Impact of Model-Driven Standards

(Rev. 1.7, clean format)

David Flater
National Institute of Standards and Technology
100 Bureau Drive, Stop 8260
Gaithersburg, MD 20899-8260
U.S.A.

Contact: David Flater, dflater@nist.gov, +1 301 975 3350

Abstract

The Object Management Group™ (the consortium that issues the Common Object Request Broker Architecture (CORBA®) and Unified Modeling Language™ standards) is making the transition from a standards architecture in which only interface definitions are normative to one in which the *model* is the primary normative artifact. This paper discusses the impact of this move for standards-based interoperability, technological change, reuse, and the state of the practice of application integration. Most significantly, the Model-Driven Architecture™ represents an opportunity for users of standards and standards workers, in the Object Management Group and elsewhere, to protect their investments of time, capital, and expertise to a greater extent than was previously possible.

Note: All referenced figures appear at the end of the document.

1 Introduction

The Object Management Group (OMG™) [8] was formed in 1989 to hasten the deployment of standards useful for integrating object-oriented software applications. Presently it includes about 800 members and is home for many standards, including the Common Object Request Broker Architecture (CORBA®) [3] and Unified Modeling Language (UML™) [9].

OMG's original framework for object-oriented integration, the Object Management Architecture (OMA) [4] (see Figure 1), supported the integration of enterprise applications with standard object services and each other using CORBA. An architecture in the sense of OMA is a complete context for and approach to designing systems that helps to ensure that high-level goals are met.

The CORBA standard specified most everything needed to enable the integration of applications, from a language- and platform-neutral Interface* Definition Language (IDL™) down to the communication protocol. For several years CORBA matured and grew, and many stable implementations, both commercial and open-source, became available. However, over time the emphasis shifted. While OMG remained the world's largest venue for standardizing object-oriented technology, OMG members increasingly found themselves under pressure to utilize OMG standards in non-CORBA environments. The OMG policy that IDL was the sole normative artifact in all OMG standards became increasingly inconvenient.

This inconvenience eventually led to a formal proposal to bump CORBA from its central role and instead let UML be the center of the OMG universe. The new architecture, known as the Model-Driven Architecture (MDA™) [1][10] (see Figure 2), was endorsed by the Platform and Domain Technical Committees of OMG in March, 2001.

As defined in Reference [1], "A model is a formal specification of the function, structure, and/or behavior of a system." Within the MDA, an interface standard – for example, a standard for the interface to a manufacturing execution system – would include a Platform-Independent Model (PIM) and at least one Platform-Specific Model (PSM). The PIM captures the conceptual design of the standard, untainted by the special features or limitations of a particular software technology. The PSM represents a "realization" of the PIM within the context of a particular software technology – for example, CORBA – wherein the choices among many possible ways of implementing different aspects of the conceptual design have been made. Additional PSMs can be standardized separately at a later time. The same PIM can therefore be realized within the context of any number of disparate software technologies without invalidating the standard.

In Reference [1], platform is defined as "technological and engineering details that are irrelevant to the fundamental functionality of a software component." Throughout this paper the broad term "software technology" is used to avoid identification with hardware concerns, but the intent is similar.

Though at first glance it might look like bad news for people who invested heavily in the OMA, the MDA actually represents a significant opportunity for users of standards and standards workers, in OMG and elsewhere, to protect their investments of time, capital, and expertise in standards and standards-based systems to a greater extent than was previously possible. Standards can be protected against premature obsolescence, and the cost of maintaining interoperability in the face of software technology change can be reduced. In the following sections, these assertions will be supported through discussion of the motivating forces, implications, results, and risks associated with the MDA.

2 Motivating forces

2.1 "Churn" of software technology

The word "churn" will be used hereafter to refer to the "software technology-of-the-year" behavior of the information technology market.

