Distr.
GENERAL

CES/SEM.47/19
21 February 2002

ENGLISH ONLY

STATISTICAL COMMISSION and ECONOMIC COMMISSION FOR EUROPE

COMMISSION OF THE EUROPEAN COMMUNITIES

CONFERENCE OF EUROPEAN STATISTICIANS

EUROSTAT

Joint UNECE/Eurostat Seminar on Integrated Statistical Information Systems and Related Matters (ISIS 2002)
(17-19 April 2002, Geneva, Switzerland)

Topic III: Object-oriented technologies, component architecture

SEMANTIC INTEROPERABILITY: TALES FROM A TECHNOLOGY-LIBERATED WORLD

Invited paper

Submitted by Eurostat [1]

Abstract: This paper presents a survey of technology used to support component-based development, namely middleware, application servers and application integration brokers. It analyses the problems associated with such component hosting platforms and assesses the adequateness of current technology in the light of new requirements for software application interoperability over wide-area networks like the Internet. It presents an approach for software interoperability where components base their interactions on the semantics of the business they support. It describes how such semantic interoperability approach is applied in the context of the STATOBJECT IST project.

I. INTRODUCTION

1. Object-orientation is presently considered as the mainstream approach in modern software construction activities. Reuse as well as the smooth integration of objects and software components are some of the key arguments in favour of object-oriented technologies. Object-oriented technologies are the natural choice in the case of IT projects aiming at developing new software systems. However, it is for new systems to be created from scratch. Usually, IT projects are bound by various constraints related to integration with heterogeneous hardware, including mainframe systems, as well as existing software applications that are not using object-oriented technology. Giving that context, and taking into account the large spectrum of software technologies available today, it is not very realistic to expect software solutions that are "purely" based on object-oriented technologies.

2. Application Integration and Middleware technologies developed over the last five years make an important step towards interoperability of software technologies. Mainly backing component-based development, such technologies can be seen as the software layer that hosts application components and

[1] Prepared by Denis Avrilionis (denis@darcedge.com).

ensures a certain quality of service during execution of these components. The success of such *component hosting platforms* has created a new category of software that did not exist in a general-purpose, commercially available form prior to 1996. Companies selling their products under the "Enterprise Application Integration", "Business Process Management", "Application Server", "Integration Broker" or "Middleware" banners have evolved from niche players to important market makers that can rival the big traditional software vendors like IBM, SUN, or Oracle. Giving that this segment of the market is young and still under consolidation, it is difficult and often risky to make technological decisions on the basis of the results of the latest market research studies.

3. This paper supports the idea that current component hosting platforms are too complex to suit the evolving needs of component-based development. In the last few years we have observed a shift from enterprise-wide software solutions, to software solutions that suit the needs of individual knowledge workers. As this tendency is increasingly confirmed in the field, component hosting platforms must shift their focus from technical integration solutions to solutions that enable business people to interact using high level concepts, close to the ones used in their day-to-day business. Based on the assumption that the technology is there to serve as a solution to a well defined problem, we argue that the key for improving reuse and interoperability of software as well as to building scalable and easily extended software architectures is not technology per se, but a thorough knowledge of the problem to be solved. In other words, putting the focus on the content and the semantics of the information can lead to a better application of object-oriented technology and interoperability between software components and systems. Shifting the focus from technology to the semantics of the information manipulated by an application makes it possible to produce better quality software. This is the approach we have adopted in the context of the STATOBJECT project.

4. Section II of this paper sets the vocabulary of terms often used to describe component-based software engineering. Section III presents a taxonomy of the various technological alternatives available today for hosting software components, namely middleware, application servers, and integration brokers. Section IV presents the problems associated with today's component hosting platforms and identifies some key issues to watch when adopting such technologies. Section V gives our view on the reasons of the complexity of current component hosting platforms. Section VI presents some of the new requirements that component hosting platforms would have to cover in the near future. Sections VII and VIII present the approach we are adopting for component-based development in the context of the STATOBJECT project. Section IX draws some conclusions.

II. COMPONENT-BASED DEVELOPMENT

II.1 The Theory

5. Component-based development is based on the idea that software can be created through the assembly of run-time functionality by integrating both custom developed and commercial off-the-shelf (COTS) modules. The original motivation of the approach is that re-use of preexisting *Components* would ease and accelerate software development effort. In addition to ``run-time" components, *Patterns* and *Frameworks* are promoting reuse at the level of software design and specification. Patterns are simple abstract solutions to common design problems. The solution they express is language and implementation independent, and usually applies across many different types of application. Frameworks are (often quite large) sets of co-operating classes or components that make up a design for a specific class of application. Using frameworks often implies that most of the design is complete at a high level and developers need to “fill the details”.

