# Integration Through a Practitioner's Eye

Srinivasa Narayanan, Subbu N. Subramanian, and the Tavant Team
Tavant Technologies
3114 Scott Blvd., Santa Clara, CA 95054, USA
{srinivas.narayanan, subbu}@tavant.com

## 1 Introduction

Publications in the area of data integration address several important data model, query processing, and data semantics issues. This paper presents integration from a practitioner's perspective – those practical pain points that are not discussed in typical research literature, but big enough to warrant careful study in order to realize practically useful integration systems. The paper discusses several challenges that arise in real-life integration scenarios, some techniques for addressing them, and identifies several key areas that require further study.

This paper is based on our experiences with building Tavant InterConnect, the crucial integration component of the Tavant Solutions Framework [NST01]. In Section 2, we set the stage for our discussions by providing an overview of the Tavant solution with special emphasis on its integration component. In Section 3, we discuss the nature of integrated data in practice and issues related to its representation. We discuss the use and manipulation of integrated data in Section 4, present several practical challenges that arise in real-life integration in Section 5, and finally conclude in Section 6.

## 2 An Overview of Tavant Integration System

The Tavant system is a framework for collaborative commerce solutions that help companies bring their sales and distribution processes to the Web. It leverages the internet as a medium to enable collaboration among various partners (such as manufacturers, dealers, distributors, and end customers) in the sales and distribution channels of large corporations. The systems, data, and business processes of the channel partners are usually autonomous and heterogeneous. Achieving a seamless integration towards providing the end user with a unified "single point of access" user experience is a key feature of the Tavant system.

Tavant InterConnect, the integration component of the Tavant system, provides a hub-and-spoke model for integrating the data and business processes of the channel partner systems. As illustrated in Figure 1, the Tavant integration hub integrates the various partner spoke systems and enables existing business processes on the spoke systems to be accessed via the hub. In addition, it provides new collaborative business processes enabled by the integration of the partner systems.

Invoking a business process that is exposed on the integration hub involves accessing data and business logic on the hub and from one or more spoke systems. The hub stores meta-data about the various systems and acts as an intelligent orchestrator by pulling data local to the hub as well as data resident on the spoke systems, invoking