It is sad, but true, that few of the revolutions in software technology that the market so frequently endures actually represent fundamental advances in efficiency or effectiveness for any given application. Nevertheless, whether it is motivated by a search for the mythical silver bullet with which to slay the enterprise integration "werewolf" [2], by a fear of being left behind by technological change, or simply by a boundless enthusiasm for new and interesting things, a critical portion of the market migrates periodically from one technology to another, dragging the rest along with it as support for the previous technology dwindles.

---

* An interface "describes a set of potential requests in which an object can participate meaningfully" [4]. In programming language terminology, it is essentially the contents of the "header file," "package declaration," or like construct. It is the minimum information necessary for software integration.

Though the fear of being left behind by technological change might be the primary cause of technological change, the change inexorably occurs and cannot be ignored. Regardless of whether a particular change is beneficial or harmful for their applications, software vendors are obliged to migrate to new technology or be abandoned by their customers. Churn even affects technology whose purpose is to bridge between disparate technologies – "middleware," like CORBA.

The point is not to rail against possibly gratuitous changes, but to observe that these changes occur at a rate that creates a fundamental problem for standards organizations. By the time that a group of domain experts has assembled, reached consensus on a standard, implemented it, tested it, revised it, and made it work in practice, the chosen technology may have come and gone, leaving no opportunity to recover the costs of standardization.

The MDA represents a positive effort to make OMG standards "churn-proof" (or "churn-friendly," depending on one's viewpoint). By separating out the most valuable, most reusable part of standards – the conceptual design – the MDA enables existing standards to be realized on new technologies quickly and cheaply instead of being reinvented from scratch.

2.2 Primacy of conceptual design

The MDA evolved out of OMG experience that getting the conceptual design right is the hardest part and the most important part of standardization. Conceptual design includes not just the model of a system, but also its ontology – the context needed to understand the model in the way that was intended.

Given that the best, brightest, and most experienced practitioners are assembled for the purpose of creating a standard, one might think that a correct conceptual design should appear almost instantly. In fact, quite often an initial consensus is reached within a matter of days. But then, over time, a series of revelations occurs that shows that some aspect was misunderstood or did not adequately capture someone's intent, and the model is revised over and over. The outcome of this process, ideally, would be a conceptual design that represented *true* consensus, rather than the *apparent* consensus that occurs in the presence of misunderstanding. However, one can never be certain that no significant misunderstandings remain.

It is difficult to validate a conceptual design because different people use the same terms to refer to completely different things without realizing it. An observer naturally interprets terms and entities in a model according to the semantics that apply in his or her own experience, and with no evidence to the contrary, tends to assume that others are using the same interpretation. Most people are prepared to believe that "technical terms" used within a narrow domain of discourse by a group of experts are sufficiently unambiguous to avoid this problem. Unfortunately, experience has shown this to be a dangerous misconception. General semantics was explored in detail by 20th century philosopher Alfred Korzybski, who concluded that "we cannot identify the different abstractions of different individuals" [6] ("identify" in the sense "regard or treat as identical" [7]) no matter how exhaustively we think we have enumerated and defined our concepts. And indeed, it is this enumerating and defining of concepts in an attempt to make everyone understand in the same way that makes conceptual design so exhausting and so critical to the success of standardization. If the conceptual design is complete, consistent, and clear, any other problems (e.g., technical incompatibilities) are easy to fix.

Part of the motivation for the MDA was the realization that writing IDL – generating syntax that specified only the signatures of operations, not their intended usage or behaviors – was a distraction to resolving the more important semantic problems. Though their mapping to the programmatic level can be highly ambiguous, UML models capture more information at the concept level and more of the intent of the designer. This makes them more effective tools of communication to assist the committee members in understanding each other and to assist users of standards in understanding the intent of the committee.

2.3 "Interoperability"

Within the information technology domain, the jargon word “interoperability” signifies different combinations of various concepts depending on who is using it. The author has found it constructive to use the following terms in lieu of "interoperability."

- "Compatibility" – the ability of systems to cooperate meaningfully for some purpose and/or to coexist without interfering with one another. Compatible systems work together as-is, not requiring any integration effort beyond some trivial configuration to let them know that they are supposed to work together.

