WHERESCAPE

Understanding The
Data Warehouse Lifecycle Model

WhereScape Software Limited

Revision 2
December 2003

ABSTRACT

Despite warnings made by W.H. Inmon and others at the outset of the data warehousing movement in the early 1990s, data warehousing practice for the past decade at least has been prefaced on the assumption that, once in production, data warehouses and data marts were essentially static, from a design perspective, and that data warehouse change management practices were fundamentally no different than those of other kinds of production systems.

The pace of business change, and its discontinuity, combined with the ongoing search for competitive advantage through better decision-making in a climate characterized by commodity transactional systems and (increasingly) commodity decision support infrastructure, has underscored, for many firms, the extent to which an organization's understanding of, and control over, the entirety of the data warehousing lifecycle model can mean the difference between competitive differentiation on the one hand, and millions of dollars in cost sunk in brittle dead-end data warehousing infrastructure on the other.

The Data Warehouse Lifecycle Model: Nothing New

In *Building The Data Warehouse*, published in 1991, W.H. Inmon made the observation that:

> The classical system development life cycle (SDLC) does not work in the world of the DSS analyst. The SDLC assumes that requirements are known at the start of the design (or at least can be discovered). However, in the world of the DSS analyst, requirements are usually the last thing to be discovered in the DSS development life cycle (p. 23).

At that time, Inmon advocated a data-driven approach to designing data warehouses, pointing out that (a) DSS analysts frequently understood their requirements, and the data available to them, only after they had the opportunity to perform various kinds of analysis on that data, and (b) the traditional waterfall-oriented models of software development (particularly those enforced by high-end computer-aided software engineering, or CASE, tools) were unlikely to produce workable data warehousing environments.

One of the earliest – and to this day the most effective – responses to the data-driven nature of decision support systems was the dimensional schema design methodology pioneering by Ralph Kimball and others. Dimensional modeling sought to interact with DSS analysts at the business vocabulary and business process level, to design inherently-legible star schema based on the key nominative elements of those business vocabularies and processes. The population of those schema was then largely a technical matter of matching available data elements in transactional source systems to the designed schema, creating or synthesizing data elements when they were not available natively in the systems of record. The fundamental notion behind dimensional modeling was, we believe, that while it might not be possible to gather data requirements from a community of DSS analysts, it was in fact possible to gather analytical requirements from a community of DSS analysts, and subsequently to map available and/or synthesizable data in the organization to those analytical requirements, as embodied in a dimensional modeler's star schema designs.

By the end of the 1990s, however, dimensional modeling practitioners found, repeatedly, that dimensional modeling exercises depended, ultimately, on the ability of designers to prototype their dimensional designs quickly, and expose DSS analysts to those designs, populated with actual data, **before the designs were put into production**. This rapid-prototype-and-iterate cycle was necessary, dimensional designers discovered, because – in support of Inmon's original point – a DSS analyst's understanding of her decision-making needs and capabilities was often crystallized only by seeing what she had asked for during an initial requirements-gathering process.

The pattern of behavior that drove the dimensional modeling community to a recognition of the need for rapid-prototype-and-iterate cycles was, by the end of the 1990s, quite widely reported, and cross-cultural.

- Asked in open-ended fashion to describe their information needs, DSS analysts frequently responded with one of two generic positions: *What data is available?* and *I need everything we have,* which good designers recognized as being fundamentally the same answer.

- Asked to review and approve a populated dimensional model based on their stated analytical needs and business models, DSS analysts frequently responded with variants of *Yes, this is what I asked for, and now that I see it, I'd like to make some changes*, followed by requests for often fundamental design modifications or entirely new kinds of schema.

