January 2014

Exception Management Application

Ran Tian
Worcester Polytechnic Institute

Exception Management Application

A Major Qualifying Project Report

Submitted to the faculty of the
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements of the
Degree of Bachelor of Science

By:

Spenser Haddad, Electrical and Computer Engineering
Ran Tian, Industrial Engineering

Project Sponsor: Barclays

Submitted to:
On-site Liaisons: Gregory Friel
Mayank Nainwal
Terrance Snyder

Project Advisors: Professor Arthur Gerstenfeld, Department of Business
Professor Xinming Huang, Department of Electrical and Computer Engineering

Submitted on
January 3, 2014

Abstract

iProcess is a TIBCO[1] product that is standardized for business process management (BPM) in Barclays. It is used to automate business processes where human activity is required. The team was tasked with researching into iProcess and building a framework for developing applications to manage exceptions, which are cases that deviate from normal situation and need to be handled by human intervention, for example missing static data or wrong trade date (in trade information). This framework will allow Barclays to develop programs to detect and close many types of exceptions over multiple systems.

[1] TIBCO Software Inc. (NASDAQ: TIBX) is an American company that provides infrastructure and business intelligence software

Acknowledgements

Our group would like to acknowledge everyone that helped to provide an opportunity to take part in this project and the great success that resulted from it.

Foremost we would like to thank Mr. Gregory Friel, our sponsor, who made this project possible and oversaw the project from the Barclays side. His support provided us with a great opportunity to work for Barclays and apply the knowledge we have gained through our studies at Worcester Polytechnic Institute.

We also would like to thank Mr. Mayank Nainwal and Mr. Terrance Snyder, the project manager, who managed the day-to-day operations of the project. We truly appreciate the many hours of guidance and support he provided during our time at Barclays. His assistance was especially vital to the completion of this project.

Moreover, we would like to thank Lorenzo Ciaravola and Pansy Verma for all of their help in maintaining the iProcess server system, which was the foundation of our project, and in helping to accommodate us to the iProcess environment. Their knowledge was greatly beneficial to us learning about the iProcess environment.

Additionally, we want to extend our gratitude towards Professor Arthur Gerstenfeld, Director of the Wall Street Project Center and also our faculty advisor, for creating such an enriching experience. Also we would like to thank Professor Gerstenfeld and Professor Xinming Huang, as being our faculty advisors, for their continued support, advice, and assistance throughout the project.

Lastly, we would like to thank Worcester Polytechnic Institute and Barclays for providing us with the opportunity to come to Jersey City and complete this project.

Without the assistance of these two organizations and all of these individuals, this project would not have been the success that it was. Thank you!

Authorship

This report was developed through a collaborative effort by both members of the project team: Spencer Haddad and Ran Tian. All sections were created and edited as a team, with equal contributions made by each member.

Executive Summary

An exception is sent from a source system, it will be routed to one or more end entity, which is responsible for the exception, according to our rule of routing. This exception will then become an item in the user's work list (called work queue in iProcess Workspace). There are features allow user to apply filter and sort parameters, forward work items to different user or group and so on.

We use TIBCO Studio as our process modeling tool and exception flow controller; using the client application called iProcess Workspace (Browser), which also belongs to the iProcess package, as the interface for end users to participate the process.

After researching and experimenting with the capabilities of iProcess, our team decided that the most efficient method of developing an exception management system would be to create a framework in iProcess. This framework would be responsible for catching exceptions, and assigning them to a sub-routine to be resolved. Using iProcess, our team modeled this framework as a workflow to catch and route exceptions to a custom-made procedure for closing them.

Table of Contents

List of Figures

Introduction

Barclays PLC is a British banking and financial services company headquartered in London. With its more than 300 years of history, it now became a multinational company and also a leader in innovation on improving business process.

Our project team worked with Barclays' ADS Future team on an exception management project. A business exception means a case that deviates from normal situation in a business process and need to be taken care of in a special manner, which is typically by human intervention. The team aimed to use a BPM (Business Process Management) engine called TIBCO iProcess to manage the business exceptions, which means to automatically catch the exception from source systems, route them to the right entity to get it fixed, which involves human participants. TIBCO iProcess is the firm-standard BPM (Business Process Management) tool for human-based and system-integrated workflow in Barclays.

The goal of this project was to create a framework of exception management using iProcess. To attain this goal, the team followed a modified scrum approach and completed the following four objectives:

1. collect requirements for the application;
2. conduct research on TIBCO iProcess;
3. design and implement the application framework;
4. recommend future directions.

In the following sections, we will talk about our project in two main topics: research on iProcess and the development of the framework.

Background

Barclays

Barclays PLC is a British banking and financial services company headquartered in London, United Kingdom. Founded in 1690, Barclays has become the third largest banks in United Kingdom with total assets of $2.422 trillion and a total employee number of 140,000. As a multinational company, it has an extensive international presence in Europe, the Americas, Africa and Asia. In the 50 countries they operate, Barclays offers business including personal banking, premier banking, business banking, wealth and investment managing, corporate banking as well as investment bank.

They values respect, integrity, service, excellence and stewardship as the most important aspects in terms of achieving their goal of "Helping people achieve their ambitions – in the right way" and "Becoming the 'Go-To Bank'." They measure and reward the employee, not just on commercial results, but on how they live Barclays' Values and bring them to life every day.

Business Exception Management

Business Exception

An exception is a case that deviates from normal situation and need to be taken care of in a special manner, typically by human intervention. A very general example of exception in our daily life would be an incorrect charge on your credit card bill. Your bank's history has a record that you do not recognize, which is an exception need to be resolved by at least one phone call to the customer service center. An exception in a business process is similar to the exception happened in our daily life – it could be caused by human error, for example a trader mistyped trading volume of a stock; it could be also caused by missing /wrong data, like an invalid business date for a trade.

A logical lifecycle of an exception is shown in the diagram on the right. After an exception being raised by system, there will be an entity as its next stop. This entity could be a person, a group of people or a trigger of an event such as printing

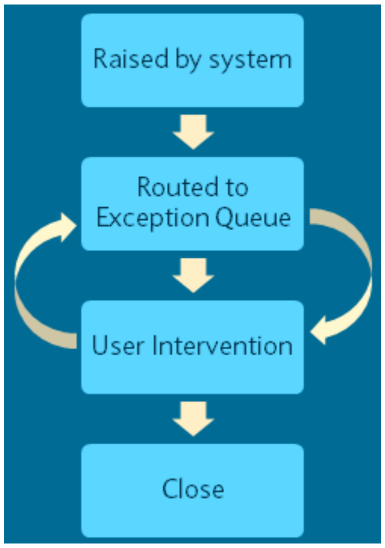


Figure 1: Exception Lifecycle

a document or sending an email. Then the next step is the human intervention, where the exception is being resolved. Notice the arrow pointing from “Routed to Entity” to “User Intervention” and backward represents a possible loop within these two steps. Imagine when you call a customer service hotline, you would enter a phone tree that leads you to different branches after each time you press a number. This is a typical situation when the exception is routed from one entity to another. Then you are connected to a representative who helps you look into the issue, but unfortunately he/she is not the expert for the issue and route you to another representative. And finally, your problem got solved; the exception reaches to its final status - “Closed”.

Exception Management

We have been talk a lot about exception, so what is exception management for our business process? To explain it in a simple way, exception management is to automatic the exception lifecycle shown above. It includes catching exception from source systems, routing of the exception to the entity which is responsible for it. The user will mark the exception as “Closed” after the problem got taken care of. The whole process would be recorded as an audit trail.

The exception management would bring the following benefits to both business and technology:

- Transparency and accountability

  Automated business processes enables you to record, monitor, measure, and accurately account for every activity within a business process effortlessly. Complete audit trails ensure you know who did what when. This drives transparency and accountability into the organization, highlighting improvement opportunities.

- Agile response to change

  Process Automation allows you to dynamically involve appropriate resources, driving speed of execution, and enables to meet customer and market requirements faster than the competition- applications built in days or weeks, not in months or even years.

- Increase Efficiency

  Exception management helps to reduce the unit cost to execute tasks by streamlining business processes and allowing you to better orchestrate time and resources to non-redundant tasks and manual processes.

Moreover, for technology side itself, the exception management provides suitable frameworks to integrate with many external applications. For business side, since we keep everything in record, the risk contained by the exceptions would be under control. Also, to prevent people from "finger pointing" in front of certain issue, exception management would help increase internal collaboration by increase the transparency and efficiency of information.

Business Process Management (BPM) Software

Business Process Management coordinates the flow of tasks, access to resources and the exchange of information among employees, customers and partners in order to enhance the efficiency of the end-to-end business process. The idea is that by understanding the processes undertaken to fulfill a business need across multiple applications, and then putting those processes on one integrated process layer, new efficiencies can be exploited without changes to how the business works or the existing IT structure. TIBCO's iProcess has been standardized as Barclays' BPM tool, so our project chose iProcess as the tool to build the exception management framework.