II.2 The Practice

6. Although the theoretical benefits of component-based development make the approach promising, both the development methodology as well as the technology needed to deliver this vision is still incomplete. The component communication standards or architectures (for example, COM, Corba, Enterprise Java Beans) exist, but achieving maximum re-use requires the standardisation process to extend further into middleware and application integration functionality than is the case at present. Practically, the whole process of architecting, designing, building and deploying component-based software solutions can be reinforced by the capabilities offered or, hindered by the constraints imposed, by the underlying component hosting platform. By *Component Hosting Platform* we refer to the software layer that is responsible for developing and run the components. Choosing the correct component hosting platform that fits the needs of a particular problem is the hidden complexity, and often the high price to pay, when adopting component-based development approaches. Below we give a quick overview of the different types of component hosting platforms that are currently available in the market.

III. COMPONENT HOSTING PLATFORMS

7. Component hosting platforms have been built for the last 10 years in order to support enterprise computing environments. Coping with enterprise requirements means that the platform would host components that model a core part of the business, and that business operation would not be able to run effectively without it. Such platforms are deployed for long enough and they must be able to deal with significant changes in the business without major impact in the services they are offering to the rest of the software infrastructure. They must be able to support a large number of users as well as a high transaction throughput, or both. Component hosting platforms are central to the effective operation of the business in which they are deployed. To do this they must be able to interact with pre-existing systems and databases, which also fulfill business-critical functions. They must also be robust and reliable. Moreover, with the advent of the Internet, component hosting platforms must be able to cope with widely varying loads, while maintaining an acceptable performance profile, and they must be secure. They must be able to allow or disallow components to access certain parts of the infrastructure (e.g. databases with confidential information). They also must facilitate administration and deployment of components and applications, for local and remote locations.

8. As far as the core services are concerned, although the technology is rather heterogeneous, the following list of services is usually covered by the majority of component hosting platforms currently available in the market:

- **Naming/directory.** A directory is a database, or set of reference files, which contains all the information needed by the component hosting platform to operate at run-time.
- **Memory management/life-cycle.** A memory management service supplements those provided by the operating system, and is aimed at making better use of available memory, freeing memory when it is no longer needed, allocating and re-allocating blocks of memory as processes change.
- **Multi-threading/multi-tasking.** Where many requests are made for a single component, the component execution platform can handle the situation by queuing the requests, or by using multi-threading or multi-tasking.
- **Load balancing.** Load balancing bears some similarity to multi-tasking. The service balances the request load by creating duplicate processes to perform a requested service, but these duplicated processes are on different machines.
- **Finder.** The purpose of a finder service is to provide a way for a client to locate a specific component instance without already having a reference to it.

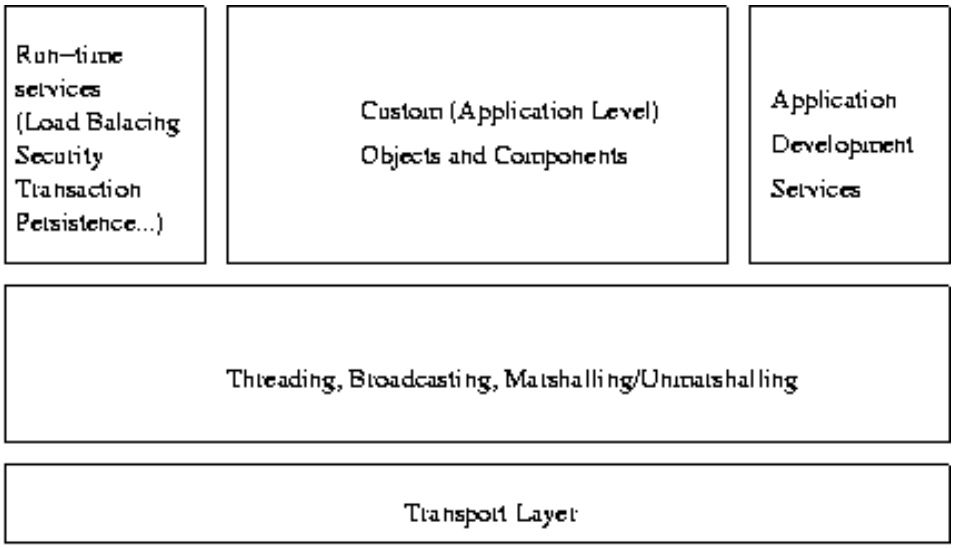
- **Security.** Security services usually implement three extremely important functions: authentication, authorization, and encryption
- **Persistence.** Business components represent managed elements of a corporate data model. Therefore, component hosting platforms usually provide a facility for transferring components from the main memory of a computer system to secondary storage, and vice versa, to allow components life-spans to exceed the length of time that the computer is switched on, and to ensure a degree of reliability in the case of system failure.
- **Transaction processing.** A component hosting platform often provides a transaction processing service to enable distributed business components that store their data on heterogeneous Database Management Systems and machines to change their state consistently and reliably.

