# A MODEL OF DATA CURRENCY IN MULTI-CHANNEL FINANCIAL ARCHITECTURES

(Research Paper)

**Cinzia Cappiello**
**Chiara Francalanci**
**Barbara Pernici**
Politecnico di Milano, Milano, Italy
{cappiell, francala, pernici}@elet.polimi.it

**Abstract**: Data quality is a critical factor for financial institutions where information is a primary and critical resource. Financial services are offered through multiple channels, such as branches, ATMs, telephone, and Internet channels, and may be supported by multi-functional software systems, to provide services such as, for instance, customer management, account management, credit management, which are characterized by complex software architectures. Different functional modules share data, which are stored in multiple source databases. Functional modules are usually not integrated across channels, as channels are implemented at different times, since they are realized in independent software projects, and are subject to stringent requirements of availability and performance. This lack of channel and functional integration raises quality problems in information products. In particular, in complex systems in which data are managed in multiple databases, timeliness is critical. In the paper, we focus on currency and we present a mathematical model to evaluate currency of data in software architectures with different degrees of integration across channels and functionalities.

**Key Words**: Data quality, Financial information systems, Multi-channel architectures, Timeliness, Currency, Volatility.

## 1. INTRODUCTION

The quality of information products is influenced by a number of factors. Enterprises providing both window-based and on-line services are faced with a variety of contrasting requirements. Services are provided through a variety of different channels, such as branches, ATMs, call centers, web, cellular phones (through GSM and WAP technologies), and other technology based channels. In general, an enterprise provides a variety of services to its customers, and the customers can access all or most of services through all channels. Information management should guarantee that data are accurate and up to date independent of the access channel. In addition, contrary to traditional window-based services, office hours tend to be extended to a 24x7 service availability requirement, so that customers are able to access the services at any time. Therefore, traditional modes of operations typical of service enterprises have been changed. For example, it is not possible anymore, to perform backups and checks on data outside office hours, assuming that no operation is being performed on the system. In addition, for on line services, performance is also an important requirement.

The primary aim is create products and services, which are in line with the market expectations, not only in terms of time but also of good quality. The impacts of poor data quality on enterprise are various. They include customer dissatisfactions, increased operational cost, less effective decision-making, and reduced ability to make and execute strategy [10]. The guarantee of correctness of processing activities is one of the main issues of the organizations [2].

For a technical point of view, the contrasting requirements illustrated above have originated a variety of different architectural solutions offered by vendors, ranging from completely centralized systems, to completely distributed systems, where each system is dedicated to a single channel and a single functionality. In fact, while a centralized system integrates all data and helps guaranteeing data that are accurate and up to date, a distributed system is easier and faster to implement, and provides better availability and performance in most cases. Therefore, usually information services are provided on separate systems first, and data are fully or partially integrated at a latter stage. Integration technologies include the use of datawarehouses for integrating data from different database [9]. Methodologies for partitioning and replication of data to providing more efficient services have also been studied in the in distributed database systems area [13].

The goal of this paper is to study implications on data quality of the different software architectures. We will focus in particular of measuring timeliness of data, based on its currency. In particular, we study data quality in multi-channel financial services, and we provide results of simulations obtained comparing different architectures with different characteristics of provided services, and different patterns of behaviour of customers.

Based on this analysis, it is possible to define criteria for design choices in the integration process: based on the characteristics of services and customers it is analyze which architecture provides better data in terms of timeliness. Physical and virtual institutions are compared, and channel and functional integration analysed.

Different definitions of timeliness and currency have been provided in the literature. In [11] currency and related dimensions are discussed. Up to date and outdated data are defined, and currency refers to the degree to which a datum in question is up-to-date. Currency is measured in terms of time from which the data are not anymore up-to-date.

In [2] currency is measured in terms of age, delivery time, and input time of data in the system. Input time is the time at which the data unit is obtained, delivery time is when the information product is delivered to the customer, and age states how old is the data when it is received. Currency is a characteristic of capture of data, while volatility is defined in terms of how long the data item remains valid, Timeliness is defined as a function of currency and volatility.

In the present paper, we elaborate the concept of currency, and we define a method to measure currency given the characteristics of the architecture of the system, refresh times for data in databases when multiple databases are realigned periodically, and the patterns of use of the system in terms of access to services through different channels.

In Section 2, we discuss more in detail the characteristics of multi-channel financial institutions and provide a definition of currency, volatility, and timeliness in this context. In Section 3, we present the different types of architectures for multi-channel services and we compare their characteristics. In Section 4, we define a method to measure currency based on the specific characteristics of this type of systems. In Section 5, we provide simulation results and analyze how the currency measures can be used to define strategies for systems integration.

## 2. DATA QUALITY ISSUES IN MULTI-CHANNEL FINANCIAL INFORMATION SYSTEMS

Financial institutions offer an increasing number of services to their customers. Traditionally, services could be exclusively delivered to customers through branches. The development of the Internet and the convergence between computing and communication technologies have opened opportunities to reach customers through a variety of technology-based channels, such as personal computers, mobile phones, televisions, kiosks, advanced ATMs and so on. Financial information systems of multi-channel institutions are comprised of complex and heterogeneous applications. This research focuses on the software components of a multi-channel architecture that are responsible for data quality. At a high level, a multi-channel software platform can be viewed as a black box that receives elementary data extracted from multiple sources and provides integrated information for different financial services and distribution channels. Data constitute a critical resource in financial information systems and data quality issues are particularly relevant for the following reasons:

1. Financial information is inherently critical and errors are consequential.
2. The quality of financial data has a strong economic impact.
3. Customers perceive the quality of financial data as an essential component of the quality of service.
4. Multi-channel software platforms extract data from a high number of databases and, thus, are more liable to inconsistencies.