At roughly the same time as practitioners were discovering the need for rapid prototype-and-iterate cycles (and the need for tools to support that process), teams operating and managing production data warehouses and marts were discovering a number of additional problems with then-conventional notions of how to build warehouses and marts:

- Prototyped data warehouses and marts often could not be moved into production without either significant modification to the technology infrastructure used to build the prototype or wholesale rehosting, because the tools and technologies used to manage production data warehouses and marts – including but not limited to extraction, transformation and load (ETL), scheduling and monitoring tools – were different that those used to build the prototypes.[1]

- Data warehouses and marts were often incredibly expensive to manage in production. It was not uncommon for a significantly-sized production data warehouse to require 5-7 full-time-equivalents on an on-going basis to keep that warehouse stable, in production and available. This high second-order operating cost was most often attributable to the brittleness of the technology infrastructure of the production warehouse, the high rates of ETL failure, and the inability of data warehouse operators to recover from ETL failures gracefully and quickly.

- Once in production and stable, the technology, processes and procedures wrapped around the typical data warehouse or mart were so complex, and the technology delivering data into the target warehouse and mart schema so carefully balanced (often down to the point release of each revision level of each software product contributing to the infrastructure), in order to protect the production environment's overall stability, that change of any sort, and particularly changes to the core warehouse or mart schema needed to reflect changes in the business environment (and therefore in the DSS analysts' needs) were impossible to implement.
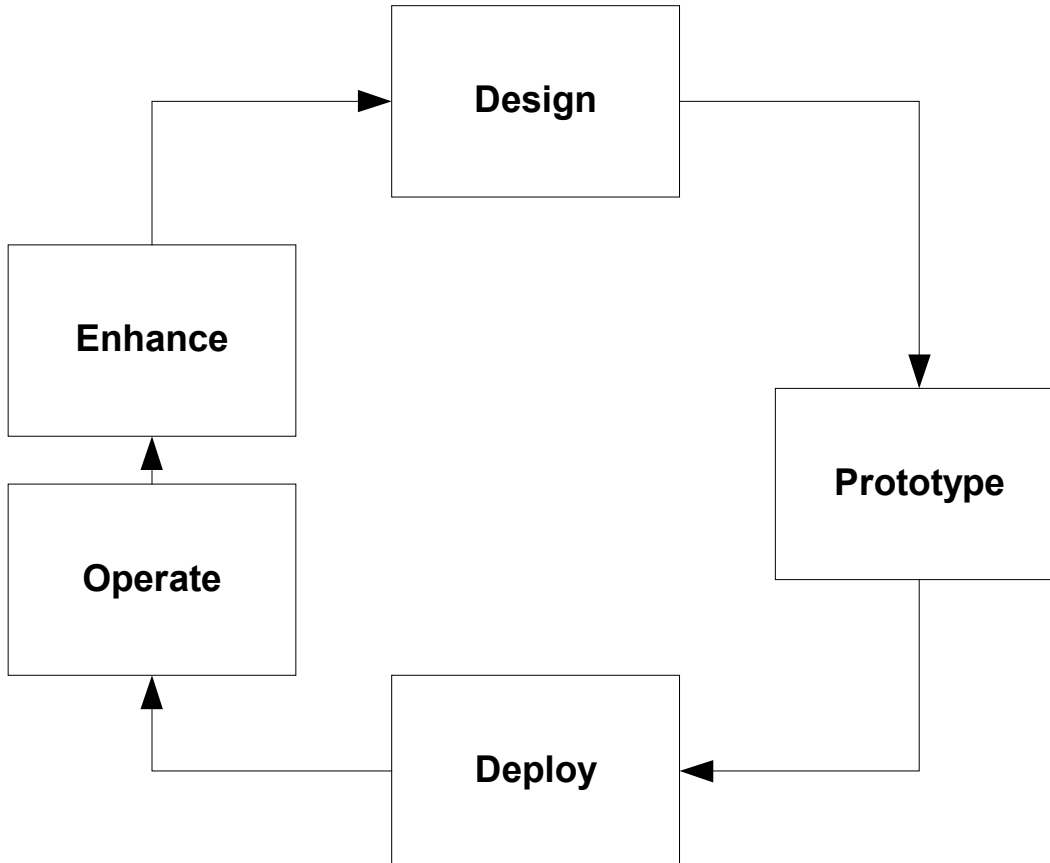
---

[1] In WhereScape's estimation, a substantial number of the specific project failures reported in the late 1990s – when data warehouse/data mart project failure rates were held by analysts to be as high as 70% -- were attributable to the inability to take prototyped warehouses or marts into production in a timely fashion.

