Embedded AI for Sales-Service Negotiation

Mike Carr, Chris Costello, Karen McDonald, Debbie Cherubino
Bell Atlantic Corporation
1717 Arch Street
Philadelphia, Pennsylvania 19103

Pamela Trusal Kemper
Inference Corporation
550 North Continental Boulevard
El Segundo, California 90245

Introduction

Bell Atlantic Service Representatives are the primary interface between the company and its many customers. They are responsible for negotiating orders to establish, transfer or disconnect telephone service as well as answer questions and process changes to a customer's service or billing account. To support this wide variety of tasks, the Service Reps use a number of computer applications, mostly hosted on mainframe systems. Each application has it's own arcane language for data entry and retrieval which must be mastered by the user. In addition, large amounts of required information, such as rates, service restrictions, etc. are documented on paper and must be memorized or "looked up" during the customer contact.

This information overload has a number of undesirable consequences. First, as Bell Atlantic increases the type and number of services it offers, it is increasingly difficult for a Service Rep to keep up with all of the information required to both provide customer service and to proactively present and sell Bell Atlantic's services. In addition, the complications of accessing large quantities of external data throughout the customer contact negatively impacts the length of the contact. This inconveniences the customer and further limits the Rep's ability to provide accurate and efficient service. Finally, the training necessary to provide the Service Reps with the tools they need to navigate this sea of information has reached unacceptable levels.

In order to address these problems, Bell Atlantic has developed a distributed software application called the SaleService Negotiation System (SSNS.) The goals of the system include:

- improvement of accuracy and efficiency
- automated support of needs-based selling
- increased sales efficiency
- decreased training requirements
- optimization of customer contact time

In addition, it is critical that any new automated system be able to address both future-oriented technology and the current mainframe-based systems which will continue to perform the bulk of Bell Atlantic data processing for the near term.

SSNS System Overview

The SSNS system provides a number of high-level functions to support it's stated goals. These functions include:

- A **Graphical User Interface** utilizing both menu-based and English-language order input. The interface process prompts the Rep, based on the geographical location of the customer, for all required customer data that must be obtained to place an order while enforcing all relevant service restrictions.
- **Mechanized reference information,** including rate calculations, on request throughout the contact.
- An **automated credit check** based on data obtained during the contact from both external sources as well as Bell Atlantic customer records.
- A **SaleService tool** which suggests appropriate Bell Atlantic products based on customer data,
- A **Service Request translator** which translates and transmits the resulting customer order data in the language and format required by the existing downstream Service Order Processors.

Figure 1 illustrates the major SSNS functions and their relationships to one another.