Data quality has been defined in the literature as "the measure of the agreement between the data views presented by an information system and that same data in the real world" [6]. Data quality can be measured along the following dimensions:

- Relevance, granularity and level of detail, which are associated with data views.
- Accuracy, consistency, currency and completeness, which are associated with data values.
- Format and ease of interpretation, which are associated with the presentation of data.
- Privacy, security, and ownership, which are general dimensions.

This paper focuses on quality dimensions associated with data values and, in particular, on *currency*. As discussed in the following, the ability to distribute services through multiple channels raises data currency issues that are specifically related to software design choices.

We define currency as a property of data values that are not out of date with respect to their actual value in the real world. It also represents a factor of accuracy, which is a more general property of data values that are consistent with their actual value.

Currency can be measured as the time interval between the latest update and the time at which data are accessed by users. With respect the definition given in [2], we assume that in on-line systems the age of data is always zero, i.e., that the data are immediately available to be stored in the system, as soon as they are provided. For instance, if a withdrawal of money is performed from an ATM machine, the relevant information is stored in the system before the money is actually delivered to the customer.

Data are also associated with a time interval measuring their *volatility*, that is the average time length for which data remain valid [2]. Volatility is considered a static property that is independent of architectural design choices. For example, the quotation of a stock remains valid for only a few seconds irrespective of architectural choices.

Currency should always be smaller than volatility in order for users to access timely (or up-to-date) data, that is:

$$Volatility - Currency > 0 \qquad (1)$$

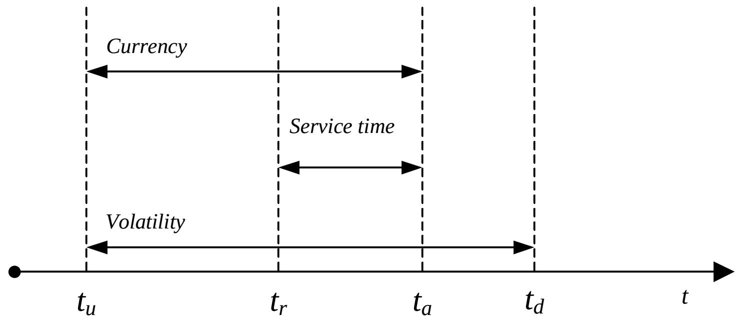


Figure 1 – Relationship between volatility and currency.

Figure 1 shows a graphical representation of the relationship between volatility and currency. In Figure 1, a user is supposed to request access to data at time $t_r$ and to be provided data at time $t_a$ after a *service delay* $T_s$. The time of the latest update is referred to as $t_u$ and, therefore, currency $T_c$ can be measured as $(t_a\text{-}t_u)$. Given a measure of data volatility $T_v=(t_d\text{-}t_u)$, condition (1) can be expressed as:

$$T_v - T_c = t_d - t_a > 0 \qquad (2)$$

While volatility is a static property of data, software design choices at an architectural level can change data currency, and therefore satisfy condition (2) to a different degree. As shown in Figure 1 currency includes service time, which is not considered in isolation in this paper.
We define *currency level* as the percentage of up-to-date information products delivered to customers, and, in turn, the next sections discuss four different types of software architectures and provide an operational measure of currency level that supports their comparison.

## 3. MODELS OF INFORMATION SYSTEMS ARCHITECTURES IN THE FINANCIAL INDUSTRY

As noted before, financial services are offered through multiple channels and are supported by multi-functional software systems. As discussed in the introduction, functional modules are seldom integrated across channels; therefore, they operate on operational databases that are not shared with other modules and need to be periodically realigned with databases of other modules.
From a data standpoint, this lack of integration across channels and functionalities raises currency issues. Users can access financial services from different channels within the same refresh period. For example, a user may withdraw money from an ATM and soon afterwards check his or her account's balance from the Internet. As money has been withdrawn from a channel different from the Internet, the user may be returned obsolete account information. This type of data currency problems has been found to constitute a significant limit to the overall quality of service. To obviate currency problems, financial institutions should increase the degree of integration of their information system architecture. However, full integration is usually considered too expensive and they tend to implement incremental integration strategies [5].

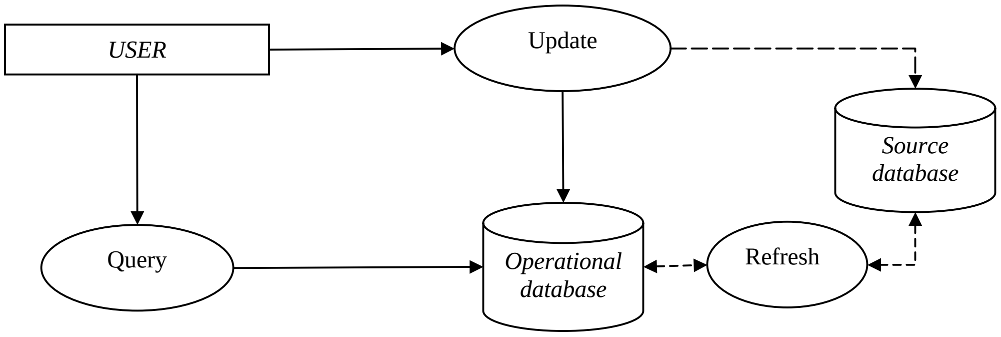


**Figure 2 – General data interaction for a user of a specific functional area and channel (front-end).**

The following sections discuss a classification of integration strategies. Strategies are classified by assuming that users both read and update data from operational databases and do not access source databases directly, that is they cannot follow the dashed path in Figure 2. This assumption is a simplification with respect to practice, since also sources can be updated directly, but can be removed without affecting the model of currency proposed in Section 4, where updates of all operational databases are considered. This paper focuses on currency problems due to a lack of integration among operational databases. Data misalignments between operational and source databases will be considered in future work.