- "Integratability" – the quality of possibly *incompatible* systems that makes it easier to adapt them or their exchanged data so that they will cooperate meaningfully for some purpose. Systems that are integratable can be made compatible with some nontrivial amount of effort, for example, by writing "glue code."

- "Composability" – the quality of a collection of components that enables them to cooperate meaningfully for the purpose of assembling larger systems.

- "Substitutability" – the quality of a group of things (call them "servers," though they could be any sort of thing) that makes it possible to replace one with another without adverse effects on things that depend on them (call them "clients").

- "Portability" – the quality of a "client" that enables "servers" to be substitutable. (Substitutability and portability are tightly interrelated.)

The only thing common to all of these interoperabilities is the vague notion that *separate* things will do the "right" thing when they are put together. What is "right" depends on context and the goals of the user.

In any event, "interoperability" has always been a goal of OMG technology. The precise intent of the term as it has been used in OMG for some years is clarified by context in the template that identifies requirements and evaluation criteria for all OMG specifications [5]:

> From section 1.1: "Objectives include the *reusability, portability,* and *interoperability* of object-oriented software components in heterogeneous environments."
>
> From section 5.1.1: "Proposals shall express interfaces in OMG IDL."
>
> From section 5.1.9: "Proposals shall specify interfaces that are *compatible* and can be used with existing OMG specifications. Separate functions doing separate jobs should be capable of being used together where it makes sense for them to do so."
>
> From section 5.1.11: "Proposals shall allow *independent implementations* that are *substitutable* and *interoperable*. An implementation should be replaceable by an alternative implementation without requiring changes to any client."

The above requirements clarify that this OMG "interoperability" exists within the context of standards for software component interfaces, which among other things, shall be defined in IDL, and shall be composable within the OMA. But as the emphasis has shifted increasingly towards conceptual design, there has been an increasing role for specifications that deal with "meta-level" concepts, or that relate to the mapping from these concepts down to the implementation level. These specifications do not fit naturally within the requirements of standards for software component interfaces as described above, yet the need for them is undisputed. This was another part of the motivation for the MDA. The MDA brings with it a broader viewpoint that is more inclusive of all of the "supporting" specifications that help us to achieve "interoperability."

3 Impact for users of standards

3.1 Applicability and longevity of standards

The notion of realizing a standard conceptual design with a software technology other than the one on which the standard was originally deployed is now endorsed and supported by the OMG. The users of OMG standards will therefore have an easier time applying them in the diverse environments that exist now and in the environments to

which they will migrate in the future. When technological change does happen, much of the cost traditionally incurred will be avoidable because the old system and the new will be able to share a common, standard conceptual design. Less effort will be needed to re-integrate software, because instead of having to refactor entire ontologies, integrators will only have to translate between different representations of the same concepts. Less retraining of personnel will be required because the fundamental functionality of the system will follow the same design.

3.2 Expressiveness of standards

With the shift from IDL to UML as the primary normative artifact, OMG gains the capability to include many more constraints and clarifications within the normative part of its standards. Structural and behavioral information that was beyond the scope of IDL, such as the relationships between different types of objects, cardinality constraints, and the legal ranges of attributes, can be specified in UML. UML-based standards can achieve a higher level of completeness and rigor than is possible in standards where this structural and behavioral information must be specified in non-normative, descriptive text. Consequently, they can be more effective at reducing the cost of achieving and maintaining compatibility and more valuable to users.

3.3 Conformance

Conformance to a well-designed standard within the context of the MDA implies integratability. Though they may utilize incompatible technologies or incompatible mappings to the same technology, two implementations of a standard PIM will share a common conceptual design. The effort required to integrate implementations of the same PIM that used different PSMs will be less than the effort required to integrate systems developed in isolation from each other, in the absence of a common standard.