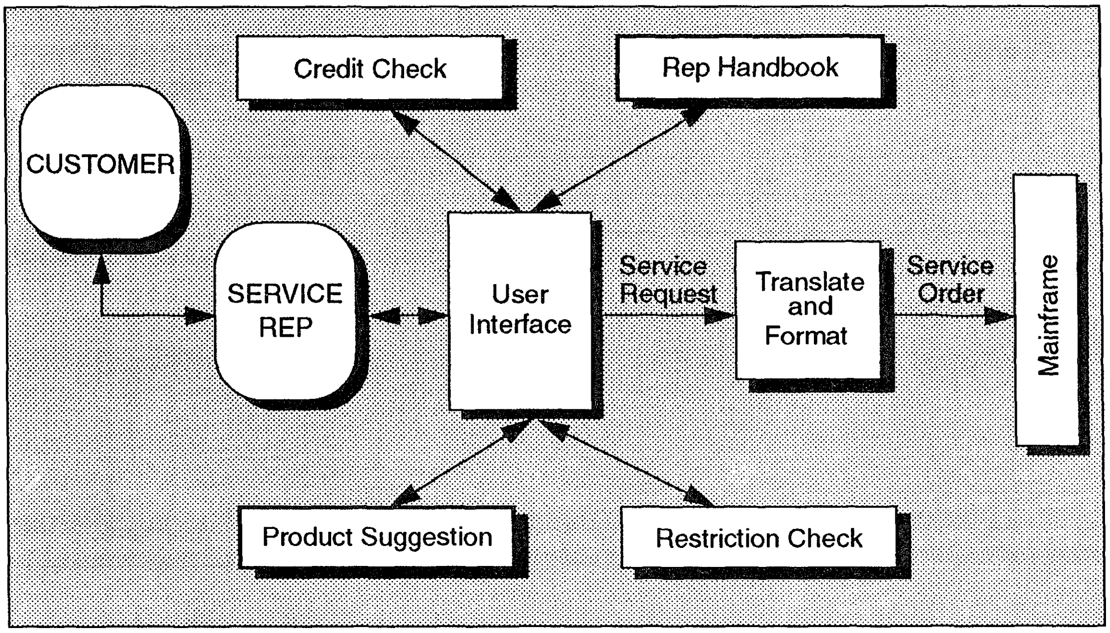

Figure 1

SSNS utilizes an object-oriented architecture where each portion of application code which performs a distinct function is bundled with relevant input and output data structures into a building block. Building blocks have no knowledge of one another's internal structure and communicate solely by passing messages. System building blocks are organized into three layers: a User Layer; Process Layer and Data layer which, along with a set of infrastructure services loosely referred to as the Contract Manager, make up the SSNS system.

This object-oriented architecture provides a number of important benefits. First, the usual benefits of an object-oriented system, such as maintainability, extensibility, etc. are realized. In addition, it provides an extremely clean and straight-forward method for embedding various software technologies, including AI-based functions within a larger conventional application.

AI in SSNS

SSNS is the first application developed and deployed within Bell Atlantic to use AI technology on a production basis, and is among the first among the Regional Bell Operating Companies to embed it in a large conventional system. A number of the SSNS functions have been implemented using AI technology, specifically knowledge-based systems. These are shown in bold in Figure 2, and include:

- RIDS - This process receives data throughout the customer contact from the User Interface and returns messages concerning any product **R**estrictions, **I**nteractions or **D**ependencies which may apply.
- CREDIT TOOL - This process provides automated credit checking based on customer credit information. **Deployment planned for 3rd quarter 1994.**
- CSOP - At the conclusion of the customer contact, this process takes the set of English-language Service Request data and translates it into the appropriate Service order language, then formats it for the mainframe-based Service Order Processors.Although each of the functions differ somewhat they have a number of common features that lend them to a knowledge-based implementation.

First, each of the functions is data-driven. In the case of RIDS, the particular customer data or set of products is developed gradually throughout the contact and can be changed at any point in a number of ways. The analysis needs to respond immediately as the data changes. In the case of CSOP, the data-driven nature is less obvious, since it is a backend batch process. However, the task of efficiently building a Service Order out of almost infinite subsets of possible data benefits greatly from a data-driven approach. Only the appropriate data is examined for translation and processed during formatting.

