Functional Specifications of the Group 4 Inc. Workflow Interpreter and Management System (WIMS)

Group 4:
Deepak Nalla
Joseph Kotzker
Elliot Linder
Corentin (Corey) Rejaud

Instructor: Professor Alexander Borgida
Team Name: Group 4 Inc.
Date: 09/30/2016

1. Non-Functional Specifications

1.1 Introduction

According to a Gartner report, Workflows are a $2.3 billion industry [1]. Also, a publication by the University of Albany defines a ‘Workflow’ simply as the movement of documents and tasks through a business process. Furthermore, they say ‘Workflow Management Systems’ allow organizations to define and control the various activities associated with a business process[2]. Workflows are used in many industries including scientific, business, medical, Information technology and education.

The aim of our project is to create a workflow language based on set workflow patterns. The language supported by our platform will allow an administrator to create workflow templates and end users to instantiate new workflows or act on existing workflow instances.

The main goal of our workflow language is to allow administrators to have a quick, easy, and simple way of systematically orchestrating repeated patterns of business activity for their end users. We want to help administrators assist their end users with transferring materials/data, services, and information processing.

Some examples of workflows that can be created with our language:

- Special Permission Number (SPN) request workflow for students in universities
- Package/order tracker workflow for electronic commerce companies with shipping abilities

We expect our language to be:

- Extensible
- Configurable
- Easy to read / human-readable
- Easy to learn and understand - by reading our documentation
- Simple
- Robust

References: [1] Professor Borgida for (Gartner Report, Workflows)

[2]

https://www.ctg.albany.edu/publications/reports/workflow_mgmt?chapter=2&PrintVersion=2

1.2 What does it do?

Someone proficient in this workflow language (i.e. an administrator) will be able to use this language to create workflows for its end users. Administrators must first look at our documentation of our language to learn it, then write up a text file with our language, and upload it to our platform to create a workflow template. The administrator's end users will be able to log in and act upon these workflow templates.