Technology Review

This section details some of the other technologies used in development of the exception management system in addition to Tibco iProcess

JAVA

Java is a class-based object-oriented programming language originally developed by Sun Microsystems, and currently maintained by the Oracle Corporation. It derives heavily from the C programming language, but is designed as a higher level alternative for making enterprise-level applications.

One of the major features of Java is its mission to have as few dependencies as possible, allowing programs written in the language to be cross-platform by default.

Oracle Database

Oracle Database is an object-relational database management system created (ORDBMS) and maintained by the Oracle Corporation. It provides database control and access using a series of table spaces and data files. Oracle databases can be accessed and manipulated using Procedural Language SQL (PL/SQL), a proprietary query language created by Oracle Corporation as an extension to the commonly-used structured query language for database management (SQL).

JAVASCRIPT

JavaScript is a programming language developed by the Netscape Corporation. It is a scripting language developed to be used with websites and online applications. It allows scripts written by the client to run in a web browser, letting it communicate with an end user and update a webpage and interact asynchronously with the client. JavaScript has also been adopted for use in developing smaller desktop applications and as a general-purpose scripting language.

Methodology

The project is mainly focusing on creating a framework for developing exception management applications to handle exception. Since TIBCO iProcess is a newly implemented tool for the ADS future team we worked with, we spend large amount of time researching on TIBCO iProcess engine, discovering the functions and features provided by the applications in iProcess engine and considering how they could be used to build the framework. We later integrate Oracle Database when developing the prototype. In addition to hand-off this application, the project team provided recommendations for further directions and improvements.

Research Goals and Objectives

For this project, the team delivered an application framework during their eight weeks with Barclays in New Jersey. The development team can use this framework to integrate with different applications used by different team for their own specific purposes. To attain this goal, the team followed a modified scrum approach and completed the following four objectives:

1. collect requirements for the application;
2. conduct research on TIBCO iProcess;
3. design and implement the application framework;
4. recommend future directions.

Data Collection

In order to understand Barclays' needs and set objective throughout the project period, the team used two essential methods: research and meetings. The team researched the software available to develop an application by reading application documentations and watching tutorial videos; understand the firm's requirement by having weekly meetings with our two project managers. Furthermore, the team met with several Barclays' employees who are familiar with the product or have using experience of other BPM software.

Development Approach – Scrum

Our project team followed a development methodology called Scrum. The scrum approach is a "framework for organizing and managing work" (Rubin, 2012). It guides the project team to work on one small piece with the highest priority at a time within a certain time period, so that the team could respond to feedback and change easily. Since our project time is only eight weeks, we set out feedback cycle time – called “sprint” in Scrum – to one week to fit in our timeline. People related to the project falls into three categories, or roles: product owner, scrum master and development team. At the beginning of each sprint, the product owner determines that needs to be built in the next sprint; the development team builds what is needed and then demonstrates what they have built. Based on the demonstration, the product owner determines goals for next sprint. Scrum masters ensure this process happens as smoothly as possible, and continually help improve the process, the team and also the product being created.

For our project team, these roles are relatively vague since the size of the team is small. Gregory Friel and Mayank Nainwal are the product owners. Our project manager Terrance Snyder held the role of both product owner and also scrum master. Also the team member Spencer Haddad and Ran Tian will take the position of scrum master in turns in each sprint. During our weekly scrum meeting on Friday, Terrance and Mayank would sit with us and discuss our accomplishment of this week’s sprint and guide us to set a milestone for the following week (sprint). Also, every morning the project team and scrum master would meet to plan for that day.

Research on TIBCO iProcess

Tibco iProcess is a suite of business process management software designed to create applications that comprise several discrete steps, either using automated routines or user-defined inputs. Within iProcess, these applications are referred to as procedures. The individual steps inside a procedure may be referred to as either tasks or steps.

iProcess was the software used for the majority of the team's research on exception management, and as such is a major component of the project.

Deployment

The entire suite of iProcess applications is run as server-driven software. All of the procedures, the engine which runs them (known as the iProcess engine), and all the required data to operate these procedures are stored together on a server. This gives iProcess all of the benefits of server software, including remote access, deployment across multiple networks, and easier maintenance and monitoring for administrators.

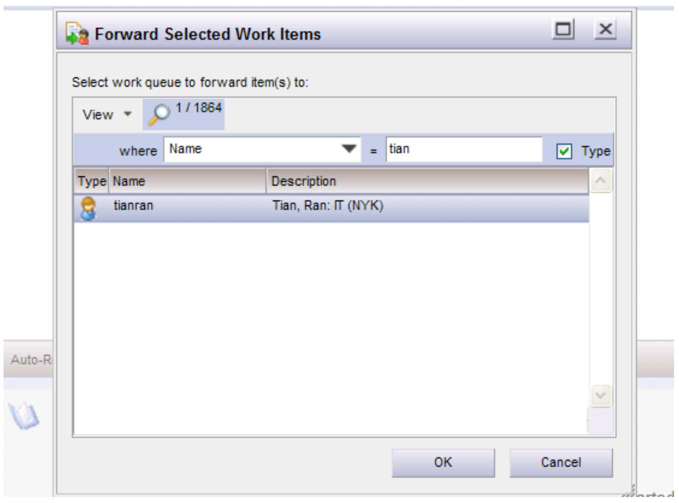
When a procedure is developed in iProcess and ready for use, it is deployed to the server for online access. Once deployed, individual instances of a procedure, called cases, can then be started by a developer or an external application. It is possible for there to be multiple cases of the same procedure running simultaneously. IProcess will then iterate through the steps in the case, automatically performing any automated tasks. Tasks which require human input are assigned to a user or a group of users, as defined in the procedure and the procedure waits here until the user completes the task. The case continues until it reaches an end task, at which point the case is closed.

iProcess Workspace

Tibco provides a default interface for interacting with cases, called iProcess Workspace, which can either be run in an internet browser or as a Microsoft Windows application. End users can use Workspace to view all of the cases assigned to them and the groups they are a member of. The cases assigned to the individual and to the groups are divided in workspace into separate queues. The user can view these queues separately, and open the cases to complete the required task. Completing the task can require the user to do a variety of jobs, ranging from inputting data into a form to opening an external application and performing some work in there.

Workspace also provides some additional features along with allowing users to view and complete tasks. The interface provides some basic filtering, and allows the user to sort the cases in a queue by several values, including task status (either unopened, or opened but not yet completed), case number (automatically generated by iProcess), procedure name, step name, and more. Users also can perform more actions on a case in addition to opening and closing it. They can forward a task to another user's or group's queue, giving them responsibility for finishing the step. However, this functionality must be explicitly enabled in a procedure first. Users may also lock a case so it cannot be completed or continued unless unlocked by the same user, useful for preventing others in a group queue from altering any data input to the task. Finally, iProcess provides an audit trail for each case, which displays all of the steps that the case has completed up until the user received it.

iProcess also generate two graphical history, one for a single work item, showing the time spend on each steps; another one is for the work queue, showing the status of work items within the queue, show in Figure 3 below.

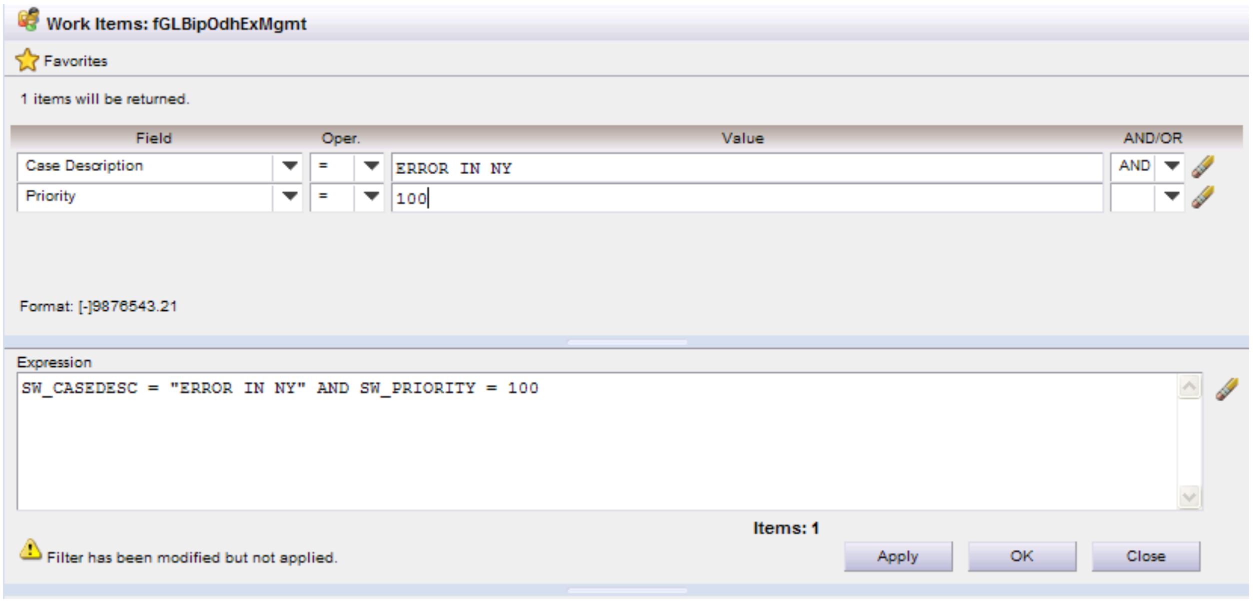


Select work queue to forward item(s) to:

where Name = tian ✓ Type

Type	Name	Description
	tianran	Tian, Ran: IT (NYK)

Figure 2 iProcess Forward Work item Window


Work Items: fGLBipOdhExMgmt

Favorites

1 items will be returned.

