process instance and chosen not to complete it right away. In this case, the process instance would have only ended when the process owner opted to complete the process. Our primary analysis involved the inclusion of human agents within the business process. The basis was adding human involvement into business processes, as opposed to fully automated process instances.

The aim of this study was to improve exception handling techniques in workflow management technology in order to decrease most of the average values of given performance indicators. Business process improvement is measured in terms of lead time, service time, wait time, and resource utilization. In terms of exception handling, lead time is the time between the initial occurrence of an unhandled exception and its completion (also known as cycle time, completion time, and turnaround time). Service time is the period of time in which resources are used to resolve an exception. Wait time is the period of time in which a workflow is idle due to an unhandled exception.

The utilization of human resources is regarded as the ratio of activity versus their availability.

Despite the limitations noted above, our use of the SWIMs approach allowed us to show that human interaction in business processes would allow people to spend less time on coordination and on the transfer of work, which would lead to a decrease in service time. When the supply of work and resources remain constant, work load and resource utilization decreases as a result. Therefore, the averages of all four performance indicators could potentially decrease as a result of our improvement of the exception handling tools in workflow management systems.

6.3 Future Work

There is minimal distinction between tasks and task instances within BPEL4People and WS-Human Task. In order to invoke a specific process instance, one has to be aware of the process name and task name. A much clearer distinction is needed in cases in which we need to navigate to a specific running instance.

Tasks within BPEL4People currently do not have auto-starting capabilities, which would be beneficial to analyze performance within workflows. A number of workflows could be staged and auto-initiated, allowing for more effective process statistics. Additionally, more support is needed when restricting tasks. Currently, tasks are

assigned and restricted by the user. Once a user is assigned a task, the user becomes the task owner, and restrictions are then placed on the user. These number of restrictions could be increased to include restrictions on specific aspects of the business process at the task level.

REFERENCES

Akram, A., Meredith, D., & Allan, R. (2006). *Evaluation of BPEL to Scientific Workflows.* Paper presented at the IEEE International Symposium on Cluster Computing and the Grid, Singapore.

Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., & Orchard, D. (2004, 11 February 2004). Web Services Architecture - W3C Working Group Note Retrieved March 25, 2014, from http://www.w3.org/TR/ws-arch/

BPEL4People. (2010). WS-BPEL Extension for People (BPEL4People) Specification Version 1.1 Retrieved February 2, 2013, from http://docs.oasis-open.org/bpel4people/bpel4people-1.1.pdf

Cain, C., & Haque, S. (2008). Organizational Workflow and Its Impact on Work Quality. In R. G. Hughes (Ed.), *Patient Safety and Quality: An Evidence-Based Handbook for Nurses* (pp. 217 - 244). Rockville, MD, USA: Agency for Healthcare Research and Quality (AHRQ) Publication No. 08-0043.

Cardoso, J., Luo, Z., Miller, J., Sheth, A., & Kochut, K. (2001). *Survivability Architecture for Workflow Management Systems.* Paper presented at the ACM Southeast Conference, Athens, GA, USA.

Casati, F., Ceri, S., Paraboschi, S., & Pozzi, G. (1999). Specification and implementation of exceptions in workflow management systems. *ACM*

Transactions on Database Systems (TODS), 24(3), 405-451. doi: 10.1145/328939.328996

Chiu, D. K. W., Li, Q., & Karlapalem, K. (1999). A meta modeling approach to workflow management systems supporting exception handling. *Information Systems, 24*(2), 159-184. doi: http://dx.doi.org/10.1016/S0306-4379(99)00010-1

Chiu, D. K. W., Li, Q., & Karlapalem, K. (2001). Web interface-driven cooperative exception handling in ADOME workflow management system. *Information Systems, 26*(2), 93-120. doi: http://dx.doi.org/10.1016/S0306-4379(01)00012-6

Derbali, M. A. (2011). *A framework proposal for intelligent management of unexpected exceptions in workflow.* Paper presented at the Confederated International Conference On the Move to Meaningful Internet Systems (OTM), Crete, Greece.

Fakhroutdinov, K. (2013). Online Shopping Activity Diagram Retrieved February 3, 2013, from http://www.uml-diagrams.org/activity-diagrams-examples.html#online-shopping-activity

Gerber, J.-A. (2006). Intalio Tempo Creating And Completing a Task Retrieved February, 2013, from http://www.intalio.org/confluence/display/TEMPO/Creating+and+Completing+a+Task

Han, M., Thiery, T., & Song, X. (2006). *Managing exceptions in the medical workflow systems.* Paper presented at the International Conference on Software engineering, Shanghai, China.

