Advanced fulfillment capabilities enhanced with technology agnostic provisioning control resource components

Damir Medved[1], Hrvoje Tutman[1], Dominik Periškić[2], Dr. August-Wilhelm Jagau[3]

[1]GDi GISDATA, R &D Department, Petra Kobeka 13/1, HR-51000 Rijeka, Croatia

[2]Hrvatski Telekom d. d., IT Department, Kupska 2, HR-10000 Zagreb, Croatia

[3]Telcordia Germany GmbH, Herriotstrasse 1, 60598 Frankfurt am Main, Germany

damir.medved@gisdata.com, hrvoje.tutman@gisdata.com, dominik.periskic@t.ht.hr, ajagau@telcordia.com

Abstract

Nowadays telecom operators are working in a complex and demanding environment characterized by the notion of *pervasive computing*. This fact has a tremendous impact on the operations of established operators. A unified approach to the delivery of products at any time and over any media (fiber, copper, wireless) requires a vastly different approach to customer order fulfillment.

The Key question is: is it possible to have a unified approach to this process by building a finite number of modular and flexible "atomic" functional modules that will provide elementary functions for all operations necessary within the fulfillment domain?

During the last decade one of important prerequisite for such an approach was achieved: the creation of a unified *service and resource inventory* on the basis of a common approach to the modeling of different technologies. Now it is possible "find" components of a solution in a technology agnostic way - and this is basis for the introduction of what are called Provisioning Control Resource Components (PCRC).

This paper describes key aspects of improving the Croatian Telekom NGOSS infra-structure with new capabilities introduced by the COMPASS project. Capabilities which will enhance the customer experience and allow Croatian Telecom to maintain a leading technological position in the Croatian market. The COMPASS project brought a new approach to Croatian Telekom order management: Catalog Driven Order Management and Provisioning Control Resource Components.

1. Introduction

An optimally designed and rigorously defined Product-Service-Resource (PSR) model is of great importance for both the efficient and rapid introduction of new services and the evolution the existing ones. It is very important to ensure that the model on the one hand is simple and on the other portrays efficiently and accurately the technical capabilities of the underlying telecommunications infrastructure. Another important objective is the optimization of the business processes through which telecom operators meet the demands of users for new product and services.

GDi and HT teams have conducted an analysis and assessment of the implementation of generic provisioning control resource components with the assumptions and requirements in mind that need to be met by typical telecom operators. This primarily refers to the fact that the model proposed for resource components will fully capture the required behavior (multiple uses, ability to be included in complex products, etc.). Further, the modeling and likely changing of resource components in the future should be feasible in a sufficiently simple and rapid manner.

The paper will provide an overview of the basic principles underlying the creation of PCRS components on the basis of existing recommendations (reference list attached). And it will review the most important of the applicable governance standards.

Subsequently the paper will provide an overview and description of the properties of the resource components that are needed to meet typical formal requirements occurring in a telecommunications network. With a focus on technological independence, the paper will describe the advantages and shortcomings of the components. It will also provide examples in which they are used.

The small number of required properties and the assumptions that are associated with each particular resource component model will be discussed also.

Finally, using a practical example key concepts will be explained that demonstrate the functionality and the effectiveness of using the selected PCRC-a. In this context the exploration of efficiencies or inefficiencies when using the component method for aggregating resource components in complex product structures will be considered.

2. COMPASS Architecture

The goal of the COMPASS project [9, 10] was the introduction of systems that enable a faster response to changing HT business requirements in terms of business process improvements and associated implementations. COMPASS had its focus on the order fulfillment processes for mass market services. As one of its consequences it was necessary to introduce new functionalities so as to allow a large degree of reuse for the introduction of new products or the changing of existing ones. The end result yielded the desired faster time-to-market.