## 3.1 Goal of integration strategies: Fully integrated multi-channel architecture

Let $c_1$, $c_2$,…,$c_N$ be a set of N channels and $f_1$, $f_2$,…,$f_M$ be a set of M functionalities. Figure 3 shows the blueprint of a fully integrated architecture where all channels share the same set of functionalities and all functionalities access the same operational database *od*.

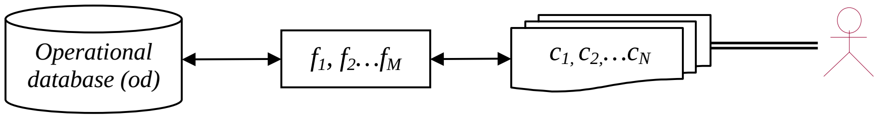


Figure 3 – Fully integrated architecture.

This architecture represents the goal of incremental integration strategies, as it has no currency problems, since data which is provided is always up-to-date at request time. Users accessing any functionality from any channel will certainly read and update the same information retrieved from a common operational database.

## 3.2 Starting point of integration strategies: Multi-channel architecture with the lowest degree of integration

The architecture with the lowest degree of integration is an architecture where each channel-functionality combination has access to a separate operational database $od_{ij}$. As noted before, this is a likely scenario as financial institutions typically implement new channels with dedicated functionalities. As new services are added over time, corresponding functionalities will be designed ad hoc for different channels and may not be integrated with existing functionalities.
Data are periodically realigned with a given refresh period. The number of operational databases to be periodically aligned is potentially high. Thus, refresh is a costly activity and the affordable refresh interval may be long. Although less expensive, this architecture presents data currency problems. In fact, within the same refresh period users may be returned obsolete data if they use different functionalities through the same channel or access the same functionality from different channels.
In the following, this architecture will be considered as the starting point to discuss integration strategies.

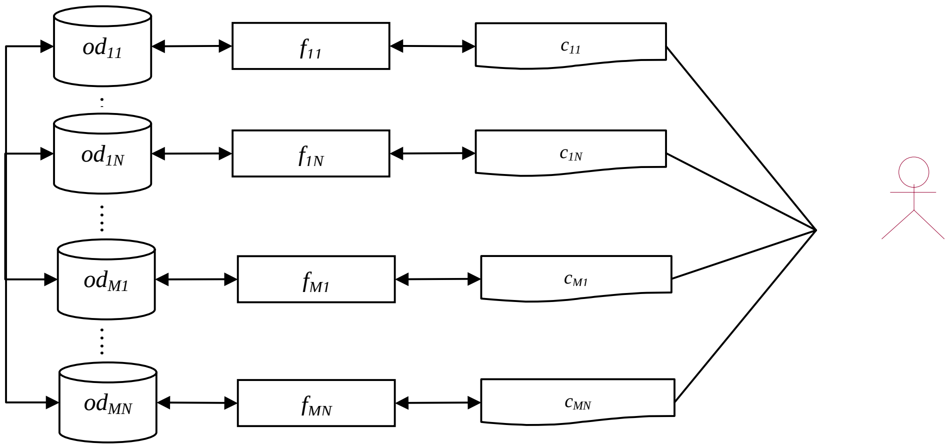


**Figure 4 - Architecture with the lowest degree of integration**

## *3.3 Channel integration strategy*

The degree of integration of the starting-point architecture discussed in Section 3.2 can be increased by integrating all functionalities managing the same channel or by integrating corresponding functional areas across channels. Both strategies are followed in practice. The first strategy will be referred to as channel integration strategy, the latter as functional integration strategy. Figure 5 shows the blueprint of the architecture that is obtained by implementing the channel integration strategy on the starting-point architecture. Note that the architecture in Figure 5 is fully channel integrated, since the channel integration strategy has been implemented for all channels. Intermediate degrees of integration are also possible.

Users of a fully channel integrated architecture may be returned obsolete data only if they change channel, while if they access different functionalities through the same channel are provided up-to-date data. The number of operational databases to be periodically aligned is lower that in the starting-point architecture and, accordingly, refresh will be less costly.

