Agile Workflow Development

Agile Workflow Modeling Simplifies Workflow Development

Reading Time: 7 minutes

Developing solutions involving workflows is by no means trivial. Workflows can be large, complex, and highly intricate. Often it's difficult to determine where to even start when developing workflows to integrate into production-ready deployable solutions. Utilizing the concepts of agile development can provide direction and organization to your workflow development, making it more productive, yielding faster delivery, creating more flexible and reusable workflows, and increasing stakeholder satisfaction.

Go Agile

Agile development involves building solutions in iterations or sprints. Each sprint involves specific entry and exit criteria and periodic status checks (often daily). Sprints tend to be short in duration – generally 1-2 weeks, seldom but possibly longer. Sprints drive focus to delivering results in defined intervals (i.e., a "time box"). Successive sprints often build on one another to achieve a desired deliverable such as a product release or a production-deployable system.

Agile techniques are very useful in developing workflows. Early sprints can focus on developing workflow models that are successively refined into increasingly complex executable workflows. While many workflow tools do not explicitly mention support for such a development method many provide at least some limited support for such agile techniques.

But before actually building workflows in sprints, the sprints first need to be planned. One useful planning approach in workflow development is that of risk-driven planning. Workflows and workflow models are developed to explore areas of the solution that are the least understood or the most problematic, e.g., the most risky areas. As larger risks get modeled and mitigated, the next level of risk is delved into. Once the key risks are understood, development becomes more straightforward and the sprints shift focus from risk-driven models to more conventional production-ready deliverables.

Workflow Modeling

The early sprints in an agile workflow development effort generally involve modeling workflows without committing to defining the low-level details of specific workflows. In these early sprints the deliverables may be directed toward mitigating risks such as operations staff usability or exceptional event handling and associated error recovery.

In a workflow effort involving file processing for example, understanding exactly how the actual processing of files is to occur is often not as important in early sprints as understanding the attributes, dependencies and general timing of processing. Modeling errors and exceptions encountered during processing is also useful early in the workflow development cycle.

Early sprint workflow models often include illustrating and evaluating:

- General flows (i.e., happy paths)
- Error and exception flows
- Operational views (e.g., how to identify, troubleshoot, and resolve processing issues)
- Performance
- Scalability

Additional sprints often construct workflow models to better understand and mitigate risks that include, for example:

- Risks associated with developing workflows consistently, quickly, and completely
- Risks associated with maintaining workflows over time
- Risks associated with promoting workflows from development to test to production
- Risks associated with managing, monitoring, and troubleshooting workflows
- Risks associated with scaling out the execution of workflows

Many of the above risks can be mitigated in early without committing to a complete workflow development effort. Facilitating this process requires constructing some basic workflow templates that are engineered for modeling. These template are then used to quickly build the above mentioned models without a large commitment of time and effort upfront.

So what does a workflow look like that is specifically engineered for modeling? Many workflow tools support the ability to run scripts and start workflows, as well as providing components to access databases, make web service requests, look for files, and perform mail actions. In a workflow model the scripting component is used heavily to simulate the running and execution duration of the other components, as well as emitting static results that are similar in nature to the other components. For example, a component that tests for the existence of files and passes a list of filenames found through the workflow is replaced in the model by a scripting component that passes a list of hard-coded filenames through the workflow. A modeled mail component simply returns a success message similar to the actual mail component's success message. The model workflow components in essence "mock" the behavior of their real counterparts.

Once these mock components are constructed using the available script component (or other components available in the workflow palette), these components can be readily assembled into a variety of workflows for modeling. Latency, fail-over, high-availability, timeout conditions, and numerous other conditions can be quickly modeled and assessed using this approach.

Not all of the workflow components in the workflow tool's palette necessarily need to be mocked. Many tools offer some "primitive" components that are useful as is, such as those that write console messages, initiate child workflows from within a parent workflow, and as stated previously, those components that execute scripts or executable code. More advanced components such as those related to mail, database, or web service may need to be mocked, especially if the underlying infrastructure these components must interact with are not yet available to use (e.g., web services still need to be developed or database tables not fully defined). And of course, application-specific code encapsulated into scripts or components that call executable code may be mocked until the code is completely developed.

