5

Service creation environment as a software development platform

Andy Bihain
GTE
PO BOX 152092
Irving, TX 75038
USA
214-718-6670
arb1@Gte.com

James White
Tandem Telecommunications Inc.
1255 W.15th Street
Suite 900
Plano, TX 75075
USA
214-423-5383

1. INTRODUCTION

This paper documents the process view of the Service Creation Environment(SCE) and how this view can be physically implemented to support the end-to-end process of the Intelligent Network(IN) as viewed by GTE. GTE has defined the base sets of capabilities within IN as the Products and Services Infrastructure. This allows any new developments to draw upon the base set of capabilities without having to rewrite services that are needed by the application.

1.1 Vision

The vision of the SCE is to build software that can be compiled, tested, provisioned and run in multiple target platforms in an automated fashion.

The following goals should allow GTE to achieve the vision show above:

- Vendor Independence
- Service Independence
- Separation of "What" from "How" of application
- Separation of application from the data
- Separation of "customer" view and "network" view of application
- Reuse of software components

J. Harju et al. (eds.), *Intelligent Networks*

Vendor Independence--By the use of open system concepts and standards GTE seeks to achieve vendor independence at the applications level.

Service Independence--The infrastructure must be kept independent of the services offered so it will not have to be alter for eve service that is developed like today's services.

Separation of "What" from "How" of application--This goal allows GTE to build reusable components without worrying about the underlying platform structure from the application.

Separation of application from data--This goal allows GTE to build applications without regard for embedding data in the application which would require rewriting the application each time data fields changed.

Separation of "customer" and "network" view of the application--This goal allows GTE to write applications so that the customer has a view of the applications that he can relate to and the network has it's view of the application for execution.

As you can seen by these goals GTE is building a very standards based open systems environment that will allow us to accomplish our vision so we can break the main frame life cycle paradigm for applications.

2. PROCESS VIEW OF SERVICE CREATION ENVIRONMENT

Service Creation Environment Function(SCEF)-Provides a work-bench to create AIN services that utilize resources within the GTE environment to produce Service Independent Element(SIE) (Note: difference between SIE and SIB's is that SIE's contain all functions not just call processing)

Service Description(SD)-Ideas for a new service are received from any source and a description is drafted from a customer perspective with the call processing aspects of service behavior. An approach to service pricing and customer billing is proposed and an estimate made of potential market size. Other service descriptions are compared to see if would be a service enhancement or a new service needs to be built. Potential feature interaction screening on a static level will take place here.

Service Specification(SS)-The SS addresses all aspect of service behavior from real-time call processing logic to support issues, service provisioning, maintenance and billing. This service is then viewed from different user perspectives(customer, sales/marketing, service assurance, etc.) which is used to generate a consistent service profile. The service profile is used to specify all views and requirements as part of the SS.

Service Analysis(SA)-A Service Specification/Service Profile is analyzed in detail from a number of different perspectives. The purpose of the analysis is multifold:

- Ensure completeness of SS/SP
- Consider impact of proposed new service on existing, deployed services
- Estimate cost of deploying service
- appreciate impact of Service Deployment upon GTE support structure.

Service Approval(SAP)-At this point in the process flow, a detailed SS/SP has been defined, Service Combination profiles have been generated for the new service executing in combination with existing services, detailed analysis of these profiles are then put forward for management approval.