9. Since the technology evolves in a rapid pace and new standards and architectures are emerging, it is quite difficult to define a taxonomy of the technology used to build component hosting platforms. Below we distinguish three types of platforms, namely middleware, application servers, and application integration brokers. This typology is more based on the three generations of component hosting platforms that emerged the last five years. However, it should be noticed that all three types provide similar functionality, but often rely on heterogeneous technology. As the technology evolves it is highly probable to see a consolidation of the architectures towards a general purpose component hosting environment for the enterprise.

III.1 Middleware

10. Middleware products have emerged as the software layer supporting leading-edge users of technology to build sophisticated distributed applications. In many cases, these applications have been implemented to support relatively extreme requirements, for instance airline booking systems or telecommunications billing systems. Today, middleware is the technology most companies are using in implementing IT solutions that must integrate wide ranges of distributed, heterogeneous resources. Middleware is the fundamental tool when it comes to dealing with integration issues. Middleware products are now equally likely to be embedded within other products or solutions, rather than be used stand-alone, however, understanding which business benefits they offer (and how they offer them) is critical to large companies and organisations.

11. Given the diversity of middleware products available in the market today it is hard to build a generic model describing component hosting platforms based on middleware technology. The picture below gives a high-level view of the common architecture blocks that can be found in most middleware products of the market. As shown in the picture, components that rely on middleware hosting platform usually do not have to deal with low-level data/object transport issues. Components also can use various application programming interfaces in order to deal with common distributed computing problems like load balancing, distributed transactions, authentication, and authorisation, etc.



12. Among the different middleware paradigms Object Request Broker middleware are message-oriented middleware are the most popular ones.

III.1.1 ORBs

13. Although the ORB acronym is often used as a synonymous to Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA) specification, an *Object Request Broker* (ORB) is a broader architectural paradigm than CORBA and has been implemented by various products over different operating systems. Although OMG's CORBA ORB offers the most technologically neutral architecture, Microsoft's DCOM as well as Java's RMI can be though as some kind of object request brokers supporting communication between objects. The ORB communication paradigm relies on a communication between a client and a server object through the use of local copies (proxies) of remote objects that reside on the address space of the client object. Clients locate server objects at runtime and then build local proxies of server objects. It is the proxy that communicates with the server. An Interface Description Language (IDL) is usually used to describe the structure of the data exchanged between objects in a language-independent manner.

14. ORB-based middleware often define a set of value-added services implemented on top of the basic object communication mechanism provided by the ORB. These services cover a broad range of distributed computing problems. CORBA's common services is a representative list of such ORB-based value-added services (e.g. naming, life cycle, transaction, persistence, event, trade, etc.[1]). As the market is evolving, Java enterprise specifications (Enterprise Java Beans, Java Transaction Service, Java Naming and Directory Interface) are relying on OMG's CORBA specification. Moreover, CORBA specification version 3 and late Java Enterprise Specifications are converging to a common architecture paradigm for software components.

III.1.2 Message-oriented middleware (MOM)

15. The MOM paradigm is based on the principle whereby applications run in different operating system processes communicate with each other by passing messages over the network. Messages contain control and application data. Control data is used by the component hosting platform. The component hosting platform is agnostic of the nature of the application data. Application data may be an arbitrarily long sequence of bytes

that contains data such as arrays, video images, simple data structures, images or sound. There are generally few restrictions imposed on either the length of the data sent, or the type of data. Unlike the Interface Description Language (IDL) used as a way to describe the content of messages in the case of ORB, MOM middleware treats application data as an opaque block of information that is passed to the application components.