**Bulletin of the IEEE Computer Society Technical Committee on Data Engineering**

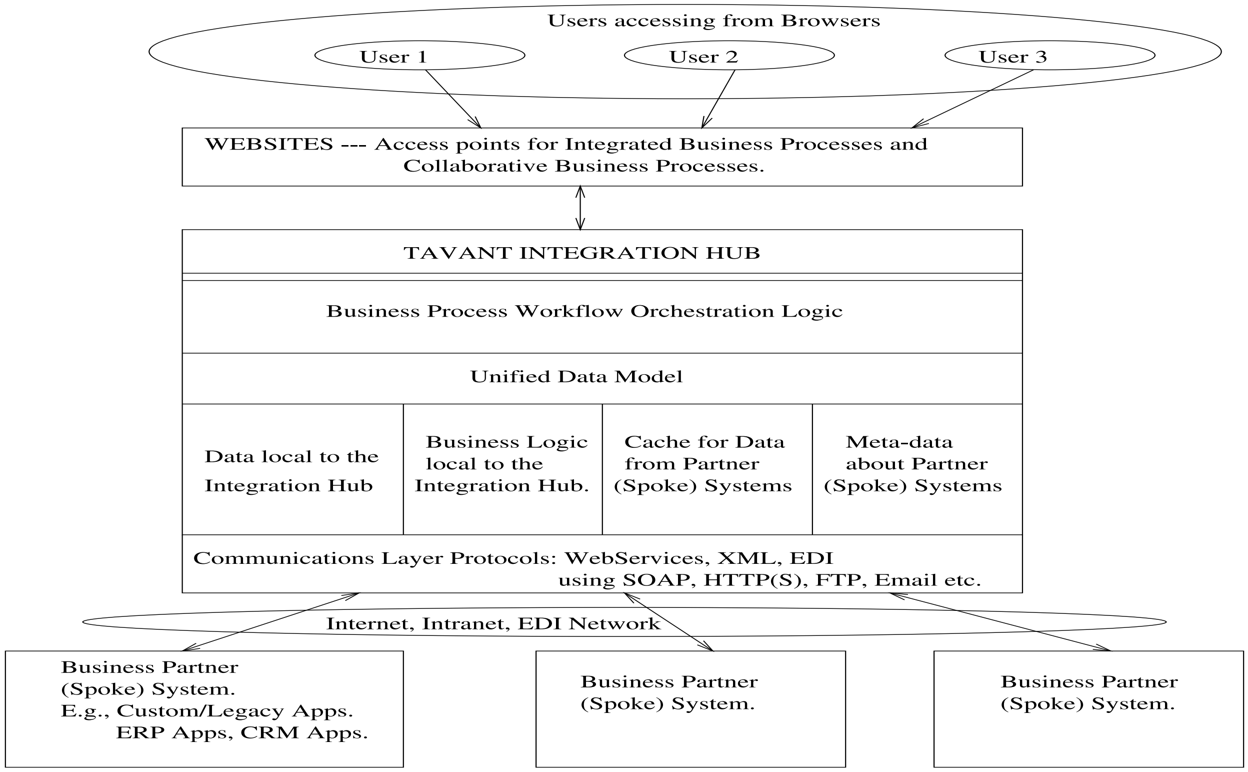


Figure 1: The Tavant Integration Architecture

appropriate business logic on the various systems, and composing the result of all these transactions into a result that is meaningful for the end user.

# 3 Integration in Real-life – Nature and Representation

In this section, we discuss the nature of integration as well as its influence on the representation of integrated data.

As mentioned in Section 2, the key goal of many real-life integration systems, including Tavant InterConnect, is to bring together related, but isolated business processes belonging to multiple component systems in such a manner that they can be seamlessly accessed over a common medium (such as the web). Thus, the ultimate focus is on *process integration* as opposed to data integration alone. We elaborate on this distinction below (Section 3.1). We discuss how business needs and real-life requirements motivate the need for integration in the presence of partial data and the challenges they impose on data representation (Section 3.2). We also motivate the need for extending the canonical data model to handle component system-specific needs and present some techniques to address this challenge (Section 3.3).

## 3.1 Process Integration *vs* Data Integration

The key goal of the Tavant system is to Web-enable existing business processes and to provide new collaborative business process among business partners. To achieve this goal, the main focus of the system is to integrate the business processes of the partners rather than achieve deep semantic integration of business data belonging to the component systems.

Data integration in the Tavant system involves mapping the data model of the component systems to Tavant hub's *canonical data model.* The canonical data model is powerful and allows for achieving a well-defined mapping among concepts belonging to the component data models whenever these concepts are precise and unambiguous in all systems. However, in real-life, there exist concepts that are unique to component systems or whose semantics vary among the component systems. In order to account for these, the canonical data model has the flexibility to allow for concepts to be modeled without necessarily associating any deep semantics with them. Though the model does not "interpret" these concepts, the hub system is still aware of them. This allows for the hub system to act as a conduit for these uninterpreted data to be passed (as part of a business process flow) to the specific component systems. The component systems, which are aware of the semantics of the data, individually interpret the data for further processing.

We now illustrate this with an example. The Tavant integration hub has a generic *Order Management module* that models the order processing functionality of various partners in the sell-side channel. The canonical data model for this module includes concepts such as *buyer, seller, shipping address,* and *billing address*. These concepts are usually precise and unambiguous across all the channel partners. However, there exist concepts that are unique to a channel partner. For example, consider a channel partner that has a proprietary set of *order processing codes* – codes such as urgent, normal, or ground that determine the shipping priority of the order – that the buyer can associate with an order. Since these order processing codes are unique to that channel partner, they are not modeled as concepts in the canonical data model. However, the model allows for capturing this information as part of the order processing workflow so that the codes can be sent to the channel partner component system for interpretation and further processing. As can be seen, the Tavant hub system focuses on integrating the *Order Submission process* across the various channel partners while respecting differences among the component systems. A simplified view of this integrated order submission process consists of the following steps – (1) *Product Selection*, (2) *Buyer Information and Preferences Entry* (that includes entering the priority code in the case of the channel partner discussed above), (3) *Shipping Information Entry*, (4) *Billing Information Entry*, and finally (5) *Order Submission.* Thus, the Tavant integration layer focuses on integrating these workflow processes to allow the channel partners to submit and receive orders, than on attaining deep data integration.