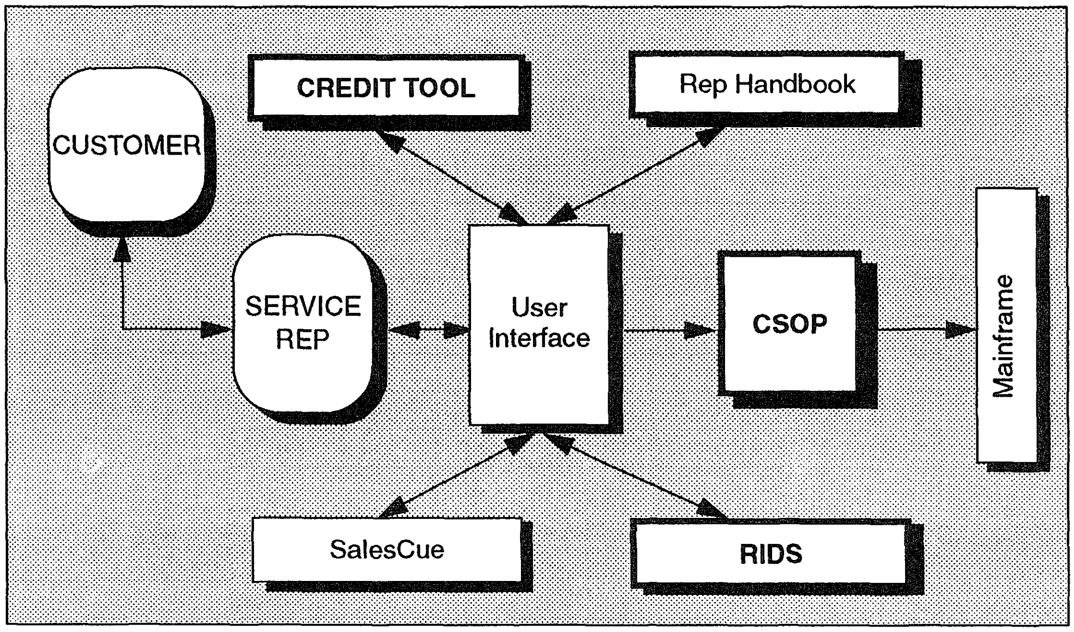

Figure 2

Second, the knowledge required to perform each of the functions is constantly changing to greater or lesser degrees. By explicitly representing product restrictions, data translations and Service Order Language variables as rules or objects, system maintenance is greatly simplified, and in some cases automated.

Finally, the system must be easily extensible. SSNS implementation plans call for a phased introduction over a number of years. Various order types: New Connect, Disconnect, Transfer, etc. will be added gradually, as will the geographic areas that the system covers. This means that changes in product restrictions, credit guidelines and Service Order language and format, which all differ across the Bell Atlantic region, will have to be incorporated seamlessly without significant changes to system design.

RIDS

As the Service Rep negotiates a Service Request with the customer, products such as Call Waiting or Answer Call are requested by the customer. The Service Rep then selects the product from the appropriate window and the data is immediately passed from the User Interface to the RIDS process running in the background. If any restrictions, interactions or dependencies are relevant to the product selected, a message is returned to the User Interface, e.g., "You have selected Answer Call; Answer Call requires Call Forwarding" and displayed in a pop-up window. Otherwise RIDs waits for the next piece of data. If the customer makes a change and "deselects" a product, that information is also passed to the RIDS process, where the absence of a product may also trigger (generally dependency) rules.

The RIDS process consists of 5 functional modules, shown in Figure 3.RIDs is implemented using a modified rule-based approach. This approach utilizes a schema system for declarative representation of application knowledge and a limited number of generic rules to select and process that knowledge, rather than using rules to both represent and apply application knowledge. In addition to the rule-based analysis, the RIDs process utilizes a series of procedural "wrappers" to assert and extract data into and out of the knowledge base.

RIDS Knowledge Base

The RIDS knowledge base is used to store declarative knowledge about product restrictions, interactions and dependencies, as well as incoming product selection or deselection data. As all restrictions, interactions and dependencies among Bell Atlantic products are considered to be corporate data, they are stored and maintained in corporate database tables. Rather than duplicate this information by hand in the knowledge base raising potential inconsistency issues, the majority of the information in the RIDS knowledge base is downloaded from these corporate database tables each time an SSNS session is started. When the RIDs process is initially