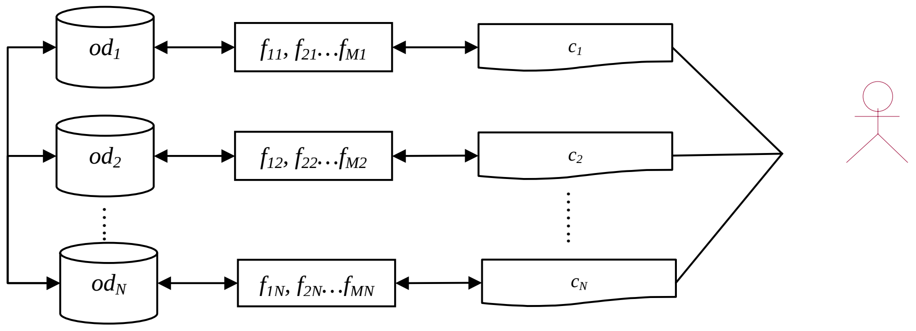

$od_1$
$f_{11}, f_{21}\ldots f_{M1}$
$c_1$
$od_2$
$f_{12}, f_{22}\ldots f_{M2}$
$c_2$
$od_N$
$f_{1N}, f_{2N}\ldots f_{MN}$
$c_N$

**Figure 5 – Channel integrated architecture.**

## *3.4 Functional integration strategy*

Figure 6 shows the blueprint of the architecture that is obtained by implementing the functional integration strategy on the starting-point architecture. The architecture in Figure 6 is fully functionally integrated, since the functional integration strategy has been implemented for all corresponding functionalities of all channels. Similar to the channel integration strategy, this strategy also allows intermediate degrees of integration.

Users of a fully functionally integrated architecture may be returned obsolete data only if they access different functional areas, irrespective of the channel that is selected to interact with their financial institution. As in the channel-integrated architecture, operational databases have to be periodically realigned.

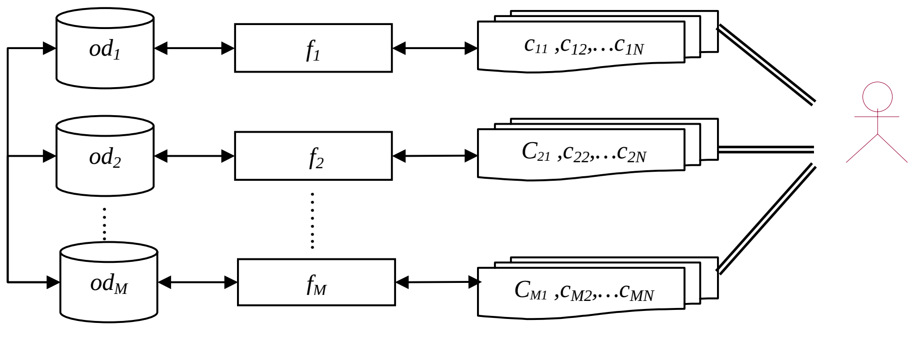

$od_1$
$f_1$
$c_{11}, c_{12},\ldots c_{1N}$
$od_2$
$f_2$
$C_{21}, c_{22},\ldots c_{2N}$
$od_M$
$f_M$
$C_{M1}, c_{M2},\ldots c_{MN}$

**Figure 6 – Functionally integrated architecture.**

The next section presents a formal model of currency level for the architectural blueprints presented in this section. In Section 5, simulations will compare levels of currency across architectural blueprints with different patterns of users' access to channels and functionalities.

## 4. TIME QUALITY MEASURES

The goal of this section is to present a model of currency quality attributes, defined in the following as model variables (Table 1). The model provides a mathematical definition of currency that can be compared across the architectures discussed in Section 3. Architectural choices result in changes of currency quality attribute values and therefore variations of currency levels. Consequently, the model supports the comparison of different architectures and the evaluation of corresponding currency levels in different application contexts that involve varying values of model variables.

The model is defined on the basis of the characteristics of the architecture with the lowest degree of functional and channel integration, which represents the most complex case (see Section 3). The definition of currency levels for architectures with a higher degree of integration is then derived from these initial definitions.

| ***Variable*** | ***Symbol*** | ***Description*** |
|---|---|---|
| Operational database | $od_{ij}$ | Set of data supporting the i-th set of functionalities on the j-th channel. |
| Refresh time | $rt_{ij,mn}$ | Time interval between alignments of operational databases $od_{ij}$ and $od_{mn}$ ($rt_{ij,mn}= rt_{mn,ij}$) |
| Operational frequency | $of_{ij}$ | Average frequency with which users of the ith functionality of the jth channel change one data unit of operational database $od_{ij}$. |
| Combined frequency | $cf_{ij,mn}$ | Average frequency with which users of the ith functionality of the jth channel change one data unit in $od_{ij} \cap od_{mn}$. |
| Volatility | $v_{ij}$ | Average volatility of data in $od_{ij}$. |

**Table 1 – Model variables.**

Let us consider an architecture with M functionalities $f_i$ and N channels $c_j$. As discussed in Section 3, this architecture has $M \times N$ operational databases, referred to as $od_{ij}$, where i and j indicate the ith functionality and the jth channel, respectively. Operational databases are defined as sets of data. Each operational database, as shown in Figure 2, is either directly updated by the user through a service, or periodically refreshed from the sources (or other operational databases as discussed in previous section). The refresh module in Figure 2 is implemented through extraction, aggregation and transformation operations. Two operational databases $od_{ij}$ and $od_{nm}$ may overlap, that is:

$$od_{ij,mn} = od_{ij} \bigcap od_{mn} \neq \varnothing$$

As shown above, data sharing across operational databases raises a need for periodic data alignment. To be as general as possible, we consider realignment parameters for each pair of databases: each pair of operational databases $od_{ij}$ and $od_{mn}$ is aligned with a refresh period $rt_{ij,mn,}$ which can be seen as the time interval before the data used by the mth functionality of the nth channel is updated with data created or modified by the ith functionality of the jth channel. Refresh periods $rt_{ij,mn}$ can be represented in a matrix RT. An element of RT is not null if corresponding $od_{ij,mn} \neq \varnothing$ and if a data unit, which belongs to $od_{ij,mn}$ is changed in $od_{ij}$.