Field	Oper.	Value	AND/OR
Case Description	=	ERROR IN NY	AND
Priority	=	100	

Format: [-]9876543.21

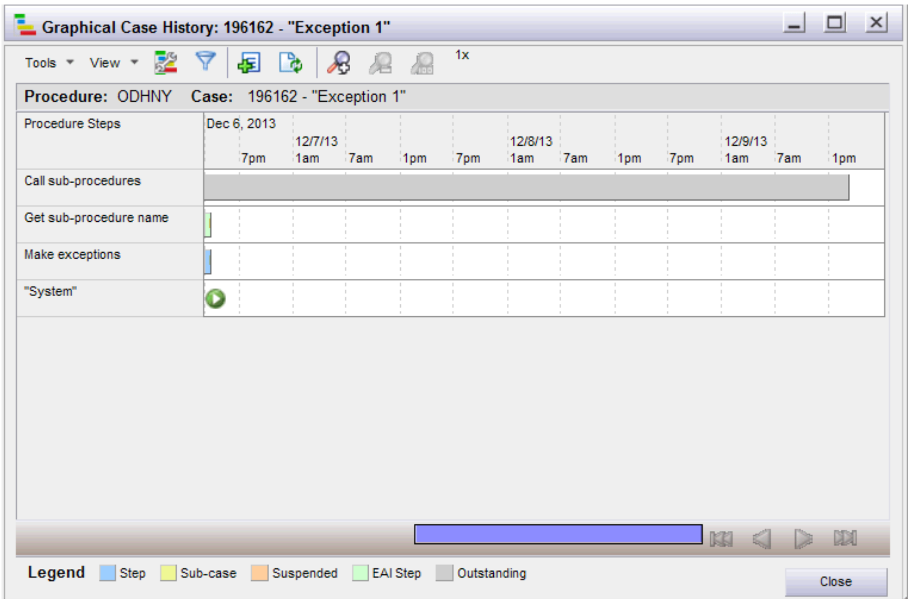
Expression

```
SW_CASEDESC = "ERROR IN NY" AND SW_PRIORITY = 100
```

Items: 1

Filter has been modified but not applied.

Figure 4 View of the Filter Work items Window in iProcess Workspace


Graphical Case History: 196162 - "Exception 1"

Procedure: ODHNY Case: 196162 - "Exception 1"

Figure 3 Graphical History - Case

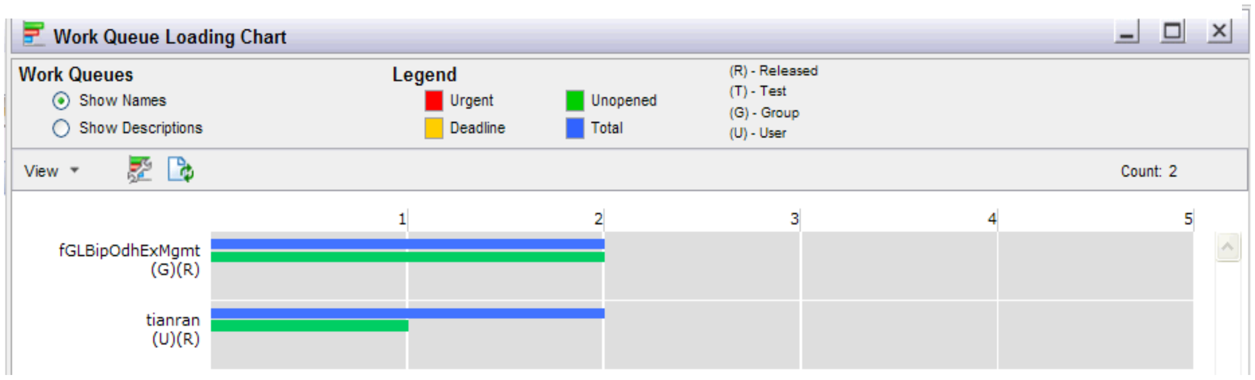
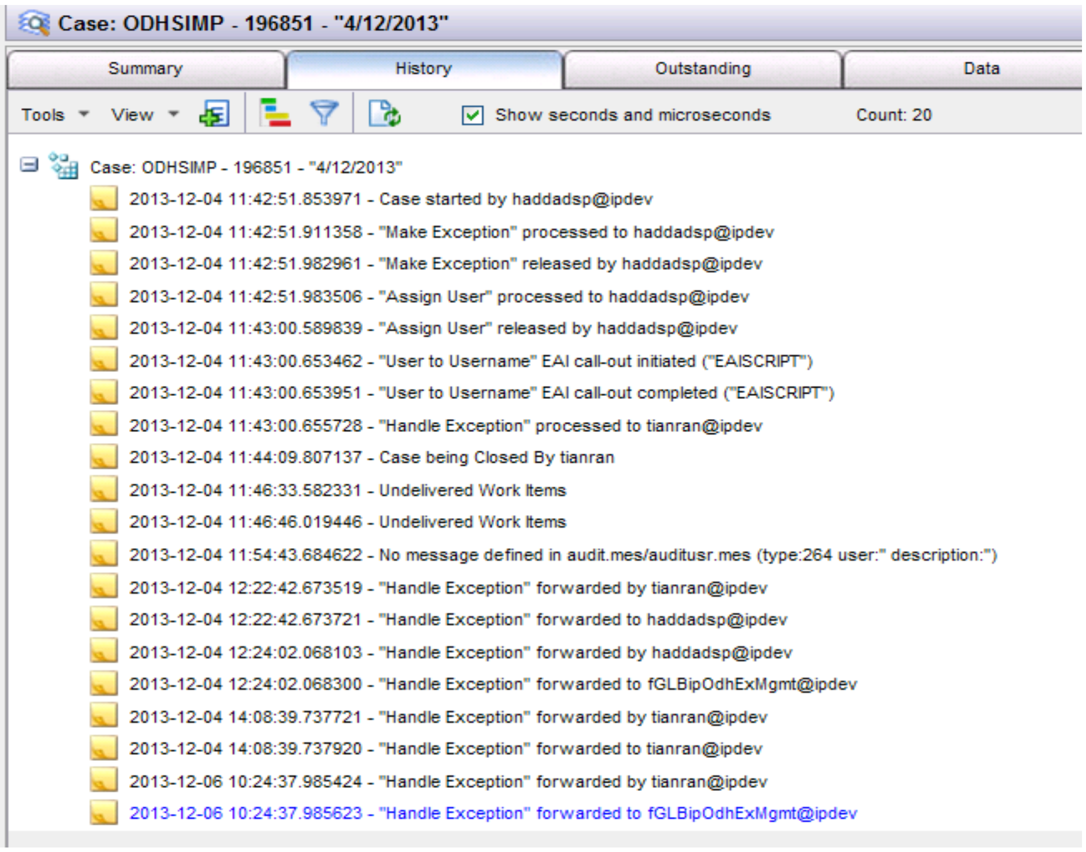

Work Queue Loading Chart

Figure 5 Graphical History – Work Queue

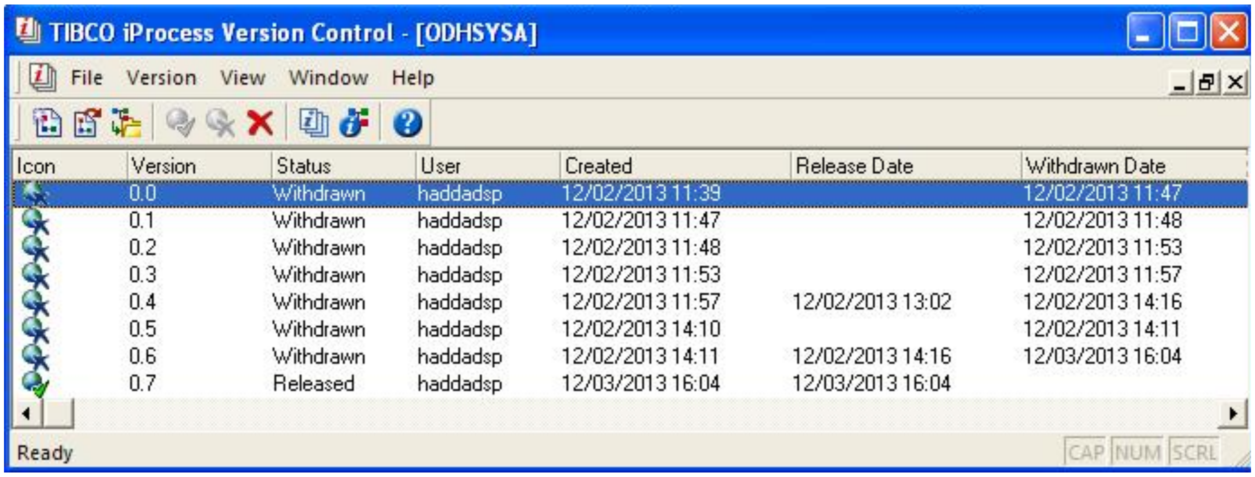
Developers and administrators also have special tools in Workspace in order to assist them in developing and maintaining procedures. They have access to a list of all procedures saved on to the deployment server, and can manually start a case for testing. Developers can also deploy procedures to a server without releasing it for general use. Ordinarily, when a procedure is deployed it is simultaneously released. A released procedure replaces any older version of the procedure on the server, and developers or applications can start cases of it. However, developers may choose to not release a procedure when deploying, which hides it from the list of procedures on a server, and only permits the developer to create cases from it. When these cases are created, it is sent to a "test" queue, which creates simulated users to receive the tasks. This allows a developer to easily test a procedure without having to rewrite their procedure to route each task to themselves.