Considering the OSS angle, the solution had to provide all functions needed in support of any kind of MACD operations on service orders coming from the CRM system. This yielded an architecture consisting of the following building blocks:

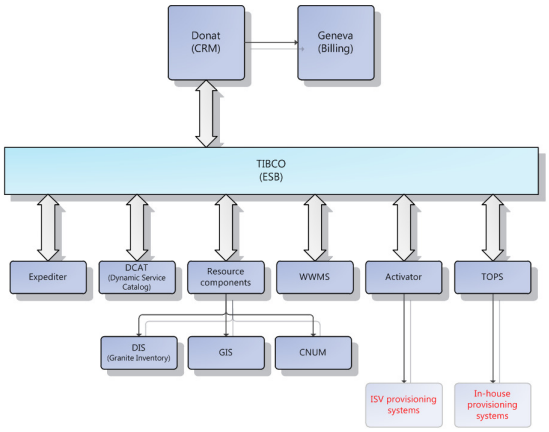

Figure 1: COMPASS SOA Architecture

BSS building blocks:
DONAT – proprietary CRM solution completely developed internally by HT
Geneva – Billing System by Convergys Corporation

OSS building blocks:
Expediter – Order management system by Telcordia
DCAT – Dynamic Service Catalog by Telcordia
Resource Components – Provisioning Engine components (resource finding and reservation) by GDi
Granite Inventory – telecom inventory system by Telcordia
WWMS – workflow and workforce management system by Fornax
Activator – activator system by Telcordia

As part of the overall architecture COMPASS introduced Expediter, DCAT and PCRCs, all of which had influence on other systems.

In order to fulfill CRM requests, OSS systems have to search and reserve some sort of resources. Depending on the resources found the associated order needs to be consigned to the Workforce system as well as the Activator system. When all orders are completed, a result message can be returned to CRM to indicate that the OSS part of the fulfillment process has finished.

This paper focuses on finding and reserving resources in the inventory.

In traditional environments, finding and reserving resources was managed by implementing for each type of service proprietary applications dedicated to a specific telecom technology. With that approach every technology dependent change led to a change in the application code. This had a huge impact on time-to-market for the service. Moreover, broadband services introduced a very complex, technology overarching matrix allowing telecoms to provide the same service using different technologies. To deal with said matrix the applications had to become both very complex and hard to maintain.

COMPASS introduced a new approach by which the finding and reserving of resources is divided up into a number of technology agnostic elementary components. The *Provisioning Control Resource Components* (PCRC) form a complete set of functions needed to fulfill any kind of resource search and reservation needs for all kinds of technologies. Since the components are technology agnostic and generic there is a need to provide a number of technology dependent parameters as part of a call to a particular PCRC. Also, the sequencing of component calls may be crucial. This information needs to be stored somewhere outside the PCRC but certainly inside the OSS since it is derived from inventory models. This use case provides another compelling argument for the deployment of the Telcordia® Dynamic Service Catalog. Inside DCAT all resource components are defined with all of the necessary, technology dependent attributes. The resource specifications in DCAT are usually part of a DCAT product model which corresponds to a CRM product. Some of the parameters defined for resource components are static (e.g., categories, statuses, or template names), others are filled by the DCAT-Engine from data in incoming CRM orders (e.g., customer installation location, bandwidth, specific service attributes).

With this kind of architecture, a very generic process can be implemented in the order management system. For less complex cases it becomes possible for the order management system to process different products in the same order and let the DCAT-Engine worry about all product inter-dependency operations. More complex cases introduce additional complexity, which is hard to track using only DCAT business rules. In such cases it is generally recommended to move the complexity out to the order management.

3. Provisioning Control Resource Components

In this section more details will be presented explaining the functionality of each of the resource components that are operating across the Inventory interface.

As depicted in Figure 1, Resource Components are SOA objects that can be invoked by Expediter (Order Manager) or DCAT (Dynamic Service Catalogue) during the fulfillment process. When resource Components are invoked by the DCAT-Engine, order and sequence of the execution is determined by DCAT where the Product-Service-Resource hierarchies are defined (Figure 2) along with appropriate business logic and rules.

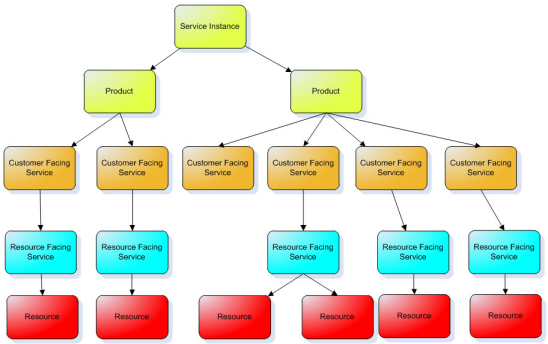

Figure 2: P-S-R Topology

Every resource components receives in an XML message sent from DCAT to Expediter all of the values binding the parameters defined in a DCAT model. A resource component exposes it results as an XML message that consists of two parts. One represents a list of tasks that need to be completed in order to fulfill the service (these tasks can be either manual or automated) and the other those results from the resource component result that encode the dependencies with respect to resource components being evaluated later in the sequence. A call to the next resource component will have to include all parameters defined for that particular resource component and, in addition, all results from all previous resource component calls. The resource component up for execution takes all parameters and extracts only those data from preceding resource component calls that is relevant.