Another crucial component of the integration hub is a canonical business process flow model that acts as a master workflow template from which the component systems' workflows can be derived. The canonical workflow accounts for the minor differences in the component workflows. For example, the order submission process described earlier may differ slightly across the various channel partners. Some channel partners may not need the "Billing Information Entry" step of the workflow because their existing accounts and invoicing procedures do not require this information. Similarly, some channel partners may not want a specific section of the "Shipping Information Entry" (such as the one that allows the buyer to choose a "Shipment Carrier") because this information is not applicable to them. To address these needs, Tavant's Order Management module defines a master workflow template for the Order Submission process consisting of a set of components that can be customized to the individual channel partner needs as outlined above.

In practice, to facilitate collaboration, business partners maybe willing to make limited changes to their existing workflows. The definition of the canonical workflow must take into account the current workflows of the partner systems and the changes that are permissible on these workflows. If business processes can be represented using workflow definition languages, an interesting challenge is to automatically create the canonical workflow from the existing workflows and the changes permissible on these workflows.

## 3.2 Partial Integration

Many businesses are web-enabling their business processes in a phased manner by doing them one by one instead of doing it all at once. The business reasons are obvious – minimize risk, see early return on investment, and synchronize technology changes with user adoption rates. Hence, the integration hub system needs to deal with integrating partial business data and processes belonging to the component systems.

Consider again the example of the order management hub system from the previous section. Recall that the hub system's canonical model comprises of concepts such as Orders, Invoices, and Shipping as well as the business processes involving these concepts. Now, consider a scenario where a business wants to achieve integration and collaboration involving its invoicing system first and wants to do the same for its shipping system in a later phase. To address this requirement, the hub system needs to represent Invoice data in the canonical model *when the corresponding data for Orders and Shipments is not available.* This causes interesting technical challenges: (1) Data modeling becomes a more complicated task – *e.g.*, foreign key restrictions in the canonical model need to be relaxed. (2) Business logic needs to be made inherently aware that only parts of the data in the data model graph may be genuinely available. (3) With no clear models for specifying unavailability of data in relational model, queries on the data model must be careful not to return incorrect answers because of the partial availability of data. We discuss these issues further in Section 4.2 as well.

### 3.3 Representing Component System Specific Information In A Canonical Data Model

In section 3.1, we motivated the need for information that is specific to a component system to be represented in the canonical model without necessarily associating any deep data semantics in the model. However, the hub system needs to be able to process such "custom" information in some ways to help towards the goal of process integration. This includes providing the ability to *pass-through* this information in workflows, allowing for simple querying in the business logic, and performing UI manipulations in the presentation layer. Tavant system uses a *Custom Attribute* framework to represent information unique to a component system, that need not be interpreted in the canonical data model.

The Tavant integration hub caters to the needs of many component systems (belonging to several partners) each of which can potentially have a large number of custom concepts. In order to scale easily across a number of such partners, it is impractical to represent the custom concepts as first-class attributes (i.e. columns) in the relational model. In the Tavant custom attribute framework, such pieces of information are stored as $< attr\text{-}name, value >$ pairs in relations. For example, in the case of the order processing code example of Section 3.1, the "Order Code" is not explicitly modeled as an attribute of the Order relation. Instead, the code in the context of a specific order $O_1$ flowing to the component system $C_1$ is represented as a tuple $< C_1, O_1,$ "$OrderCode$", "$Urgent$"$>$ in the "Order Custom Attribute" relation.