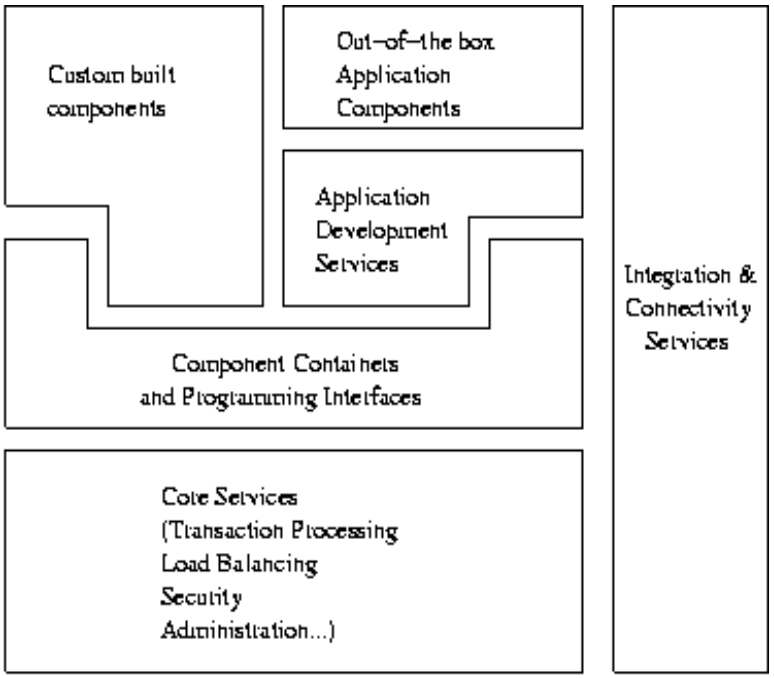
16. MOM is used in high performance and availability environments where the quantity of information to deliver is quite important (e.g. trading rooms receiving on line stock quotes from the stock markets). In the last few years MOM products are progressively include guaranteed delivery, transaction facilities, as well as security in their platforms.

III.2 Application Servers

17. Application servers became very popular as component hosting platforms for web-facing applications. In a more broad sense, application servers provide the runtime infrastructure and development services necessary to deploy applications or components in a multi-tiered architecture supporting web and other client interfaces.

18. The application server market has been created by the strong demand for transactional and secure web-facing architectures during the early dot-com era. Giving such a broad area of functionality, vendors in the area of Database management systems, enterprise middleware, or client-server development frameworks have enlarged their product offering to cover this new market opportunity. Consequently, application server products that are currently available in the market are reflecting the initial orientation of the vendor and thus have different strengths and weaknesses.

19. Application servers are built for a specific purpose, i.e. to provide easy-to-program transaction-processing environments for web-based and other thin-client applications. Using an application-server hosting platform often implies strong requirements in multi-user server-based transactional applications. The figure below shows the main architectural blocks of an application server hosting platform. Although transactional processing was one of the primary requirements in developing application servers, presently, almost all applications servers available in the market are complete component hosting platforms providing extensive capabilities for distributed computing, integrating middleware technology, as well as various connectivity links with data store mechanisms and custom enterprise applications.



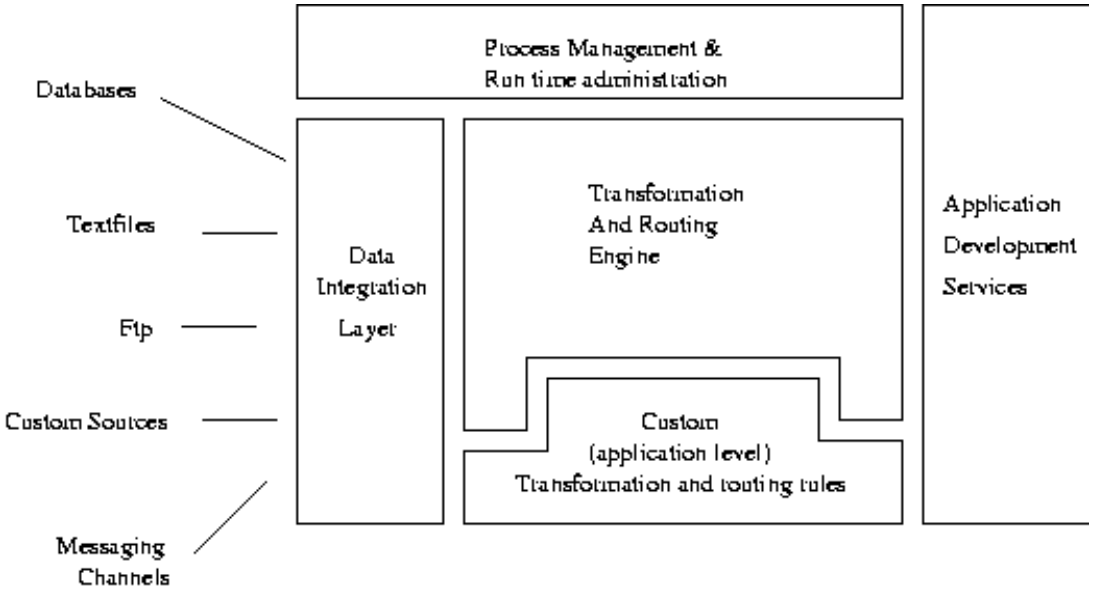
III.3 Application Integration Brokers

20. Enterprise Application Integration (EAI) technology combines a set of technologies that help automate the integration of custom-built and/or packaged business applications, so that they interact seamlessly, and exchange business-level information in formats and contexts that each understands. EAI solutions are not always single products. In fact, they often comprise many products, each of which focuses on automating a particular aspect of integration, or a particular type of integration problem. Such products lines, often marketed as ``Enterprise Application Integration Brokers" make initial implementations of integration solutions quick and easy to carry out by automating many steps of the integration effort.

