Paper ET21

# Business Event Triggers – the Next Level of Automation

Marc Jantke, Entimo AG, Berlin, Germany

## ABSTRACT

Automation in clinical trial data management and analysis is traditionally designed as a set of pipelines: rigid sequences of steps, often initiated by manual input or by the completion of another pipeline. While effective for linear workflows, this approach lacks flexibility. Loopbacks, conditional branching, or constructing dynamic networks of activities are cumbersome or impossible.

This presentation explores a new paradigm: automation through business event triggers. Every system activity produces events that can be captured and used to trigger subsequent actions. These events form a loosely coupled automation network rather than a fixed chain. Pipelines can still be mimicked where needed, but the event-driven model enables richer branching, context-sensitive reactions, and user-specific automations.

Key benefits include handling diverse outcomes more elegantly, supporting complex workflows, and decoupling permissions. Automations can be designed for individual users, teams or global settings within the corresponding authorization boundaries and isolation from other automations.

## INTRODUCTION

The transformation, analysis and reporting of clinical trial data towards submission is a labor- and time-intense yet highly repetitive sequence of activities. Pharma companies are striving to get new medications to the market faster, and CROs are supporting this endeavor to the best of their ability. Common approaches to optimize and speed up these processes entail standardization and automation.

The standardization of data models driven by CDISC such as SDTM and ADaM, but potentially also USDM, is essential to enable any kind of process automation. Standardization goes beyond data models and also applies to standard libraries for macros and programs, standard workflows, standard templates, and guidelines for outputs and documentation. While the term standardization implies a one-size-fits-all approach, this is typically not the case. Sponsors have to deal with different data collections, data transformations, analysis and reporting needs for different therapeutic areas, often even for different trials in the same area. Organizations, processes and systems have to manage this flexibility in standards consumption also in scope of automation.

Today, automation for such processes is often implemented as pipelines, where a single pipeline is a sequence of computational steps. Modern pipeline technologies support conditional step executions and branching for parallel processing, as well as a multitude of parameterized pipelines to construct complex networks of process automations. While this is an adequate and state of the art approach to reduce manual and error-prone executions of repetitive tasks, it comes with a variety of challenges that need to be addressed when implementing such an automation system in a regulatory context.

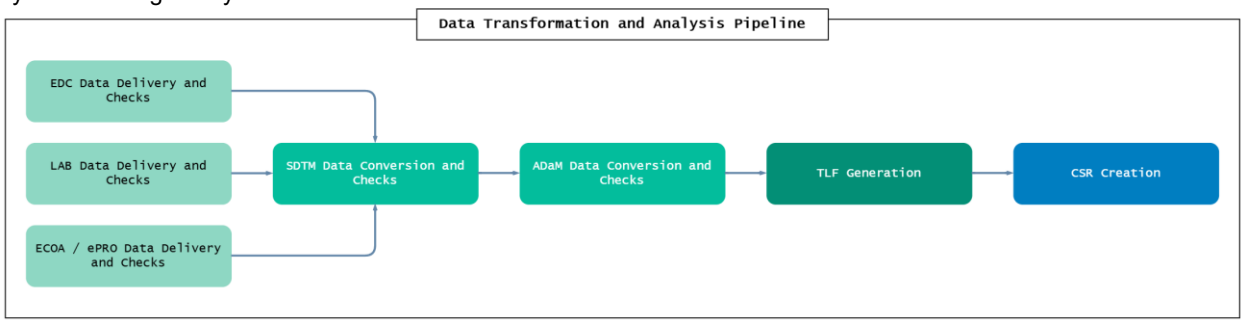


*Figure 1: Data Transformation and Analysis Pipeline*

An automation pipeline, like the example of CRF/nonCRF → SDTM → ADaM → TLF automation as shown in Figure 1, must tackle the following challenges:

**Responsibility**

Different individuals are contributing as authors and reviewers to the individual pipeline steps; in many cases still other individuals are managing the pipelines. It must be clear who is responsible for creating, maintaining and potentially troubleshooting each single step in a pipeline and the pipeline as a whole.