$$RT = \begin{vmatrix} 0 & \dots & \dots & rt_{11,MN} \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ rt_{MN,11} & \dots & \dots & 0 \end{vmatrix} \qquad \forall i, j, m, n : i = m \wedge j = n \Rightarrow r_{ij,mn} = 0$$

As discussed in Section 2, data quality also depends on data volatility. Data has to be timely when users receive it. In this paper, refresh periods $rt_{ij,mn}$ are assumed to satisfy the following condition:

$$\forall i, j, m, n : rt_{ij,mn} < \min\lfloor v_{ij}, v_{mn} \rfloor$$

Intuitively, to guarantee given levels of currency, refresh periods should decrease as data change frequency increases. To tie refresh periods with the frequency of data changes, operational databases $od_{ij}$ are associated with two parameters:

- *The operational frequency $of_{ij}$*, defined as the average frequency with which users of the ith functionality of the jth channel changes one data unit of operational database $od_{ij}$ in a time unit.
- *The combined frequency $cf_{ij,mn}$*, defined as the average frequency with which users of the ith functionality of the jth channel changes one data unit in $od_{ij} \cap od_{mn}$ in a time unit.

The average number of data units that are modified by users of operational database $od_{ij}$ and need to be aligned with $od_{mn}$ after the refresh period $rt_{ij,mn}$ evaluates to:

$$du_{ij,mn} = cf_{ij,mn} \times rt_{ij,mn}$$

If $du_{ij,mn} < 1$, alignments occur, on average, after each change in a data unit and users always access current data. Otherwise, users may retrieve data from an operational database that have already been changed within another operational database and are therefore obsolete. In the following, we use average values for our computations. The implication is that data currency is not guaranteed 100%, but occasional out-of-date values can be generated even if the condition above is valid.

It is hypothesized that frequencies of data changes and frequencies of data accesses coincide. In fact, in financial systems all operations are registered and therefore correspond to updates. Under this hypothesis, from combined frequencies $cf_{ij,mn}$ it is possible to calculate the frequency $ef(t)_{ij,mn}$ with which at time $t$ a user of the ith functionality changes one data unit in $od_{ij} \cap od_{mn}$ that has already been changed in $od_{mn}$.

If $t_0$ is the time of the most recent refresh, the portion of $od_{ij} \cap od_{mn}$ that is modified by a user of the mth functionality of the nth channel between $t_0$ and $t$ is:

$$\frac{\int_{t_0}^{t} cf_{mn,ij}\, dt}{\left| od_{ij,mn} \right|} = \frac{cf_{mn,ij} \times (t - t_0)}{\left| od_{ij,mn} \right|} \qquad t_0 \le t \le t_0 + rt_{ij,mn},$$

where $\left| od_{ij,mn} \right|$ indicates the cardinality of $od_{ij,mn}$, that is the total number of data units in $od_{ij} \cap od_{mn}$.

The frequency $ef(t)_{ij,mn}$ with which a user of the ith functionality of the jth channel changes one data unit in $od_{ij} \cap od_{mn}$ that has already been changed in $od_{mn}$ can be estimated as follows:

$$ef(t)_{ij,mn} = \min\left( cf_{ij,mn} * \frac{\int_{t_0}^{t} cf_{mn,ij}\, dt}{\left| od_{ij,mn} \right|} ; cf_{ij,mn} \right),$$

where $ef(t)_{ij,mn}$ evaluates to $cf_{ij,mn}$ whenever the combined frequency of users of $od_{mn}$ are such that data in $od_{ij} \cap od_{mn}$ are modified multiple times within the refresh period $rt_{ij,mn}$.

Note that a user of the ith functionality of the jth channel reading or modifying an obsolete data unit does not necessarily generate an error. When $od_{ij}$ and $od_{mn}$ are refreshed, alignment operations can restore data correctness in most cases. However, users accessing obsolete data are offered a lower-quality service and, in financial systems, they typically experience functional restrictions [Redman 1996]. A measure of the average number of obsolete data accessed within a refresh period by all users of the ith functionality of the jth channel is:

$$\overline{n}_{ij,mn} = \min\left(\int_{t_0}^{t_0+rt_{ij,mn}} ef\left(t\right)_{ij,mn} dt; \left|od_{ij,mn}\right|\right)$$

If every $od_{ij,mn}$ was independent from the other, the fraction of out-of-date data units in $od_{ij}$ could be obtained as the summation of $\overline{n}_{ij,mn}$ for all $od_{mn}$ that share data with $od_{ij}$ divided by the cardinality of $od_{ij}$, that is:

$$outofdate_{ij} = \frac{\sum_{n=1}^{n=N}\sum_{m=1}^{m=M}\overline{n}_{ij,mn}}{\left|od_{ij}\right|}$$

