MYRROR

Briefing Note

October, 2025

Static Process Models to Continuously Updating System Representations

Introduction

Whilst all large scale financial services organisations have significantly digitalised their systems, fragmentation remains widespread across the banking and financial services industry. Core systems remain as collections of silo'ed business functions, typically, batch processing area-specific data on business unit specific infrastructure.

Over the next 5 years these systems will need to be modernised. This will be driven by the accelerating adoption of AI and the efficiencies this enables. The challenge to be addressed in managing this migration lies in bridging the gap between the core capabilities of legacy implementations and the demands of the AI engines.

The full potential of AI can only be realised by providing the newly introduced AI engines with consistently formatted, high quality, continuously streaming data. Meeting this requirement will require legacy implementations to migrate from batch processing data in a plurality of formats into unified, real-time, continuously streaming architectures.

This is not to suggest the existing systems will be abandoned. Rather the existing infrastructure will be leveraged to progressively accommodate the increasing demands of AI adoption. This transition will need to be managed carefully.

To manage this transition, precise observability of the "*as-is*" state of the current implementation needs to be extracted. The key word here is "observability". This has a meaning beyond "visibility". It will be insufficient to simply identify the movement of data; an understanding of cause, context and consequence of the movement will be required — and this is essence of *observability.*

Once system observability has been secured, mechanisms may be introduced to ensure proposed system changes do not result in unintended consequences across the implementation.

The first step in process is modelling the existing infrastructure. Whilst most organisations have made some progress in modelling system behaviour, typically this behaviour has been captured as sets of historic static system diagrams, manually and periodically updated.

The following presents a solution for converting these static process models into continuously updating real-time system representations.

Keywords: Artificial Intelligence, legacy implementations, silo'ed business functions, continuously streaming data, system modernisation, static process models, real-time system representations.

The Current "State of Play"

Extracting and maintaining system visibility is a non-trivial undertaking. The current approach typically involves examining existing documentation, interviewing subject-matter experts, observing system activity, and using process mining software to extract elements of system behaviour. This fragmented assembly of information is then compiled into a static representation of a subset of system activity using one of numerous process modelling tools.

The outcome is typically a library of thousands of static models,' each describing only a portion of overall system behaviour. Maintaining these models becomes a time-consuming, resource intensive, error prone manual process needing to be undertaken periodically, given the live implementation is a dynamic environment subject to near-continuous change.11

What is required is a capability to automate this process - a capability that converts the library of static models into continuously updating system representations, that remain synchronised with the current state of the implementation.

HELIXsystem Process Assembler™

The HELIXsystem Process Assembler™ ("Process Assembler") delivers this capability.

The Process Assembler works by observing the systems end-point behaviour as this executes and compiling the observed behaviour into a continuously updating process representation. This is a fully automated function at run-time requiring no human intervention.

The resulting representation is

- "**Complete**" - capture of all system activity, however infrequently executing
- "**Unambiguous**" - does not require conformance, fitment or any other form or recalibration
- "**Machine readable**" - generated to any standard format able to be consumed by any standards- based software

For the purpose of converting static system models into continuously updating representations, the Process Assembler only requires visibility of the system's end-point behaviour. This visibility may be provided by monitors already installed on the live implementation; all that is required is a branch feed from these monitors to be provisioned to the Assembler Utility.

The Assembler Utility assigns standardised temporal information to the observed end-point activities, being the Initiation (send) and Termination (receive) events. This temporal data is stored in the Assembler Utility Temporal Library along with the native timestamps to ensure there is no loss of fidelity. The Assembler Utility uses the standardised timestamps to identify latency across messaging activities and generate alerts when this latency exceeds previously observed parameters.

To ensure minimal impact on production, the Assembler Utility runs in its own isolated environment, held within the client environment. Coupled to the Assembler Utility is the database of end-point behaviour. This database is owned and controlled by the client. The Assembler Utility continuously interrogates this database to extract attributes from pre-defined message fields. These attributes are assembled into a "**synthetic identity**" that uniquely identifies each message on each messaging environment.

As a message migrates across a heterogenous implementation, the Assembler Utility compiles an end-to-end representation of the executing process and generates this to standard notation. This is achieved with ultra-low latency, minimal system impact, without requiring:

- monitors to be installed on data stores
- change to the services or applications
- modification or alteration of the message formats
- any system change that may require any part of Production to be referred to QA

System Complexity and Existing Tooling

System Complexity

Myrror Corporation recognises that large-scale distributed systems are inherently complex. The first step in extracting an understanding of these systems is to secure observability of the system's behaviour.

Currently aspects of system behaviour have been captured in historical, static process representations. The proposed solution is the implementation of the Process Assembler to progressively convert the existing library of static models into continuously updating real-time system representations.

This incremental transition is not intended to model the complexity of the existing implementation; it is intended to introduce automation designed to contain and eliminate the time-consuming,

resource hungry and error prone manual processes currently required to maintain model concurrency.

Existing Tooling

Myrror Corporation also recognises the significant investment made by the model maintenance teams to familiarise themselves with process modelling tools, or a small number of tool-suites.

It is not proposed that these tool-suites be replaced by the Process Assembler. As the Process Assembler is fully standards compliant and all outputs conform to a standard graphical formalism, any output generated by the Process Assembler may be seamlessly integrated into the existing tool-suites already in use.

This approach eliminates the need for retraining or altering the established analytical procedures already in use. The only change is the conversion of static models to contemporaneous system representations. The model maintenance teams will continue to use the tools with which they are familiar but will be freed from the time-consuming task of manual model updates, enabling these teams to focus on higher value-add activities.

Concluding Comments

Just as Service Orientated Architecture and Cloud Computing drove the migration to the “next generation” of distributed computing, the introduction of Artificial Intelligence is heralding another generational transformation of large-scale distributed systems.

This “next generation” will be defined by the migration from silo’ed business functions, batch processing business unit specific data to unified, continuously streaming data management platforms.

With this migration comes a new imperative:

“Continuous streaming will demand continuous observability”

Managing the next generation of distributed systems cannot be achieved using historical, static process representations assembled from a fragmented collection of outdated system information. The emerging challenge is to transition from static process models to contemporaneous system representations delivering real-time, continuously updating system observability.

The first step in this transformation is converting the existing system knowledge into real-time continuously updating system models. Once real-time system observability has been secured, managing the transition to the “next generation” architecture becomes a progressive, carefully managed, and auditable process.

The HELIXsystem Process Assembler™ is protected technology by US patent grant No 8,554,594 “Automated Process Assembler”.