The Current and Future Role of Data Warehousing in Corporate Application Architecture

Robert Winter
Institute of Information Management, University of St. Gallen

Abstract

Data warehouse systems are widely accepted as a new middleware layer between transactional applications and decision support applications, thereby decoupling systems tailored to an efficient handling of business transactions from systems tailored to an efficient support of business decisions. For companies maintaining direct contact with large numbers of customers, however, a growing number of novel, channel-oriented applications (e.g. e-commerce support, call center support) create a new architectural challenge: Traditional transactional applications have to be decoupled from channel-oriented applications to allow for sufficient flexibility of assigning access / distribution channels to products / services. This paper analyzes the possible extension of the data warehousing approach to application integration, i.e. the potentials of reusing data warehousing methodology and management concepts for decoupling traditional transactional applications and channel-oriented applications. As a foundation for that analysis, an application architecture model along the dimensions 'business process', 'business unit', and 'business function' is proposed, and relevant application types are located within that model. The extended role of data warehousing that reflects the emergence of novel application types as well as the necessity of on-line data integration by operational data stores is discussed. Models and findings are based on a close collaboration with several large companies that is institutionalized by the competence centers 'Data Warehousing Strategy' and 'Banking Architectures of the Information Age'.

1. Introduction

Data warehousing has established a new middleware layer in corporate application architecture. Such a middleware layer is necessary because the direct, individual access of decision support applications to data of operational, transaction oriented applications has proved to be technically or economically infeasible: Data quality problems and complex integration requirements usually make it impossible to supply consistent, integrated data real-time to various decision support applications. Even if technically feasible, the development and maintenance of mn interfaces between m decision support applications and n transactional applications cannot be economically useful. As an intermediate systems layer, the data warehouse system is decoupling decision support applications and operational applications, thereby reusing integration mechanisms and derived data for various decision support applications and allowing maintenance to be focussed on few, well-defined interfaces. The role of the data warehouse system as middleware layer is illustrated by Figure 1.

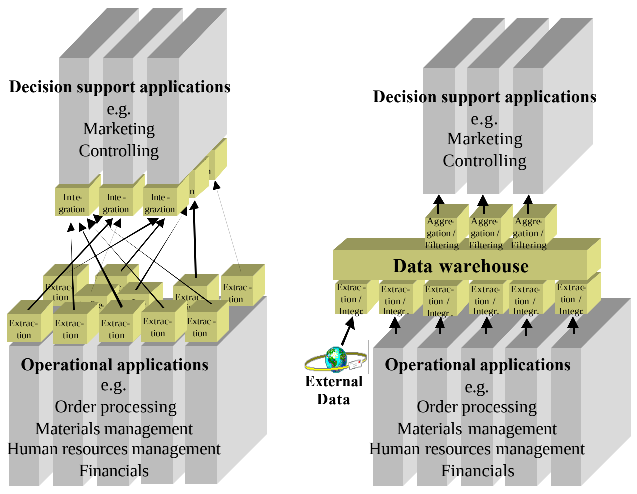

Figure 1. Data warehouse system as a middleware layer

This paper analyzes the possible extension of the data warehousing approach to application integration, i.e. the potentials of reusing data warehousing methodology and management concepts for decoupling traditional transactional applications and channel-oriented applications. As a foundation for that analysis, an application architecture model along the dimensions 'business process', 'business unit', and 'business function' is proposed, and different integration scenarios are located within that model. The extended role of data warehousing that reflects the emergence of novel application types as well as the necessity of on-line data integration by operational data stores is discussed. The paper closes with research questions that result from the proposed evolution of data warehousing in corporate application architecture. Models and findings are based on a close collaboration with 18 large Swiss and German companies, mostly from the insurance and banking sector. This collaboration is institutionalized by the competence centers 'Data Warehousing Strategy' (January 1999 – December 2000) and 'Banking Architectures of the Information Age' (January 2000 – December 2001).