There are several challenges that arise in the context of querying and manipulation of information stored as $< name, value >$ pairs. Lack of support in commercial databases for SchemaSQL-style querying ([LSS01]) makes even simple querying of custom attribute tables a complicated task. For example simple selection queries involving custom concepts need to be expressed as join queries on the custom attribute table. Another challenge is the creation of indexes on the custom attributes for performance reasons. It is not possible to build indexes on custom concepts due to the fact that the information is modeled as data and not schema. There is a need for commercial database products to address these issues.

## 4 Use and Manipulation of Integrated Data

In this section, we study the various contexts in which integrated data is used in real-life integration systems. We also discuss how integrated data gets further manipulated to provide for "function integration" and other post-integration requirements.

### 4.1 Integration Hub as a Cache

The Tavant integration hub provides seamless access to information from multiple component systems and offers a single-system view to its users. In most cases, the system hides from the user the fact that the integrated information and processes actually belong to multiple component systems. However, the overhead of pulling information from and orchestrating workflows among a multitude of component systems for every user request can cause slow response time and may not be acceptable from a system performance standpoint.

To address this performance issue, the Tavant integration hub has the ability to cache data locally from the component systems. This is useful especially in cases where the data on the component systems do not change frequently and when the timeliness of the data is not crucial for the end user. In this scenario, the cache acts a "transparent cache" where the user is unaware of the existence of the cache. The cache on the hub is maintained by explicit update requests from the component system (push model) or via an automatic refresh initiated by the hub system on a periodic basis (pull model).

Besides helping with performance, the cache on the integration hub allows for *function integration* by enabling the creation of completely new features or processes that do not exist in the component systems. For example, consider a product catalog from a component system that is cached on the integration hub. Let us say one of the component systems does not have support for an operation such as keyword search, while all other component systems are capable of performing this operation. If the integration hub has support for the keyword search operation, the users of the integration hub can seamlessly perform keyword search over *all* integrated data, including over data belonging to the "keyword search challenged" component system.

The above technique of compensating for the component systems' data and functions at the integration hub is particularly applicable in the context of integrating legacy systems. We revisit this issue in Section 5.2.

The Tavant integration hub also allows for scenarios where data changes occur frequently on the component system and/or timely and accurate information is crucial. In these cases, user requests are answered by pulling data dynamically from the component systems.

### 4.2 Leveraging Integrated Data to Handle Connectivity Challenges

Query processing in a distributed systems environment is a well-studied subject [OV99]. While many of the ideas are relevant in the context of a data integration setting as well, there are key differences due to the fact that these two environment differ in some fundamental ways.

While the distributed systems environment is tightly coupled, the data integration environment comprises of loosely coupled, autonomous component systems. In particular, the reliability of the component systems as well as reliable communication among them cannot be guaranteed. Thus in practice, the integration system should be able to cope with unavailable component systems and serve useful purpose even when some of the component systems are unavailable.

A technique for addressing the above problem makes use of the integration hub cache discussed in the previous section. This cache can be used to substitute for the component system when the component system is unavailable. This approach is based on the belief that it is better to present a possibly stale result to the user rather than not present any results at all. In these instances, the hub system may want to inform the user about the existence of the cache and warn the user that the data may not be current.

To continue with the example from previous section, let us say the product catalog in the integration hub cache contains pricing information for the various products. Consider a scenario where, while answering a user query regarding product price and availability, the hub fails to obtain the latest pricing information from the component system (say due to connectivity issues). The integration system may still want to present the user with the pricing information from the cache along with an indication to the user that the information may not be accurate.

In scenarios where the cache substitution technique may not be applicable (because the component system information is not cached or because security considerations prohibit the availability of pieces of information required for query processing), techniques of query relaxation ([BT98, GGM92]) can be applied to answer queries. The challenges here are that the hub system should be able to deal with missing and/or approximate information (at both the data and schema level) every step of the way till the results are presented to the user.

The hub's data integration model needs to be cognizant of and be able to deal with unavailable data and schema. Lack of proper models in commercial systems for dealing with unavailable or approximate information makes this a challenging task.