Case: ODHSIMP - 196851 - "4/12/2013"
Summary
History
Outstanding
Data
Tools
View
Show seconds and microseconds
Count: 20

```
Case: ODHSIMP - 196851 - "4/12/2013"
2013-12-04 11:42:51.853971 - Case started by haddadsp@ipdev
2013-12-04 11:42:51.911358 - "Make Exception" processed to haddadsp@ipdev
2013-12-04 11:42:51.982961 - "Make Exception" released by haddadsp@ipdev
2013-12-04 11:42:51.983506 - "Assign User" processed to haddadsp@ipdev
2013-12-04 11:43:00.589839 - "Assign User" released by haddadsp@ipdev
2013-12-04 11:43:00.653462 - "User to Username" EAI call-out initiated ("EAISCRIPT")
2013-12-04 11:43:00.653951 - "User to Username" EAI call-out completed ("EAISCRIPT")
2013-12-04 11:43:00.655728 - "Handle Exception" processed to tianran@ipdev
2013-12-04 11:44:09.807137 - Case being Closed By tianran
2013-12-04 11:46:33.582331 - Undelivered Work Items
2013-12-04 11:46:46.019446 - Undelivered Work Items
2013-12-04 11:54:43.684622 - No message defined in audit.mes/auditusr.mes (type:264 user:'' description:'')
2013-12-04 12:22:42.673519 - "Handle Exception" forwarded by tianran@ipdev
2013-12-04 12:22:42.673721 - "Handle Exception" forwarded to haddadsp@ipdev
2013-12-04 12:24:02.068103 - "Handle Exception" forwarded by haddadsp@ipdev
2013-12-04 12:24:02.068300 - "Handle Exception" forwarded to fGLBipOdhExMgmt@ipdev
2013-12-04 14:08:39.737721 - "Handle Exception" forwarded by tianran@ipdev
2013-12-04 14:08:39.737920 - "Handle Exception" forwarded to tianran@ipdev
2013-12-06 10:24:37.985424 - "Handle Exception" forwarded by tianran@ipdev
2013-12-06 10:24:37.985623 - "Handle Exception" forwarded to fGLBipOdhExMgmt@ipdev
```

Figure 5 Example Audit Trail for Procedure ODHSIMP

One of the most important tools available to developers is the version control tool. When a procedure is selected in Workspace, its version history may be displayed, which details all of the previous major and minor versions, the times they were deployed and withdrawn from the server, and any notes the developer wrote for them when created. In the Windows version of Workspace, developers also have the option to open any previous version, edit it, and save it as a new version of the procedure. This gives developers the ability to quickly create updates to their procedures by slightly modifying previous versions or to rollback to a previous version entirely by opening the old procedure and saving it as a new version without any edits.


TIBCO iProcess Version Control - [ODHSYSA]
File Version View Window Help

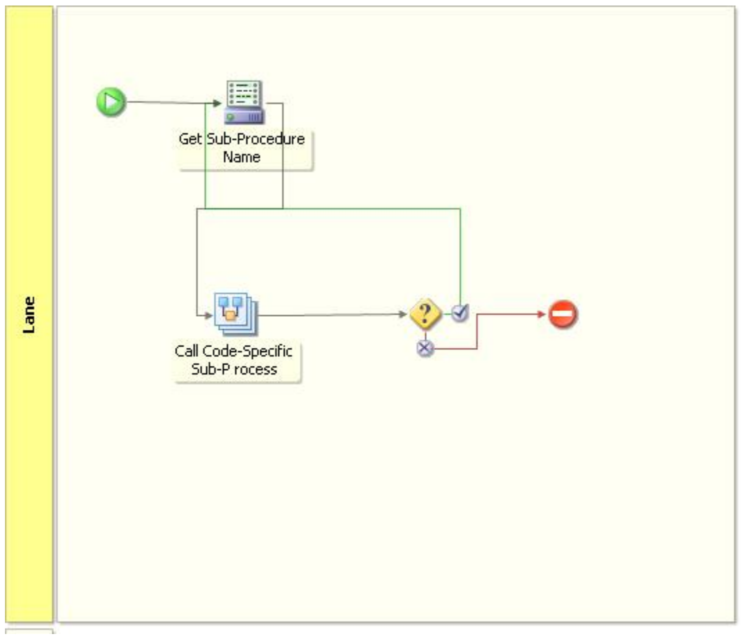
Icon	Version	Status	User	Created	Release Date	Withdrawn Date
	0.0	Withdrawn	haddadsp	12/02/2013 11:39		12/02/2013 11:47
	0.1	Withdrawn	haddadsp	12/02/2013 11:47		12/02/2013 11:48
	0.2	Withdrawn	haddadsp	12/02/2013 11:48		12/02/2013 11:53
	0.3	Withdrawn	haddadsp	12/02/2013 11:53		12/02/2013 11:57
	0.4	Withdrawn	haddadsp	12/02/2013 11:57	12/02/2013 13:02	12/02/2013 14:16
	0.5	Withdrawn	haddadsp	12/02/2013 14:10		12/02/2013 14:11
	0.6	Withdrawn	haddadsp	12/02/2013 14:11	12/02/2013 14:16	12/03/2013 16:04
	0.7	Released	haddadsp	12/03/2013 16:04	12/03/2013 16:04	

Figure 6 View of the iProcess Workspace Version Control Window

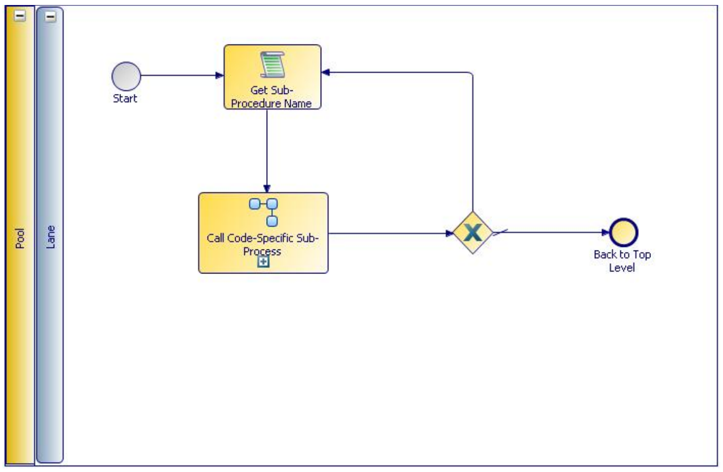
iProcess Modeler

Development in iProcess is done using software called Tibco iProcess Modeler, which is accessible through the Windows version of Workspace. IProcess Modeler, or simply Modeler, allows developers to create procedures by creating a flowchart of several tasks, and specifying the details of these tasks. Modeler then allows the developer to save the procedure to the deployment server, where it then can be released as either a new procedure or an updated version of an already existing procedure.

Procedure developers also have an alternative environment in which they can work. Tibco Business Studio, a series of plug-in for the Eclipse Integrated Development Environment, provides support for creating iProcess procedures. Compared to Modeler, Business Studio provides a more refined interface for modeling procedures, and gives access to several of Eclipse's features, including remote database access, connections to multiple deployment servers, and the ability to write Java and JavaScript code which can then be integrated with the procedure. However, Business Studio does not support all of the features of iProcess, and as a result some of the functionality present in Modeler is impossible to replicate. Some examples of features Business Studio lacks include: the ability to customize forms for user tasks, the ability to update a procedure with a new major version (only minor version updates are permitted, and the ability to create and add Memo fields.



(a)



(b)

Figure 7 Comparison of the same procedure in (a) iProcess Modeler and (b) Tibco Business studio

Fields and Tasks

This section describes the function of the most commonly used tasks in iProcess, along with the various fields which are used to provide functionality to these tasks.