A Workflow Modeling Example

Here is a workflow modeling example using Flux, a workflow and file orchestration tool. This example workflow model is intended to illustrate the operation and monitoring of a certain workflow to IT support staff. This workflow model illustrated here is basic. Each file goes through a set of processes that perform the following activities:

- Validates the file,
- Splits the file into its header, body, and trailer portions,
- Scans the file portions and if the scans are successful,
- Ingests the file into a database.

The workflow model logs the workflow initiation event, validates the file, splits it into file portions, then ingests, with a final message that the workflow has completed. The specifics of exactly how the file is validated, split (and then scanned), and ingested are delegated to child workflows.

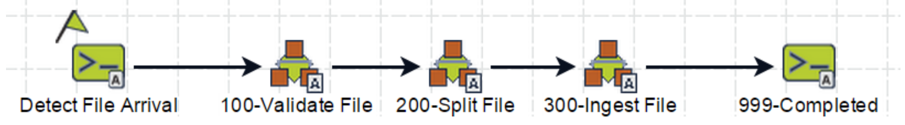


Figure 1. A workflow model to illustrate some conventional file processing.

Flux has many components, including those that move and manage files, read and send mail, update and query databases, send and receive web service requests, schedule and fire jobs, initiate child workflows, print messages, and run scripts and executable code. So using Flux, the top-level workflow shown above only uses two Flux workflow components, console messages the log output to the console, and flow chart actions that initiate child workflows.

The 200-Split File process is a little more complex, introducing a third component. The Flux Null Action is a component that can run interpreted scripts. A Flux Process Action could also be used, which runs Windows batch or Unix/Linux bash scripts or command line executables. Your workflow tool of choice most likely has similar components.

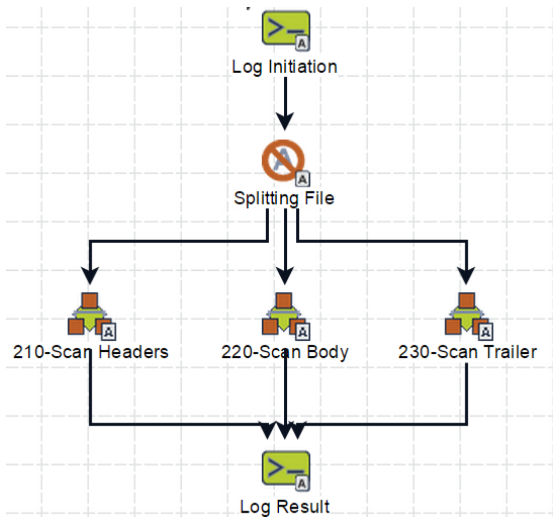


Figure 2. A child workflow illustrating the splitting of a file into its component parts.

The Splitting File activity is actually a very simple script that only models the expected execution time of the splitting file process. All it does is sleep for a random interval between 20 and 40 seconds.

Since this is a very preliminary model, 210-Scan Headers, 220-Scan Body, and 230-Scan Trailer all initiate instances of the same child workflow.

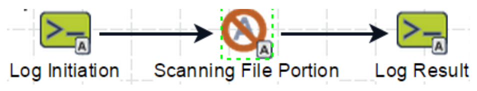


Figure 3. Another child workflow illustrating the scan processing associated with a portion of a file.

In later sprints these workflows will be replaced with more specific workflows. Additionally new models may be constructed to model other risks. Shown below is the above workflow extended to model timeout conditions. This workflow adds two additional components, a Java action that runs Java code, and an Error action that forces an exception – in this case when a process times out when running too long.