**Accountability**

GxP compliance requires accountability for all relevant artifacts. This includes not only the creation of programs and scripts which are used as pipeline steps, but also the individual execution of these steps in case relevant submission deliverables are produced. Pipelines are often executed using central, technical accounts, and companies have to take additional precautions to assign executions to identifiable and accountable individuals.

**Complexity**

Individual pipeline steps come with individual preconditions, such as the availability of required input data in an expected state. Companies have to find a fine balance between fewer, larger pipelines with more assembled preconditions or more, smaller pipelines with fewer individual preconditions. While more pipelines add to the overall automation complexity and thus maintenance efforts, it could be a better choice to run smaller portions of the overall process depending on the fulfillment of individual preconditions. The design of pipelines and dependencies also affects the ability to rerun certain parts of the pipelines in case of changed input conditions.

**Timing**

The overall goal of such pipelines is to produce the submission-ready deliverables. But neither programmers nor pipeline administrators can wait until the final data is delivered before starting to work. Thus, all individuals have to put additional effort into dealing with early, potentially incomplete and faulty input data deliveries. While this is daily business for experienced data scientists, it is often overlooked when constructing the overarching pipelines.

It is important to keep in mind that automation does not nullify the need for GxP compliance in these processes. Process steps must remain access-controlled and traceable; the results produced must be audit-trailed and accountable to human individuals. These requirements stand true for manual processing and automated systems alike.

This paper discusses how the reactive nature of modern software systems could contribute to process automation. Current systems are often designed as a (micro) service architecture utilizing a mixture of APIs and events for the communication between system components. APIs are typically used for synchronous request-response interactions where an immediate response is crucial, e.g. when displaying information to the user. Events are used for asynchronous communication where components are publishing events upon finalization of subprocesses, which can be consumed by subscribers to trigger downstream activities. APIs are often made publicly available as Open API to allow consumers (even end-users) to make use of these APIs in scope of integration and automation. Depending on the purpose, these APIs are typically wrapped and presented as higher-level business APIs built upon low-level technical system APIs. The same could be offered for events, wrapping technical system events into consumable business events, which would open up a new world of asynchronous/synchronous automation beyond currently available APIs and pipelines.

While Agentic AI can be a great tool for automation, it finds its use more in areas that relate to guidance provision, insight generation and natural language processing. By contrast, data transformation, analysis and reporting are factual, evidence- and rule-based activities that can be automated with established processes and technologies, and do not necessarily require the use of LLMs. That being said, the use of business event triggers in combination with automation pipelines could include the utilization of Agentic AI as recipient of events or in scope of individual pipeline steps.

## SOFTWARE ARCHITECTURE AND AUTOMATION

Automation of software-supported processes is tightly bound to the capabilities of the software systems to be automated. Thus, it is important to roughly understand the history of software architectures because it directly relates to the automation capabilities provided by such software systems.