Error! Reference source not found. shows a procedure containing all of the types of steps escribed below. From left to right, they are: a start step, a user task, an EAI call to a script, a conditional splitter, an EAI call to a database (top), a complex router (middle), a trigger step (bottom), and wait step, a sub-procedure call, and an end step.

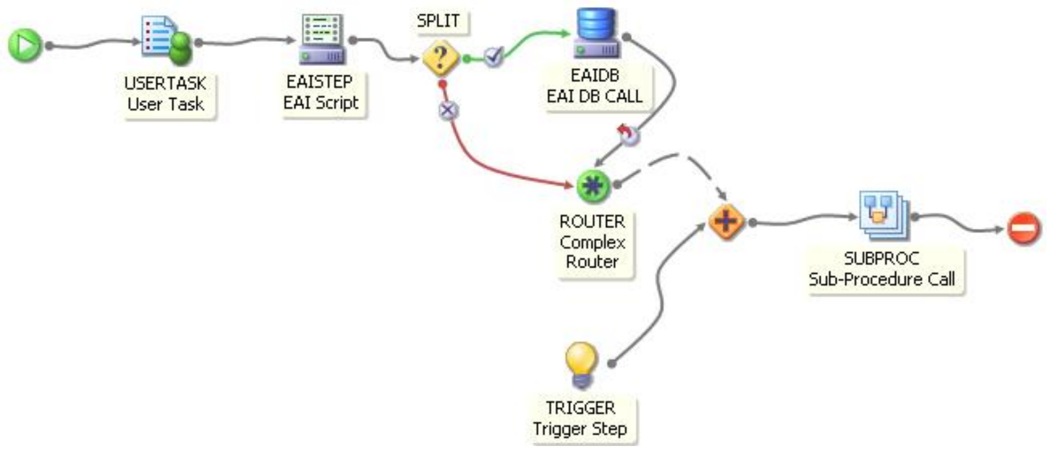


Figure 8 Procedure depicting the different steps in iProcess

Fields

Fields are the various data items stored in a procedure used to pass data between the various tasks. A procedure can have as many fields as is necessary. When a field is defined, it can be mapped as an input or output to many of the different types of tasks, which allow them to be read, edited, and updated.

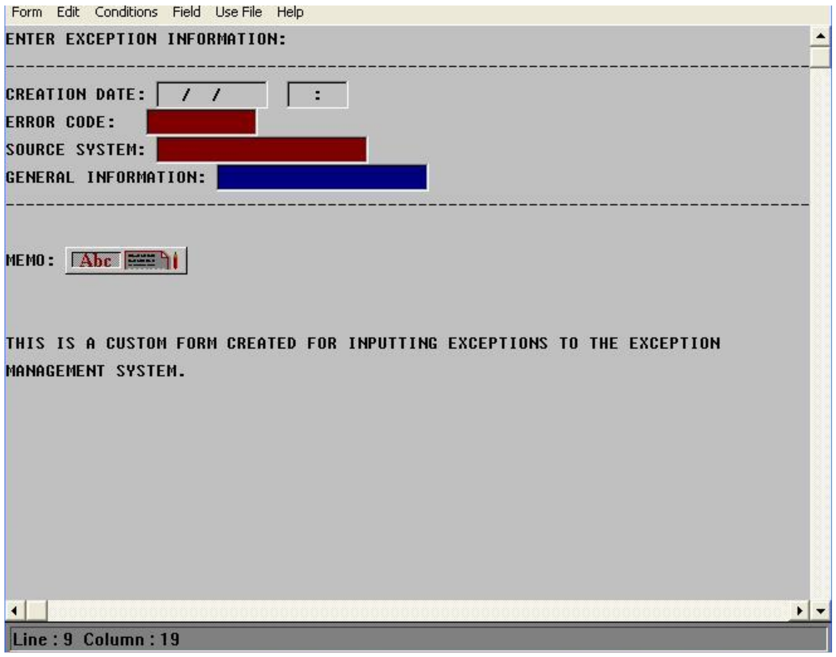
The types of fields are text, numeric, comma separated numeric, date, time, memo, and composite. Text fields are similar to strings in programming languages, and are groups of characters that must be under a certain length determined when the field is defined. This length can be anywhere from 1 to 255 characters. Numeric fields are analogous to floating point values in programming languages, and are, "Any number, positive or negative, up to the length and number of decimals you select (maximum length 18 including decimals, decimal point and sign, with no more than 8 decimals)."(Reference To iProcess Modeler Basic Design) Comma separated numeric fields are identical to numeric fields, but the thousands are separated by commas (for example: 123,456). Date and Time fields are used for storing dates and times, respectively. The formats for each are "dd / mm/ yyyy" for Date fields and "hh:mm" for Time fields (using a 24 hour clock). Memo fields are used to store larger amounts of text than a text field, which is stored in a separate file from the main procedure file, unlike the other fields. Composite fields refer to iProcess tables, which can hold data of different field types within one table to be accessed later.

Fields can either be defined as a single instance, or as an array of that field type. When an array is created, an integer field is also defined as the index for the array. This is used to help iterate of arrays in tasks. The names for these index fields follow the convention <Field Name>_IDX. In addition, every process automatically defines a general index, SW_GEN_IDX, which can be used to iterate over multiple arrays.

User Tasks

User tasks are the visible tasks of iProcess. These are the tasks which are sent to work or group queues to be completed by humans. The specific queue(s) which the task is sent to can either be predefined by entering the name in Modeler, or be input as a text field which will contain the name of the queue if the proper user or group will be determined at run-time.

When the user task is opened in Workspace, it generates a form which the user can use to view and edit data. A developer can specify fields from the procedure's workspace to populate the form, allowing the user to view the value of the field, edit it, or calculate a new value by inputting some additional data that is then used by an internal script. In addition, a developer can write text directly onto the form, in order to create messages, or dynamically hide and show various sections of the form based on rules defined using JavaScript scripts embedded in the form. These are features are useful in cases where a user may have to enter different values into a form based on previous inputs. For example, a sales application may show different fields depending on whether the user specified they wanted to look in a certain department.



ENTER EXCEPTION INFORMATION:

CREATION DATE: / / :

ERROR CODE:

SOURCE SYSTEM:

GENERAL INFORMATION:

MEMO:

THIS IS A CUSTOM FORM CREATED FOR INPUTTING EXCEPTIONS TO THE EXCEPTION MANAGEMENT SYSTEM.

Line : 9 Column : 19

Figure 9 Modeler View of an iProcess Form

Error! Reference source not found. shows the developer's view of an iProcess form. It displays the date and time for the exception's creation date, and asks the user to input the error code, source system, and optionally some information about the exception.

After a user has completed all the steps inside a form, they can either "keep" a form, closing it temporarily but still saving the data they entered and keeping the user task in their queue, or "release" the form, ending the user task and progressing on to the next step, updating any fields that were edited in the form.

EAI Steps

Enterprise Application Integration (EAI) steps are tasks which allow iProcess to communicate with external software, and use their functionality to enhance iProcess'.

EAI steps are comprised of a number of sub-types, which provide functionality with different types of software. These subtypes include: database calls, script calls, Java client calls, and emails. These tasks can be performed immediately when the procedure reaches them, or they can have a delayed release where iProcess waits some predetermined amount of time before performing the task.

Database calls allow iProcess to access an external SQL database by calling a stored procedure in the database. Parameters in the stored procedure can be mapped to fields in the iProcess workspace, allowing a procedure to update tables or receive information via queries inside the stored procedures. In order to access a database, the target schema must have a database link that iProcess can access at runtime.

Script calls are automated tasks which allow a procedure to automatically perform some action without user input. These scripts can be written either in JavaScript (when using Tibco Business Studio) or in iProcess' own scripting language (when using the Windows version of Workspace). These scripts can use the fields in the procedure's workspace as variables, and automatically return any edited values when the step is completed. This allows procedures to calculate field values in the background.

Java client calls give procedures the ability to implement Java code by calling a method within a class. When a Java client EAI step is created, the developer must first specify a Java class or factory to call at the step. When the EAI step is reached at runtime, the constructor for the class or factory is called, and a new object of the class' type is created. Classes that are to be called by iProcess must implement one of the custom interfaces provided by Tibco. Once the class is created, the specific method within the class is called. Inputs and outputs from the method may be mapped to fields in the procedure. The Java method then runs until completion, at which point the task is complete and the procedure continues.

Emails allow iProcess to send automated emails as part of a procedure. The body of an email is created in development, and fields can be embedded as to customize the email based on data in the procedure. The recipient can either be hard-coded as to send an email to the same address in every case, or can be mapped to a text field to create a dynamic destination.

Trigger Steps

Trigger steps act as parallel paths in a procedure, creating a new sequence of events in addition to the path started by a start step. The trigger step, and the tasks that follow it, are not began until the case is "triggered", either by manually triggering the procedure in Workspace, or by another task in the procedure. When a procedure is triggered, any trigger step inside it is automatically started, and the tasks following it are performed. Trigger steps are useful for handling rare cases that a procedure ordinarily would not have to be concerned about. In addition, triggering a closed case will cause the case to reopen at the trigger step, which can be employed to restart or review a certain case.