The measure of currency *currency_level* $_{ij}$ of the ith functionality of the jth channel can then be obtained as:

$$currency\_level_{ij} = 1 - outofdate_{ij}$$

Commonly, in real cases intersections between operational databases $od_{ij,mn}$ are not null. In these cases it is necessary to consider data as belonging to the intersections. For these data items, the frequencies have to be added because data has a change frequency that is the sum of all the frequencies deriving from other operational databases. However, in out-of-date data units calculation these data must not be considered twice, so the formula is:

$$outofdate_{ij} = \frac{\sum_{n=1}^{n=N}\sum_{m=1}^{m=M}\left(\overline{n}_{ij,mn} - \left(\sum_{\substack{h>m\\k=n}}^{M}\left(\frac{\overline{n}_{ij,hk}}{od_{ij,hk}} * \left|od_{ij,mn}\bigcap od_{ij,hk}\right| - \sum_{\substack{y>h\\z=k}}^{M}\frac{\overline{n}_{ij,yz}}{od_{ij,yz}} * \bigcap_{\substack{v=m\\v\neq z}}^{y} od_{ij,vz}\right) + \sum_{\substack{k>n\\y=m}}^{N}\left(\frac{\overline{n}_{ij,hk}}{od_{ij,hk}} * \left|od_{ij,mn}\bigcap od_{ij,hk}\right| - \sum_{\substack{z>k\\y=h}}^{N}\frac{\overline{n}_{ij,yz}}{od_{ij,yz}} * \bigcap_{\substack{v=n\\v\neq y}}^{z} od_{ij,yv}\right)\right)\right)}{\left|od_{ij}\right|}$$

Other architectural models discussed in Section 3 have a higher degree of functional and channel integration. Operational databases of architectural models with a higher degree of integration can be obtained as the union of a subset of operational databases $od_{ij}$. For example, if the ith and mth functionalities of the jth channel are integrated, the new operational database is $od_{ij} \cup od_{mj}$. Due to these relationships among operational databases, estimates of currency for architectural models with a higher degree of integration can be derived from the values of *currency_level*$_{ij}$.

In Section 3, two integration strategies have been distinguished, referred to as functional and channel integration, respectively. Operational databases $od_j$ of the functionally integrated architecture (Section 3.4) can be obtained from $od_{ij}$ as:

$$od_i = \bigcup_{j=1}^{N} od_{ij}$$

Conversely, operational databases $od_i$ of the channel integrated architecture (Section 3.3) can be obtained from $od_{ij}$ as:

$$od_j = \bigcup_{i=1}^{M} od_{ij}$$

In general, the intersection between two operational databases $od_j$ storing data of the same set of functionalities for different channels will be greater than the intersection between two operational databases $od_i$ managing data for different functionalities. For example, let $od_{11}$, $od_{12}$ , $od_{21}$, $od_{22}$ be four operational databases corresponding to two functionalities and two channels. The following relationships hold if channel (a) and functional (b) integration are implemented:

| **(a)** |
|---|
| $od_1 = od_{11} \bigcup od_{21} \qquad od_2 = od_{12} \bigcup od_{22}$ <br> $od_1 \bigcap od_2 = (od_{11} \bigcup od_{21}) \bigcap (od_{12} \bigcup od_{22}) = (od_{11} \bigcap od_{12}) \bigcup (od_{11} \bigcap od_{22}) \bigcup (od_{21} \bigcap od_{12}) \bigcup (od_{21} \bigcap od_{22})$ |
| **(b)** |
| $od_1 = od_{11} \bigcup od_{12} \qquad od_2 = od_{21} \bigcup od_{22}$ <br> $od_1 \bigcap od_2 = (od_{11} \bigcup od_{12}) \bigcap (od_{21} \bigcup od_{22}) = (od_{11} \bigcap od_{21}) \bigcup (od_{11} \bigcap od_{22}) \bigcup (od_{12} \bigcap od_{21}) \bigcup (od_{12} \bigcap od_{22})$ |

The intersections between operational databases will coincide in cases (a) and (b) if the following condition is satisfied:

$$(od_{11} \bigcap od_{12}) \bigcup (od_{21} \bigcap od_{22}) = (od_{11} \bigcap od_{21}) \bigcup (od_{12} \bigcap od_{22}) \cdot$$

Typically, the condition above is not satisfied, while functional integration increases the cardinality of overlaps, that is:

$$\left|(od_{11} \bigcap od_{12}) \bigcup (od_{21} \bigcap od_{22})\right| > \left|(od_{11} \bigcap od_{21}) \bigcup (od_{12} \bigcap od_{22})\right| \cdot$$

However, operational and combined frequencies and refreshment periods also affect the level of data currency achieved with different integration strategies. As a consequence, the choice between functional and channel integration as well as the degree of integration in both cases should be based on users' behaviour. Simulations will compare different architectures in different application contexts characterized by different operational and combined frequencies and will try to infer the architectural integration strategy that maximizes data currency and minimizes refresh periods. Future work will collect empirical data to describe real application contexts.

## 5. SIMULATION RESULTS

Simulations compare currency levels of different architectures in different application contexts. Different types of financial institutions are considered:

– Global, national and regional institutions, depending on their size and geographical presence.
– Physical or virtual institutions; the latter exclusively operate through technology-based channels, while the former also operate through physical branches.

Note that virtual institutions typically operate at a global or national level. Accordingly, regional virtual institutions are not considered. Users are classified into three categories depending on the average number of transactions that they complete over time [7]:

– Active users, executing more than 200 transactions per year.
– Moderate users, executing more than 20, but less than 200 transactions per year.
– Sleepy users, executing less than 20 transactions per year.