3.1. PCRC Messaging

Figure 3 shows an example of an XML message input to a PCRC. This input method is replicated for all implemented PCRCs, thusly creating a clear and "readable" framework. This framework is easy to maintain and "future proof" in terms of the flexibility and ability to add additional parameters.

Figure 3: PCRC input XML sample

The PCRC input messages comprise five important sections:

I. Specifies name of the target PCRC (*FindFreePort* in the example) that is going to be executed and the action on it (CREATE in the example)

II. From the DCAT product decomposition tree (c.f., Fig.2) defines the CRM input products from which the call to the PCRC is derived. This is important in terms of a *back-mapping* capability when some of the parameters from the input are altered during PCRC execution and need to be mapped back in the original CRM request

III. Input parameters for the PCRC as defined in DCAT and filled during DCAT-Engine execution

IV. Results from previous PCRC calls. Since, PCRC calls have dependencies; the PCRC pulls the necessary data from input parameters and previous PCRC results

V. Original CRM order input (excerpted in the sample); some information PCRCs have to read data from original the CRM request

Similarly Figure 4 below shows an example of an XML message output from a PCRC:

I. The information copied from the request (excerpted in the example).

II. Result of the PCRC concatenated with previous result (in the example, first *PathElementAttribute* section was concatenated as the PCRC result)

III. Task list. In this case *FindFreePort* component returns single task for activation of DSLAM port found.

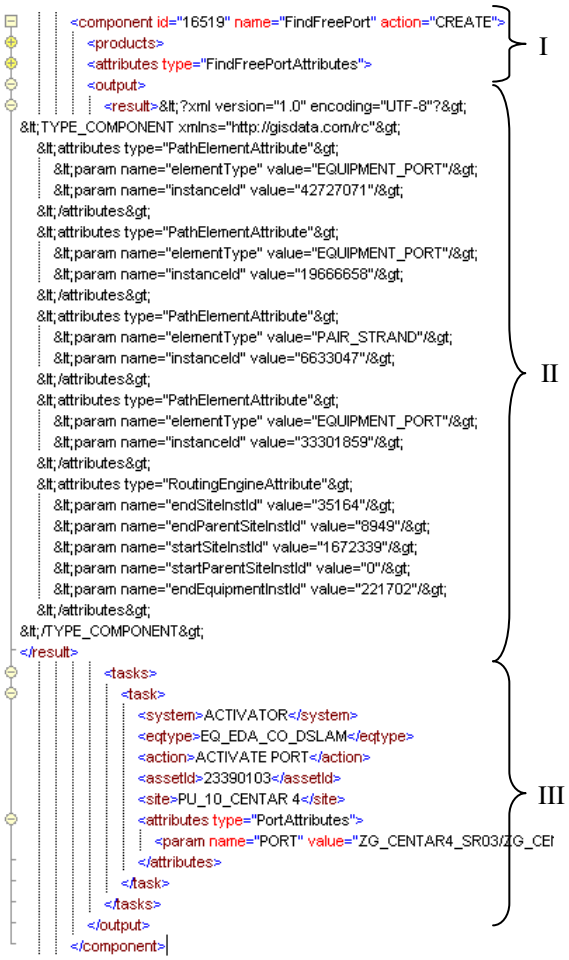

Figure 4: PCRC output XML sample

3.2. PCRC Execution

Depending on the incoming CRM order action, each of the components needs to have a slightly different behavior. The most straightforward one is the "Create" action when a new *Path* instance has to be created without having to inherit anything from other existing path instances. In general however, inventory components need to expose several different actions to support the life-cycle of an Expediter work order. This is to say when a new "Create" action order comes into the system; first a new path is created with all inventory resources that are defined in DCAT for the ordered product. The path status, however, is created as pending (yellow). After the workflow reaches the appropriate status, a message is sent to the DCAT-Engine to trigger component calls for updating the status from yellow to green (Live). So, all components need to expose methods for creating new instances and for changing status. In situations where Expediter invokes the Resource Components these rules of execution remain the same.