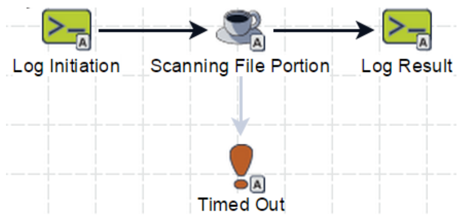
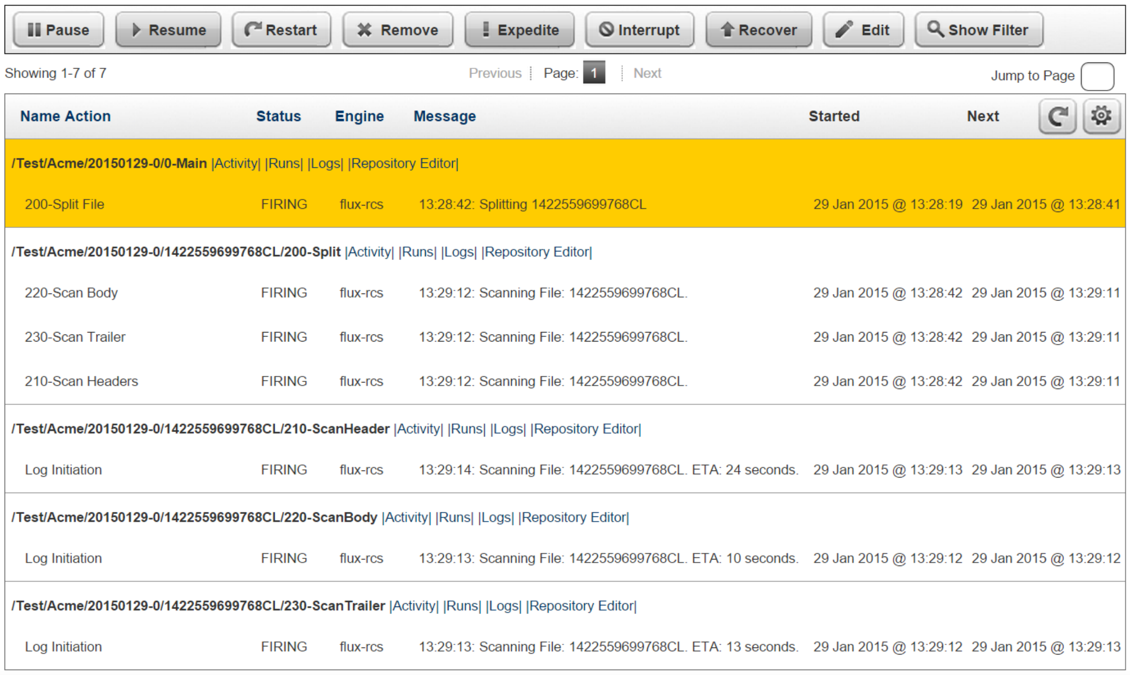


Figure 4. Extending the scan processing associated with a portion of a file to include a timeout condition.

Useful models can be quickly created and then run with just a few components (in this case five) and a few lines of script. In these models we did not even need to resort to any "mock" components! Running an executable model of a workflow that does not have any errors (i.e., the "happy path") allows developers and operations personnel the opportunity to evaluate and get familiar with the developing workflow.