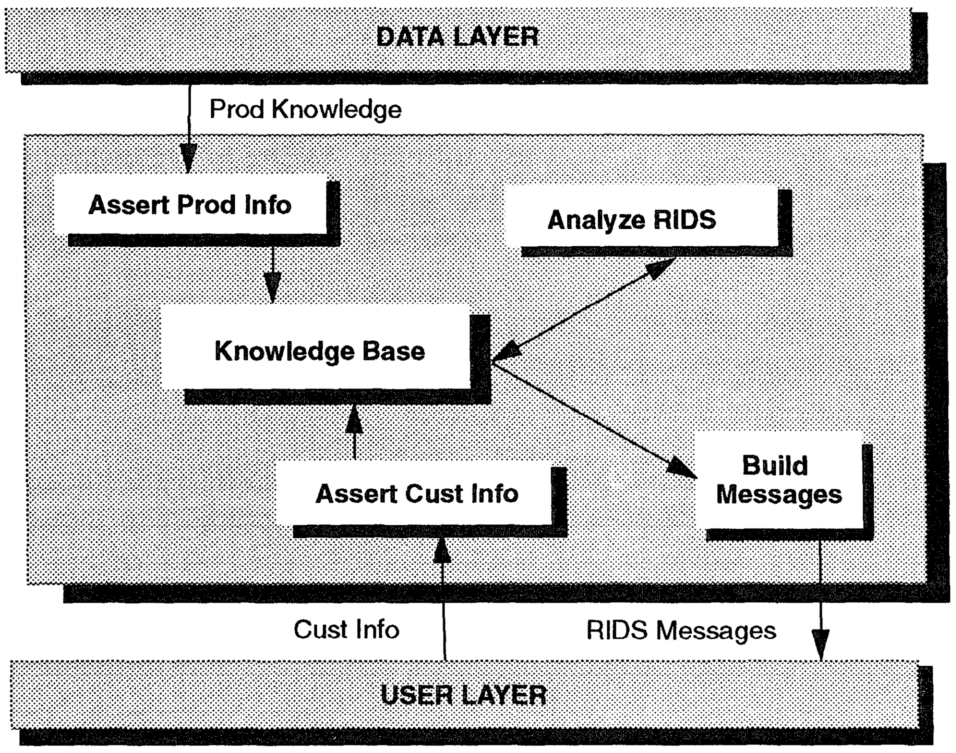

Figure 3

invoked, the knowledge base contains only a limited number of definition-type schemas noted below. (A instance of each type is shown for clarity)

Prod-knowledge. Prod-knowledge schemas contain the restriction, interaction and dependency knowledge organized on a product by product basis.

```
(Defschema prod-AC
    (instance-of prod-knowledge)
    (prod-id 1)
    (prod-name Answer Call)
    (restricts prod-UF)
    (interacts prod-CFV)
    (depends node-1)
```

The restricts and interacts slots will reference other prod-knowledge schemas, while the depends slot will reference node schemas.

Node. In some cases product dependencies involve more than one other product and may be AND or OR relationships. For example: Answer Call requires Call Waiting AND either Touchtone OR a tone signalling set. To address this, product dependencies are modelled using a set (or tree) of dependency nodes. Each node is either complex, containing children that may themselves be nodes, or atomic, containing a single product. Complex nodes also identify the relationship (AND/OR) among their children. Each dependency node is represented by a schema with the tree implicit in the relationships.

```
(Defschema node-1
    (node-id 1)
    (node-description AC-depends)
    (left-child node-2)
    (right-child node-3)
    (node-relation AND)
    (atomic-node-type complex))
```

Cust-info. Each time a product is selected (or de-selected) by the Service Rep, information is sent to RIDS and stored in a schema.

```
(Defschema cust-info-1
    (state select)
    (time-stamp 00123)
    (contact-id cust-1)
    (prod-id 1))
```

These schemas are used by the Assert Prod Info and Assert Cust Info functions as templates for further knowledge base population. Both Assert Prod Info and Assert Cust Info are strictly procedural processes.

Assert Prod Info

Assert Prod Info begins by sending a request-for-data message to the System Contract Manager which in turn invokes the appropriate data layer building block (DLBB). The DLBB queries the appropriate database tables and returns a TVO[1] containing the data. The data is "massaged" by a series of functions which generate the format represented by the prod-knowledge and node schemas and assert the data into the knowledge base as instances of those schemas. After the download function is completed, the RIDs process is available to receive User Layer messages.

Assert Cust Info

Each time a product is selected, the data is passed to RIDs, in TVO form, by the User Interface. The Assert Cust Info function reads the TVO and asserts a schema into the knowledge base as an instance of the cust-info schema. If a schema representing that product selection already exists (e.g. a product has been previously selected but is now de-selected), then its values will be updated with the new data. If it does not yet exist, it will be created dynamically and populated.

Analyze RIDS

RIDS processing is accomplished by a series of rules which match the incoming data stored in cust-info schemas with the restriction, interaction and dependency information stored in prod-info schemas. As noted earlier, the rules are generic containing no product-specific information, but match on prod-knowledge and cust-info schemas as appropriate. Each of the rules generates an error message identifying the type of restriction, interaction or dependency and the product or products affected.

Restrictions/Interactions. Restriction and interaction rules are triggered by the presence of two cust-info schemas (or product selections), where a prod-knowledge schema shows the first product in the restriction or interaction slot of the other. (Question marks indicate variable names.)