The notion that data warehouses and marts had a lifecycle, and that that lifecycle involved a return to design at the schema level, was thus well-established as a notion among practitioners by the end of 1990s.

Yet today, a Google search on the phrase "data warehouse lifecycle" reveals relatively few content-rich sites, and data warehouse lifecycle models are still often presented using waterfalls or thread models that end with deployment – which is in fact where real-world data warehousing – in terms of ongoing business benefit -- begins.

The Data Warehouse Lifecycle Model Today

Although specific vocabularies vary from organization to organization, the data warehousing industry is in agreement that the data warehouse lifecycle model is fundamentally as described in the diagram below

```
              ┌──────────┐
    ┌────────►│  Design  │──────────┐
    │         └──────────┘          │
┌────────┐                          ▼
│        │                     ┌──────────┐
│ Enhance│                     │          │
│        │                     │Prototype │
└────────┘                     │          │
    ▲                          └──────────┘
┌────────┐                          │
│        │                          ▼
│ Operate│          ┌──────────┐
│        │◄─────────│  Deploy  │◄────────
└────────┘          └──────────┘
```

The model, which is a cycle rather than a serialized timeline, and which based on WhereScape's experience repeats every 12 to 18 months, consists of five major phases:

1. Design – the development, from both available data inventories and DSS analyst requirements and analytical needs, of robust star-schema-based dimensional data models. Practically speaking, we believe that the best data warehousing practitioners today are working simultaneously from (a) the available data inventory in the organization and (b) the often incomplete expression of needs on the part of multiple heterogeneous analytical communities, making data warehouse and data mart design a complex and intricate process more akin to architecture and diplomacy than to traditional IT systems design. Key activities in this phase typically include end-user interview cycles, source system cataloguing, definition of key performance indicators and other critical business metrics, mapping of decision-making processes underlying information needs, and logical

and physical schema design tasks, which feed the prototyping phase of the lifecycle model quite directly.

2. Prototype: the deployment, for a select group of opinion-makers and leading practitioners in the end-user analytical communities, of a populated, working model of a data warehouse or data mart design, suitable for actual use. The purpose of prototyping shifts, as the design team moves back and forth between design and prototype. In early prototypes, the primary objective is to constrain and in some cases reframe end-user requirements by showing opinion-leaders and heavyweight analysts in the end-user communities precisely what they had asked for in the previous iteration. As the gap between stated needs and actual needs closes over the course of 2 or more design-prototype iterations, the purpose of the prototype shifts toward diplomacy – gaining commitment to the project at hand from opinion leaders in the end-user communities to the design, and soliciting their assistance in gaining similar commitment.

3. Deploy: the formalization of a user-approved prototype for actual production use, including the development of documentation, training, O&M processes and the host of activities traditionally associated with enterprise IT system deployment. Deployment typically involves at least two separate deployments – the deployment of a prototype into a production-test environment, and the deployment of a stress-tested, performance-tested production configuration into an actual production environment. It is at this phase that the single most often-neglected component of a successful production warehouse – documentation – stalls both the transition to operations and management personnel (who cannot manage what they cannot understand) and to end-user organizations (who after all have to be taught at some level about the data and metadata the warehouse or mart contains, prior to roll-out).

4. Operation: the day-to-day maintenance of the data warehouse or mart, the data delivery services and client tools that provide DSS analysts with their access to warehouse and mart data, and the management of ongoing extraction, transformation and loading processes (from traditional drop-and-reload bulk updates to cutting edge near-real-time trickle-charging processes) that keep the warehouse or mart current with respect to the authoritative transactional source systems.