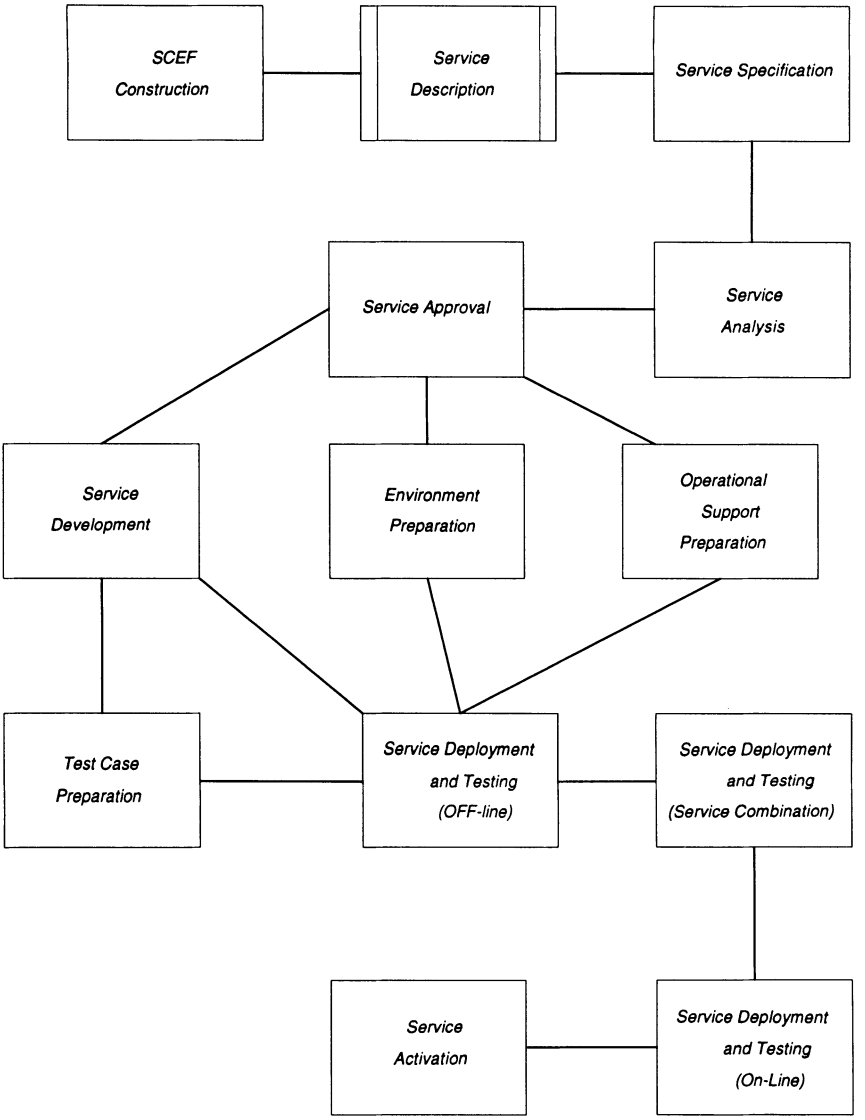

Figure 1. SCE process

Service Development(SD)-After approval, an implementation program has been put into place and now Service Development can take place. Service Development is defined as the process of creating all elements that will run across each of the target platform environments in support of all processes. Part of the SD process includes building test cases for the testing

Environment Preparation(EP)-Prior to deployment of the new service package to the GTE environment several preparation activities must be completed:

- Local Resources are deployed in target environment
- Both the off-line and on-line test environments are prepared for service testing.

Operational Support Preparation(OSP)-The support environment is prepared ahead of time for the introduction of the new AIN service and all outstanding non-technical issues are resolved to ensure that nothing will prevent the timely deployment of the service.

Service Deployment and Testing-Off-Line(SDTOFF)-There are two distinct phases of off-line service testing:

- Service testing in an off-line test environment
- Service Combination testing in an off-line environment

The off-line test environment is a mirror of the physical network elements but isolated in an off-line environment for testing.

Service Combination Deployment and Testing-Off-Line(SCDTOFF)-After testing the service in an stand alone mode it is necessary to test it in combination with other deployed Services for the following reasons:

- Magnitude of feature interactions considered during SS/SA were correctly noted.
- Uncover feature interactions at functional level not discovered during analysis.
- Discover feature interactions resulting directly from structural design of new service.
- Verify correctness of combination packages developed for new services.

Service Deployment and Testing On-Line(SDTON)-The processes involved in the deployment of a service to an on-line environment are essentially the same as for the deployment of that service to an off-line environment, however, the set of test to be performed in the on-line environment represents a small subset of the tests performed in the off-line environment.

Service Activation-Upon successful completion of service testing, the service is formally activated, if it is a revision of a previous service then all the service data, subscription data, and pending updates will be converted. Configuration data will be different for each version of the service and should not be converted.

3.0 TOOL SELECTION PROCESS

The complexity of this environment and what we are trying to build with reuse concepts dictates this development must be an Object Oriented(OO) development. The OO paradigm allows GTE to accommodate a very large development and design effort across multiple target platforms and systems while achieving a high degree of code module reuse.

3.1 OO Analysis and Design Tool

Rumbaugh's Object Modeling Technique(OMT) was chosen as the Object-Oriented analysis(OOA) technique. GTE selected VisualWorks from Martin-Marietta as the initial design tool running on a PC. After initial analysis, the model is being input into Soft Ware

through Pictures(SWP) by IDE. SWP allows us to generate C++ header files that are imported into an Object-Oriented Design Code(OODC) toolset. Rumbaugh's methodology was chosen because most of our designers are familiar with it and our OODC environment has built a direct import interface into SWP. We have also developed extensions to OMT to offset some of the weaknesses in design of the OMT methodology so we can use recursive design with class and class structure charts using state transition diagrams.

3.2 OO Model Code Development

We are taking the C++ header files generated by SWP and then inserting them into a toolset called Teknekron Communications Systems BaseWorx Object Services Package(OSP). OPS features include generating graphical user interfaces, object servers, object communication and distribution, external communications, GUI and managed objects code generators.