Different types of financial institutions have a different mix of active, moderate and sleepy customers, with different patterns of access to services across channels. Table 2 reports the composition of customers for different types of financial institutions used for simulations [4].

| | active | moderate | sleepy | %online user |
|---|---|---|---|---|
| Regional Physical Bank | 25% | 40% | 35% | 10% |
| Global Physical bank | 68% | 18% | 14% | 10% |
| Global Virtual bank | 50% | 30% | 20% | 100% |
| National physical bank | 50% | 25% | 25% | 10% |
| National virtual bank | 50% | 30% | 20% | 100% |

**Table 2- Composition of customers for different types of financial institutions**

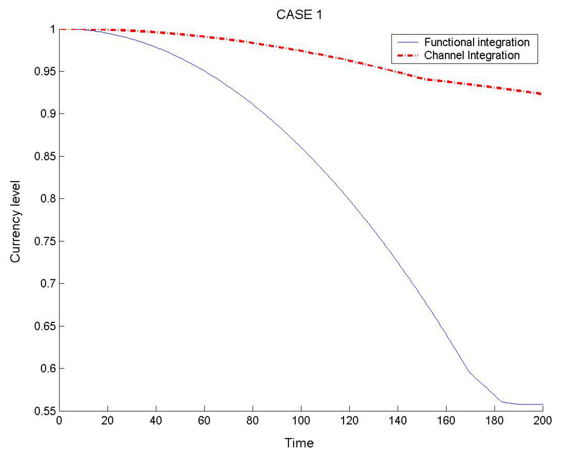
Four categories of financial services are considered: home banking, trading, credit card and insurance. Each category of services can be offered through four types of channels: Branch, Internet, Call Center and GSM (SMSs). Table 3 and Table 4 specify the distribution of customers' transactions across services and channels for 7 types of institutions (cases 1-7) listed as rows. Values in Table 3 and Table 4 are consistent with empirical benchmarks in the financial literature, but will be verified and refined in future work.

| | Home banking | Trading | Insurance | Credit card |
|---|---|---|---|---|
| CASE 1- Regional Physical Bank | 55% | 30% | 6% | 9% |
| CASE 2- Regional Physical bank | 40% | 50% | 4% | 6% |
| CASE 3- Regional Physical bank | 55% | 30% | 6% | 9% |
| CASE 4 - Global Physical bank | 50% | 33% | 7% | 10% |
| CASE 5 - Global Virtual bank | 50% | 33% | 7% | 10% |
| CASE 6 - National Physical bank | 53% | 32% | 6% | 9% |
| CASE 7 - National Virtual bank | 53% | 32% | 6% | 9% |

**Table 3 - Distribution of operational frequencies across services**

| | Branch | Technology-based channels | Internet | Call center | GSM SMSs |
|---|---|---|---|---|---|
| CASE 1- Regional Physical Bank | 90% | 10% | 60% | 25% | 15% |
| CASE 2- Regional Physical bank | 90% | 10% | 60% | 25% | 15% |
| CASE 3- Regional Physical bank | 50% | 50% | 60% | 25% | 15% |
| CASE 4 - Global Physical bank | 90% | 10% | 60% | 25% | 15% |
| CASE 5- Global Virtual bank | 0% | 100% | 60% | 25% | 15% |
| CASE 6 - National Physical bank | 90% | 10% | 60% | 25% | 15% |
| CASE 7 - National Virtual bank | 0% | 100% | 60% | 25% | 15% |

**Table 4 - Distribution of operational frequencies across channels**



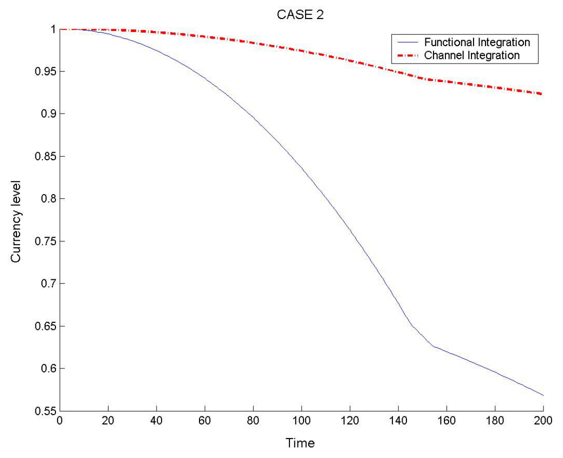


**Figure 7 – Currency levels in cases 1 and 2.**

Simulations compare the channel and functional integration strategies in cases 1-7 with varying refresh periods. Refresh periods are considered identical for all pairs of local databases. Figure 7 compares currency values in cases 1 and 2. In both cases, the channel integration strategy shows higher values of currency as refresh period increases. These results are consistent with the distribution of users' transactions across channels, as in cases 1 and 2 only 10% of all transactions is hypothesized to be executed through technology-based channels. As this percentage increases to 50% (case 3, Figure 8) the difference of currency between the channel and the functional integration strategies significantly decreases.

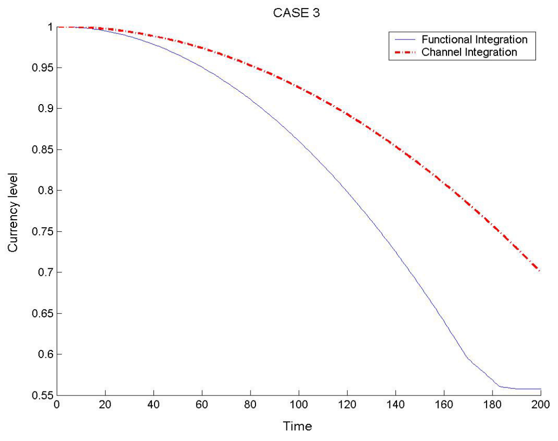
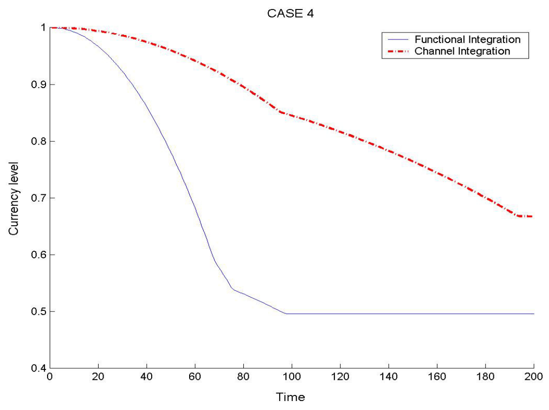