Router, Condition, and Wait Steps

Procedures in iProcess have the capability to create branches in their step order, allowing them to split their workflow between one of two paths, or to run multiple simultaneous chains of tasks. To achieve this, iProcess provides three steps for splitting and merging paths: the complex, condition, and wait steps.

Complex router steps are steps which can take in and send out paths from multiple tasks at once, compared to other tasks and steps which are only capable of only receiving and sending paths to one task at a time. Paths that are output from a complex router are progressed through simultaneously, allowing a procedure to process multiple tasks at once. However, if one of the parallel branches reaches an end step, the entire case is closed and other tasks are resolved automatically. Using complex routers to merge split paths causes whatever branch that reaches the router first to continue, and other tasks will end at the router. This is to prevent multiple instances of the same task from being released.

Condition steps are special splitters which divide one path into two. Unlike complex routers, however, condition steps only choose one of the two paths to proceed upon. The decision of which path to choose is based upon a simple script, which returns a Boolean value. If the value is true, the procedure continues across one path. Otherwise, it proceeds along the second path. Condition steps have a large amount of uses, from creating loops in procedures to deciding which of two sub-processes to call.

Wait steps are another type of in iProcess which can have multiple input and output paths, along with complex routers. Unlike complex routers, however, the wait step does not automatically continue when one path reaches it. Instead, the wait step holds the procedure until all of the parallel branches have reached it. This ensures that multiple parallel tasks are all completed by the time the wait step is reached in the procedure.

Sub-procedure Calls

In iProcess, there is support for two types of procedures. First are main procedures, which can be started by creating a new case in workspace. The second are sub-procedures, which can only be started from within a main procedure or another sub-procedure by a sub-procedure call task. When a sub-procedure call task is reached, the designated sub-procedure is begun, and the main procedure then waits at this step until the sub-procedure continues running. Using sub-procedures, it is possible to repeat several tasks more easily than using a loop, or to call a specific sub-procedure depending on data in the case.

A major difference between main and sub-procedures is that when a sub-procedure is called, the main procedure is capable of passing along some fields as input to the procedure, which can be used to populate fields inside the sub-procedure. Likewise, sub-procedures can return data to the main procedure once they are complete, updating the main procedure's fields. This is created by mapping fields in the main procedure to the inputs and outputs of the sub-procedure in the sub-procedure call.

There are two types of sub-procedure calls. The first are static sub-procedure calls, which are calls to the same sub-procedure every time the main procedure is run. These are useful for breaking down larger procedures into smaller sections for easier maintenance, or for when there are multiple procedures which carry out similar tasks. In addition, sub-procedures can be altered and maintained independently of the main procedure, which allows a development team to have greater control over updating a system of procedures.

The second variety of sub-procedure call is the dynamic sub-procedure call. Dynamic calls allow a main procedure to call a variety of different sub-procedures, or even multiple sub-procedures simultaneously, based on events within a case. Dynamic calls, along with the benefits provided by static calls, also give a procedure greater flexibility, allowing it to handle a wider range of circumstances by calling specific sub-procedures for specific needs. Dynamic calls are done in iProcess by passing an array of sub-procedure names to the sub-procedure call task. When the task is reached, it calls all of the sub-procedures listed in the array, and waits until they have all completed to continue in the main procedure. The inputs and outputs to the sub-procedures must be provided as arrays of the same type as the parameters defined in the sub-procedures.

When multiple sub-procedures are called, they must all have similar inputs and outputs. To achieve this, iProcess uses an element called an interface, which defines the inputs and outputs that a sub-procedure implementing it must provide. When a dynamic sub-procedure call is made, it checks the interface it was given, and uses this to find the list of appropriate sub-procedures it can call.

The use of dynamic sub-procedures was a major component in the creating the exception management framework. But defining a dynamic call based on data within an exception, it allows the framework to properly route the exception by simply calling the sub-procedure designated to handle that specific exception.

Accessing iProcess from external systems

A major feature of the iProcess suite is the ability to access the business process manager's internal from non-iProcess applications. This is accomplished in iProcess by providing a variety of plug-ins, server objects, and application programming interfaces (APIs) that enable communication with various types of applications.

Following are two of the more frequently used tools for communication between iProcess and external applications.

JAVA/.NET Server Objects

In order to allow applications created to access the full functionality of iProcess' procedure management, it provides a group of objects which are stored on the server which the iProcess engine and procedures are kept. External applications can be implement these objects, which creates an API for it to communicate with the iProcess server. These objects encapsulate several methods that allow programs written in the Java programming language or using Microsoft's .NET framework (for example, programs written in the C# programming language), to view procedures, start cases, and complete user steps. The objects also allow a program to obtain any data about a procedure that it may need.

The server objects encompass classes which represent the various items that can be found on the iProcess server. This includes procedures, their cases and steps, any work queues and the work items inside them, the users and groups on the server, and a node object which encompasses all of the objects on the server. By calling the methods inside the correct class, a program can manipulate workflows and cases within iProcess with a large degree of flexibility. The classes provide support for creating cases, forwarding, locking, and unlocking work items and steps, viewing data about a step (for example, the user assigned to the step or the fields it has access to), managing users and groups, and more. The result is now an external program can use iProcess with the same amount of freedom as a user with Modeler and Workspace.

Web Services Plug-in

Like the Server Objects above, iProcess also provides two APIs that allow online applications to use iProcess' functionality to expand their own. These APIs provide access for web services which use the Representational State Transfer (REST) and Simple Object Access Protocol (SOAP) protocols to interact with an iProcess server, and perform similar functions to those provided by the Server Objects. This allows a web service to run and manage procedures independently of iProcess, which presents a method of creating a customized user interface.

Limitations of User Interface

After researching about iProcess Workspace (Browser) and its implementation during the prototype, we identify the following limitations:

- Cannot customize exception list

The list that shows exception, or "work items" in a "work queue" in iProcess, has fixed columns which does not describe an exception in an efficient way.

- Inefficient to manipulate work items

When a user forwarding a work item (exception) to another work queue (user or group), the work item would disappear from users own work queue so that he/she is not able to see the information about that item unless someone forward it back to the original user. Similarly, after an exception is closed by user, it would be released from the user's work queue so that the user will not be able to retrieve the information.

- Customization of Access Level

iProcess Workspace (Browser) have access levels that are already set-up. To customize the access level, developers must connect to the central server, which our project timeframe does not allow us to. We would like to have a special access level for team managers, who could see the teams work queue, an access level for team members, who could only see their own work queue. Certainly, different group may customize their access level to meet their business needs.

Framework for an Exception Management System

The iProcess business process manager application is designed with the purpose of creating a flexible suite of procedures and sub-procedures for handling the numerous and complex workflows that a large business requires to function. This capability makes iProcess well-suited to being adapted as an exception management system.

The basic framework for exception management is made of three components: a collector, a router, and a sub-routine. The collector is responsible for scanning the source systems and detecting when one raises an exception. It then records any information given about the exception, and passes this on to the router. The router then analyzes the information about the exception, and from it determines which sub-routine to pass the exception along to. Finally, the sub-routine is a custom workflow specifically meant to handle this particular type of exception, or a particular group of exceptions. From here it can either go through the steps necessary to close the exception, or route the exception into yet another sub-routine. This is then repeated until the exception is successfully resolved. Performing the latter allows a management system to have a tree-like network of sub-routines, which the exception will traverse until reaching the correct one. This has the benefit of making the exception management system modular, making it easier to maintain and expand.

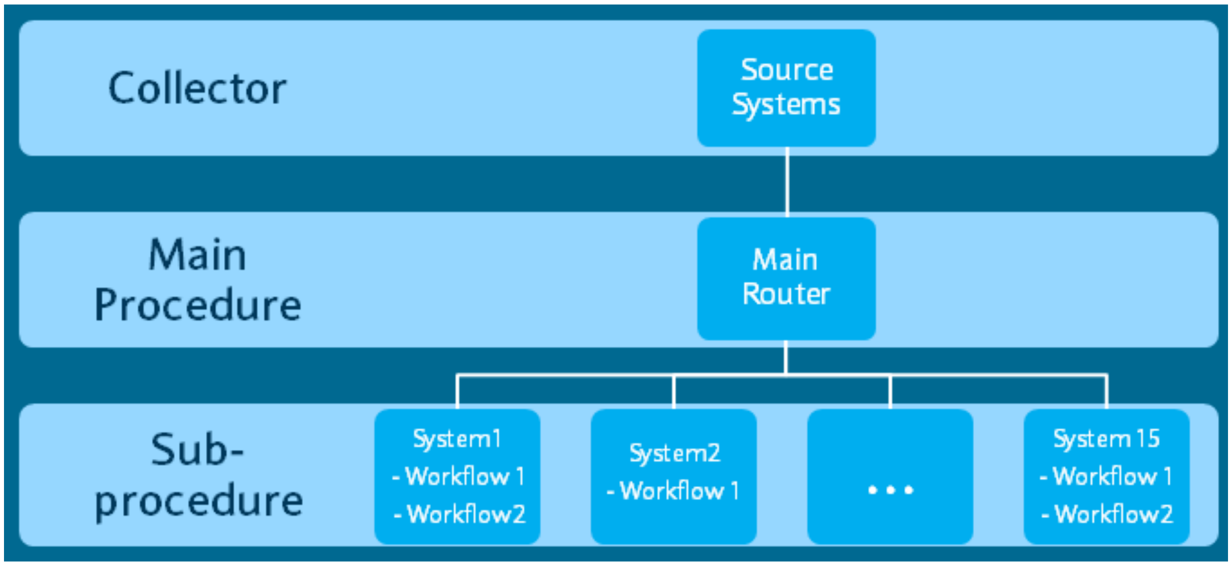


Figure 10 Diagram of the Exception Management Framework, and its analog in iProcess

In iProcess, the described above is possible to implement by utilizing main and sub-procedures to act as the router and sub-routines, respectively. The collector section would be developed as a separate application which is included as part of the source system. This application would be called when an error was raised by the system, and then collect data on the error to form the exception. Using either iProcess' Java or .NET API, the application could then start a case of the main procedure for exception management, and pass along the information on the exception as data inputs.