1. In SSNS all data is passed in C data structures known as TAG-VALUE-OBJECTs or TVOs. Within the hierarchical TVO structure, each individual piece of data has a name (or tag) and a value. Utilities to read from and write to TVOs are provided by the SSNS infrastructure services.

```
(Defrule prod-prod-restricts
"This rule prevents the selection of two products one of
which restricts the other"
    (schema ?cust-select1
        (prod-id ?id1))
    (schema ?cust-select2
        (prod-id ?id2))
    (schema ?prod-knowledge-A
        (prod-id ?id1)
        (prod-name ?name1)
        (interacts ?Prod-knowledge-B))
    (schema ?Prod-knowledge-B
        (prod-id ?id2))
        (prod-name ?name2)
        =>

(generate-error-message Interation1 ?name1 ?name2))
```

For example the customer requests Answer Call and Call Forwarding-Variable. The interaction rule is processed as:

```
(Defrule prod-prod-restricts
"This rule prevents the selection of two products one of
which restricts the other"
    (schema Prod-knowledge-AC
        (prod-name Answer-Call)
        (prod-id 1)
        (interacts Prod-knowledge-CFV))
    (schema Prod-knowledge-CFV
        (prod-name Call-Forwarding)
        (prod-id 12))
    (schema Cust-selects-1
        (prod-id 1))
    (schema Cust-selects-2
        (prod-id 12))
    =>

(generate-error-message Interaction1 Answer-Call Call-
Forwarding-Variable))
```

Dependencies. Dependencies are implemented by a series of rules. An initial rule is triggered by the presence of a customer selection which matches a prod-knowledge schema with a depends slot containing a node-id. Additional rules then perform a tree walk to find all possible dependencies, then walk back up the tree checking for the absence of customer selections which match the products represented by each node to avoid returning a dependency that has coincidentally satisfied. The final result is the generation of one message containing only the dependencies that were not satisfied. Although a complete example is beyond the scope of this paper, the generation of dependencies is accomplished as follows:

```
(Defrule start-looking
"Based on a product selection, this rule locates the root
node of the dependency tree and asserts that it is required."
    (schema ?prod-knowledge-A
        (prod-name ?name1)
        (prod-id ?id1)
        (depends ?node1))
    (schema ?cust-info1
        (prod-id ?id1))
=>
(assert (need-prod-node ?node1)))

(Defrule generate-required-node
"This rule takes a required prod node and asserts that each
of it's children is also required."
    (need-prod-node ?node1)
    (schema ?node1
        (node-type complex)
        (left-child ?node2)
        (right-child ?node3)
=>
(assert (need-prod-node ?node2))
(assert (need-prod--node ?node3)))
```

Process Output

Process Output is a series of procedural functions which extract error messages from the knowledge base and write them to a TVO in order to pass them back to the User Interface.

CSOP

At the end of the customer contact, the Service Request containing all data required to produce a Service Order is sent by the User Interface process to the CSOP process. A typical residence service request may contain 200-300 pieces of data with more complex requests containing 800-1000.

First, the English-like Service Request data must be translated into Service Order Language. Service Order Language consists of FIDs (Field Identifiers) and USOCs (Universal Service Order Codes) with their corresponding values. For example, if premises visit charges are to be waived the FID RMK with the value "WHR" should be placed on the service order. Or often less clear, if the customer is a Mary Baldwin student living off campus, the FID ZST (indicating student customer) with value O,MBC (off-campus, Mary Baldwin College) should be placed on the service order.

Then the appropriate FID and USOC data must be assembled and formatted to produce a Service Order that can be further processed by the existing mainframe-based Service Order Processors (SOPs). Although SSNS provides a single user interface throughout the Bell Atlantic region resulting in a uniform Service Request, each of the operating companies has its own SOP with its own unique Service Order Language and format, further complicating the CSOP task.

Finally the Service Order is sent to a data layer process to be relayed to the appropriate SOP.

The CSOP process consists of 5 functional modules, shown in Figure 4.

CSOP is implemented using a variety of reasoning techniques, both object-oriented and rule-based as well as a series of procedural "wrappers" to assert and extract data into and out of the knowledge base. As described in the RIDs process, a schema system is utilized for declarative representation of application knowledge while generic rules to are used to select and process that knowledge. In CSOP, however, the selection of appropriate knowledge is accomplished via rule-based pattern-matching, but often the processing of that knowledge is accomplished by methods attached to the schemas, and triggered by message-passing from the right-hand side of the rules. This combination of reasoning approaches allows the knowledge to remain highly segmented and very easily maintained. CSOP also utilizes a wrapper process similar to that previously described to assert and extract information into and out of the knowledge base.