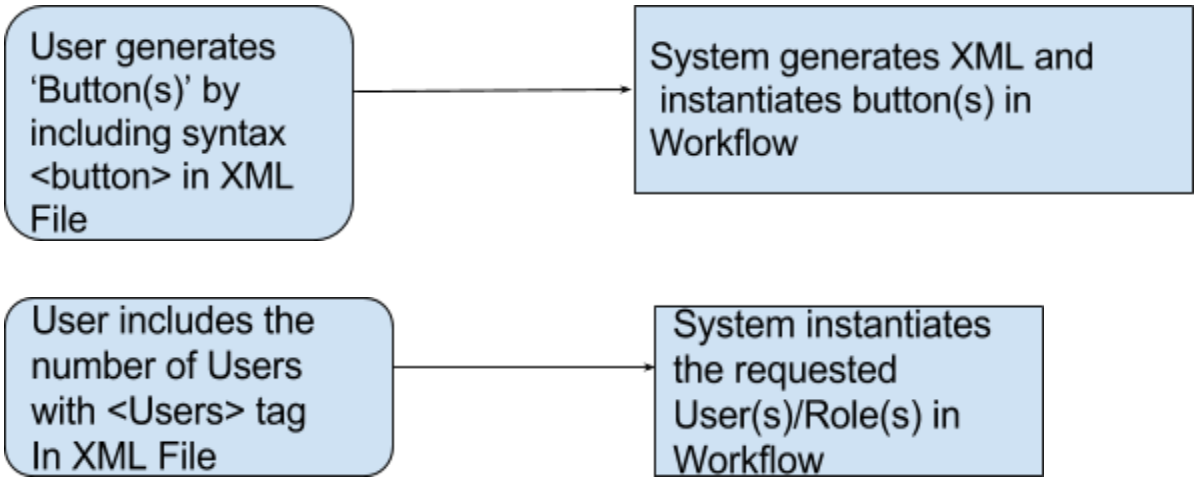
1.3 Non-functional requirements

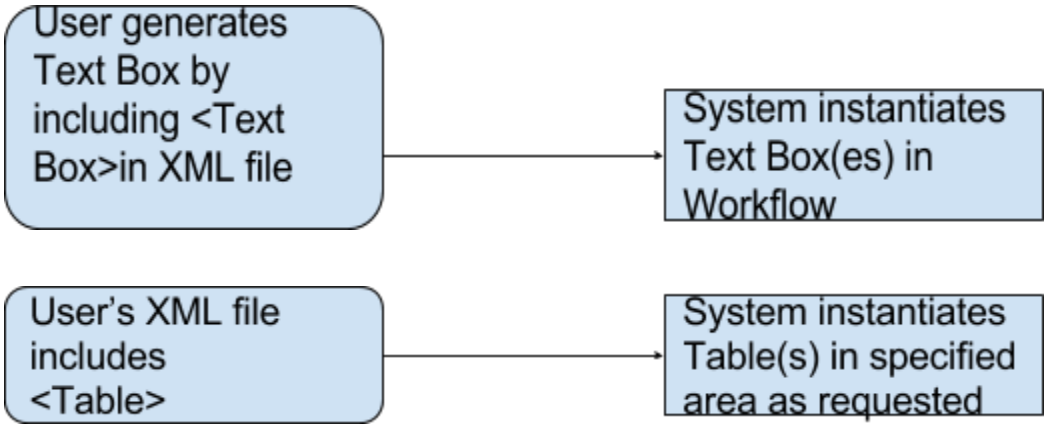
Our project will work on all main operating systems: Windows, Mac, and Linux. Extensibility is important because we want this to work on all main platforms. Specifically, the operating environment will need to have Java 8. Any computer should work: Desktop,

Notebook/Laptop, Servers, Smartphones, and Tablets. This device will need internet connection in order to log in and complete a task for a specific workflow (e.g. handing off a workflow to another user or completing a task), since a task will talk to the database via internet connection. Persistent internet connection is not required, but is recommended. Our project will let the user know if a transaction has not been successful due to a fault with the internet connection.

1.4 Designing the User Interface for a Workflow

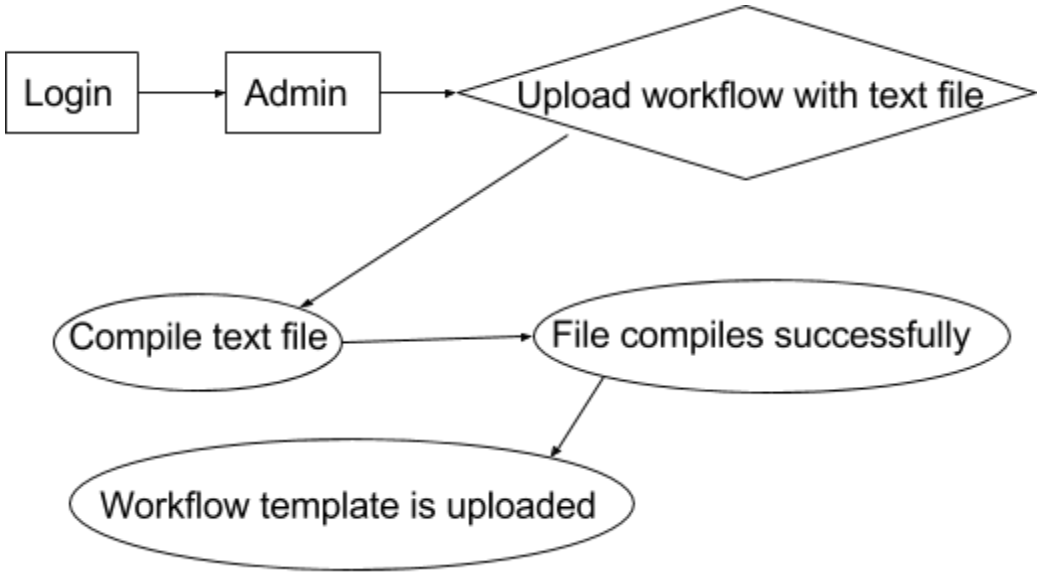
The workflow programmer will be able to design the User Interface (UI) for the other users who are going to be also accessing the workflow. Our Workflow Interpreter and Management Software will allow the workflow programmer to upload an xml document with the exact specifications of the UI using a given list of objects to work with including buttons, text boxes, list views, and tables.