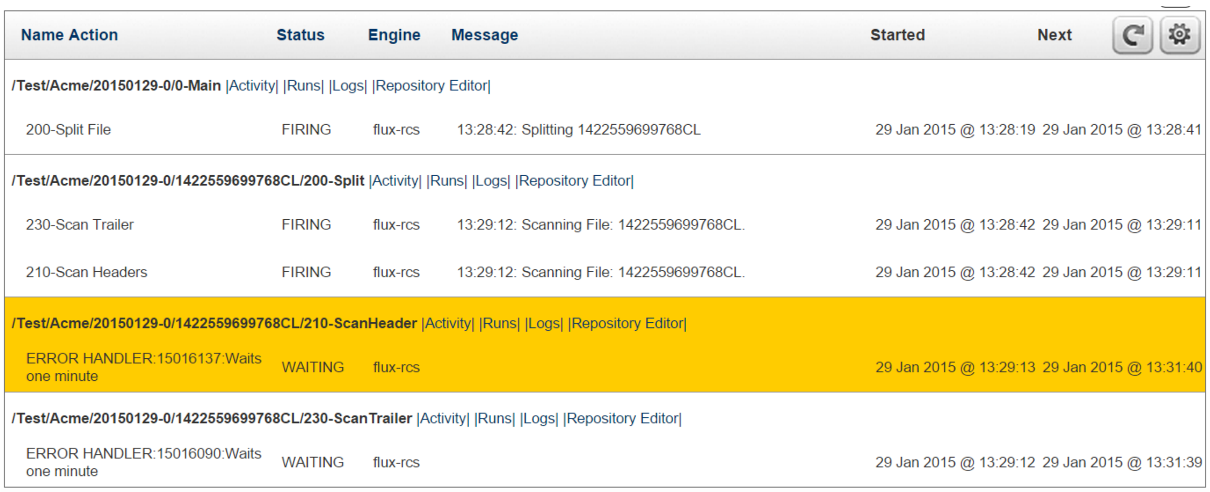


Showing 1-7 of 7

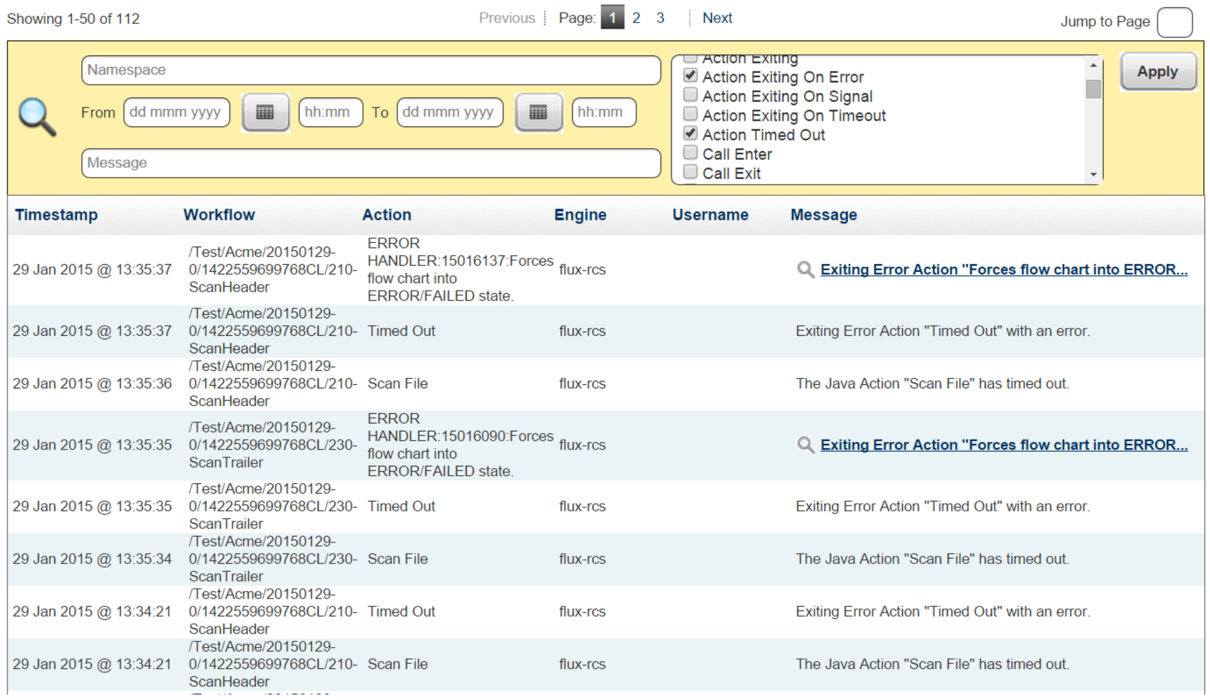
Name Action	Status	Engine	Message	Started	Next
/Test/Acme/20150129-0/0-Main |Activity| |Runs| |Logs| |Repository Editor|					
200-Split File	FIRING	flux-rcs	13:28:42: Splitting 1422559699768CL	29 Jan 2015 @ 13:28:19	29 Jan 2015 @ 13:28:41
/Test/Acme/20150129-0/1422559699768CL/200-Split |Activity| |Runs| |Logs| |Repository Editor|					
220-Scan Body	FIRING	flux-rcs	13:29:12: Scanning File: 1422559699768CL.	29 Jan 2015 @ 13:28:42	29 Jan 2015 @ 13:29:11
230-Scan Trailer	FIRING	flux-rcs	13:29:12: Scanning File: 1422559699768CL.	29 Jan 2015 @ 13:28:42	29 Jan 2015 @ 13:29:11
210-Scan Headers	FIRING	flux-rcs	13:29:12: Scanning File: 1422559699768CL.	29 Jan 2015 @ 13:28:42	29 Jan 2015 @ 13:29:11
/Test/Acme/20150129-0/1422559699768CL/210-ScanHeader |Activity| |Runs| |Logs| |Repository Editor|					
Log Initiation	FIRING	flux-rcs	13:29:14: Scanning File: 1422559699768CL. ETA: 24 seconds.	29 Jan 2015 @ 13:29:13	29 Jan 2015 @ 13:29:13
/Test/Acme/20150129-0/1422559699768CL/220-ScanBody |Activity| |Runs| |Logs| |Repository Editor|					
Log Initiation	FIRING	flux-rcs	13:29:13: Scanning File: 1422559699768CL. ETA: 10 seconds.	29 Jan 2015 @ 13:29:12	29 Jan 2015 @ 13:29:12
/Test/Acme/20150129-0/1422559699768CL/230-ScanTrailer |Activity| |Runs| |Logs| |Repository Editor|					
Log Initiation	FIRING	flux-rcs	13:29:13: Scanning File: 1422559699768CL. ETA: 13 seconds.	29 Jan 2015 @ 13:29:12	29 Jan 2015 @ 13:29:13