2. Modeling Application Architecture

2.1. IS Architecture vs. Application Architecture

Although both scientific community and practitioners agree on the importance of information systems architecture (e.g. [1][8][14][21]), widely accepted architecture models are hard to find. We focus on the information systems level so that industrywide architectures that are focussing on technology, software, or design patterns (e.g. IBM's Open Blueprint [17], OMG's CORBA, Microsoft's .NET) as well as business architectures (e.g. IMG's Business Model of the Information Age [4]) are not regarded. Even IS architecture models differ widely:

- Scheer defines IS architecture as a model of IS components and their relationships that also comprises methods for planning, development, and coordination [15, p.3].
- Österle defines IS architecture as a high-level model of organization, business functions, business data, software systems, and databases [12, p.26] which can be used as a foundation for organizational and application development [12, p.69].
- Zachman's framework provides standards for planning, building, using, and maintaining information systems, covering various perspectives from planner's view through subcontractor's view and comprising data, functions, network, organization, schedule, and strategy [8, chapter 3][21].

As a common denominator, IS architecture should comprise not only specifications and documentation of IS components and their relationships with regard to all relevant aspects (e.g. data flow, control flow, roles and responsibilities), but also appropriate construction rules [14]. As a consequence of the complexity of real-life enterprise-wide IS architecture, no single model is able to represent all relevant aspects [20][21]. Thus, IS architectures comprise several partial models, each covering a relevant view or dimension (e.g. application architecture, conceptual and logical design, data/object/process models [14]).

Being an important partial model for IS architecture, enterprise application architecture allows to document the most important applications of an enterprise and their relationships as well as rules for creating and maintaining appropriate application structure.

In many companies, application architecture has grown over a long period of time. In the worst case, applications have been introduced unsystematically. If rudimentary construction rules have been followed, application structure mostly reflects organizational structure, i.e. shifted from functional decomposition (e.g. sales, order processing / production, procurement, financials) to object oriented decomposition (e.g. according to products or product groups, to customer segments, or to a business units as a combination of these).

As an example, a retail bank usually has developed separate applications for loans, cash deposits, custody, etc., and a telecommunications company usually has developed separate applications for fixed line telephony, cellular telephony, data communications, internet access, etc. In most cases, every functional cluster or business unit maintains separate customer data and product data. As a consequence, customers receive different bills from different applications, have to report inquiries or address changes to different contact points, and are bothered by marketing campaigns for products that they already bought. From the viewpoint of the company, redundant data create inefficiencies and inconsistencies, cross-selling potentials cannot be exploited, and customer knowledge is distributed over numerous applications.

2.2. Application Architecture Model

Due to its particular importance for large companies, several application architecture approaches have been proposed. While regarding organizational units temporarily, IBM's Business Systems Planning (BSP, see [3]) assigns

- business processes (e.g. financial planning, product development, order processing) to
- data clusters (e.g. product, customer, bill of material, order)

to support the identification of integration areas based on data flows.

IMG's Promet Systems and Technology Planning (STP, see [4, p.41]) assigns

- functional clusters (e.g. sales, financials, procurement) to
- business units (e.g. private banking customers, retail customers, asset management)

to represent integration scenarios and 'as is' application architecture as well as to support the identification of 'should be' application architecture.

Analysis of application architecture practice in several companies participating in the above mentioned competence centers shows that relationships between applications are graphically documented, usually without referring certain dimensions and without comprising explicit construction rules.

A basic assumption of the proposed model is that the 'city planning view' [14][16, p.6] of data corresponds widely to the city planning view of business units (in contrast to the city planning view of business functions). If that is true, BSP's benefits of visualizing data/process integration can be combined with STP's benefits of visualizing function/business unit integration by combining the two two-dimensional models into one single, three-dimensional model.

In the resulting three-dimensional model, each application refers the dimensions 'business process' (from BSP), 'business unit' (from STP, corresponds to BSP's data dimension), and 'business function' (from STP). By locating applications in that three-dimensional model, various integration concepts can be visualized, overlaps and gaps become apparent, and application redesign can be guided. Since the model's primary purpose is to guide application design by visualization, dependencies between dimensions are not obstructive.