CSOP Knowledge Base

The CSOP knowledge base is used to store both the incoming Service Request data as well as knowledge about the Service Order language and format into which the data is to be transformed. This information is stored in three types of schemas: tag, service order language, and service order template.

Tag. As noted earlier, all SSNS data including Service Request data is passed by tag name and value. The CSOP knowledge base contains a set of schemas, each of which corresponds to one of the possible tags that may be part of a service request.

If a given tag value must be translated, information about either the new value to be assigned, or how to obtain that new value (e.g., perform a calculation) is also stored in that schema. If data from more than one tag schema is required for a translation, the name(s) of those schemas are included in the translation information along with a target schema name to hold the combined data.

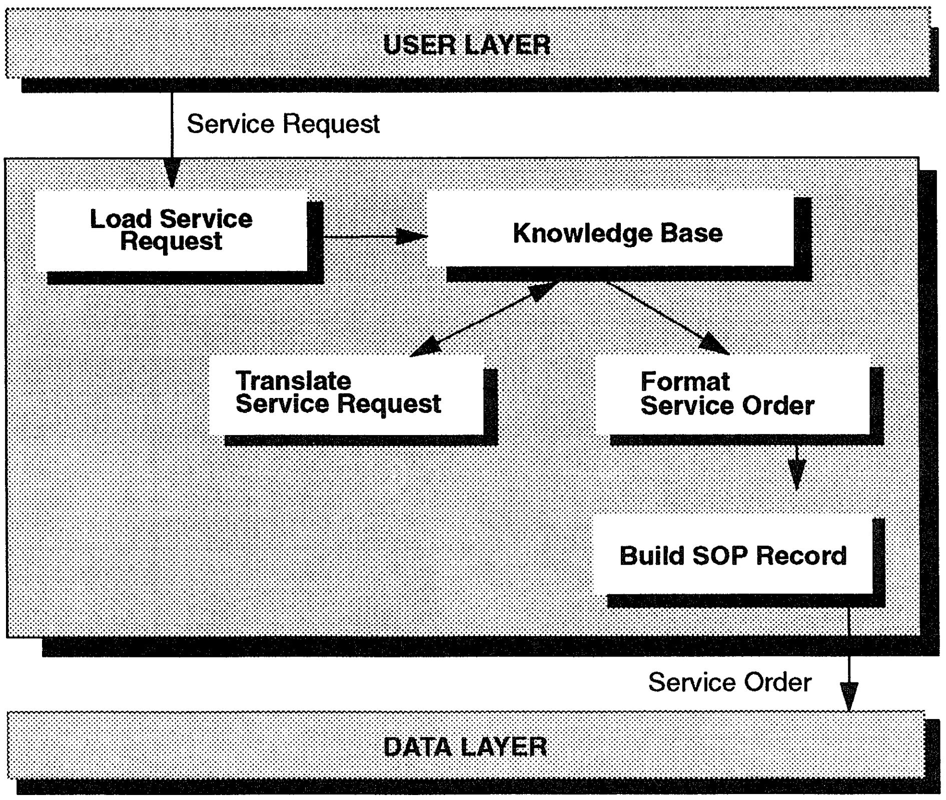

Figure 4

```
(Defschema cust-tel-nbr
    (instance-of multi-occr-tag)
    (tag-name SVC_RQ.SVC_RQ_CUST.
            CUST_POC.TEL_NBR)
    (value)
    (value-format format-npa-nxx))
```

Service Order Language. The knowledge base contains a schema for each FID or USOC that may be used in a service order. The schemas contain formatting parameters along with relational links to the tag schemas that will contain their values and information about how the value should be processed.

```
(Defschema LN "Listing Name"
    (instance-of left-handed-FID)
    (label-start 2)
    (label "LN")
    (get-value concat-value)
    (value-start 6)
    (values (d-listing-last-name d-listing-first-name
    d-listing-middle-name)))
```

Service Order Template. The service order template represents the service order in terms of sections, and each FID/USOC that may appear in each section. The template reflects the order of the sections and the FID/USOCs within each section.

```
(Defschema service-order
    (has-sections (ident-section listing-section billing-
    section)))
    (defschema billing-section
    (has-entries (bill-name bill-address number-of-
    bills)))
    (defschema bill-name
    (has-FIDS (BN1, BN2, BN3)))
```

Load Service Request

The Load Service Request function is a procedural function which processes the incoming TVO which contains the Service Request data and asserts it into the knowledge base. As each individual tag is read, the schema name corresponding to that tag is located in a hash table of tag-schema pairs (generated automatically from the knowledge base code at system start-up) and the tag value is placed into the schema. If the tag can occur more than once with different values a child schema is created for each instance and the value stored in that schema. When the entire TVO has been processed, the rule-based translation process is invoked.