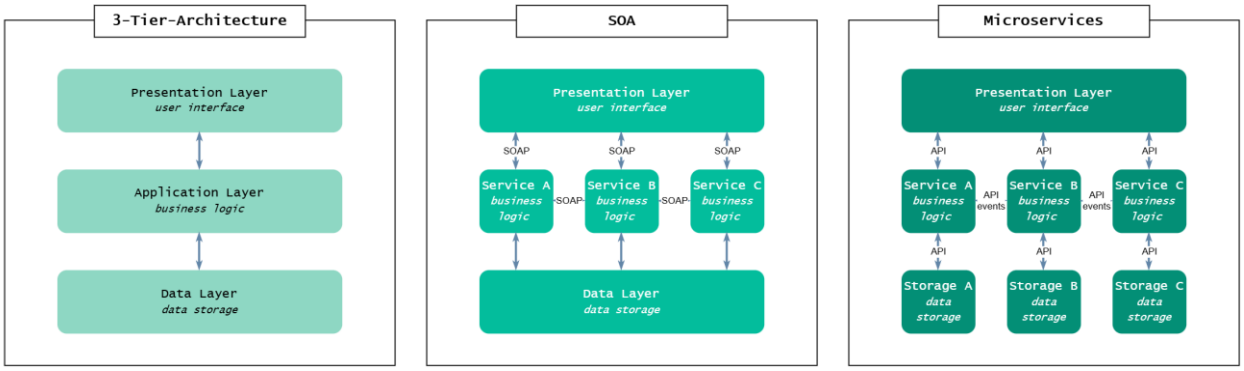


*Figure 2: Software Architecture Comparison*

With the original focus of software on computation, it was a natural development that complex software systems in the early 2000s were monolithic, often following a three-tier architecture for presentation, business logic and data storage. This monolithic architecture is not designed for dynamic scaling, which became popular with growing computation capacities and the cloud adoption. This inherent limitation became worse with growing computational demands, growing data volumes and user numbers because the synchronous nature of such systems led to increasing waiting times for processes and users.

The automation capabilities of monolithic systems are typically limited and non-standardized. Vendors were building custom interfaces to individual upstream and downstream systems and processes that could be used to trigger data and information exchange. The integrated automation capabilities, if any, were typically either limited to simple sequential executions of predefined steps, or complex administrative tasks inaccessible for normal users. End-to-end automation of complex business processes including upstream and downstream integrations was hardly possible.

Based on these lessons, software vendors invested in breaking down the monoliths into standardized services following the Service Oriented Architecture (SOA) approach, where systems are constructed as a variety of individual services connected by standardized interfaces, often using the highly secure, yet complex Simple Object Access Protocol (SOAP). Each of these services encapsulates a certain functional business capability, and the combination of all of these capabilities forms an overall system. The distribution of system capabilities over different services with clearly defined interfaces enabled better scalability on the service level, and better designed and more accessible user interfaces.

The automation capabilities of SOA systems are comparable to monolithic systems. While the SOAP interfaces generally provide standard interfaces to system capabilities, these were often not designed for integrations and automation but for the necessary system communication. Thus, SOA systems either provided proprietary solutions for automation just like monolithic systems did (or did not), or users had to find ways to interact with the complex SOAP interfaces for their automation needs.

Modern cloud-native microservice architectures are a natural evolution of SOA. Services have been further disassembled from complex business capabilities to individual business tasks, replacing single SOA services with a series of microservices. These services are loosely coupled, and the communication has been simplified with moving from SOAP to REST. This increasing need for reactive communication between services, because of network bandwidth and latency, opened the way for event communication in addition to the established REST APIs. In API interactions, calling services typically send a request and synchronously wait for a response. With event messaging, services can subscribe to events of interest that are published by other services for consumption, enabling fully asynchronous inter-service communication.

The automation capabilities of microservice architectures based on their APIs are plentiful. Many systems are designed following an API-first approach, putting well designed, documented and stable APIs in the center of the software architecture. Such APIs are crucial for the isolated service management and yield the additional benefit of reliable automation capabilities. While the aforementioned events are a common second means of communication in modern systems, these are rarely exposed to the community for integration and automation. Techniques such as event storming support an event-focused software design, but a consequent "Events-first approach" is still missing.

## AUTOMATION WITH BUSINESS EVENT TRIGGERS AND APIS

An example of end-to-end automation for a clinical trial reporting process might look like the pipeline above. CRF and non-CRF data are delivered into the central Clinical Data Repository (CDR) and further processed inside the Statistical Computing Environment. Once data quality and conformance checks for the delivered data have passed, data transformations from raw to SDTM are executed. Once quality and conformance checks for the SDTM data have passed, data transformations from SDTM to ADaM are executed. Once quality and conformance checks for the ADaM data have passed, TLFs are created. Once TLFs are complete and approved, the CSR can be written or generated.