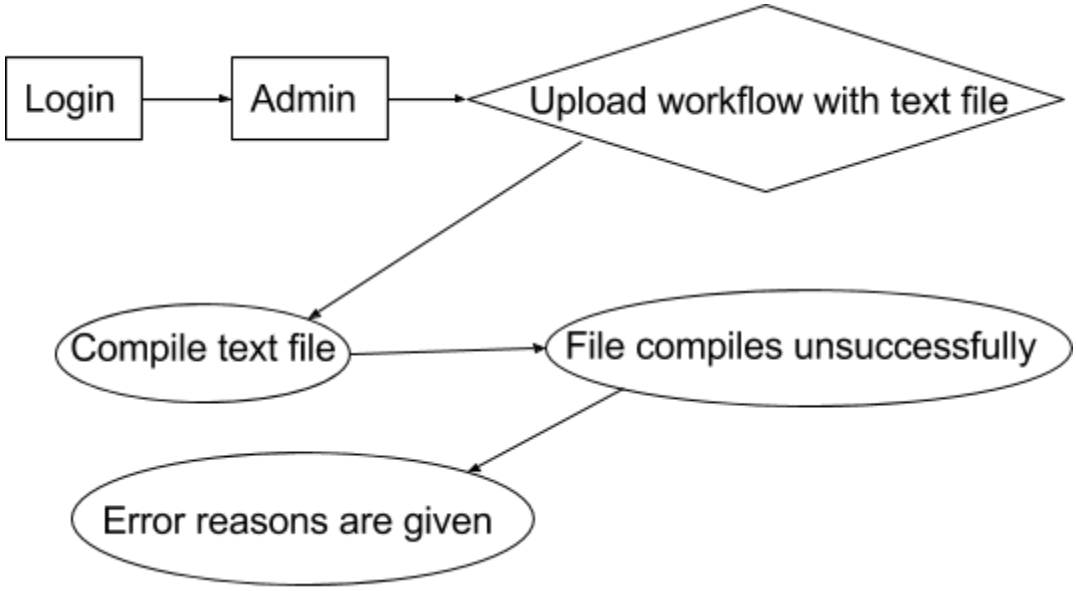


2.1 Use case #1: An administrator uploading a new workflow template

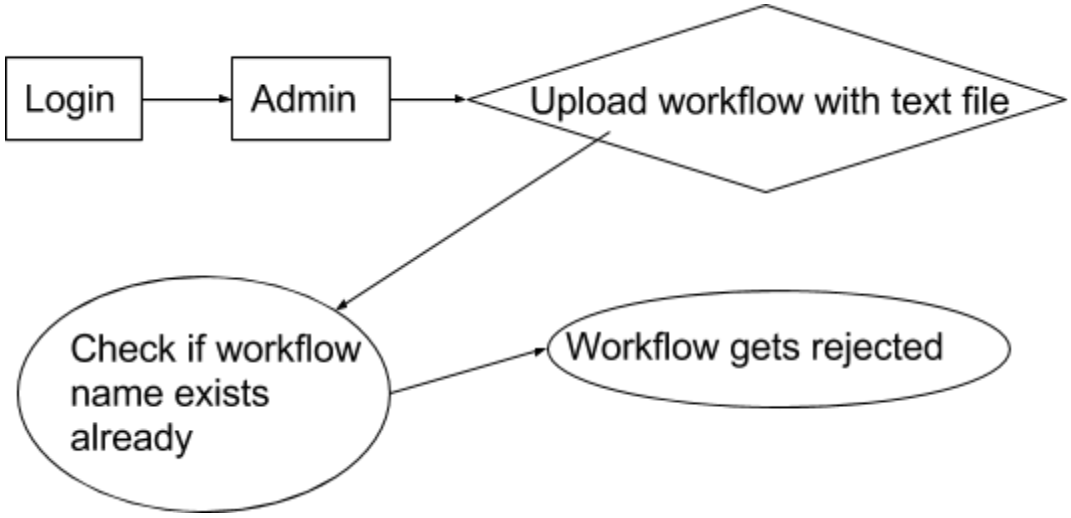
2.1.1 Sub-use case #1: An administrator successfully uploads a new workflow template



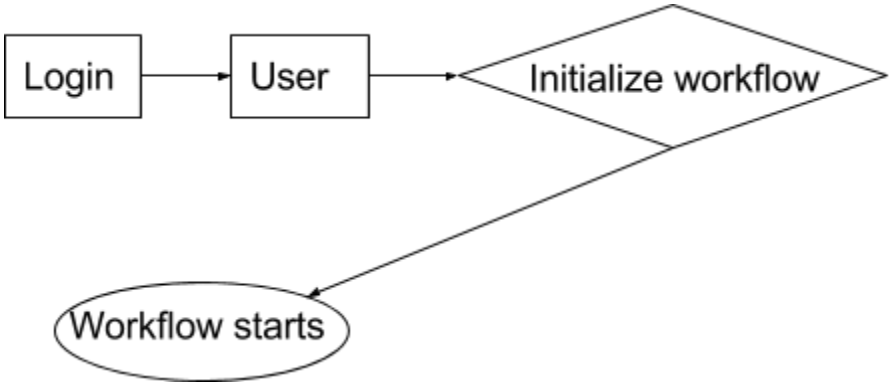
2.1.2 Sub-use case #2: An administrator unsuccessfully uploads a new workflow template