Translate Service Request

The translation function incorporates both object-oriented and rule-based processing. As in the RIDS process, translation rules are generic. They contain no specific translation information but rely on pattern-matching to both respond to incoming data and locate appropriate translation information along with any additional data required. Methods are invoked on the right-hand side of the rules and are used to perform data manipulation such as calculations or format changes, as well as any knowledge base update resulting from the translation.

For example, the following rule is triggered by the presence of two tag values. The first tag can have any value, the second tag must have the name and value specifically stored in the eval-transform slot of the first. Messages are then sent, first to tag1 to execute a particular function which manipulates its value replacing it in the value slot, then to the same tag to pass its new value to its target schema.

```
defrule eval-transform
(schema ?tag1
    (value ?v1)
    (eval-transform ?tag2 ?value2 ?target)
    (eval-function ?function1))
(schema ?tag2
    (value ?value2))
=>
(send ?tag1 ?function1)
(send ?tag1 put-new-value))
```

When all translations have completed, the format function is invoked.

Format Service Order

Format Service Order uses a series of procedural functions to place the appropriate FID/USOCs in the correct position on the service order, represented by an output stream. These functions "walk" the service order template as represented in the knowledge base, utilizing the relational links between the FIDs and the incoming tag data to see if each FID should appear on the order. (i.e., the appropriate tag(s) have current values.) If so, a message is sent to the FID to write its label and value on the Service Order output stream using the placement information contained within it. If the FID has no data associated with it is skipped and the process continues until the complete template has been processed.

Build SOP Record

For efficiency, the formatted Service Order is finally placed in a record format understandable by the mainframe SOP. No additional data fomatting is done, the stream is simply broken into pages with header information associated with each.

Application Implementation

SSNS is a distributed system, utilizing a number of software and hardware technologies. The individual User Layer processes are implemented in MOTIF and hosted on SUN workstations. Infrastructure services are implemented in C and are compiled into each individual building block. Process layer building blocks (other than CSOP and RIDS) are also implemented in C and are hosted (including CSOP and RIDS) on a SUN server. Sybase is used for data storage and access. The Sybase process and related Data Layer building blocks are hosted on an additional SUN server.

The RIDS and CSOP building blocks contain both knowledge-based and conventional subprocesses. The knowledge-based subprocesses, such as RIDS Analyze, and Translate Service Request, along with their associated knowledge bases were developed primarily in the ART-IM/UNIX language. Data input and output processes, such as Assert Cust Info or Load Service Request, and Build RIDS messages or Build STI Record were implemented directly in C. The C-based I/O processes called SSNS infrastructure services as required. In addition, a number of C functions were written to perform data manipulations more appropriate to that language such as string parsing. These functions were called directly from ART-IM rules, or invoked by messages. Both C code and ART-IM code are compiled into the application executable and can be called either internally from ART-IM, for development purposes, or externally by the SSNS architecture, during deployment.

The ability of ART-IM to provide a clean interface between its internal language and C was critical to allowing CSOP and RIDS to be embedded transparently in SSNS.

Application Development

SSNS development began in July of 1991 with an eventual staff of approximately 45 software developers, 20 business requirements analysts, and 10 deployment and testing specialists along with a number of support personnel. CSOP development began with a prototyping effort using two developers. Full system development followed in March of 1992 with a development staff of five. The RIDS development effort was accomplished in parallel to the full system effort, utilizing two of the five developers as required.

SSNS was (and is being) developed according to the Bell Atlantic Software Engineering (BASE) methodology. This methodology adheres to current practices in software engineering and includes the following phases: Project Planning, Requirements Definition, General and Detailed Design, Software Development, Testing, Quality and Configuration Management, Installation, Maintenance and Documentation. The development of the AI subprocesses, although historically an activity often hostile to Software Engineering Methodology, followed BASE as well. Some development deliverables such as function call-trees, or structured english were replaced by more "AI-like" deliverables, such as object hierarchies, or knowledge representation schemes, but the majority of the development processes were compatible.