The dimension 'business function' lines up aggregate functions that are supported by applications, e.g. create offers, accept orders, create contracts, calculate prices, billing or plan / control resource utilization. The dimension 'business unit' lines up aggregate organizational units that result from customer segmentation, product grouping, or a combination of both. Example scales for this dimension are:

- Retail banking units, private and corporate customer units, investment banking units, and asset management units for an universal bank (structured according to customer segmentation),
- Fixed line telephony units, cellular telephony units, data communications units, and internet access units for a telecommunications company (structured according to product grouping)
- Life insurance units, health / accident insurance units, liability / home insurance units etc. differentiated for domestic private customers, domestic group policies, foreign policies etc. (structured according to a combination of customer segments and product groups)

The dimension 'business process' lines up aggregate processes that are supported by applications, e.g. customer relationship management, order processing, product development, risk management, or corporate planning.

2.3. Typical Application Architecture in Service Industries

In all insurance and banking companies participating in the above mentioned competence centers, an application architecture evolved that comprised from about 10 to over 100 transactional application clusters, each covering most functional aspects of a more or less complete business process for a specific product group or for a specific business unit. The predominant integration strategy is cross-functional, ***product oriented integration*** [5, p.2-3] (or business unit oriented integration, respectively).

From these 'vertical' applications, most companies transferred certain business functions into dedicated cross-product applications with huge efforts. E.g., customer data management was transferred from various product-specific applications into one single, cross-product 'partner management' application to avoid the problems of redundant customer data management and create opportunities for cross-selling and customer bonus programs. Similar effects were created by transferring all product configuration and pricing functions into one single 'product management' application or by transferring reporting functions into a single reporting application. Although the data managed by cross-product applications are processed by all other applications and thereby become 'core' or 'reference' data, they should be treated as operational data [2, pp.141-142]. Cross-product applications can therefore be considered as transactional applications, i.e. sources of transaction data for decision support applications.

By creating dedicated cross-product applications, product oriented integration is complemented by ***core data integration***. A typical application architecture comprising vertical applications and cross-product applications is illustrated by Figure 2.

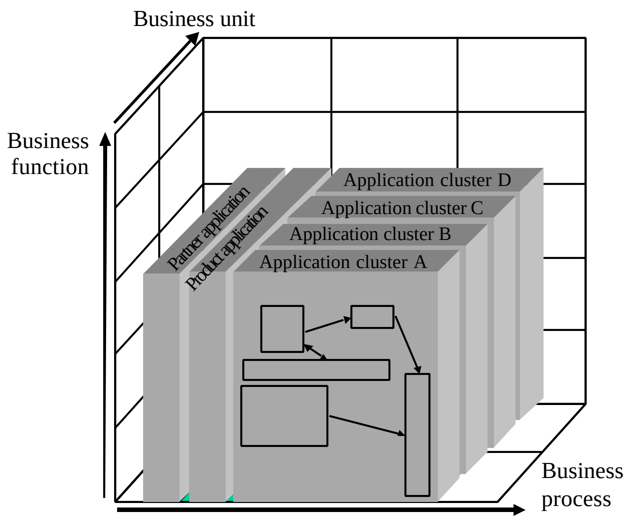

Figure 2. Application architecture found in most service industries

In such an application architecture, data warehousing creates an intermediate layer by which subject-oriented information for decision support applications is derived from transaction data. To avoid the development and maintenance efforts for numerous individual interfaces between transactional applications and decision support applications, source data are integrated into a (logically) centralized, consistent database. This database is then used by all decision support applications as a single source of consistent data. For simplification, the data warehousing layer is not further structured in Figure 3. In reality, however, usually several architectural components are created for data extraction, data staging, data transformation, data integration, data correction, data quality control, data load, data security, data selection and filtering, data aggregation, etc. Moreover, the data warehouse does not have to be implemented as a single, centralized system, but can also be partially or entirely implemented in a decentralized or even virtual way.