2.1.3 Sub-use case #3: Workflow name already exists

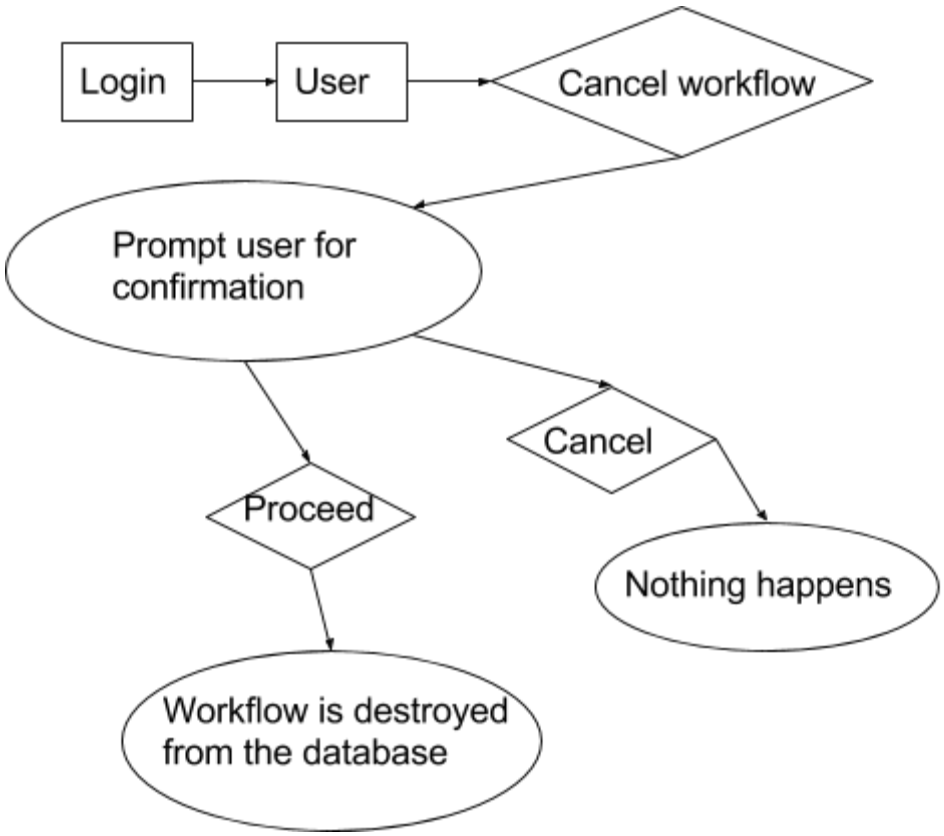


2.2 Use case #2: A user instantiating a workflow

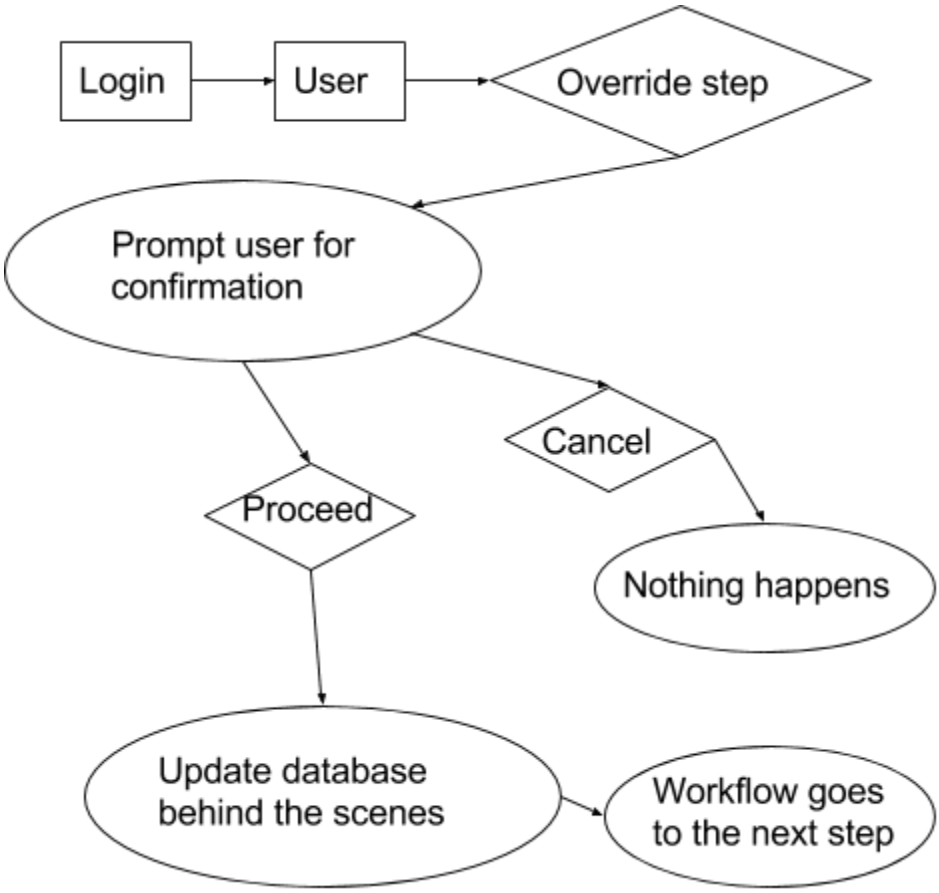


2.3 Use case #3: An end user managing an existing workflow they own

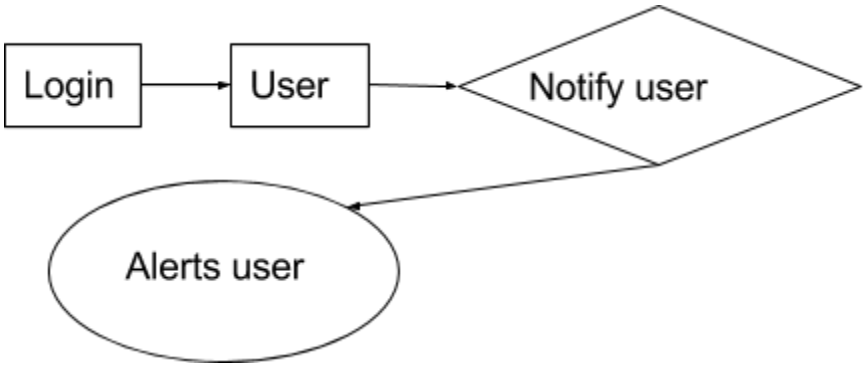
2.3.1 Sub-use case #1: Cancel workflow



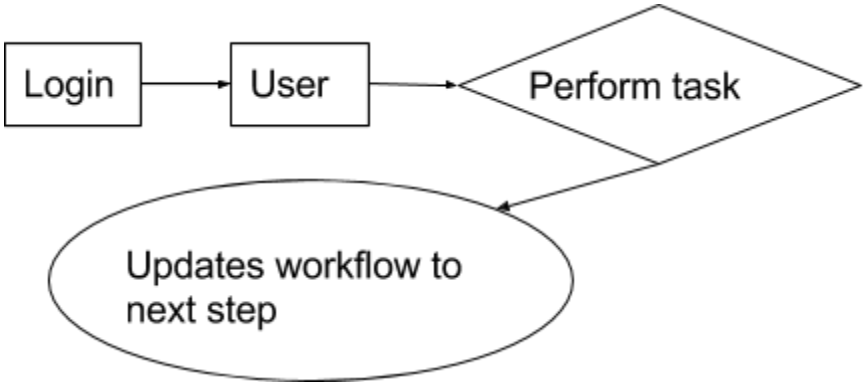
2.3.2 Sub-use case #2: Override steps



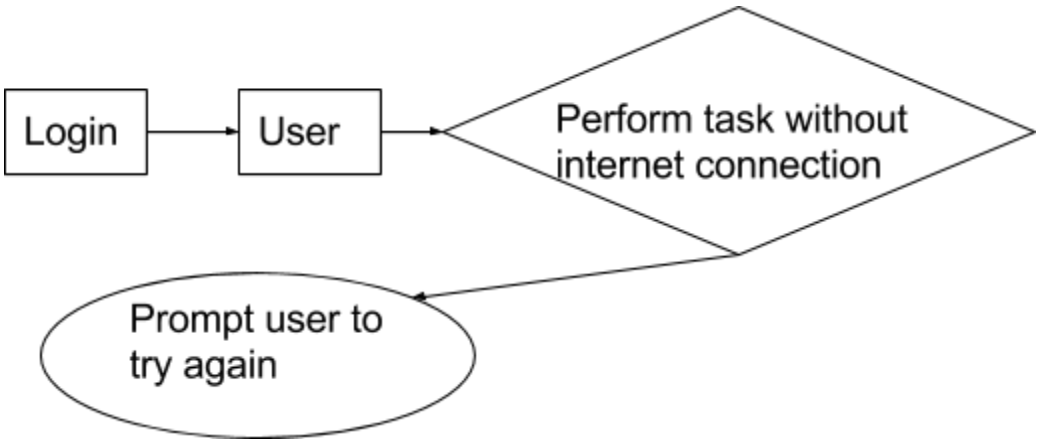
2.3.3 Sub-use case #3: Notify users



2.4 Use case #4: An end user participating in an existing workflow (they don't own)

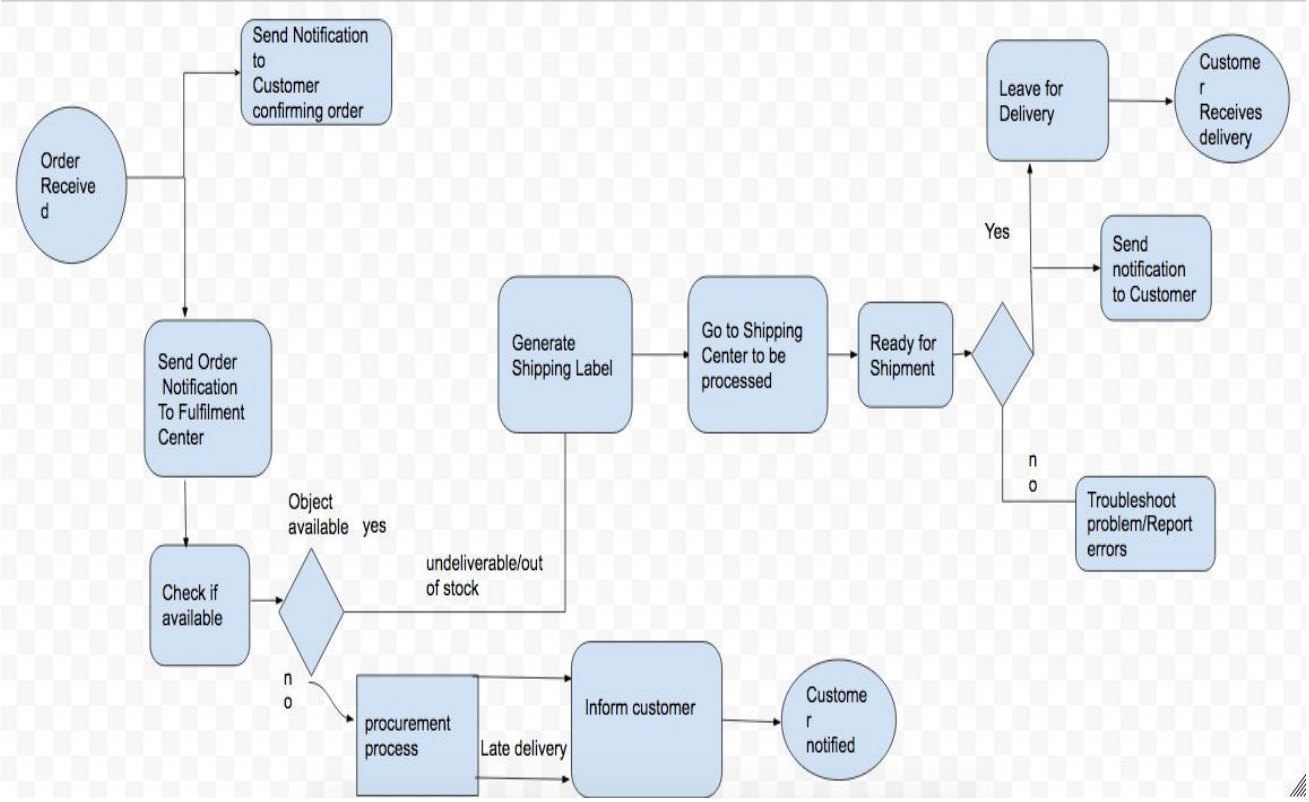


2.5 Use case #5: A user tries to perform a task without internet connection



2.6 Use Cases for Package/Order Tracking example

The following diagram depicts our use cases for the package/order delivery tracking example in UML notation. It shows the various actions that can be performed taking the system from one state to another.



3. Examples (in depth)

3.1 Special Permission Number (SPN) request