The router would be a main procedure in iProcess. The procedure would parse the inputs from the collector, and generate the name of the proper sub-procedure, and call it. The sub-procedure then would either route the exception further to another sub-procedure, or resolve it, ending the process and closing the exception.

Developing the system in iProcess allows exceptions to be managed individually simply by adding a sub-procedure designed specifically for one type of exception. The version control feature makes it possible for a sub-procedure responsible for a particular exception is able to be easily added or updated, without compromising the rest of the exception management system. The ease of creating procedures and sub-procedures in iProcess makes it ideal for implementing this framework.

Prototype Project

In order to demonstrate the capabilities of iProcess and how it could be implemented as a tool for business exception management, a library of procedures and sub-procedures were developed. This library was created as a method of presenting some of the facets of iProcess, especially those that were most relevant to creating an exception management framework. The library consists of one main procedure and two sub-procedures. In the presentation, the team hoped to demonstrate the function of user steps, iProcess scripts, sub-procedure calls, and how the iProcess engine progresses through a procedure.

The main procedure of the library, ODHDEMOP, was designed to emulate the "collector" and "router" portion of an exception management system. Because the team did not have access to a system which could provide actual exceptions for use in testing, the main procedure was designed to be completely closed-loop; having no communication with any system besides iProcess. As a result, the collector section had to be manually performed in the demonstration. This was done by adding a user task to the main procedure, where the fields of an exception (the error code, source system, and creation date and exception information) are input manually. In a full exception management application, this step would be replaced with a program which would monitor a system for exceptions, then create a new case of the exception manager procedure with the fields as an input. While not indicative of how an actual system would behave, during the final presentation this had the benefit of showing viewers exactly what the components of an exception were.

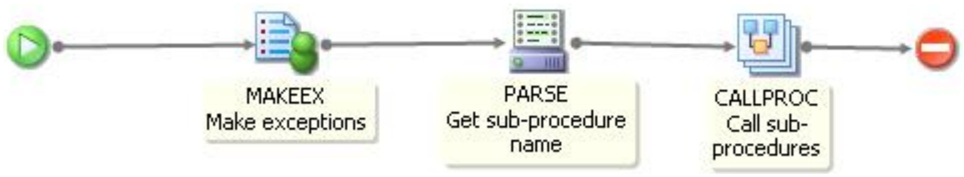


Figure 11 Modeler view of ODHDEMOP

The router section of the procedure, however, was implemented similarly to how it would be in a full application. The section consists of two steps in iProcess; a script step and a sub-procedure call. The script step takes the source system name from the collector, and concatenates it to the general prefix assigned to all the procedures developed in this project: “ODH”. The resulting string is then added to the array of text fields used in the sub-procedure call step. The sub-procedure call step is then reached. This sub-procedure implements an interface which takes the four fields that make up the exception’s data as input. Because the array of sub-procedures to call can only ever have one item in it, due to the nature of the procedure, the step only calls the one sub-procedure specified in the collector section. When the sub-procedure is complete, this signals that the exception has been closed, and the main procedure ends.

The two sub-procedures, ODHLDN and ODHNY, are meant to simulate two systems, LDN and NY, that each raise an exception. Both of the sub-procedures have a workflow meant to show different forms of data manipulation in iProcess. ODHLDN is a single path that has three steps. First, a script task assigns the name of the default user (in the presentation, “haddadsp”) to a text field. Next, a user task creates a form asking for the default user to reroute the task to a different user by entering their name into the form. This is meant to simulate a manager reassigning an exception to another user or group (without forwarding), who then performs some manual task to resolve the exception and confirms it with iProcess. It also gives the option to change the status of the exception from “OPEN” to “CLOSE”, by editing a text field. Once the form is released, it is passed a condition step that checks whether the user assigned to the form is different from the default, and whether the status of the exception is “CLOSE”. If both of these conditions are met, the sub-procedure reaches the end and is complete. Otherwise, it loops back into the user step. This cycle then repeats until the conditions are fulfilled.

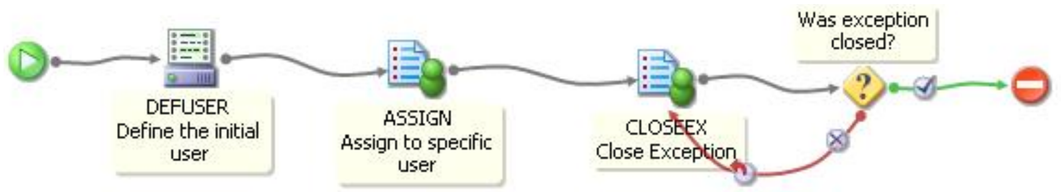


Figure 12 Modeler view of ODHLDN

ODHNY, the other sub-procedure, is designed to represent an application for detecting and fixing incorrect trade dates being entered into the NY system. The first step is a script which records the exception's creation date into a separate field for use later in the procedure. The next step is a conditional splitter. This condition uses the exception's code to choose which path to follow. If the code is "d404", the procedure determines the exception is for a wrong trade date. The procedure then moves to a user task which takes the correct input as a date. If the date entered is the same as the original incorrect date (which was saved to a separate field in the script task), the procedure loops back into the user task until a valid date is entered. The exception is then closed. If the code is any other value, the procedure concludes that the exception was falsely sent, and asks a user to review the exception information before closing it.

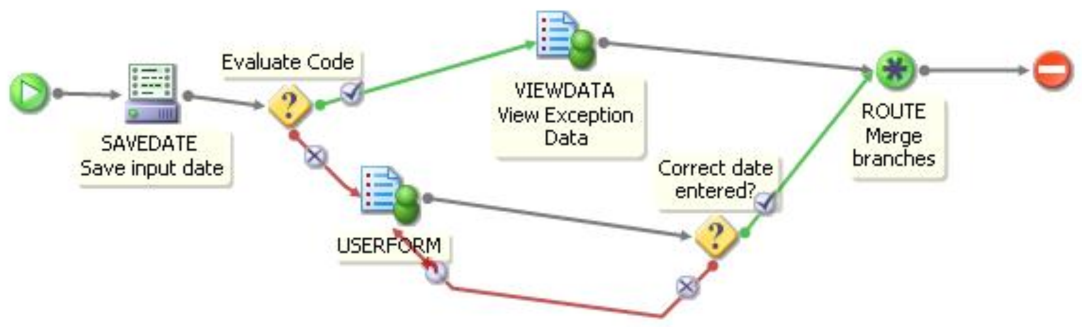


Figure 13 Modeler view of ODHLDN

A full demonstration of these three procedures gives a practical demonstration of how to create an exception management system in iProcess. It shows how a case in a procedure is started, the function of the more commonly used tasks, and how a procedure progresses through these steps. The demonstration also gives the opportunity to show some other capabilities of iProcess. A memo field can be added to the procedures to show how they can be edited within a form. The work queues can be locked, either manually or by one user in a group queue accessing the form. And lastly, the work items can be forwarded to demonstrate this functionality. All in all the demonstration gives a comprehensive list of iProcess' features, and advocates using its use as a framework for creating exception management application.

Recommendations and Future Directions

Code assigned to exceptions

In order to completely integrate the framework the team designed for an exception management system, a group will have to define error codes for any exceptions they wish to monitor. This entails listing all of the exceptions they would want to handle with the exception manager, and creating an error code to each. This code could be any alphanumeric value, from simple integers to detailed names. For example, a source system coded with “ABC” could have an exception “ABC001”, indicating missing static data; or “ABC002”, representing wrong data within trade information. This code would then be used by the router as part of the parameters for parsing the exception and finding the correct sub-routine. We could set a rule, by programming, such as “group A will take all the exceptions from source system ABC”, and then exceptions with code prefixed by “ABC” would be routed to group A.