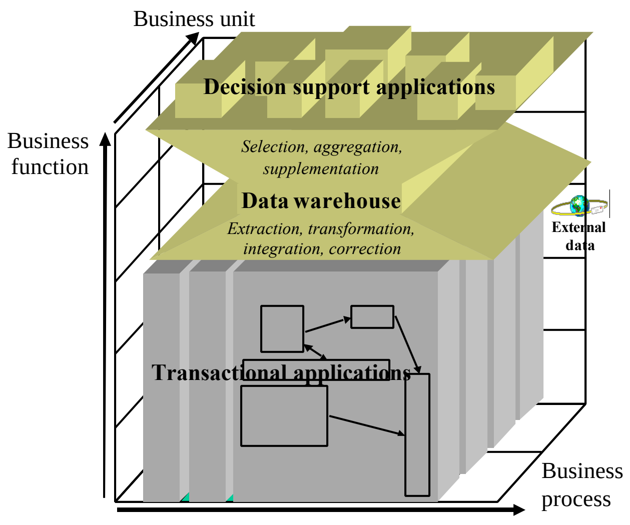
Business unit
Business function
Decision support applications
Selection, aggregation, supplementation
Data warehouse
Extraction, transformation, integration, correction
External data
Transactional applications
Business process

Figure 3. Role of data warehousing in traditional application architecture

3. Application Integration

3.1. Multi-Channel Management and IS Architecture

Some business functions (e.g. device-specific presentation, device-specific communication) are closely linked to an access channel or a distribution channel, respectively. Like cross-product business functions mentioned above, channel specific business functions have often been replicated in vertical applications as a consequence of rapid development projects (e.g. quick introduction of new products, mergers and acquisitions) or a missing architectural vision. However, in contrast to the transfer of cross-product business functions which started in the early 1990ies, channel-specific business functions that have not been widely transferred yet to dedicated applications. As a consequence, product (or business unit) specific business logic is tied to access or distribution channel logic, and no flexible assignment of products / business units to access / distribution channels ('multi-channel management') can be supported.

The support of multi-channel management is essential for information systems of service oriented companies: Customers demand multiple access channels (e.g. face-to-face, letter-based, phone-based, and electronic) to products / services to be supported. Management demands to decide which channels to use for which products / services without being restricted too much by IS/IT restrictions.

The clustering of channel-specific business functions into dedicated applications is determined by the respective access media: Customers demand access to information and products/services by phone, by Internet, by cellular phone and WAP, by traditional written communication, by using self-service terminals (e.g. ATMs), or via sales representatives. As a consequence, vertical applications and cross-product applications have to be complemented by channel-specific applications like Call Center support, WWW portal, WAP portal, Letter Center/Document Management support, ATM support, and traditional, in-house transaction applications. These applications may differ not only by supporting a different access and/or distribution channel, but also by different security mechanisms. Hence, product oriented integration and core data integration are complemented by access/distribution ***channel oriented integration***.

In the proposed application architecture model, channel integration is represented by a new set of application clusters which cover the distribution / customer access portion of business processes over all products (or business units), but for only one specific distribution / access channel (i.e. for selected business functions). Due to their optical appearance, channel-specific applications can be designated as 'horizontal' applications also in this model (see Figure 4). Channel-specific applications are usually created by transferring and integrating selected business functions from vertical applications. This process can be supplemented or partially substituted by acquiring standardized applications.

Since the number of core data applications is relatively small, it is technically feasible and economically useful to create individual interfaces between core data applications and vertical applications. A growing number of distribution / access channels and the flexibility requirements of multi-channel management, however, require an explicit middleware layer to be created for decoupling horizontal and vertical applications [5, p.5]. This middleware layer may be primarily data-oriented, primarily component-oriented, or primarily message-oriented [16, p.7]. All of these forms have its own strengths and limitations, but data-centric approaches are the traditional and best-proven solution [16, p.7].