21. In many ways, the client-server revolution set the stage for the coming of EAI. Before the rise in popularity of packaged enterprise applications and of departmental LAN computing, IT departments of many large organisations attempted to create a corporate data model that would be used to solve this difficulty. Such initiatives were rarely successful, because the complexity associated with creating, verifying and tuning such models meant that, by the time the modelling group had completed its effort, the rest of the IT organisation had created more systems to accommodate changing business needs. With the introduction of LANs, the client-server technology revolution and the attendant shift of responsibility for IT spending (from central IT to departmental lines of business), even the few successful projects ended up being irrelevant. The result is that in all but the most opportunistic, short-lived integration projects, the hand-crafted approach to integration yields solutions that are too costly, complex and fragile. In addition, given that many integration projects that start out as temporary solutions often stay in use far longer than the team's original expectations. Enterprise Integration Brokers made a successful entry into the software market by introducing a structured component-based approach for enterprise integration.

22. The figure below gives a high-level model of the most common architectural blocks of current EAI brokers of the market. Most often such products propose connectivity with a wide variety of data storage mechanisms ranging from databases to files containing proprietary data formats. Components that are hosted

by such platforms aim at transforming data entities from one format to another (e.g. positional records to XML, relational database records to proprietary message formats, etc.). In addition, higher level components are describing the process flow between enterprise applications.



IV. DO CURRENT COMPONENT HOSTING PLATFORMS FIT THE NEED OF COMPONENT-BASED DEVELOPMENT?

23. The benefits of component-based development have been put forward for many years and come in pair with the wide adoption of Object-oriented technology. Object-oriented technologies are the natural choice in case of IT projects aiming at developing new software systems. Component-based development is a valuable approach for new systems to be created from scratch. Unfortunately, creating systems from scratch is an ideal situation that becomes rare in the present global economic slowdown. Commonly, IT projects are bound by various constraints related to integration with heterogeneous hardware, including mainframe systems, as well as existing software applications that are not using object-oriented technology. Giving that context, and taking into account the large spectrum of software technologies available today, it is not very realistic to expect software solutions that are "purely" based on object-oriented technologies. However, giving the heterogeneity and the distributed nature of today's software applications, it is highly probable that some sort of middleware, application server or integration broker technology would be required. The adoption of such a component hosting platform requires a lot of attention, since it is a critical factor of success (or failure!) of an IT project.

24. Many companies used to build their own middleware rather than using an off-the-shelf solution. Some of them are still unaware of the existence of component hosting platforms, some are unaware of their capabilities and some think they can do better than the software vendors. Unfortunately, the main added value of component hosting platforms is to remove the complexity associated with the development of component connectivity services. There is also a large group of companies where responsibility for building an application is entirely left to developers with no teams to provide infrastructure software or tools. In this case purchases of component hosting platform come from a project budget, which often forces the project leader to choose what he or she perceives to be the cheapest option. Unfortunately, developers often appear to be cheaper than component hosting platforms. This can be a route to disaster. The development of in-house middleware, application server, or integration brokers is a significant burden on an organisation,

requiring specialised and costly skills. There are examples of IT departments where over half the staff have ended up supporting connectivity services developed in-house. Some recent figures have placed the estimate of the cost to support in-house middleware at 70% of some companies IT budgets.

25. The evolution of the needs and the market offering the last five years have been a great catalysts in making the broader IT community aware of the benefits of component-based software construction. As presented in the previous section, a whole catalogue of component-based hosting platforms and their associated development approaches are now available in the market. However, early reports from the industry reveal an alarming view on component-based development. The complexity in developing with current component hosting platforms is already reported within the community of software developers[2]. As the market pressure for offering complete turn-key platforms for enterprise-wide component-based development increases, software vendors are not ready (at least in the short run) to re-assess the complexity of their products. Mergers and acquisitions add to the complexity of software products: companies trying to protect their market share acquire and integrate competitors' products into their product lines. The main driver for integration in most cases is time to market, often with disastrous impact on the overall architecture of the initial products.

26. It is our belief that, giving the complexity of current market products, it is not possible to expect out-of the box solutions for software integration and component-based development. The technology for building component hosting platforms is presently under consolidation so Application Programming Interfaces are not stable enough to guarantee portability, even across two successive versions of the same product. The complexity of building components that run in current component hosting platforms is such that companies and organisations they would have to keep a pool of developers that maintain home-built re-usable components over a long period of time. As a paradox, end-users of technology like IT departments of companies and organisations do not benefit from component-based development as the software industry it self (through standardisation of interfaces between services within the component hosting platforms).