Haynes, S. R., Purao, S., & Skattebo, A. L. (2009). Scenario-Based Methods for Evaluating Collaborative Systems. *Computer Supported Cooperative Work (CSCW), 18*(4), 331-356. doi: 10.1007/s10606-009-9095-x

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly, 28*(1), 75-105.

Hochmüller, E., & Dobrovnik, M. (2005). *Flexibility Issues in Workflow Management Systems.* Paper presented at the Workshop on Business Process Modeling, Development, and Support (BPMDS), Porto, Portugal.

Holanda, H. J. A., Hernáiz, J. M., Barroso, G. C., & Serra, A. B. (2010). Performance Evaluation of Web Services Orchestrated with WS-BPEL4People. *International Journal of Computer Networks & Communications (IJCNC), 2*(6), 117 - 134.

Jordan, D., & Evdemon, J. (2007, April 11). Web Services Business Process Execution Language (WS-BPEL) Version 2. Retrieved July 5, 2014, from http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

Looker, N., Webster, D., Russell, D., & Xu, J. (2008). *Scenario Based Evaluation.* Paper presented at the IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC), Orlando, Florida, USA.

Louridas, P. (2008). Orchestrating Web Services with BPEL. *IEEE Software, 25*(2), 85-87.

Mourão, H., & Antunes, P. (2007). *Supporting effective unexpected exceptions handling in workflow management systems.* Paper presented at the ACM symposium on Applied computing, Seoul, Korea.

Nie, P., Seppälä, R., & Hafrén, M. (2010). Open Source Power on BPM - A Comparison of JBoss jBPM and Intalio BPMS Retrieved February 3, 2013, from http://jannekorhonen.fi/project_report_final_BPMS.pdf

Patnaik, S. (2011). A Mathematical Modeling of Exceptions in Healthcare Workflow. In N. Meghanathan, B. Kaushik & D. Nagamalai (Eds.), *Advanced Computing Communications in Computer and Information Science* (pp. 120-129). Berlin: Springer

Peffers, K., Tuunanen, T., Rothenberger, M., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems, 24*(3), 45-77. doi: 10.2753/mis0742-1222240302

Potti, P. K., Ahuja, S., Umapathy, K., & Prodanoff, Z. (2012). Comparing Performance of Web Service Interaction Styles: SOAP vs. REST. *Journal of Information Systems Applied Research (JISAR), 6*(1), 4-26. doi: http://jisar.org/2013-6/N1/JISARv6n1p4.html

Shi, M., Yang, G., Xiang, Y., & Wu, S. (1998, 22-24 Oct 1998). *Workflow management systems: a survey*. Paper presented at the International Conference on Communication Technology, Beijing, China.

Smith, H., & Fingar, P. (2004, January 6, 2004). Workflow is just a Pi process Retrieved February 2, 2013, from http://www.bptrends.com/publicationfiles/01-04%20Workflow%20is%20just%20a%20Pi%20Process%20Smith-Fingar.pdf

Song, X., Han, M., & Thiery, T. (2007). Representing and Handling Workflow Exceptions in Highly Dynamic Healthcare Environments. *The International Journal of Intelligent Control and Systems, 12*(1).

van der Aalst, W. M. P. (2001). How to Handle Dynamic Change and Capture Management Information: An Approach Based on Generic Workflow Models. *Computer Systems: Science & Engineering (CSSE), 16*(5), 295 - 318.

van der Aalst, W. M. P., & Hee, K. v. (2002). *Workflow Management: Models, Methods, and Systems* (First Edition ed.). Cambridge, USA: The MIT Press.

van der Aalst, W. M. P., & Kumar, A. (2003). XML–Based Schema Definition for Support of Interorganizational Workflow. *Information Systems Research, 14*(1), 23 - 46.

Vojevodina, D., & Kulvietis, G. (2004). *Office Activity Procedure Exception Handling Realization Difficulties.* Paper presented at the Conference on Advanced Information Systems Engineering (CAiSE) Workshop on Business Modeling, Development and Support, Riga, Latvia.

WS-HumanTask. (2012). Web Services – Human Task (WS-HumanTask) Specification Version 1.1 Retrieved February 2, 2013, from http://docs.oasis-open.org/bpel4people/ws-humantask-1.1.pdf

VITA

Mwaka Mahanga is a Computer IT professional with experience in software analysis and development. He holds a bachelor's degree in Computer Science and has more than 8 years of experience in Java and Enterprise Portal applications. His experience spans from software development and QA to configuration support. He currently holds a software configuration role with a major financial company. He is an experienced Build and Release Engineer and has supported a number of application releases for companies in defense and financial industries. He is very passionate about the advancement of new tools and technologies, and dedicated to local user groups. He continues to be active in generating innovative ideas for improving continuous build integration processes. He currently resides in Jacksonville, Florida.