Conformance seldom, if ever, necessarily implies compatibility of software, even when standards include the specification of all underlying technologies. Ambiguities in the standard and differences in understanding among the readers make it unlikely that two implementations developed in isolation from each other would be compatible on the first attempt. After several iterations of changes and retries, compatibility can be achieved; but if this process becomes long and expensive, then the value of having attempted to produce standards-based compatibility in the first place is questioned. Nevertheless, the value of standards is in *reducing* the cost of achieving and maintaining compatibility. Whether or not that cost can be reduced to zero makes no difference to the viability of standards in practice as long as a return on investment is demonstrated.

The MDA allows for conformant implementations on incompatible technologies, while the OMA did not. Compatibility between implementations that happen to use the same technology is neither more nor less likely to emerge than it was before, but implementations using different technologies are now more integratable because they can conform to the same standard. The MDA therefore represents not an abandonment of compatibility, but an embracing of additional opportunities for integratability. This greater impact makes it easier to demonstrate a return on investment for defining and using standards.

3.4 Transience of technologies

One implication of the emergence of the MDA is that standards are not as good indicators of a technology's longevity as has previously been believed.

For end users, indiscriminately keeping up with technological fads has high costs and few measurable benefits. Few of the innovations that contribute to software technology churn actually represent potential gains in efficiency for any given enterprise, and many become obsolete or irrelevant in a very short time. Unfortunately, it is difficult to distinguish the technologies worth pursuing from the others within the time frame necessary to maintain a competitive advantage. Users therefore look for strategies to minimize the resources expended on keeping up with technology fads without getting left behind on the changes that really matter.

Users must make decisions based on the evidence that is available at the time. Often, this evidence includes the existence of, or lack of, commitment by standards organizations to create standards around the technology in question. But if one subscribes to the philosophy of the MDA, then this is the tail wagging the dog. Standards must

impact the viability of technology to some extent, but the impact of technology on the viability of standards is much greater.

4 Impact for standards workers

4.1 Pragmatism and flexibility

Some standards workers see the ultimate goal of their work as a standard that is perfected to the point that conformance would necessarily imply compatibility. The ideal outcome would be that such a standard is implemented and used universally, resulting in universal compatibility with zero integration cost. One could call this "complete" standards-based compatibility.

At best, it is an understatement to say that complete standards-based compatibility is time-consuming and expensive to achieve. If it is achievable at all for software, it is achievable only if the opportunities for changes, upgrades, or extensions to supporting infrastructure are severely constrained by the standard. Nobody actually *wants* those kinds of constraints, yet the frequency with which routine upgrades break compatibility in practice demonstrates that compatibility cannot be *guaranteed* without them.

Since it is not usually practical to constrain users in that way, most standards organizations have accepted that there will be changes to the underlying infrastructure and coped with the risks that this flexibility entails. The MDA is just the next logical step in the direction of increased flexibility. Not just changes to the infrastructure, but the very choice of *which* infrastructure will be used, is a degree of flexibility that users expect, and standards must provide.

The MDA does not prevent anyone from pursuing the ideal of complete standards-based compatibility using a particular software technology and PSM. However, it does provide a pragmatic way forward in the event that commitment to any particular software technology wanes before investment in standards that depend on it has fully paid off. If that occurs, one must build a new PSM, but one can reuse the conceptual design, which represents the lion's share of the investment.

4.2 Integratability as a goal

At one time, standards organizations did not respect integratability as a possible goal in itself. The ultimate goal was always compatibility, so a standard that did not attempt to deliver complete compatibility was not considered a serious standard.

While compatibility of systems is still the goal of users, the acceptance of the MDA suggests that, in these more pragmatic times, standards organizations could be more accepting of a standard that aims for integratability only. In some cases, this would simply be a lowering of expectations and should not be accepted. But in other cases, there is the real possibility that the costs for both contributors to and users of the standard would be reduced. In some cases, it might well cost less in the long run to accept a certain amount of integration expense than to attempt to "pre-integrate" everything through highly constrained standards.