27. In summary, on the one hand, in-house development of component hosting platforms do not solve the problem of component-based development. On the other hand, acquiring a component hosting platform would not automatically solve the problem neither. As trivial as this might sound, we believe that the only way forward is for engineers *to careful assess the problem to solve before taking any architectural decision on the technology to use*. Once the problem is understood and the core functional and non-functional requirements are identified, a meticulous assessment of the functionality offered by the component hosting platforms of the market must be carried out. By having in mind that no software tool would ever solve the problem with out-of-box functionality, it is very important to assess how much custom software must be written in order to make the component hosting platform fit the project needs. This must be done with a lot of precaution - engineers are always tempted to over-engineering the end-system (especially the non-functional parts of the software platform). It is often better to limit the features to use than to try to use the full list of applications programming interfaces offered by a component hosting platform. The critical problem to solve is not how to do the ``plumbing" in order to make technology work, but how the tailor the component hosting platform in order to fit the semantics of the interaction between components of the target application. The objective of the suggested approach is to shift the focus from technical interoperability between heterogeneous pieces of software, to *semantic interoperability* between business abstractions encapsulated in software components.

V. WHY IS COMPONENT-BASED DEVELOPMENT SO COMPLEX?

28. In the previous sections we have given our view on the current state of the art in component-based development and the underlying component hosting platforms. In summary, although component-based development sounds promising as a approach for low cost and efficient software construction, the task is

made difficult by the complexity of the underlying component hosting platforms. In order for engineers to successfully perform component-based development, we propose an approach where the problem drives the technology to use and not the other way around.

29. Below we analyse why present component hosting platforms ended up with such complex architectures.

V.1 Middleware and the “virtual mainframe”

30. The aim of early work on middleware and later on application server and integration brokers is to develop off-the-shelf connectivity software that supports distributed processing at runtime, and is used by developers to build distributed software. The ultimate objective in building component hosting platforms is to provide a set of services that enable the creation of a ``virtual mainframe" from a heterogeneous mixture of platforms and technologies. Creating a virtual mainframe means delivering an application execution environment that provides the run-time functionality of a virtual computer, similar to what can be found today when programming in mainframe environments.

31. We claim that such a vision is not adapted and not suitable in the current context of open computing environments spawning across wide area networks or the Internet. Components running in open environments have to use and communicate with other resources and components distributed over the network, often over unreliable links where latency, failures and disconnection are usual phenomena.

V.2 Where the virtual mainframe paradigm breaks: Wide Area Network programming

32. The technology that underlies all current component hosting platforms is based on the idea of a virtual address space in which components live and share data, with distributed (possibly world-wide) garbage collection, where strong typing guarantees the integrity of pointers. Unfortunately, even though pointers are allowed across machines on a Local Area Network (LAN), e.g. using the notion of network proxies as in the case of ORBs, they are generally disallowed across firewalls. Similarly, transparent distributed object systems are unrealistic: some network objects will be kept well hidden within certain domains, and reaching them will require effort. The essence of present middleware relies on the notion of action-at-a-distance computing: the idea that components are available transparently at any time, no matter how far away. While this assumption could hold in a LAN, it cannot work well in a Wide Area Network (WAN)[3].

33. In general, we can forget about the notion of action-at-a-distance computing: the idea that resources are available transparently at any time, no matter how far away. Instead, we have to get used to the notion that movement and communication are step-by-step activities, and that they are visibly so: the multiple steps involved cannot be hidden, collapsed, or rendered atomic. The action-at-a-distance paradigm is still prevalent within LANs, and this is the fundamental reason why LANs are different from WANs, where such an assumption cannot hold. If this were possible, we could then go on and program the Internet just like we now program a LAN. The reason why this is impossible is that there are phenomena observed in the WAN that remain largely hidden in LAN architectures:

- Virtual locations. Because of the presence of potential attackers, barriers are erected between mutually distrustful administrative domains. Therefore, a program must be aware of where it is, and of how to move or communicate between different domains. The existence of separate administrative domains induces a notion of virtual locations and of virtual distance between locations.
- Physical locations. On a planet-size structure, the speed of light becomes tangible. For example, a procedure call to the antipodes requires at least 1/10 of a second, independently of future

improvements in networking technology. This absolute lower bound to latency induces a notion of physical locations and physical distance between locations.

- Bandwidth fluctuations. A global network is susceptible to unpredictable congestion and partitioning, which result in fluctuations or temporary interruptions of bandwidth. Mobile computing is even more affected by bandwidth fluctuations: mobile devices may perceive bandwidth changes as a consequence of physical movement. Programs need to be able to observe and react to these fluctuations.