In this example, a student sends the SPN request form to Carol DiFrancesco (the Department of Computer Science Graduate Secretary) who instantiates the workflow.

This triggers a SPN to be assigned to the student, but not sent out yet. Once that step is completed, a trigger is tripped that sends the form to the Graduate School. Upon receipt by the Graduate School, Alex Bachmann now must approve/deny this student's request and file it in the system. Once Mr. Bachmann has signed the form or rejected the application, the workflow system is triggered to notify Secretary DiFrancesco.

If the SPN request is approved, the SPN is retrieved and sent to the student email address listed in the request form that was filed at the very beginning of this process. In a parallel process, this form is sent to Associate Dean Lenore Neigeborn for filing in her office. This workflow instance is marked as completed and is archived

There are a couple exceptions to the standard process. If the Graduate School is delaying the process, Secretary DiFrancesco has the ability to override the Graduate School's approval/denial ability and issue the Special Permission Number to the student. The other way to handle a delayed response is for a notification to be delivered to Secretary DiFrancesco with the contact information of the person who has not completed their responsibility yet such that the override is not immediately required.

3.2 Package/Order Tracking for Electronic Commerce companies

A precondition for this workflow commencing is the customer placing the order. The customer only receives messages from the workflow system and does not ever actually interact with the system. Once the order has been placed by the customer, the company initializes the workflow by inputting the order information into the workflow system.

Once the workflow is initialized, a parallel notification process occurs; a notification is sent to the customer confirming their order and a notification is sent to the fulfillment center letting them that there is a new order. Once the items are packaged and ready for shipment, the fulfillment center generates a shipping label which in turn notifies the shipping company that the package is ready for pickup.