It is important to note that the above techniques for dealing with unavailable component systems may not be applicable in the context of *all* integration scenarios. The integration system should allow for declaratively specifying the scenarios where the cached information from the integration hub can be substituted and/or query relaxation techniques can be applied. The system should make use of these specifications to apply the appropriate technique for a given scenario. Also the integration system should allow for an option where the end user is made aware of the fact that in order to generate the results to her query, one or more of these scenarios have been applied.

### 4.3 Data Semantics as a Function of User Expectation

This section discusses yet another scenario that leverages the existence of the integration hub cache. It exploits the fact that in real-life, semantics is context sensitive. For example, when a user is browsing the product catalog, it can be acceptable to present a possibly stale and hence approximate price information. But when the user is placing an order for the same item, it is important that the user is presented the current accurate price of that item. A smart integration system can take advantage of these semantical distinctions that arise in real-life. For example, in the Tavant integration system, the catalog prices are cached on the hub on a nightly basis. Thus, prices for items can be more efficiently obtained for the more frequent catalog browsing activity. For the less frequent ordering activity, current prices are obtained from the component system using real-time webservice queries.

### 4.4 Integration and OLAP

Integration allows for related data in multiple, disparate systems and data formats to be accessible via one common data model. Such an integrated model naturally serves as a basis for business analysis and reporting. While integration facilitates business analysis, in practice there are several differences between an online transaction processing (OLTP) based integration system and an online analytical processing (OLAP) system ([KRRT98]). The integration system should be cognizant of this distinction. It should facilitate OLTP queries across component systems, while allowing for building an efficient OLAP system that leverages the integration model as much as possible. Better still, the integration system should be able to infer the optimal model for the OLAP system based on the business analytics workload.

Currently, this problem is solved in practice by creating an explicit datawarehousing server that is distinct from the integration server. The datawarehousing server is managed by data replication (often involving data transformations) from component systems as well as the integration hub. Finally, for seamless OLTP and OLAP querying via the integration server, the datawarehousing system is added as a component system in the integrated environment.

## 5 Challenges and Opportunities

In this section we identify several challenges that arise in real-life integration scenarios that require further research. We also outline some practical approaches that help address these challenges in the absence of elegant solutions.

### 5.1 Fault Tolerance and Error Recovery

Fault tolerance and error recovery in the context of a transaction that spans multiple distributed systems where each component system is transactional is a well-studied subject ([OV99]). However, work flow integration in our context involves performing transactions over loosely coupled component systems that do not necessarily support transactional semantics. Currently, no models exist to handle this scenario. Thus, when a component system involved in a workflow encounters a failure (unexpected errors or server crashes), it is very difficult to even detect what happened during the accesses to non-transactional resources, let alone recovering from the errors. Offering very strict transactional semantics to the entire application layer may be too prohibitive and even impossible in most cases. However, it would be useful to have models that offer a range of capabilities from making it possible to detect what happened, to even recovering from the error situations.

The above mentioned scenarios arise frequently in Webservice calls where an application communicates with a partner system over the Web. Note that the interfaces between these systems are typically not transactional. For example, let us say the user initiated an action on system A that causes an order to be submitted (as a webservice request) from system A to system B. Let the order be accepted by system B. However, before system B could respond to system A acknowledging the successful completion of the order request, let us say that system B crashes (in the course of some post-processing task on system B) that prevents the successful delivery of the acknowledgement to system A. In this scenario, the user should ideally be informed of the error that happened in system B *as well as* the fact that the order was successfully placed before the failure occurred. To program the application to remember this fact in case of any error that happens is a very difficult task. Due to the lack of proper models and transaction systems to deal with these situations, Tavant uses an application programming model where the whole sequence of operations on system A can be defined as a workflow involving a sequence of transactional operations. In such a model, the information about any completed transaction will be automatically stored persistently by the engine executing the workflow. In addition, to deal with these scenarios in a practical way, the Tavant system also keeps the user informed about such situations. For example, in the above scenario, system A would inform the user that an error occurred while system B was processing the order, the order *may* have been successfully processed, and that the user should check later to see if the order has indeed been accepted by system B.