Figure 5. A workflow dashboard illustrating the "happy path" processing for a file.

Adding exception circumstances such as time out conditions moves the model beyond "happy path" processing into simulations of problem identification and resolution. The model can be extended further to explore and simulate a variety of scenarios regarding recovery, restart, and failover ahead of production deployment.



Name Action	Status	Engine	Message	Started	Next
/Test/Acme/20150129-0/0-Main |Activity| |Runs| |Logs| |Repository Editor|					
200-Split File	FIRING	flux-rcs	13:28:42: Splitting 1422559699768CL	29 Jan 2015 @ 13:28:19	29 Jan 2015 @ 13:28:41
/Test/Acme/20150129-0/1422559699768CL/200-Split |Activity| |Runs| |Logs| |Repository Editor|					
230-Scan Trailer	FIRING	flux-rcs	13:29:12: Scanning File: 1422559699768CL.	29 Jan 2015 @ 13:28:42	29 Jan 2015 @ 13:29:11
210-Scan Headers	FIRING	flux-rcs	13:29:12: Scanning File: 1422559699768CL.	29 Jan 2015 @ 13:28:42	29 Jan 2015 @ 13:29:11
/Test/Acme/20150129-0/1422559699768CL/210-ScanHeader |Activity| |Runs| |Logs| |Repository Editor|					
ERROR HANDLER:15016137:Waits one minute	WAITING	flux-rcs		29 Jan 2015 @ 13:29:13	29 Jan 2015 @ 13:31:40
/Test/Acme/20150129-0/1422559699768CL/230-ScanTrailer |Activity| |Runs| |Logs| |Repository Editor|					
ERROR HANDLER:15016090:Waits one minute	WAITING	flux-rcs		29 Jan 2015 @ 13:29:12	29 Jan 2015 @ 13:31:39



Showing 1-50 of 112 Previous | Page: 1 2 3 | Next Jump to Page

Namespace
From dd mmm yyyy hh:mm To dd mmm yyyy hh:mm
Message

Action Exiting
[✓] Action Exiting On Error
Action Exiting On Signal
Action Exiting On Timeout
[✓] Action Timed Out
Call Enter
Call Exit