A standard that aims for compatibility must specify the usage and behaviors of operations with enough completeness to cover all of the expected use cases and with enough constraints to be unambiguous. In many cases this can be elegantly done. But in the unfortunate event that the use cases from different customers turn out to require wildly divergent behaviors from some particular feature, then that feature becomes a standards nightmare that is nearly impossible to specify, to understand, or to use. In all likelihood, the resulting standard ends up being rejected for being too complicated, too big, or too inelegant.

On the other hand, if the same standard aimed for integratability only, then that problematic feature could be specified loosely. All of the necessary behaviors would be *feasible* with the interface and model provided, but the specification would not necessarily be complete or unambiguous. Clarifying the usage and behavior of that particular feature would be left to the implementor or user group, and bridging between the different usages and behaviors of two different implementations would be left to the integrator. Integratability would still be assured

because the two implementations would still share a standard set of concepts. The resulting standard would be simpler, more maintainable, more understandable, and probably more viable in general, simply because a particular *feature* that could not be cleanly standardized was left out. Of course, the remaining features that could be fully standardized should be.

The product data management systems used in manufacturing demonstrate why this compromise is sometimes worth considering. All commercial product data management systems are designed for a high degree of site-specific customization. This is necessary because the diverse physical constraints of manufacturing and diverse political constraints of dealing with customers and suppliers are not adaptable to the convenience of the software vendor. The "business rules" of the user of the product data management system necessarily have a great impact on its usage and behaviors, and every user's information schema is different. One cannot execute any realistic use case without touching upon some site-specific customization; yet, there is a core set of functions that every site needs, one way or another.

In that sort of environment, even with a standard interface, writing a portable client is daunting, at best. Is it then pointless to have an OMG standard for product data management? No. Different systems will implement different business rules, and these will always need to be learned by integrators, but that learning will occur within the context of a known, standard set of concepts and a known, standard computational model. There is a clear return on investment for integratability, even if *a priori* compatibility proves hard to achieve.

4.3 Reference implementations and profiles

Now that platform-specific concerns are being factored out of standards to a greater degree, there is a correspondingly greater need for reference implementations and UML profiles to clarify how formal assertions are intended to be realized in everyday practice. (A UML profile is a specification of the meaning of a particular usage of UML extensibility features within a particular context.)

For the proponents of particular software technologies, this represents an open opportunity to draw in more users. Although OMG does not standardize on reference implementations, the public availability of high-quality reference implementations, mappings, tools, and support would encourage the selection of a given technology for normative PSMs and facilitate the migration of other standards to that technology.

5 Risks

5.1 Degraded testability

Standards are more effective at reducing the cost of achieving and maintaining compatibility if they provide sufficient constraints to enable compliance to be assessed unambiguously. With the MDA, the ability to assess compliance depends entirely on the completeness and consistency of the mapping from PIM down to implementation details.

To become a standard, a submission must contain at least one PSM. If the PSM is expressed in IDL, then its testability is the same as the testability of previous OMG standards. But PSMs expressed in a modeling language must be done more carefully if testability is to be preserved. It is too easy for the details that are necessary for testing to be "abstracted away" in the interest of model clarity.

Fortunately, this risk is already being addressed through a call for proposals for a UML Testing Profile [11], a "meta-level" standard that will define how test purposes and the structural and behavioral aspects of test suites can be integrated with the system models that will be standardized within the MDA.

In cases where a PIM has been realized *ad hoc* on some platform for which neither a PSM nor a UML profile has been standardized, the worth of conformance testing is debatable. Without a normative PSM or UML profile, there can be no authoritative traceability to the standard.

5.2 Change of modeling paradigm

The MDA rests on the major assumption that modeling languages, or at least modeling paradigms, are less subject to churn than software technologies. Historically, there have been many modeling languages that have come and gone, but the unification of the most-used methods in 1996 and continuing integration of best practices into UML has proven to be a surprisingly stabilizing influence.