3.2. Operational Data Stores

The transformation of transaction data into integrated, consistent input for decision support applications by data warehousing consumes a certain amount of time and creates non-volatile, aggregate information. Many operational decisions (e.g. promotion effectiveness, customer retention, key account information [8, p.245]), however, need actual yet integrated and subject-oriented data in or near real-time. Likewise, following a 'zero-latency' or 'straight-through-processing' strategy [16, p.13], vertical applications and channel-specific applications need to be integrated in real-time. Since real-time access is not possible for integrated, consistent data warehouse data and since these data have usually been aggregated, the concept of operational data stores has been introduced for operational decision support [7, pp. 1-20] and for data-oriented application integration [5, p.4][6, pp.1-2].

As an example, Delta Airlines has implemented a zero-latency strategy by using operational data stores to integrate

- reservations system, operations applications, Federal Aviation Administration applications, onboard telemetry, and flight information display systems on the one hand with
- cross-product (or cross-business unit) exception alerts, ad hoc inquiries, and status reports on the other hand [16, p.14].

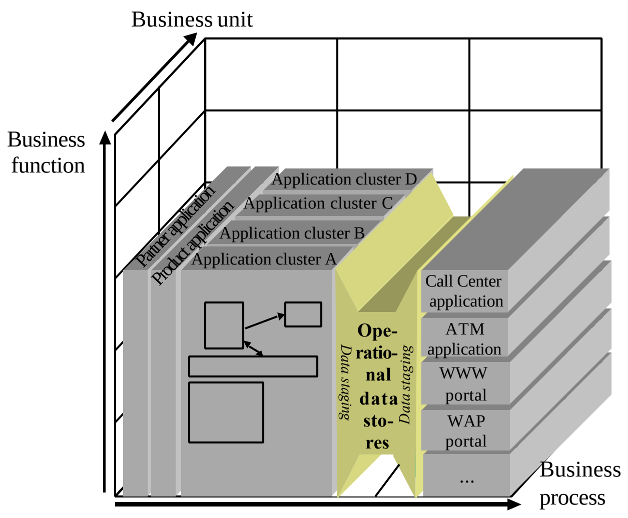

Figure 4. Role of operational data stores in application integration

Figure 4 illustrates the role of operational data stores for decoupling vertical and channel-oriented applications. Analyses of several companies participating in the above mentioned competence centers show that in mature application architectures, product oriented integration, core data integration, and channel oriented integration occur simultaneously. After being detailed and customized for a specific application portfolio, reference architectures based on Figure 4 could guide the transfer of business functions from vertical applications to channel-oriented applications and / or the integration of purchased software packages.

Data managed by operational data stores have characteristics that differ from data managed by operational applications as well as from data managed by the data warehouse (see Table 1). It becomes evident that operational data stores can be positioned between transactional applications and the data warehouse [7, pp. 12-17][8, pp. 234-235]. As a consequence, operational data stores should be used not only for application integration and as a data source for operational decisions, but also as an important (but certainly not exclusive [6, p.4]) source system for the data warehouse to avoid duplication of integration functionality [7, pp.21-25][8, pp. 250-251].

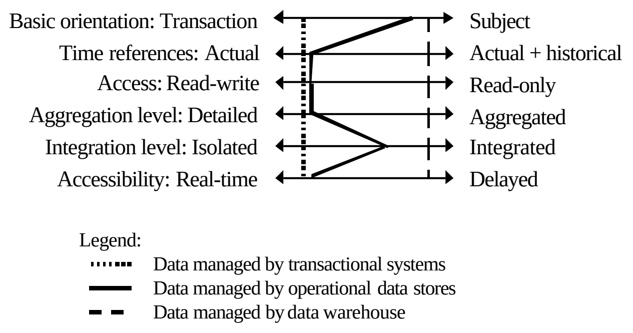
Basic orientation: Transaction
Subject
Time references: Actual
Actual + historical
Access: Read-write
Read-only
Aggregation level: Detailed
Aggregated
Integration level: Isolated
Integrated
Accessibility: Real-time
Delayed
Legend:
Data managed by transactional systems
Data managed by operational data stores
Data managed by data warehouse

Table 1: Comparison of operational applications, operational data stores, and the data warehouse

4. The Changing Role of Data Warehousing

4.1. Operational Data Stores vs. Data Warehouse

In principle, the introduction of channel-oriented applications has no influence on the positioning of data warehouse systems in corporate application architecture: Like vertical applications and core data applications, channel-oriented applications create and process transactional data, which then can be transformed into information by data warehousing.