34. While component hosting environments continue to treat network latency, security authorization failures, resource unavailability and other WAN phenomena as programming exceptions, the complexity of the solutions and thus the products that are available and will arrive into the market is doomed to increase.

VI. NEW REQUIREMENTS FOR COMPONENT HOSTING PLATFORMS

35. As presented above, component-based development and component hosting platforms have been created as a solution to the increasingly complex enterprise computing environments. Such environments have been created as an (often organic) assembly of applications and components using heterogeneous technology over a long period of time. It is not unusual to find applications that adopt various architectural paradigms, including multi-tier and single tier client-server architectures; use heterogeneous storage systems ranging from DBMSs to files; or communicate using standardized or ad hoc middleware technology. In that context the primary concern of component hosting platform was to implement the basic ``plumbing" that make applications talk to each other. At that time (especially before the Internet era) the interactions between business people were rather well defined, stable and limited in complexity. The rise of Internet and intranet technologies has opened up new opportunities for collaboration between people. This creates new requirements for component hosting platforms:

- As more hardware nodes, software applications and people are connected via a network, the value of that network to the participants grows exponentially and a positive feedback loop is created. The problem of supporting such a virtual community shifts the focus of distributed computing platforms from technical integration to the support of collaboration and related services. Supporting networked people instead of networked components pushes computing hosting platforms to become community workspaces. In such dynamic environment where virtual teams are rapidly established and equally rapidly disbanded, the ``virtual mainframe" paradigm imposing central management of services can no longer apply without limiting flexibility.

- Business-to-business knowledge sharing, where organizations demand flexibility, without compromising their internal IT infrastructures can no longer comply with the security paradigm proposed by current component hosting platforms. The ``enterprise as fortress" security model cannot cope with the need for business people to access and share business information appropriate to the context of their interaction.

- People collaborating with partners and co-workers often mix personal knowledge with corporate information. Some of this information is unlikely to be of use beyond the scope of this particular interaction and is thus unlikely to appear in a central repository. This is especially true of transient information, such as ideas that evolve as the team collaborates. In the face of ever-increasing volumes of content in diverse structured and unstructured formats, it is impractical (if not impossible) to provide centralized access to information. A central model of software component interaction may be limited in terms of the range of content types it can support. Instead of using such a centralized paradigm, new interoperability platforms should allow people to interact with other peers in a decentralized but secure manner. Such distributed architectures may provide access to

specialized information resources applicable only to specialized group of users, for example, transient working documents, or production meta-data.

36. All the above examples show that as the nature of interaction between people over the network become far-reaching, the need for centralized enterprise-wide computing infrastructures based on heavy component hosting infrastructures is decreasing. As the focus shifts from interaction between companies, organizations, and institutions to interaction between knowledge workers, the need for flexible, semantically aware software platforms becomes more and more important. Pushing this reasoning further, if people had been able to describe the nature and the semantics of the integration between their peers in the language of their business, there would be little need for such complex component hosting infrastructures as the ones available in the market today. This is the primary motivation and the driver of the technological developments of the STATOBJECT project.

VII. SEMANTIC INTEROPERABILITY AND THE STATOBJECT PROJECT

VII.1 Understanding the problem

37. With regard to the efforts undertaken by the e-Europe initiative, the new shift for short-term indicators for the EMU policy, the structural indicators for benchmarking European policies and the start up of enlargement, the European Statistical System (ESS) Network needs to review its functional infrastructure and adopt a new strategy for its development.

38. The functioning of the ESS is based upon a large co-operation structure around an important number of working parties, task forces and committees meeting frequently at European and national level. This involves thousands of persons, using as inputs pre-existing documentation, expressing opinions, and taking decisions. The output of this co-operative system is encapsulated in legislative documents, norms and standards or simple textual information.

39. Moreover, the statistical production process within the ESS is constantly impacted by new regulations at the European level. Statistical offices have to share/exchange information so far kept within their production systems with other institutions and communities of experts. Private companies are more and more solicited to supply data to the ESS.

40. Statistical organizations have long ago started re-engineering their functional architecture adopting new management methods and culture that makes workers more independent and mobile but at the same time more responsible and accountable. However legacy in IT system is an obstacle to fully implement the new environment and get all the benefits out of the effort.

41. In that context, the objective of the STATOBJECT project is to investigate the requirements, define the architecture and implement a prototype of the software infrastructure that make possible for ESS people and applications to collaborate at a ``semantic" level. We call this semantic interoperability infrastructure the *European Statistical Grid* (see below).