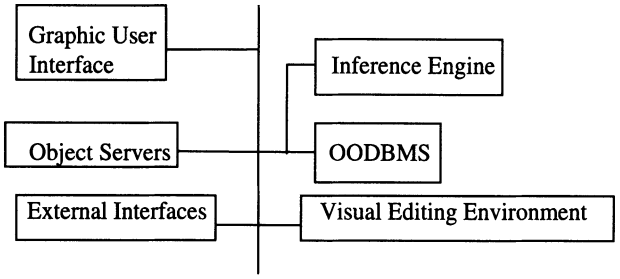

Figure 2. Software Backplane Services

The above listed diagram is a visual representation of the TCSI's OSP core and the elements that the software backplane concept is supporting. The graphical user interface is for creating code that is used by the toolsets not applications. The applications management operations, administration, and management(OAM) is supported in the Object Server Module. For the applications this includes configuration, fault, performance and security management. This server is designed to support both on-line transaction processing systems(OLTP) and network management based systems(CMIP). This approach allows us to build for any type of environment and legacy system that we must support in our applications migration.

The Object Server Module also provides for the interface into the OODBMS which is the Verstant OODBMS.

The communications module provides for message based communications using the communications core platform(CCP), a transaction manager(TM) and rapid transit(RT) paradigms. The object based communications is defined in the applications entity communications(AEC) service which provides access to object within a single process and across process boundaries. The AEC service provides a connection-oriented, asynchronous client-server protocol for intra-process, inter-process and inter-computer communications. Manager/Agent communications is provided via the CCP or CMIP protocol stack and the OSI tools provided by the CMIP stack vendor.

The OO middleware vendor provides the connectivity into the software backplane where my other toolsets are also communicating with each other.

3.3 Object Oriented DataBase(OODB)

In our application we had the following needs that had to be met by our database.

1. The network applications model has complex data structures and relationships that could be captured either via graph structure or object structures.
2. The need of extremely high performance with fine-grained concurrence control.
3. Because this application is mission critical applications must ensure data integrity, be highly available, and support very large databases.
4. The applications operate in geographically distributed environments and require support for distributed databases with the ability to move data across various nodes in the network.
5. The applications require that the DBMS maintain versioning control.
6. Because these applications are event driven, they require the ability to deliver and receive notification of network-wide events.

These were the reasons for choosing an OODB. Versant was selected because they are based around the concept of logical object identifiers(LOIDS). Because an object's LOID in Versant never changes and is never reused, the semantic integrity of database is assured. For transaction integrity, proven techniques of locking, and both physical and logical logging. Versant provides support of 281 Tera-Objects per database and 65,000 databases per network.Versant is also providing the tools to support a 365x24 operation without taking a database of line for backups and keeping the database integrity within the network.

3.4 Inference Engine(IE)

Due to the complex nature of the applications that the SCE is writing software for the ability of having applications make decisions based upon business or process rules is an absolute requirement. Also the ability of feeding back information from the applications will allow us to change business requirements only once and they will be propagated through out the systems as required.

Because of these requirements the tool ART-IM from Inference Corporation was chosen to provide these features as the software is generated and then distributed across the run-time environment. During the OO Design process business rules will be identified and process rules will be identified and then distributed across the run time environment.

3.5 Visual Programming Environment(VPE)

The use of a VPE for the SCE is a mandatory requirement. The ability of using this environment for programmers that are not fully trained is a principal that has been established by the business. Under current evaluation is the Tandem Service Creation Environment (SCE) 2000 to fulfill this role.

The SCE2000 consists of a programmer's workbench, the service designer's visual service editor, and a third layer a service provisioning environment. The SCE2000 workbench provides a structured means of managing the creation of service primitives(discrete operational components). Currently written in GNU Perl it's now being ported to GNU C++. The workbench is capable of producing API's, state machines, switch triggers, or downloads to a

batch environment. The SCE2000 relies on the GNU rcs product to manage source files and also provides for distributed management of primitives by allowing for the local and remote generation through the use of makefiles.

The SCE2000 workbench employs revision control to provide a layered philosophy of subsystems, programs, and modules. The subsystem would contain a simple service comprised of programs. From a control standpoint root would be created to own the subsystem. For each new subsystem root privilege would be required. Subsystem/Programs naming follow standard UNIX conventions. Programs represent lower-level services required to create a subsystem and programs are comprised of modules.

Modules(primitives are equivalent to Service Independent Building Blocks(SIBB), and constitute the lowest-level of functionality possible in the system. Primitives are implemented using C++ classes are wrappers, their functionality comes from C/C++ source code compiled and made available in pseudo-library format. Programmers are responsible for primitives, however, check-out occurs at the program level in order to access and/ or modify primitives.

Primitives provide the lowest level of functionality in the SCE, with a web of networked primitives constituting a service. The workbench environment provides a pre-programmed logic that eases the integration of primitives. A menu option will create a primitive template file(PTF).