3.3. Examples of Key PCRCs

3.3.1 RoutingEngine Component

The *RoutingEngine* (RE) is the most complex resource component and it is used to find a route from the A-side to the Z-side. In most cases it will search for a free route from the customer site to the first point of presence of the operator (e.g., from the customer site to the MDF site in the copper case). The resource component has a lot of parameters that can be defined. Most of them are bound to values at design-time (such as technical premises site category, pair or strand categories, statuses, whether or not to use PCM paths instead of just pairs/strands, port/card/equipment terminated categories and statuses, and then some). During runtime, the field with the customer site location has to be filled with data from the CRM system in order to run the component. The RE component supports the following actions:

- CREATE – finds route from A-side to Z-side, route is composed of path elements (access node port, pair, MDF port), input parameter is *buildingId* (customer site location).
- MODIFY – this action is used for complex scenarios like reallocation, input parameters include *buildingId* (new customer location) and existing path (in this case *RoutingEngine* tries to reuse elements of existing path when searching for a route from customer site to technical location).

3.3.2 FindFreePort Component

The *FindFreePort* component is responsible for searching for free ports on active equipment in accordance with defined parameters. It can be used for defining a port or ports on a TDM switch, a DSLAM, an OLT, or similar. Output from this resource component will be a free port documented in the inventory and will be included in a path as a path element. This immediately implies that the resource component can only be used as a *child* resource component of a path resource component. There are several fields that can be defined as search criteria. Moreover, the order of port usage

on the equipment can be defined also. One of the significant fields is the site name (or site ID). Usually, the site ID will be the destination site ID from the RE component result. Also, port bandwidths, port statuses, port card statuses, and categories are very important fields too

This component is also capable of generating tasks for port activation/deactivation.

FindFreePort supports the following actions:

- CREATE – searches for port on active equipment in accordance with input parameters
- MODIFY – this action is used when there is an existing port (as a path element), in this case *FindFreePort* checks if this port satisfy input parameters and if not searches for new port, this action is used to upgrade/downgrade a service or, in case of reallocations, to reuse an existing port if possible.
- DELETE – this actions is used to return a port on a given path to the inventory and to generate port deactivation tasks

3.3.3 EquipmentManager Component

In the context here, equipment represents mainly customer equipment that has to be created at provisioning time. For examples consider CPE equipment items, such as splitters, ONT, ISDN NT, modem, etc. To define such equipment it is necessary to define a template according to which the equipment will be created. Also, an equipment name pattern is necessary. It may well be that other fields or UDA's have to be defined on the equipment object also. Ports of the equipment are used as constituent elements of a path. It is necessary to specify which port is used in which path.

The *EquipmentManager* component supports the following actions:

- CREATE – creates new equipment instances in the inventory from defined shelf templates, Resource Component returns ports on equipment that will be used as path element.
- DELETE - deletes equipment from the inventory

3.3.4 Customer Component

This component is responsible for managing customer objects in the inventory. CRM is the master of all customer data. All modifications are effected in CRM and then propagated to the network inventory. A customer object can be created either directly or by using a template. If a template is used, the customer name is a necessary field. Without a template, all fields in accordance with inventory requirements have to be defined. Customer objects can be assigned to paths.

The actions supported by this component are:

- CREATE – creates or modifies (if customer already exists) customer data in the inventory
- MODIFY – updates customer data
- DELETE – deletes customer object from the inventory

3.3.5 Path Component

This component is used for managing the inventory *Path* object. A new path is created from a path template and the associated path elements result from the execution of other resource components: *RoutingEngine*, *FindFreePort*, *EquipmentManager* and Path component (path can use channel of parent path). Depending on the input parameters the component can assign customer and order number (asset id) to the path.

This component is also responsible for generating a manual task. This task typically calls for wiring ports on an MDF and access node (when *PathElements* are added to a Path component checks for ports that are not pre-wired and generates the task). The actual behavior heavily depends on the input parameters that can be defined in DCAT at design time.