There are a number of phased targets throughout the overall development. SSNS Versions 1.0 and 1.2 address a single order type, residence New Connect, for the PA rate jurisdiction only. Current development, SSNS Version 1.5, is focused on software that will be deployed throughout the region for residence New Connect order types. Future development phases address the spectrum of order types as well as Bell Atlantic Business Customers.

Application Deployment

The deployment of each version of SSNS follows a significant period of system and user acceptance testing in-house. Testing is accomplished using approximately 30 scripts which are executed manually by representatives from the Business organizations. Once software is installed in the business office a period of limited use precedes general release throughout the business office.

Initial deployment of SSNS Version 1.0 began in October 1992 on a limited basis in one business office with additional rollouts following hardware deployment throughout PA. Deployment of Version 1.5, planned to begin later this year will result ultimately in approximately 7500 Service Reps issuing approximately 150,000 New Connect orders per month.

Significant payoff has already been realized from SSNS. Based on a sampling conducted in late 1993 comparing approximately 100 Service Reps using the system vs. 100 Service Reps using traditional methods a 29% increase in revenue per contact was realized. The rate of errors which can significantly extend the overall time to complete a New Connect order have also been lowered.

Application Maintenance

The SSNS application is currently being maintained by SSNS team members. Whenever possible, system maintenance has been automated through the use of specialized utilities. End-user maintenance is not a goal at this time. In the future, system maintenance responsibility will be assumed by a specialized Application Administration function within the Information Systems group.

CSOP

Although not on a frequent basis, additional FIDs and USOCs are added to the Service Order requirements, while the incoming Service Request data may change due to changes in User Layer requirements. Approximately 1-10 FID/USOC changes are seen over the course of a year, each with lead time on the order of several weeks to several months. User Layer changes are less frequent than that, and are generally driven by the same changes in Service Order requirements.

The maintenance process is supported both implicitly by CSOP design and explicitly with a number of utilities. The design use of a limited number of generic rules with the majority of knowledge represented declaratively in schemas, as well as the separation of translation and formatting knowledge provides for straight-forward maintenance. Over the last year, most changes have been made to a limited set of the knowledge base schemas. Changes to the rules have been, and are expected to be extremely infrequent. Utilities such as the automatic generation of hash tables from the knowledge base and automated unit test facilities have also been developed to support system maintenance.

RIDS

RIDS knowledge is more likely to change than any in the system. Bell Atlantic offers new products with associated restrictions, interactions and dependencies frequently. Therefore, in addition to the design support mentioned above, the RIDS process has automated knowledge-base maintenance.

The use of a database download to build the majority of the knowledge-base on system start-up eliminates the need for most maintenance activity. The database tables used to store the restriction, interaction and dependency information are maintained as part of the overall corporate data maintenance process and are relied upon to be accurate. Any changes to the actual structure of the corporate data, which would affect the data massage procedures within RIDS, are possible but expected to be extremely infrequent. Changes to the RIDS rules are also expected to be infrequent. This download process also circumvents the need for building block recompilation and potential SSNS redeployment due simply to changes in RIDS knowledge.

Overall, maintenance experience over the last year has been extremely positive. CSOP (and RIDS) have among the best maintenance records in the overall SSNS system. In fact, the majority of system maintenance is performed by team members new to both AI techniques and ART-IM.

Summary

The use of AI-based technology within Bell Atlantic's SSNS has been a success. Embedding the technology within the larger system has been a straightforward process. CSOP and RIDS building blocks function like any other in the system requiring no special data communications or infrastructure service support. Maintaining and extending the processes has been accomplished with relative ease. In particular, the use of automated knowledge-base generation such as that used by RIDS eliminates a large part of the maintenance burden. Finally, and most importantly, functional goals have been achieved. The deployment of a new technology to support Service Representative activity has been accomplished with significant benefit to the company, and little or no impact to the remainder of the Bell Atlantic processes.

Based on the success of the current AI development, the use of the technology is being planned for future processes that were initially targeted for other methods of implementation.