Primitive Template Files are basically interface definitions for a primitive. This is used both when the primitive is generated and when the primitive object is used in the service design stage. When the primitive is defined a secondary file containing the primitive's icon representation in bitmap format is also created. This bitmap file contains a default bitmap and can be modified by service programmer.

PTF files define the number and types of the primitive's input and output parameters, as well as the names of the primitive's successors. Parameters can be defined using pre-defined data types, or user-defined data type can be created for the primitive(structures or blobs).

Definition and usage of the user-defined data types requires direct modification of source code files. Primitive successors are essentially valid pathways that can be traversed once the primitive has completed processing. These pathways are specified simply by providing the names of other primitives.

WorkBench also creates Service template files, which define the connections between the primitives. These files allow changes to be made to the Service without requiring a recompile of the entire system. Such changes are limited, however, to those involving the logical connections between primitives.

Wrapper code is generated from the SCE2000 workbench and consists of two C++ source code files, one header file and one C source code file. Tandem will be changing the entire system to C++ to facilitate inheritance.

Within the service design environment primitives are plugged-in to create a service. Tandem is providing a level of integration so that you will be able to modify a primitive and then dynamically refresh the design environment. Next is the Visual Service Environment(VSE).

The VSE is an object-based service definition system. Using the primitive components developed under the SCE2000 workbench, services can be created by dragging and dropping primitives from one or more palettes, and then connecting these new service components to define both logic flow and data pathways. Palettes are where primitives can be grouped together and one or more palettes being available for selection for each service created. This

is modeled after the standard UNIX library convention. Primitive palette windows work cooperatively with the service creation window to provide drag-and-drop capabilities. You cannot drag-and-drop between palettes yet. Blocks are a complexity management tool and exist at the same level as services, and can house any combination of Service, Primitive or blocks. Each block can contain up to 10,000 components and the entire system supports up to 10,000 level of nesting.

As you can now see why the tool sets were chosen and how GTE is looking to integrate them to provide a comprehensive environment for our programming staff. Now let's look at the physical implementation of the SCE process and how GTE is going to automate the Service Specification Tool.

4. SCE LOGICAL IMPLEMENTATION

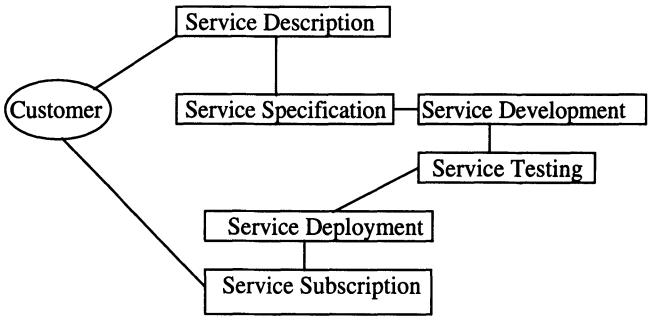

Figure 3. Logical SCE Diagram

Customer-This customer gives a service idea to a GTE representative as a set of capabilities.

Service Description-This is the first pass at capturing the idea of the customer and putting into graphical/textual description from a customer viewpoint.

Service Specification-This is the second pass at the service description putting it into the network view point to build.

Service Development-Coding the service out of the Service Independent Elements.

Service Testing-Testing the service against the customer and network criteria.

Service Deployment-When a service is deployed within the network ready for subscription.

Service Subscription-The service is network ready for the customer to subscribe , use and be billed for it.

4.1 Service Specification Tool

The service specification tool need to be built so that it can extract information from our corporate data bases and build a customer profile from the information present in the GTE databases. Then the profile can be run against an AI matrix which contains the capabilities sets

of the Intelligent Network. Then a list of capabilities questions can be generated for the SS tool. The sales engineer can then take the tool and work with the communications manager to test the capability concepts for his network. After the questions are answered then an analysis is done against the capabilities and service suggestions would be made.

The Sales Engineer(SE) would then be able to visually present service concepts to the customer and refine them there. Upon agreement, the SE would bring the model back to feed into the SCE process. All views of the service could then be built with the customer profile as the baseline information for the SCE. With a generated model looking at all of the views necessary for the SCE and rules applied from the inference engine then most of the code could be generated automatically.

5. CONCLUSIONS

Because we have looked at the Intelligent Network as an end-to-end process, GTE is developing the integration of the SCE/SMS necessary so that we can overcome the deficiencies of our legacy systems. We are migrating to a new systems environment because of Process Re-engineering. This has given us the capabilities of using corporate data as an asset by removing corporate data from legacy applications. We are building an object view which looks at each process from it's unique view point and are building capabilities and rules to accomplish this. The end-to-end process view allows GTE to build an infrastructure that allows to build our products and services in a very rapid manner with substantially less cost then our previous environment.