Importantly, most known modeling languages are integratable with UML. They share a paradigm – a "mode of viewing the world" [7] – so a conceptual mapping and a relatively painless migration are feasible. If just "yet another" modeling language were to become popular at UML's expense, some migration would be necessary, but the investment of building PIMs in UML would mostly be preserved. This would be similar to substituting PSMs, but on a different level of abstraction. On the other hand, if that language captured a modeling paradigm that was not even integratable with the UML paradigm, the premise of the MDA would be invalidated. The conceptual designs would not be reusable; they would need to be rebuilt in the new paradigm. There is nothing to mitigate this risk except for the widespread belief that a shift away from the UML paradigm is unlikely to occur in the near future.

5.3 Rejection

Increasing the level of abstraction at which something is designed tends to decrease the intuitiveness of the level at which it is implemented. A complex mapping from PIM to implementation could result in non-intuitive interfaces that scare away developers. To avoid this, the developers of UML profiles and other supporting standards must adhere to the same design principles as the developers of interface standards: confine the scope of the standard to exactly what is required, nothing more; reuse that which is reusable; keep independent functions separate; and above all, keep it simple. [5]

A completely different kind of rejection risk is that the MDA might be perceived by some as evidence that open standards are weakening, leading them to buy into proprietary, nonstandard solutions. Hopefully, by now the reader has been convinced that this perception would be inaccurate. The MDA does not weaken existing OMG standards, but strengthens everyone's ability to realize and reuse them in a much broader context.

6 Conclusions

Software technology is transient. When a standard is irrevocably bound to a particular technology, the quality work and experience that go into its conceptual design disappear along with the technology. The MDA approach makes it possible to save the conceptual design, which is the most valuable part of the investment. But even if the relevant technologies are in no imminent danger of disappearing, the MDA helps to avoid duplication of effort and other needless waste.

While the MDA originated in the OMG, there is nothing to prevent other standards organizations from benefiting from it. The primacy of the conceptual design makes it easier for previous work to be adopted in OMG and for OMG work to be realized in the contexts of other standards organizations. Moreover, by factoring out the aspects of standards that are peculiar to particular technologies, the MDA eliminates many of the bad reasons that are often used to justify "yet another" standard. One may well need to build "yet another" PSM, or perhaps many of them, but the more difficult and expensive process of conceptual design need not be repeated. Unlike so many transient technologies, the MDA represents a meaningful step towards reducing the cost of achieving and maintaining "interoperability."

Acknowledgements

The formulation "ability to cooperate meaningfully for some purpose" is due to Evan Wallace, who proposed it as his own definition of "interoperability."

The author thanks the following reviewers whose comments have improved this paper: Len Gallagher, Sharon Kemmerer, Jim Nell, Richard Soley, and Larry Welsch.

References

[1] Architecture Board MDA Drafting Team, "Model Driven Architecture: a Technical Perspective," Version 1.0, http://doc.omg.org/ab/2001-02-04, 2001.

[2] Frederick P. Brooks, Jr., *The Mythical Man-Month: Essays on Software Engineering*, 20th anniversary edition, Addison-Wesley, 1995, Ch. 16-17.

[3] *The Common Object Request Broker: Architecture and Specification*, Rev. 2.4.2, http://doc.omg.org/formal/2001-02-33, 2001.

[4] *A Discussion of the Object Management Architecture*, http://doc.omg.org/formal/2000-06-41, 1997.

[5] *Generic RFP Template*, Version 11, http://doc.omg.org/ab/2000-05-02, 2000.

[6] Alfred Korzybski, *Science and Sanity: An Introduction to Non-Aristotelian Systems and General Semantics*, 5th edition, Institute of General Semantics (http://www.general-semantics.org/), 1994, p. 425.

[7] *New Shorter Oxford English Dictionary*, 1993 edition, Oxford University Press, 1993.

[8] Object Management Group home page, http://www.omg.org/, 2001.

[9] *OMG Unified Modeling Language Specification*, Version 1.3, http://doc.omg.org/formal/2000-03-01, 2000.

[10] Richard Soley and the OMG Staff Strategy Group, "Model Driven Architecture," Draft 3.2, http://doc.omg.org/omg/2000-11-05, 2000.