### 5.2 Legacy Systems Integration

Webservices is increasingly being touted as the technology for dealing with legacy systems integration. In theory, Webservices promise increased interoperability and lower cost of application integration and data sharing regardless of operating systems, languages, or hardware platform. This is achieved by the availability of tools, techniques, and standards to easily “wrap” existing applications and enable them to communicate among each other. However, for reasons that include lack of tools in a variety of platforms, scarcity of skilled personnel, strict adherence to established standards and procedures, and businesses’ inherent resistance to change, there exist legacy systems that cannot be webservice enabled and only support legacy protocols. Towards this end, The Tavant system supports a variety of legacy protocols including FTP, EDI, and flat file upload.

Several challenges arise even in scenarios where webservice enabling of legacy systems is possible. The legacy system’s participation in the integration environment will naturally lead to increased workload on the system and greater demand for system resources. However, the application may not have been built in anticipation of the new workload. For example, consider a PC-based application in a corporate office that provides information on field inventory for select corporate users. Let us say this application needs to be webservice enabled so that dealers and distributors in the sales channel will have access to the global inventory information. Simply wrapping the application as a webservice is not sufficient to address this need because the PC-based application does not have the wherewithal to handle the huge volume of transactions initiated by the channel partners.

Scaling a webservice enabled legacy application to handle increased workload poses interesting challenges. In certain cases, clustering techniques can be used to address this issue. For example, the PC-based application of the above example can be scaled by means of a cluster of PCs running the application and a load balancer that handles transaction requests. However this assumes that the application is stateless – i.e. it processes each transaction request without any knowledge of previous requests. For the cases where this assumption is not valid, a technique used in practice is to cache both data and functions (or function results) on the integration hub. Thus, in these scenarios, the hub provides for function integration in addition to data and process integration.

### 5.3 Streaming Data – Query Processing Issues

Integrating data from multiple systems frequently involves several data transformations as the data flows through various levels of protocols among the different systems. When dealing with large data, the ability to stream data as it undergoes transformations (without the need for temporarily materializing the data) is a crucial factor in improving performance. While this concept of "pipelining" is well understood and implemented within database systems, many systems and protocols related to data transformation at the middleware or application level do not lay emphasis on transformations on streams and deal only with fully materialized representations of the data they need to convert. Important examples include commercially available tools for conversion between EDI and XML, and tools for conversion between XML and Java. While the XML SAX processor has the ability to deal with XML data streams, parsing at the SAX level has several overheads that are cumbersome for application development. The challenges in this area require more work at a level of academic research as well as commercial products.

## 6 Conclusion

In this paper, we discussed the practically useful problem of integrating disparate business data and processes and making them available on the Web. We presented the Tavant integration framework, discussed several challenges that arise in real-life integration scenarios, and our approach to addressing them. In particular, we discussed the nature of integration needs in real-life, issues related to representation, and the use and manipulation of integrated data to solve practical problems. We also identified several challenges that do not currently have elegant solutions and hence provide opportunities for further research.

## References


[BT98] Philippe Bonnet and Anthony Tomasic. Partial answers for unavailable data sources. *Lecture Notes in Computer Science*, 1495, 1998.

[GGM92] Terry Gaasterland, Parke Godfrey, and Jack Minker. Relaxation as a platform for cooperative answering. *Journal of Intelligent Information Systems*, 1(3/4):293–321, 1992.

[KRRT98] R. Kimball, L. Reeves, M. Ross, and W. Thornthwaite. *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouses.* John Wiley & Sons, 1998.

[LSS01] Laks V. S. Lakshmanan, Fereidoon Sadri, and Subbu N. Subramanian. SchemaSQL: An extension to SQL for multidatabase interoperability. *ACM Transactions on Database Systems*, 26(4):476–519, 2001.

[NST01] Srinivasa Narayanan, Subbu N. Subramanian, and The Tavant Team. Tavant system architecture for sell-side channel management. In *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, Roma, Italy*. Morgan Kaufmann, 2001.

[OV99] Ozsu, M.T. and Valduriez, P. *Principles of Distributed Database Systems.* Prentice Hall, 1999.