5. Enhancement: the modification of (a) physical technological componentry, (b) operations and management processes (including ETL regimes and scheduling) and (c) logical schema designs in response to changing business requirements. In cases where external business conditions change discontinuously, or organizations themselves undergo discontinuous changes (as in the case of asset sales, mergers and acquisitions), enhancement moves seamlessly back into fundamental design.

Seen properly, the model is actually a cycle of cycles, as identified in the diagram below:
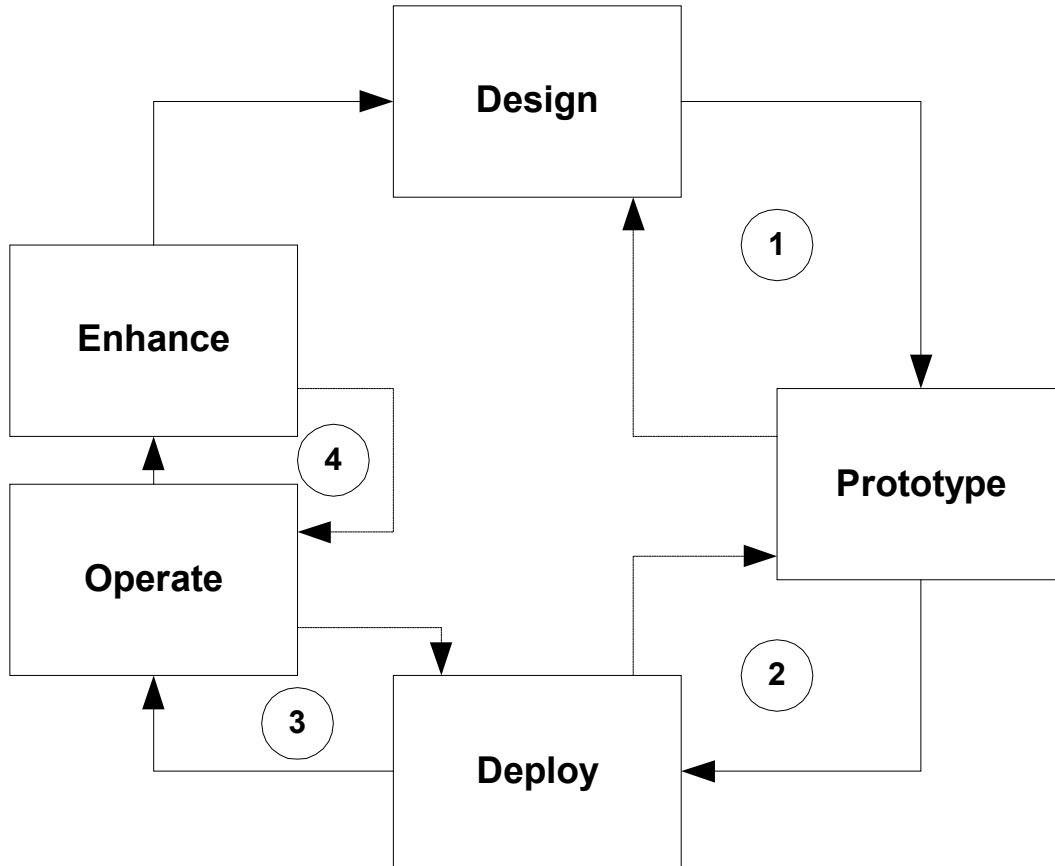
1. Rapid design-prototype-iterate cycles

2. One or several deployment cycles

3. Changes to deployment typologies to facilitate operations

4. Changes to operational regimes to provide incremental enhancements

Key issues in the first loop – design-prototype – include:

- The ability of the design toolset to create a populated prototype overnight, so that multiple iterations of a particular schema can be accomplished within a calendar week.

- The ability of the design tool to produce visual output intelligible to end-users, so that schema redesign can happen in real-time in critique sessions between designers and representatives of the end-user organization.