[11] UML Testing Profile draft RFP, http://doc.omg.org/ad/2001-04-02, 2001.

Figures

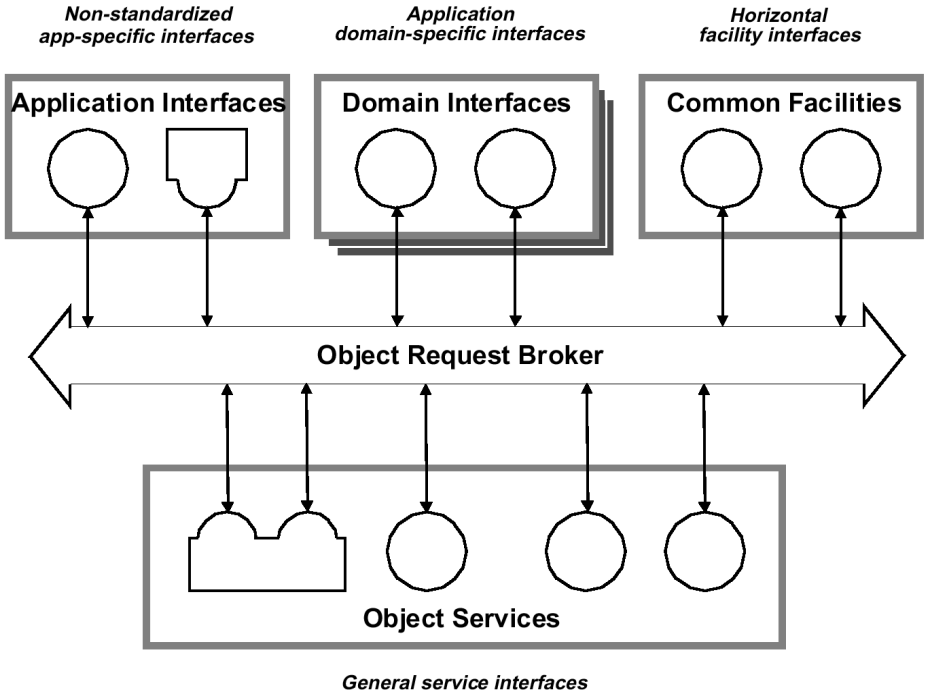

Figure 1: the Object Management Architecture

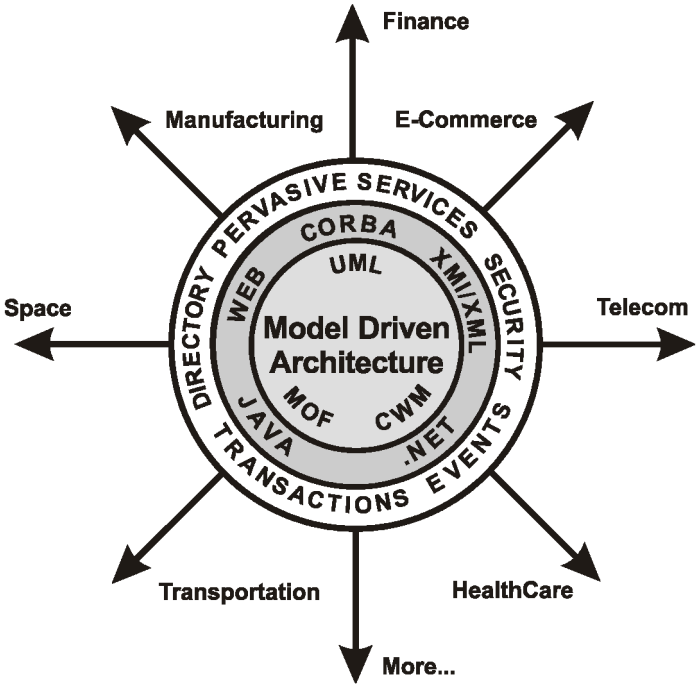

Figure 2: the Model-Driven Architecture