The next step would be to modify the source system to properly send these codes when an exception is found. This requires the system to first realize that the issue exists (possibly by an exception being returned in its own code or creating a flag to catch the mismatch), looking up the code, and then sending the call to the iProcess API with this code.

Overall, the time and labor requirements to fully implement these codes would be fairly intensive. However, the exception management system will benefit greatly from the extra parameter to parse, and for sub-procedures to use as a field.

Communication With External Applications

As found in research, iProcess has a wide variety of methods with which it can integrate itself with another system or program. It has Java and .NET APIs for other applications to start and manage cases inside it, allowing the developers to simply integrate iProcess by utilizing these objects. IProcess has EAI steps designed to call outside systems, including the ability to invoke stored procedures on external databases, so it can run queries and add data to tables from a procedure. EAI steps are also able to call a method inside a Java class to run Java code in the background. Lastly, iProcess has the ability to simply send emails as part of a step, which brings literal communication with outside groups.

Using these capabilities of iProcess will allow developers to utilize iProcess' business process management abilities without majorly compromising their already established systems, and to expand iProcess by adding the functionalities of the other systems.

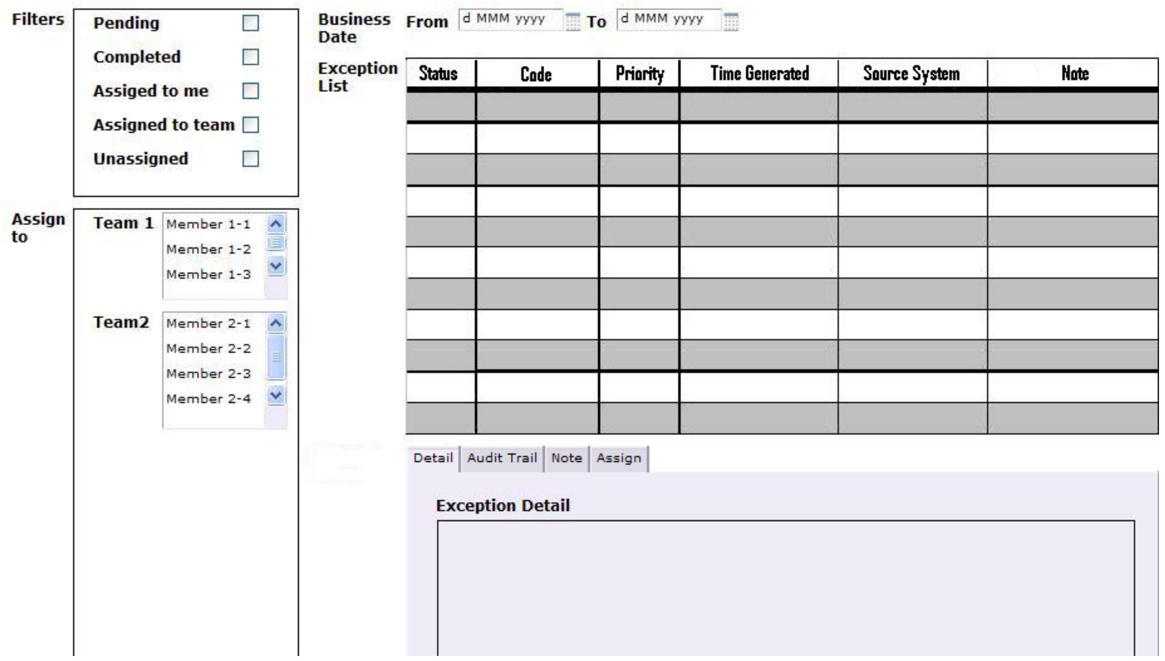
Define User Access Levels

One of the major limitations of iProcess, in particular the browser version of iProcess Workspace, is the simply defined permissions for the different types of user. Most significantly, the general user has greater access to the internals of iProcess than is necessary or practical. By default, a general user can view all of the procedures stored on the iProcess server and view activated cases. They also can view information about cases in their work queue, and can edit some of this data. Field values in a case, even if they are not used in the work item they are accessing, can be edited from Workspace. This poses a serious concern for the stability of the procedure, as incorrect field values may cause the case to terminate prematurely, or end with incorrect data. Either way, this may mean the procedure ends while the exception which started it still exists in the system.

By defining custom user access levels, administrators can prevent users from accidentally altering data they should not have access to. In addition, preventing users from accessing some of the tools in iProcess provides a cleaner, more compact user interface for them to interact with.

Customized UI

As we mentioned earlier, the iProcess Workspace has certain limitations when used as the user interface for the business users while there are more functions that are not necessary when handling the exceptions. When recommend the development team build a customized UI using TIBCO



Business Studio or other applications. Below is a form we build using Business Studio but have not implement into our framework due to the time limitation. We want to use this as an example to demonstrate the following recommendation regarding user interface for the application:

- Columns of the exception list

We want to describe an exception using these six fields: “Status”, “Code”, “Priority”, “Time Generated”, “Source System” and “Note”, in which shows the memo updated by the person who opened the exception. These six fields contain the crucial exception information that we would like to our users at a first glance.

- Business Date

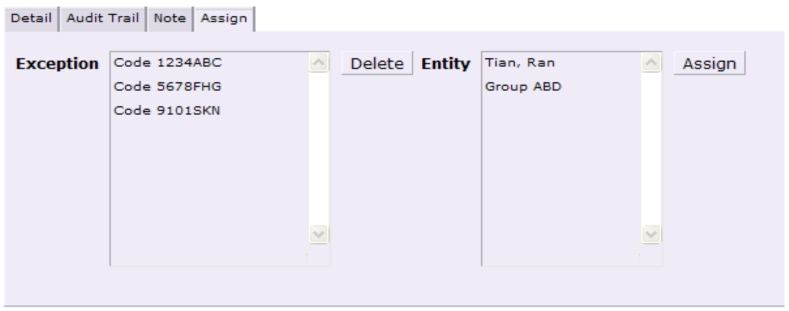
It would be handy for the user if there is a special filter for the business date right above the exception list.

- “Filters” pane

Customized the filters for different business purpose. Different user group may have their own filters apply to the exceptions.

- “Assign to” tab

This tab shown below allows user to forward multiple exceptions to more than one user or group at one time.



Code 1234ABC
Code 5678FHG
Code 9101SKN

Tian, Ran
Group ABD

Figure 15 Recommendation of UI – the “Assign to” tab

Reflection on the Project

Design component

As we discussed in the paper earlier, during our project, we designed the routing logic of the exceptions to make sure the exceptions could be routed to the right entity; also, for testing and demonstrating purpose, we designed a exception generator that creates dummy exceptions which will be send to Barclays' server; as a suggestion for future direction, we designed a user interface of the application, which is more user-friendly to both business user and development team.

Constrains and alternatives

The biggest constrain we encountered is that we had limited access to the Barclays central server of iProcess. This means that we could not use real-life case (exceptions) as our tester for the exception management application. The alternative we chose is to create the exception by ourselves – we designed the exception generator, which would create exceptions with information such as time generated, source system and description. The generator would send the exception to the server and then we could test the application and also use our dummy exceptions during the demo.

Need for life-long learning

Overall, our 7-week project experience with Barclays is priceless. The environment there was completely different from what we have in school. We had a chance to solve a real-life problem with people who work "in the field". We realized there are more factors that would influent our project progress. So we had to be very clear with Barclays' needs from time to time and make adjustment to our daily/weekly goal properly. Also, we learnt that it is important to be proactive and ask for help. Though people work there at Barclays are all very busy, they are all very patient when we asked questions. People say "It is not embarrassing when you ask questions about things you don't know; it is embarrassing when you make mistakes because things you don't know and you didn't ask." I think the 7 weeks with Barclays really help us understand this point.

As student with different academic background, we both learnt how to work with each other by "translating" the language within our major field to words that everybody understands. We also got a great chance to practice this when we preparing our final presentation to two different audience groups: business analyst and development team. We believe the all those skills we gained will be very helpful in the future when we left college and start the new chapter of life.

References

Barclays, *"Annual Report 2012"*. Retrieved 16 March 2013. <http://reports.barclays.com/ar12/servicepages/downloads/files/entire_barclays_ar12.pdf>

Barclays, *"People – Barclays Annual Report 2012"*. Retrieved 2013-07-14. <http://reports.barclays.com/ar12/governance/people.html?cat=m>

Barclays, *"Barclays Purpose and Values"*. <http://group.barclays.com/about-barclays/about-us/transform/values>

Barclays, *"Our History"*, <http://group.barclays.com/about-barclays/about-us/our-history>

Global Finance, *"World's 50 biggest banks 2012"*. August 27 2012, Retrieved 28 April 2013. <http://www.gfmag.com/tools/best-banks/11986-worlds-50-biggest-banks-2012.html#axzz2RlGUpWAO>

Rubin, K. S. (2012). *Essential Scrum: A Practical Guide to the Most Popular Agile Process.* Addison-Wesley Professional

Defining Fields. *IProcess Modeler Basic Design.18. IProcess Modeler Basic Design.* TIBCO Software Inc. Web.