- The ability of the design toolset to document changes made to the logical schema in a way that permits post-project design audits to recover the rationale for schema changes.

In the second loop – prototype-deploy – the key operational issue is straightforward: can the design team deploy using precisely the same technology that it used for its prototype? In too many cases – particularly when prototyping is done using hand-assembled office productivity tools and rinky-dink PC databases – the answer is no, and this loop elongates (sometimes into periods measured in months or quarters) while an approved prototype is reimplemented, from scratch, using production-grade technologies. In almost every case of which WhereScape has knowledge, design elements present in the prototype are lost in such a transition, either because the prototype environment made use of technology not available in the production environment, or because the reassembly of the prototype design is imperfect: desirable design elements are removed (inadvertently or otherwise) and new design elements (desirable or not) are introduced.

The third loop – operate-deploy – involves a close, tense, ongoing discussion among the operations team about the ways in which the productized design and its implementation affects operations. The classic examples of problems in this area include granularity (often of the time dimension) and loading strategies. A warehouse designed for deployment in a relational environment, with an hourly time granularity in its time dimension, five years of history and nightly incremental refresh, produces a time dimension that, logically, has 43,800 or so rows. A choice to deploy that schema using a proprietary MDDBS technology may appear, at the transition to deployment, to be a workable choice, but fails operationally when it becomes apparent that the load time for MDDBMS cubes with a time dimension of that granularity requires more than 24 hours to load. Similarly, so-called trickle-feeding or trickle-charging of large-scale data warehouses often appear practicable at the transition to deployment, and fail weeks or months later when

- the operational characteristics of the transactional source systems change, and the drain of transaction-level replication from the source systems outstrip those source systems' capacity

- the size of the target schema, and the complexity of its indexing schemes, make incremental updates generally more impractical that incremental bulk refreshes or full-blown drop-and-reload options.

The fourth loop – the enhance-operate loop – is often not discussed in the literature, but generally practitioners distinguish enhancements from redesign by the depth and breadth of changes to the target schema in the warehouse or mart. Enhancements typically involve the addition of dimensional elements or facts to schema that remain unchanged at the table level, while redesign requires the addition of new tables or new schema. Issues in this area focus around three related factors:

- The ease with which design techniques and tools can be used to create a prototype of the required enhancements for approval prior to the implementation of those enhancements.

- The extent to which the technology underpinning the production warehouse or mart is dependent on the schema remaining entirely stable in order to protect

other code – extraction, transformation, loading, scheduling or monitoring code – from change and the overall configuration from potential destabilization.

- The extent to which the end-user environment embeds expectations about the shape of the warehouse or mart schema in its environment – in for example encapsulated SQL fragments manipulated as objects by client query tools or portal applets – that may be invalidated by schema changes.

In many cases, the design tools used to deploy the initial warehouse or mart are unavailable for enhancement prototyping, testing and validation, for the reason mentioned above: they are hand-assembled PC-class tools that are discarded when the prototype is transitioned into production.

Similarly – and this is particularly the case in environments where ETL is hand-crafted and where scheduling and monitoring is performed manually – the code base required to keep the warehouse or mart up and running is often invalidated by even simple changes to the target schema, because the code 'knows' at the lowest possible level about a particular schema, and a change to that schema would invalidate hundreds or thousands of lines of code elsewhere – code that can only be changed by hand, one line at a time.

The final problem – the dependencies of client environments on particular versions of warehouse or mart schema – is a pressing one for which there is no good general solution today, largely owing to the lack of a commonly-used metadata standard. Good operators and practitioners often hide enhancements from the more brittle commercial off-the-shelf tool environments using database views that present an unchanging schema to those tools, but many organizations are forced to retest and reconfigure client environments each time schema in the warehouse or mart are changed.