After the package is picked up by the shipping company and arrives at the shipping center, the package undergoes a confirmation process (e.g. barcode or RFID scan) and at this point, a notification is sent the customer that the order has shipped. The shipping company processes the package and places it on its first hop to the user's shipping address. Subsequent arrivals at intermediate shipping/postal centers are confirmed as they happen, which triggers the next leg of the route to happen.

At some point, the package leaves the shipping company's internal network and goes out for delivery, which undergoes the same confirmation process as every other leg in the route. Upon delivery to the customer, the deliverer enters the confirmation that the package has been successfully delivered. This triggers a notification to the customer that their package has been delivered. This workflow instance is marked as completed and is archived.

Due to the complexity of electronic commerce, there are several exceptions to the standard process, a couple in the fulfillment process and a few in the shipping phase.

- One fulfillment exception is that an item that was ordered is out of stock. In this situation, the customer is informed that the order may be late or undeliverable if the procurement process fails.
- If a customer cancels the order however, nothing can be done from our side as we only deal with the workflow from the supplier side once we receive an order, meaning the customer can only request a return once he receives a package and the order is marked as completed, or for some reason the package is late or undeliverable. This will be a more advanced feature in our system, and will not be in our basic features.
- The shipment phase is not without potential exceptions, as mentioned earlier. A customer may provide incorrect shipping information. In this situation, the information can be overridden and corrected, overridden and returned to sender, or the process can be paused and the user notified. For any reason, if there is a delay and the package will be late as a result, the customer and the electronic commerce company will be notified.
- If a package requires a signature from the customer but they are not present, the shipping company will attempt to deliver the package two more times. If the package is undeliverable for a variety of reasons such as three unsuccessful delivery attempts (initial attempt and two re-try attempts) or unable to pass through security at an airport or hub, the package will be returned to the electronic

commerce company and the customer notified. If the package is lost while in transit, the shipping company will notify the electronic commerce company and they will reach out to the customer regarding what to do next.

4. Categories of importance for features

4.1 Kernel

The kernel features of our workflow system are those whose inclusion is completely necessary for the proper functioning of any system that can be called a workflow system. They are as follows:

- **Action Triggers**
    - For a workflow to work, it is necessary for actions of the system or user to be able to trigger other actions. This is what enables it to be called a "workflow." In general, our system will provide a way to designate a trigger event, usually the completion of a workflow step (but also potentially an error or external trigger), to which the system will then react by causing the specified trigger reaction to occur.
- **Alerts**
    - It is a basic feature of any workflow system that it be able to send alerts to its users. At its most basic, a workflow system should be able to send an email to users participating in a workflow instance when it is their turn to act on the workflow. A reasonable extension of this will include the ability for workflow

designers to send notifications to users at any point in the workflow process, containing any information they may wish to include from the workflow system.

- **Basic User System**
    - It is imperative for the overall system that it be able to handle multiple users. Basic functionality of a user system must allow different users, who have to complete different roles in a workflow, to log in and see the workflows in which they are participating. See below for extended functionality.
- **Addition of Workflow Templates / Instantiation of Existing Workflow Templates**
    - The workflow-participant-facing element of the system must have a way for the administrator / workflow developer to add new workflow "templates," in order for instances to be run.
    - The end-user workflow participants have to be able to instantiate workflows, so that they can be used and participated in. This also necessitates that running workflow instances be stored and tracked.
- **Modularity**
    - Every capability of the workflow system outside of its kernel framework will be implemented in a module. This will allow a more stable and functional system for both workflow developers and participants
    - Notifying participants is one module, timers are a module, opening files is a module, printing files is a module, importing or exporting any kind of data are their own modules, etc.

4.2 Reasonable

The set of features here denoted as ‘reasonable’ is a series of extensions on the kernel system, all of which are reasonable to expect in a production workflow system. All of these features should be available to users of our workflow system by the first main version release.

- **Timers**
    - Workflow developers should be able to implement timed events to occur within the workflow, either synchronously with the flow of the workflow or asynchronously (see Parallel Streams of Execution). This will serve as a specialized type of trigger where the action is the completion of a timer.
    - The most obvious use of this is for timed reminders about stages of the workflow to be sent to the instance owner.
- **Parallel Streams of Execution**
    - A workflow should not be limited to a single stream of execution, but rather should be able to support multiple streams or flows within a single workflow. A single event, say the completion of a single stage, should be able to trigger multiple events.
    - This will allow asynchronous events to occur when they are not interdependent.
- **Task Automation / Dynamic Decision Making**
    - Actions should be able to occur without user input when this is possible or desired. Triggers should be able to specify whether the triggered action requires user input or not. Actions that occur without user input should be able to access necessary information in the system without being tied to a particular user.