Whether it is necessary to introduce operational data stores in addition to a data warehouse should be decided in each case carefully: If the focus is on providing actual data for reporting, often cross-product applications or more frequent data warehouse updates are sufficient. If, on the other hand, applications have to exchange subject-oriented data in or near real-time, the introduction of an additional architectural layer is inevitable [10, p.20].

Since both the data warehouse and operational data stores can be regarded as data-oriented integration middleware, it was proposed that operational data stores should be implemented as a part of the data warehouse or that the data warehouse should be directly utilized for transactional services like customer relationship management or e-commerce ('closed loop model', e.g. [13]). Since operational data stores are fundamentally different from the data warehouse due to real-time processing needs and read-write access to data, however, these two middleware layers should be separated in application architecture [7, pp. 25-27]. An application architecture model that is extended by operational data stores is illustrated by Figure 5.

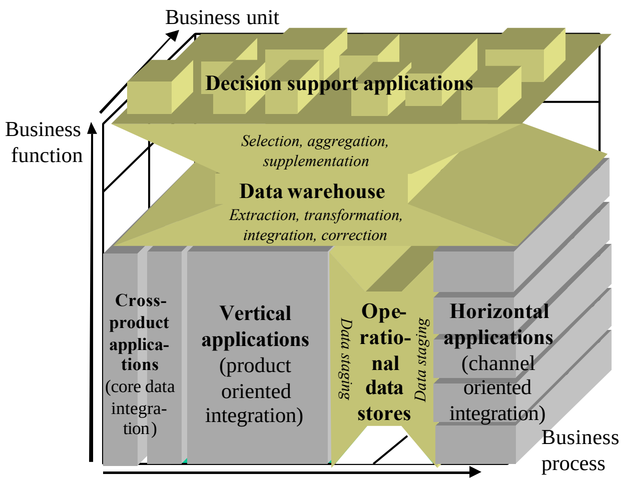
Business unit
Business function
Decision support applications
Selection, aggregation, supplementation
Data warehouse
Extraction, transformation, integration, correction
Cross-product applications (core data integration)
Vertical applications (product oriented integration)
Data staging
Operational data stores
Data staging
Horizontal applications (channel oriented integration)
Business process

Figure 5. Data warehouse and operational data stores – Different components of enterprise application architecture

By using operational data stores, an efficient 'local' closed-loop approach can be supported between vertical and horizontal transactional applications. By data warehousing, efficient information supply between transactional applications and decision support applications can be achieved. Since information 'backflows' take their way indirectly (i.e. by actions of decision makers) into the transactional systems and not directly through the data warehouse, however, only an open-loop approach can be implemented by data warehousing. As a consequence, the business case for operational data stores is much more straightforward as the business case for data warehousing which comprises many intangible, indirect benefits.

4.2. Reuse Potentials of Data Warehousing Concepts for Application Integration

Although the employment of operational data stores and data warehousing differ in architectural positioning, business benefits, and implementation, many synergies can be utilized in development and operations. The potentials and limitations of using operational data stores for feeding integrated data into the data warehouse have already been discussed in section 3.2.

In addition to that direct relationship, we recently started to analyze which additional, methodical and managerial concepts of data warehousing can be reused for application integration based on operational data stores:

- ***Project justification:*** Time savings for data suppliers and users, availability of more and better information as a foundation for better decisions, and improvement of business processes have been mentioned as most important benefits of data warehousing (e.g. [18]). Since these benefits are mostly hard to quantify, data warehousing projects are (like the introduction of integrated business application software packages) strategic projects whose payoffs cannot be exactly calculated [19]. Application integration projects usually create more tangible benefits due to their direct, closed-loop effects. In addition, project justification should benefit from data warehousing-related findings like recommended division of tasks between IT and business units, reference business cases for certain project classes, differentiation of platform-related issues and application-related issues, recommended pricing policies for quick-win projects, etc. (detailed findings from our Data Warehousing competence center can be found in [9]).
- ***Permanent organization:*** While transactional systems are fueled by masses of daily business transactions which are quite stable and bless many applications with a life much longer than initially planned, decision support applications often are abandoned when business transformations or organizational changes occur or even have never been really used by the decision makers. This is due to a missing or fragmentary organizational concept: While functional organization and / or process ownership provide a foundation for the organization of transaction systems management, no such framework traditionally exists for data warehousing [18][19]. With growing experience from successful data warehousing solutions in large companies, however, data ownership has emerged as a conceptual foundation from which recommended roles and responsibilities (e.g. sourcing and channeling, infrastructure, standards) as well as basic processes for permanent data warehousing were derived (a detailed description can be found in [11]). Since many similarities between data warehousing and application integration exist with regard to organizational issues, most of these findings appear to be reusable.
- ***Development methodology:*** Regardless of the underlying paradigm, traditional applications are developed according to certain specifications. In contrast to that, a data warehouse is usually built upon existing transactional systems with requirements being clear at most partially. This is due to the fact that transactional

systems are developed to support specific, comparably stable business transactions, while the data warehouse has to support ever-changing, individual decisions in a dynamic business setting [19]. In our experience, missing specifications and high project risks can be avoided if small, quick-win projects (usually data marts) are successively being implemented based on a corporate data warehousing vision, an existing core data management, a rudimentary organizational and technical infrastructure, and certain standards. In any phase, infrastructure-related tasks and project-related tasks should be differentiated as completely as possible. Data warehousing development starts with an initial 'infrastructure & vision' phase and continues with permanent cycles comprising an 'analysis & specification' phase, an 'implementation' phase, and a 'production & adaptation' phase [19]. Being specific to data warehousing, the focus on iterative development as well as novel sub-phases like 'motivation of project sponsor / project marketing' or 'infrastructure / project differentiation' appear to have high reuse potential for application integration, while features like 'business question analysis' are too closely linked to the decision support scenario.

- ***Meta data management***: The discussions of meta data issues for operational data stores in [7, pp. 45-56] and [8, pp. 253-268] do not significantly differ from respective discussions in the data warehousing literature (e.g. [2][10]). Nearly all meta data relevant for data warehousing are also relevant for application integration based on operational data stores vice versa. Data warehousing has created significant meta data awareness and has triggered many corporate meta data projects whose results should be widely reusable for any layer of enterprise application architecture from application integration and decision support to transactional applications.

5. Conclusions and Directions for Future Research

In this paper, we introduced an application architecture model by which various types of integration can be visualized and which – after being detailed and customized for a specific application portfolio – can serve as a reference model for the city planning view of enterprise application architecture. Being an important part of architecture models, however, construction rules are only partially implied (e.g. by recommended types of decoupling) so that further research and verification of construction rules and optimization criteria is necessary.

Analysis of application types and decoupling requirements showed that, at least for large service-oriented companies, usually two interdependent middleware layers are necessary: By operational data stores, product-oriented and channel-oriented transactional applications are connected efficiently in or near real-time. By data warehousing, decision support applications are linked to transactional data sources. Since transactional data are being integrated according to information subjects in operational data stores, these data stores are an important yet not exclusive data source for the data warehouse.

The co-existence of operational data stores and the data warehouse leads to the question whether operational data stores – like the data warehouse – have to be developed more or less from scratch or whether, if not infrastructural components, at least methodological and managerial concepts can be reused. Preliminary analyses promise methodological reuse potentials for project justification and systems development. Moreover, reuse potentials exist for permanent organization (structures as well as processes) and meta data management. While being different in nature, data warehousing and operational data stores share certain roles so that significant synergies in development and operations should be exploited.

Models and findings are currently based on case-based analyses in tight collaboration with 18 large Swiss and German companies, mostly from the insurance and banking sector. In many of these companies, the proposed application architecture model was applied, and insights from large data warehousing and, more recently, application integration projects could be acquired. Future research will be aimed in broadening the empirical base by including companies from other sectors and by including selected quantitative aspects.