The last, implicit, loop in the warehousing lifecycle model is the transition from (enhanced) operation back into design, and this is the transition that most conventional models of the data warehousing lifecycle model neglect entirely, as though a warehouse or mart, once taken into production, never undergoes major renovation at either the schema or technology infrastructure level thereafter. Those assumptions may have been true a decade ago, when the commercial world experienced so-called "punctuated equilibrium": relative short and infrequent periods of discontinuous change, followed by much longer periods of business model and competitive stability. However, since at least the end of the 1990s, the commercial environment for most firms has been, we think, much closer to 'perpetual whitewater' than to punctuated equilibrium – discontinuous change, with a cycle measured in months, is the norm rather than the exception. This implies that any warehouse that cannot be redesigned from the schema out, incorporating new infrastructure and new designs, every 10-14 months is useful for a single business change cycle, and its total acquisition costs must be amortized over a period of time measured in months in order to get a true picture of its commercial value. This also implies that organizations building warehouses and marts according to the waterfall lifecycle model are in effect committing themselves and their organizations to perpetual deployment-redesign cycles, since, in most organizations today, it takes more than 6 months to get a typical mart or warehouse into production.

What seems clear to us is this: the closed loop warehousing lifecycle model has to become the norm for commercial organizations with short business change cycles if they expect to deliver business value from their DSS projects, and to recoup their costs for such projects in a reasonable amount of time.

Conclusions

Where conventional systems design, deployment and operation models are essentially waterfall models that consign change management to the operational or production phase of a system's lifecycle, the data warehouse lifecycle model is a loop-of-loops, moving ceaseless from design and prototype through production and operation and back, into design, as DSS analysts refine their short-term information needs and analytical models, and as the business climate within which those needs and models are refined changes, discontinuously and with increasing rapidity.

In such an environment, the existing data warehousing marketplace – which is characterized on the one hand by one-size-fits-all software packages and on the other by best-of-breed technology components – must be integrated and disciplined by **integrated data warehousing lifecycle management tools** that frame and govern the entire process of warehouse and mart creation, production, operation and (re)deployment, in order to avoid:

- The creation of dead-end DSS environments that replicate the state of an organization's decision-making capability as it was two or three business cycles ago, and which cannot be modified because each technology component is inextricably bound up in a particular dimensional model that no longer meets business decision-makers needs for breadth, depth or focus.

- The cost of replacing such dead-end DSS environments with new technology ensembles designed using the same methods and processes that created the problem in the first place.

- The temptation to abandon business decision-making as a source of competitive advantage, and install work-alike packaged analytical software products, in order to avoid the risk of brittle custom data warehouses and marts, which leads organizations into a dead-end decision-making environment in which they have less, not more, germane data for decision-making, and an outside vendor, rather than an in-house DSS team, to manage in order to drive necessary changes and enhancements into their environment.

The DSS market analysts – Gartner, META, IDC and ComputerWire among them – are in agreement that data warehouse lifecycle management is the next phase in the evolution of decision support systems infrastructure, and that forward-looking organizations will include DWLM toolsets in their data warehousing and data marting projects in future.

About WhereScape

Founded by dimensional data warehousing veterans, WhereScape is the market's leading independent source of enterprise-class tools for the design, construction and operation of dimensional data warehouses, data marts, business intelligence environments and decision support systems.

Our flagship product, WhereScape RED, allows you to design, prototype and deploy a production-quality, full-scale dimensional data warehouse in a matter of days or weeks, to spend less than an hour a day managing that warehouse once it is in production, and to enhance that warehouse as often as you like – making changes on a daily basis if necessary, and redeploying those changes overnight.

WhereScape designs its products to do precisely what data warehousing professionals need from their toolset – design, prototype, deploy, operate and enhance dimensional data warehouses – without the exorbitant price tags and complexity associated with database management system CASE tools or extraction, transformation and load (ETL) environments.

Whether you're an experienced data warehousing designer, a seasoned database administrator, or a business expert with a need to bend technology to your commercial needs, WhereScape gives you the tools to build real-world data warehouses, fast.