Apply

Timestamp	Workflow	Action	Engine	Username	Message
29 Jan 2015 @ 13:35:37	/Test/Acme/20150129-0/1422559699768CL/210-ScanHeader	ERROR HANDLER:15016137:Forces flow chart into ERROR/FAILED state.	flux-rcs		Exiting Error Action "Forces flow chart into ERROR...
29 Jan 2015 @ 13:35:37	/Test/Acme/20150129-0/1422559699768CL/210-ScanHeader	Timed Out	flux-rcs		Exiting Error Action "Timed Out" with an error.
29 Jan 2015 @ 13:35:36	/Test/Acme/20150129-0/1422559699768CL/210-ScanHeader	Scan File	flux-rcs		The Java Action "Scan File" has timed out.
29 Jan 2015 @ 13:35:35	/Test/Acme/20150129-0/1422559699768CL/230-ScanTrailer	ERROR HANDLER:15016090:Forces flow chart into ERROR/FAILED state.	flux-rcs		Exiting Error Action "Forces flow chart into ERROR...
29 Jan 2015 @ 13:35:35	/Test/Acme/20150129-0/1422559699768CL/230-ScanTrailer	Timed Out	flux-rcs		Exiting Error Action "Timed Out" with an error.
29 Jan 2015 @ 13:35:34	/Test/Acme/20150129-0/1422559699768CL/230-ScanTrailer	Scan File	flux-rcs		The Java Action "Scan File" has timed out.
29 Jan 2015 @ 13:34:21	/Test/Acme/20150129-0/1422559699768CL/210-ScanHeader	Timed Out	flux-rcs		Exiting Error Action "Timed Out" with an error.
29 Jan 2015 @ 13:34:21	/Test/Acme/20150129-0/1422559699768CL/210-ScanHeader	Scan File	flux-rcs		The Java Action "Scan File" has timed out.

Figure 6 and 7. A workflow dashboard and audit trail view illustrating a circumstance where the scan processing of a file has timed out.

Over time, the workflow model can be further refined to illustrate more and more use cases. Sometimes the modeling effort takes on a life of its own, proceeding in parallel to the larger workflow development effort. This may occur if the effort is highly complex and there are many dependencies needing a tight coordination between multiple workflows. In such instances the models assist in excavating issues ahead of encountering them in the development effort itself. Additionally such parallel development can provide operations staff and stakeholders many opportunities to explore and evaluate use cases in advance of the finally deployed solution.

Eventually there comes a time where the models are replaced by the actual workflow deployment, but even then the models may continue to be of value for future training and documentation of the overall solution.

Proceeding in an Agile Fashion

Workflow development is best done in an iterative fashion, gradually refining workflow models into more specific and complex workflows that can eventually be integrated into a production-deployable solution. Driving the iterations based on mitigating key risks provides a path to ensuring the eventual solution meets all stakeholder needs, not just the needs of a select few.

Over successive sprints, the workflow models are refined, replacing mocked components with their real counterparts and injecting application specific scripts and code where needed to complete the workflows.

Agile methods facilitate workflow development, adding control and checkpoints to the iterative development process. Early agile sprints deliver low-effort, executable workflow models that explore stakeholder needs and key risks. These models can then be extended in later sprints production-deployable solutions.

About Flux

Flux assists enterprises in provisioning, onboarding, scheduling, tracking, and reporting an enterprise's file orchestration processes. These orchestrations vary from simple file transfers to highly complex workflows involving extensive processing, many routes, varied alerts, and complex decisioning. Flux facilitates finance, healthcare, and software providers in effectively orchestrating files to create new revenue opportunities and facilitate expense reductions. First released in 2000, Flux has grown into a file orchestration platform that hundreds of enterprises rely on daily for their mission critical systems.

Contact Flux

Contact sales@flux.ly for information and browse flux.ly for other white papers and Flux download link.