In contrast to most of the vertical applications, many decision support applications, and nearly all data warehouses, horizontal applications are often implemented by deploying standardized application software packages. Hence, it is possible from a technical point of view and desirable from a business point of view to propose reference models for important parts of the transactional layer of information logistics. The existing collaboration with partner companies seems to be a good foundation for addressing this issue in future research.

References

[1] Bernus, B., Mertins, K., Schmidt, G. (eds.): Handbook on Architectures of Information Systems, Springer: Berlin etc., 1998.

[2] Devlin, B.: Data Warehouse – from Architecture to Implementation. Addison-Wesley: Reading etc. 1997.

[3] IBM Corp. (ed.): Business Systems Planning – Information Systems Planning Guide, Application Manual GE20-0627.

[4] IMG AG (Hrsg.): Promet® STP, Methodenhandbuch für die System- und Technologieplanung, Version 1.1, St. Gallen 2000.

[5] Imhoff, C.: The Corporate Information Factory, DM Review, December 1999, http://www.dmreview.com/editorial/dmreview, 29-03-2000.

[6] Imhoff, C.: The Operational Data Store: Hammering Away, DM Review, July 1998, http://www.dmreview.com/editorial/dmreview, 29-03-2000.
[7] Inmon, W.H.: Building the Operational Data Store, 2nd edition, Wiley: New York etc. 1999.
[8] Inmon, W.H., Zachman, J.A., Geiger, J.G.: Data Stores Data Warehousing and the Zachman Framework, McGraw-Hill: New York etc. 1997.
[9] Jung, R., Winter, R.: Justification of Data Warehousing Projects, Research Report, Competence Center Data Warehousing Strategy, University of St. Gallen, St. Gallen 2000.
[10] Kimball, R., Reeves, L., Ross, M., Thornthwaite, W.: The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouses. John Wiley & Sons: New York etc. 1998.
[11] Meyer, M., Winter, R.: Organisation des unternehmensweiten Data Warehousing, to appear in: Jung, R., Winter, R. (Eds.): Data Warehousing 2000, Physica: Heidelberg, 2000.
[12] Österle, H., Brenner, W., Hilbers, K.: Unternehmensführung und Informationssystem: der Ansatz des St.Galler Informationssystem-Managements, 2. Aufl.; Teubner: Stuttgart, 1992.
[13] OVUM Evaluates: CRM Strategies: Technology Choices for the Customer-focussed Business; OVUM Ltd., London 1999.
[14] Pezzini, M.: Architecture and IS „City Planning“, in: Gartner Group (Ed.), Application Integration – Making E-Business Work, London, 6-7 September 2000
[15] Scheer, A.-W.: Architecture of Integrated Information Systems, Springer: New York etc. 1992.
[16] Schulte, R.: Application Integration Scenario: How the War is Being Won, in: Gartner Group (Ed.): Application Integration – Making E-Business Work, London, 6-7 September 2000.
[17] Tibbetts, K.: Enterprise Architectures: A Comparison of Vendor Initiatives, http://www.software.ibm.com/openblue/kx95/cover.htm, 02-09-1998.
[18] Watson, H.J., B.J. Haley: Managerial Considerations, Communications of the ACM, 41 (1998), 9, 32-37.
[19] Winter, R.: Data Warehousing beyond Tools and Data: Justification, Organization, and Structured Development of Data Warehousing Applications, in: Abramowicz, W., Orlowska M.E. (Eds.): Business Information Systems – Proc. BIS'99, Practical Sessions, pp. 197-208.
[20] Youngs, R., Redmond-Pyle, D., Spaas, P., Kahan, E.: A standard for architecture description, IBM Systems Journal, 38, 1, 1999.
[21] Zachman, J.A.: A framework for information systems architecture, IBM Systems Journal, 26, 3, 276-292, 1987; reprinted in IBM Systems Journal, 38, 2/3, 454-470, 1999.