While these steps could be combined into one extensive pipeline or a set of pipelines, it opens up the initial questions raised in the introduction:

1. Who is responsible for the management of the pipeline, and who is taking care of any pipeline errors?
2. Who is triggering the pipeline and when?
3. Who is the accountable individual associated with the individual pipeline steps?
4. What are the preconditions necessary to assure reliable and consistent pipeline executions at any time?
5. How can the pipeline be adjusted to handle interim analyses and final submission runs?

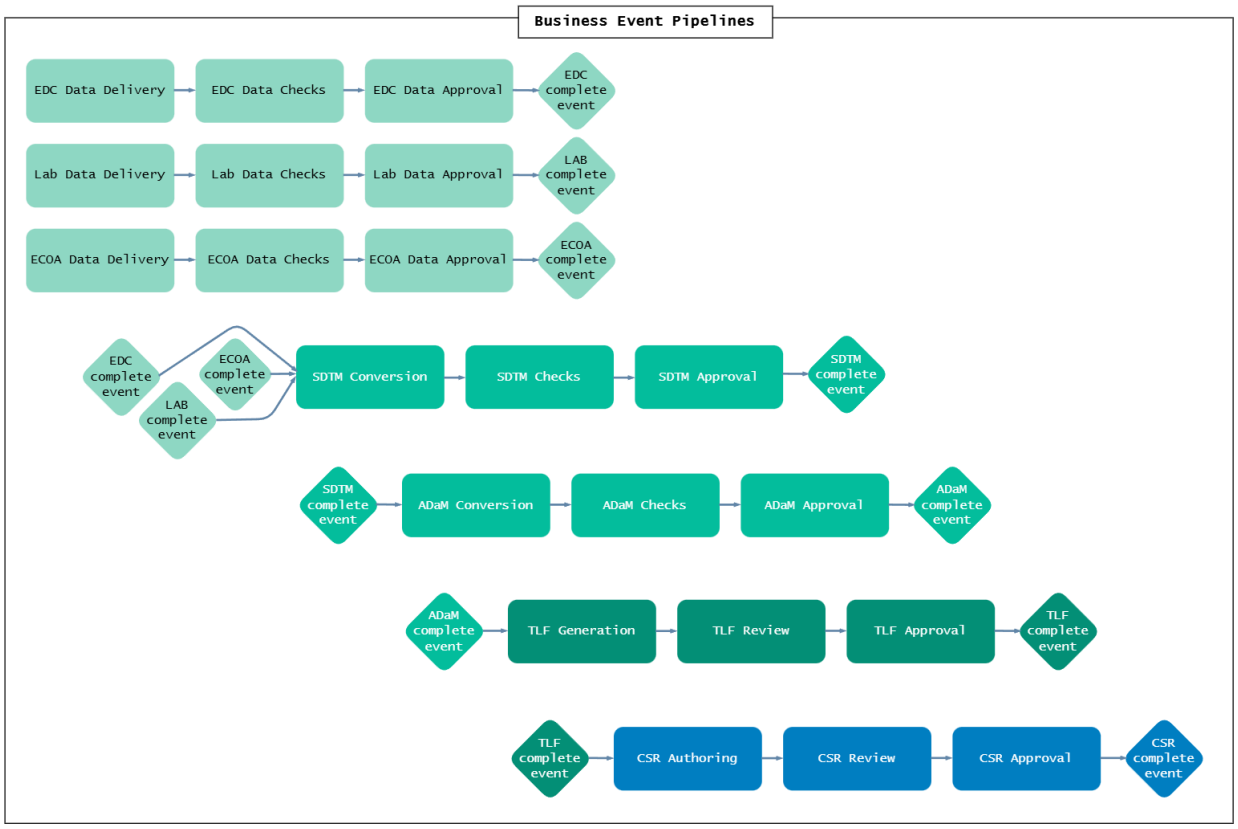


*Figure 3: Business Event Pipelines*

The utilization of business events as triggers for individual pipeline parts as shown in the graphic above can help to address these challenges. Like microservices, pipelines could be considered as micro-automations covering individual sub-processes, and the end-to-end automation is connected by business events. The responsibility and thus the accountability of these micro-pipelines lies with the actual individuals working on the sub-processes. Data Managers and their teams are configuring the CRF and non-CRF data import and the associated checks. These raw data imports could be configured in individual small pipelines per source system. The successful checks of raw data trigger a business event. Data Scientists responsible for the SDTM conversion are working on the respective programs and automation pipelines. Once ready, this team can configure the pipeline to react on the business event triggered by the successful raw data checks. Depending on the status of raw data encoded in the triggering business event, the pipeline can trigger interim analysis or final submission runs. The same or another group might be responsible for the ADaM conversion and later the TLF programs. The same rules apply; responsible individuals are empowered to maintain their own processes and automations, and trigger these based on business events published

by upstream processes. Depending on the level of automation, the final step of the process could either be the CSR generation or the notification to the Medical Writers that all data and TLFs are available for further processing.

This separation of concerns by the use of business event triggers for smaller pipelines provides answers to the questions above. The subscription nature of events enables teams to independently manage the sub-processes in their responsibility without any additional extra tasks required by the teams providing the upstream input data.

1. Each team is individually managing a pipeline for their sub-process and thus responsible to take care of any errors.
2. Each team can define on which business events and which encoded conditions the processes should be triggered.
3. The team lead could take accountability for the overall sub-process pipeline, or even individual micro-pipelines could be configured down to program-level ownerships.