    - It should also be possible for workflows to "decide" between multiple next courses of action based on conditions or statuses specified by the workflow developer. This will allow the system to choose from different actions to perform in different scenarios, thus making the workflows more adaptable.

- **Edge Cases / Exception Handling**
    - The workflow designer should have the ability to implement his/her own handling of runtime exceptions in the workflow. This includes provisions for overriding / bypassing individual actions or entire stages of the workflow.
- **User Permissions / Instance Protection**
    - The administrator of the system should be able to add new workflows written by the workflow developer as templates to the system. S/He should also be able to remove existing workflow templates once they are no longer needed. Only the administrator should be able to perform theses tasks.
    - End-users of the system (i.e., neither the workflow developer nor the system administrator) should be able to create new workflow instances from workflow templates in the system, see and participate in any workflows they are a participant in, and see the status of workflow instances they are participants in.
        - For every workflow instance, there should be one owner (the instantiator) and any number of participants. Only the owner should be able to see the exact details of the current stage of the workflow at all times; other participants should only be able to see when it is their turn to act in the workflow, and what is required of them.

- **Useful Interpreter Messages / Workflow Verification**
    - The workflow interpreter is loaded by the system when the administrator adds a new workflow template from a file. The interpreter parses and interprets the workflow written by the workflow developer, and should report whether or not the workflow specification is valid. If it is valid, it should load it into the system successfully, but if it detects any problems, the interpreter should be able to report those problems in detail to the administrator, and it should not load the workflow into the system. This will ensure safe and proper use of the system, and also provides helpful information to the workflow developer.
- **Moderately Customizable UI**
    - Our workflow system will be adaptable, but in order to provide the customer with a pre-built user interface for the system, it will be necessary to limit the elements of user interface over which the workflow developer have control. The workflow developer will be provided with certain UI elements to choose from, which they can specify to use in certain ways, take certain kinds of input, and trigger certain actions. Our system will be responsible for taking this specification and generating the corresponding user interface.

4.3 Advanced

- **Full Extensibility Support**
    - The modularity of the system should allow implementation of new modules to add new functionality on top of the system we ship. Version 1.0 of the system

should come with several modules implemented already, to allow for the implementation of workflow examples we have already described. If possible, however, an advanced version of the system should have the ability to build custom modules on top of our system, and then use those modules seamlessly in workflows designed and used with our system. If this is successfully implemented in a future version, separate documentation will need to be provided for module developers; necessary information will include specifications for interacting with the system framework, interacting with other modules, and best practices.

- **Adaptable UI**
    - In future versions of the system, the workflow developer can be provided with more options and greater customizability for the user interface design of individual workflows. This will entail providing more user interface elements that the workflow developer can make use of in his / her workflows, and finer control over how those are used.
- **Invocation of External Applications**
    - The Version 1.0 release of the system will include modules for importing and exporting data and files. A more advanced version of the system should include the ability to directly invoke applications from the workflow system to view and manage data, and should work seamlessly with user applications on local devices.
- **Usage Analytics and Process Monitoring**
    - While not necessary for the proper functioning of a workflow system, and therefore not a target for our 1.0 release, it would be highly beneficial for the

system administrator to be able to see, and generate reports on, usage statistics for the overall system; how many users, how many active users, how many running workflow instances, how many completed workflows in total, and other statistics would be extremely valuable information for administrators of the system.

- Additionally, the administrator would benefit from the ability to keep an eye on the load of the system, see what workflows are in use and if any are "dormant," (i.e., should have been completed but are clearly abandoned after a certain period of time,) and other system-wide usage and process information.

5. Concluding Summary

The target of this project is to design and implement a workflow language using workflow patterns. This language will allow administration to create workflow templates which are used by our end users to instantiate new workflows or act on existing workflow instances. This language will allow administration a simple and efficient method of coordinating common processes for their end users in any way that they require.

As seen in our document, this robust platform has many different use cases and relatable examples. The development team has maintained a list of features that are understood to be core/kernel functionality as well as other categories and aim to implement as many as possible in order to provide our customers with the best and most easy-to-use platform possible.

The development team is looking forward to building the platform and is excited for the many uses for which customers are expected to utilize the platform.

6. Acknowledgements

- ★ Group 4, Inc. would like to thank our Professor Alex Borgida for his expertise and advice as a helpful resource on this requirements analysis and report.

Deepak Nalla:

- Editor
- Workflow instantiating and design use cases for our XML file/notation
- Use cases/UML diagram of our 2nd example

Joseph Kotzker:

- Description features and division of features of importance (kernel, advanced,etc.)
- General design and plan of document

Elliot Linder:

- Wrote examples section
- Conclusion
- General editing

Corentin Rejaud:

- Non-functional specifications section
- Use cases
- Summary