VII.2 STATOBJECT Workplan

42. Work under the STATOBJECT IST project is pursuing the following objectives:

- Conceptualization of the notions underlying core processes of the European Statistical System (ESS) and description of these abstractions in terms of business object and component models

- Definition of the architecture of the *European Statistical Grid*, i.e. the software infrastructure on which such models can be implemented
- Validation of the European Statistical Grid concepts and architecture through demonstrators as well as definition of the path for transitioning to the new ESS architecture.

43. Innovation in STATOBJECT lies in the combination of the following: STATOBJECT decouples domain modelling from technological constraints and presents a statistical domain model that stands in its own right and can be used as a roadmap for implementing interoperability between statistical systems. STATOBJECT stresses the difference between statistical software systems and the infrastructure that is used to integrate these systems. By enforcing this difference at the architecture level, STATOBJECT allows the integration of heterogeneous systems using semantic interoperability protocols that are derived from the domain model. This work structure exhibits a two-step approach in which:

- A formal domain model and an architecture of the underlying software architecture are first defined,
- The above artefacts are validated on real-life examples based on existing statistical processes and statistical software systems.

VIII. THE EUROPEAN STATISTICAL GRID

VIII.1 Background on Grid Computing

44. Grid computing represents an approach to distributed computing, whereby the resources in the network are shared and extracted in order to support a common computation request. Most of today's grid applications are focused on the aggregation of CPU cycles, so that a network of enough individual PCs can achieve comparable or superior power to a supercomputer. Grids can also be used to enable large datasets to be shared and accessed by many users. The term ``grid'' in the context of the STATOBJECT project comes from the analogy between computing resources and statistical resources and applications: instead of aggregating CPU cycles, in the context of the STATOBJECT project we target a software infrastructure that connects people and applications in a network of peers. The European Statistical Grid is the infrastructure that enables ESS participants and software applications to create a distributed community where human and software agents can offer and consume services, according to the needs of the statistical production process.

VIII.2 Overall architectural approach

45. The European Statistical Grid is an infrastructure that connects distributed heterogeneous resources in a peer-to-peer fashion, in such a way that participating nodes can share documents, statistical meta-data as well as production data in a secure manner. Even though the underlying applications might be based on heterogeneous software technologies, the interoperability between such a heterogeneous collection of resources relies on the fact that the resources that are required to interoperate share some level of common semantics that is derived from the statistical domain model. The technology that enables the technical integration between heterogeneous software will be based on integration software that enables the definition of semantically rich interfaces, namely, web services and XML-based business object definitions. Such business objects would be derived directly from the statistical domain model. In other terms, the domain model forms the “semantic glue” that allow collections of diverse resources and applications to work together, and thus plays a crucial role in the construction of the European Statistical Grid. The ultimate extension of this vision is of a global grid in which all the computers connected to the European Statistical System network are able to communicate with each other directly on an application-to-application level. Web services seek to enable application-to-application connectivity across heterogeneous networks and, since grid infrastructure is designed to address issues of connectivity between disparate resources, there is a significant

level of synergy between the two approaches. In that context STATOBJECT will closely follow the work of the major web services initiatives, such as Microsoft's .NET, Sun's Sun ONE and IBM's Web Services.

IX. CONCLUSION

46. This paper presents a survey of component hosting platform technologies available in the market today. Component hosting platforms is a term used to encompass a set of software interoperability solutions such as middleware, application servers, and integration brokers - all this been technology created to allow development of reusable components for distributed software systems.

47. The main idea developed in this paper is that component hosting platforms are currently too complex to be used on a wide scale by software developers. One of the reasons for this is that they all based on the idea of the enterprise computing environment as a ``virtual mainframe" where all computing resources can be accessible as if they where on the same physical machine. Wide area network issues like accessibility problems, network latency, network partitions, etc. are treated as programming exceptions. Such view definitely cannot cope with the increasing need for lightweight infrastructures supporting semantic interoperability among distributed knowledge workers.

48. Our approach in the context of the STATOBJECT IST project is to develop a semantic interoperability layer, called the *European Statistical Grid* where participants of the European Statistical System can interact by exchanging knowledge and information on a peer-to-peer basis. Such interactions are based on the interoperability between business objects that are derived from a domain object model and validated by domain experts. The European Statistical Grid is the infrastructure that enables European Statistical System participants and software applications to create a distributed community where human and software agents can offer and consume services, according to the needs of the statistical production process.

Endnotes

[1] see ftp://ftp.omg.org/pub/docs/formal/98-12-09.pdf for a detailed description of CORBA services

[2] for example see http://www.javageeks.com/SBJP/

[3] Luca Cardelli, Abstractions for Mobile Computation. Jan Vitek and Christian Jensen, Editors. Secure Internet Programming: Security Issues for Mobile and Distributed Objects. Lecture Notes in Computer Science, Vol. 1603, Springer, 1999. ISBN 3-540-66130-1. pp. 51-94