4. The subscribed business events can be evaluated to assure that all preconditions are fulfilled before the pipeline is executed.
5. Like above, the pipeline could consider different states of input data as encoded in the subscribed business event and launch interim vs. final runs accordingly.

There is flexibility in the approach to utilize business events of modern software systems in combination with sequential automations. On the most general level a single business event (e.g. raw data delivery) could trigger a single end-to-end pipeline. On the most detailed level each individual check, conversion program and analysis program could be a single-step pipeline configured to react on business events proving the required inputs being available for the execution. Companies could use this approach to find the right balance for the individual SOP and compliance demands.

## CONCLUSION AND OUTLOOK

API-based sequential pipelines are a powerful tool to automate business processes in clinical data analysis and reporting activities. Such pipelines are an additional layer and come with additional efforts for maintenance and compliance. The benefits outweigh these drawbacks by far, but the operational procedures and precautions to maintain regulatory compliance have to be addressed.

The combination of such pipelines with business events published inside systems opens up a new set of possibilities for how to structure processes and automations while keeping complexity in check and regulatory compliance intact without additional efforts for any involved team.

Companies should look for software vendors and technology consultants providing systems and solutions that offer a sensible combination of business APIs and business events. Once the fundamental aspects of automation with APIs and events are solved, companies can dive deeper into further benefits of orchestrating such complex processes with the help of Agentic AI and other tools.

## REFERENCES

[1] https://www.reactivemanifesto.org/ (last accessed 02-Feb-2026)

[2] https://techcommunity.microsoft.com/discussions/integrationsonazure/orchestration-vs-choreography-how-to-pick-the-right-integration-pattern-in-azure (last accessed 02-Feb-2026)

[3] https://aws.amazon.com/compare/the-difference-between-soa-microservices/ (last accessed 02-Feb-2026)

[4] https://www.ibm.com/think/topics/soa-vs-microservices (last accessed 02-Feb-2026)

[5] https://solace.com/blog/evolution-of-apis-restful-event-driven-apis/ (last accessed 02-Feb-2026)

[6] https://blog.axway.com/learning-center/apis/basics/event-driven-vs-rest-api-interactions (last accessed 02-Feb-2026)

## CONTACT INFORMATION

Your comments and questions are valued and encouraged.  Contact the author at:
Author Name: Marc Jantke
Company: Entimo AG
Address: Stralauer Platz 33-34, 10243 Berlin, Germany
Email: mja@entimo.de
Website: www.entimo.com