**Figure 8 – Currency levels in case 3**



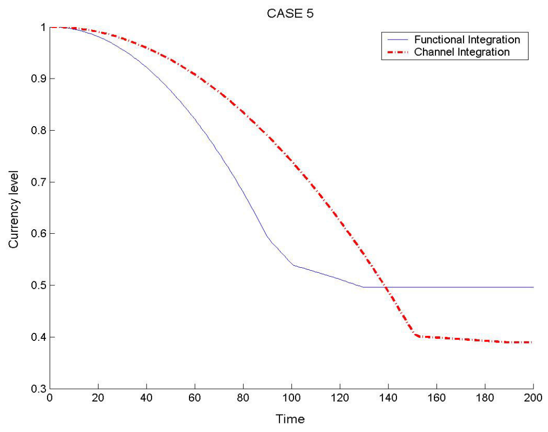


**Figure 9- Currency levels in cases 4 and 5**

Figure 9 compares currency levels in cases 4 and 5. Cases 4 and 5 represent a physical and a virtual global institution, respectively. The virtual institution shows a threshold value of refresh period above which the channel integration strategy becomes less efficient than the functional integration strategy. Similar trends are obtained in cases 6 and 7, representing physical and virtual national institutions.

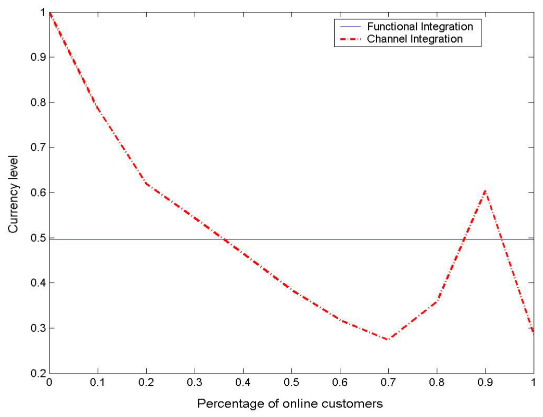


**Figure 10- Currency levels in case 5 as a function of the percentage of online customers**

Figure 10 shows a trade-off between functional and channel integration that occurs as the percentage of online customers increases. When customers are evenly distributed across channels, the functional integration strategy shows higher currency levels.

## 6. CONCLUSIONS

The purpose of this paper is to define a mathematical model to evaluate data currency levels that accounts for the characteristics of updates in distributed databases in the context of multi-channel and multi-service applications. The currency level indicator is a measure of the portion of data stored in a database that is out-of-date after a given time period from the latest update. Simulations compare different real cases of financial institutions characterized by different customers' behaviour, size, service and channels. Results show how data integration across channels increases currency levels more than integration across functionalities in the majority of cases. However, a functional integration strategy is more efficient when users' accesses are distributed evenly across channels.

Future work will examine volatility requirements and will present a model to obtain consistent values of refresh periods based on currency levels. Besides, future research will conduct a survey to collect real data on customer behaviour. Real data could be used to select the correct mix of channel and functional integration strategies depending for different patterns of customer behaviour and, possibly, provide methodological design guidelines.

## REFERENCES

[1] Ballou D. P., Pazer H.L., Modelling Data and Process Quality in Multi-input, Multi-output Information Systems. *Management Science*, vol. 31, No. 2, February 1985.

[2] Ballou D. P., Wang R., Pazer H.L., Tayi G.K., Modelling Information Manufacturing Systems to Determine Information Product Quality. *Management Science*, vol. 44, No. 4, April 1998.

[3] Ballou D. P., Pazer H.L., Designing Data Information Systems to Optimize the Accuracy-timeliness Tradeoff. *Information Systems Research*, 1995.

[4] Bracchi G., Francalanci C., Bognetti A., *La Banca multicanale in Europa*, Edibank 2002 (in italian).

[5] Escalate, Inc., Multi-channel Integration, a Retailer's Perspective. Available on line at: *www.escalate.com/whitepaper3.htm*, 2001.

[6] Kaplan D., Krishnan R., Padman R., Peters J., Assessing Data Quality Accounting Information Systems. *Communications of the ACM*, vol.41, no.2, February 1998.

[7] KPMG Consulting, *Read Your eFuture*, Autumn 2000.

[8] Orr K., Data Quality and Systems Theory. *Communications of the ACM*, vol.41, no.2, February 1998

[9] Orr K., *Data Warehousing Technology*. The Ken Orr Institute 2000.

[10] Redman T.C., The Impact of Poor Data Quality on the Typical Enterprise. *Communications of the ACM*, vol. 41, no. 2, February 1998.

[11] Redman T.C., *Data Quality for the Information Age.* Artech House, 1996.

[12] Wang R.Y., A Product Perspective on Total Data Quality Management. *Communications of the ACM*, vol. 41, no.2, February 1998.

[13] Wiederhold, G., Ceri, S. Pernici, B., Distributed database design. *IEEE Proceedings*, 1987.