The actions supported by this component are:

- CREATE – new path is created from template and path name is assigned according to given path name pattern, path elements added to path are result of execution of other resource components.
- MODIFY – modify existing path:
  - replace existing path element(s) with new path element(s) (reallocation or service upgrade/ downgrade, etc.)
  - change path category
  - create path revision
  - change path name
- CHANGE_STATUS – changes path status
- DELETE – deletes path from inventory, appropriate manual tasks are generated.

3.3.6 Numbering Component

The Numbering resource component is used as an interface to the numbering system and it is responsible for managing telephone numbers: selecting free numbers on a network element according to defined parameters, changing number statuses, handling local number portability, porting

numbers to and from other operators, assigning numbers to a path in the inventory.

3.3.7 UDC Component

This component manages UDC (User Defined Class) objects in the inventory. Mandatory parameters for creating a UDC are a name and a type. All other parameters are defined as UDA's (user defined attribute) – all of these parameters can be defined in DCAT.

3.3.8 Association Component

The Association component creates association between objects in the inventory. The definition of an association must be specified trough the Telcordia® Granite *Administrator Client*. It must have defined source and destination objects replete with their categories and relationship cardinalities. For instance, in the provisioning process associations will be used to define relationships between an Internet UDC and the BB service path instance. The association resource will be placed as a child resource of the source object and it will point to the destination resource. A required field to define the resource is the association type.

3.3.9 GIS-PolygonDiscovery Component

This resource component is able to connect to a GIS database and to perform a GIS query which returns details of a polygon defined in the GIS database. As an input field the X/Y coordinates are used and the name of the polygon layer. Also, the field name of the polygon shape from which the return value will be read must be supplied as another input parameter. The component will return the polygon field value. This information can be useful to Expediter for enabling it to create a request to the work-force management system.

3.3.10 GIS-ShapeDiscovery Component

This resource engages a similar component as the previous one. The query on the GIS database will return all GIS objects with its details that overlap with the shape constructed as a circle around the X/Y coordinates and the radius supplied as inputs. The layer name on which the query will be performed should be supplied as an input also.

4. Illustrative Example

As an example illustrating what has been said in the previous chapters considers the COMPASS implementation of the POTS+ADSL+IPTV ADD process, which represents an Activation scenario for the POTS+ADSL+IPTV service bundle. We will use this scenario to explain how PCRC's are combined and integrated to achieve the desired functionality. The following diagram represents the process which has been realized by the COMPASS project and which is executed by the order management system:

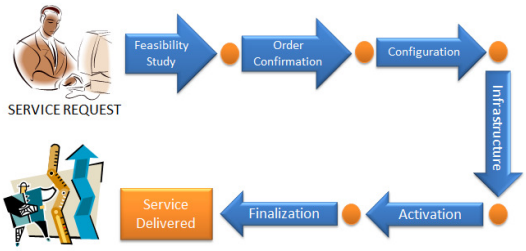

Figure 5: Service Delivery–generic process overview

4.1. PCRCs Used

The following PCRCs are used supporting both the provisioning of the POTS+ADSL+IPTV service bundle and the subsequent *Activation* process of the Expediter OM:

No.	RC name	Description
1	RoutingEngine	Finds free route from the customer site to the MDF
2	NumberingComponent	Finds free telephone number on PSTN exchange
3	FindFreePort	Finds free port on DSLAM and generates activation task
4	EquipmentManager	Creates splitter at customer side
5	PathComponent	Creates ADSL_LINE path and generates MDF task
6	FindFreePort	Finds free port on PSTN exchange
7	EquipmentManager	CPE – telephone
8	PathComponent	Creates POTS/ADSL path and generates MDF task
9	EquipmentManager	Creates modem at customer side
10	PathComponent	Creates BB/ADSL path
12	CustomerComponent	Creates customer in Inventory
13	PathComponent	Creates POTS service path
14	PathComponent	Creates ADSL service path
15	UDCComponent	Creates Internet UDC object
16	AssociationComponent	Creates association between ADSL service path and Internet UDC object
17	UDCComponent	Creates IPTV UDC object
18	AssociationComponent	Creates association between ADSL service path and IPTV UDC object

Table 1. Resource Components for POTS+ADSL+IPTV Activation

4.2. Order Management Process

The POTS+ADSL+IPTV *Activation* scenario is implemented through the POTS+ADSL+IPTV ADD process. The corresponding Expediter order management process comprises three parts:

- Feasibility Study (Figure 6)
- Confirmation
- Realization (Figure 7 and 8)

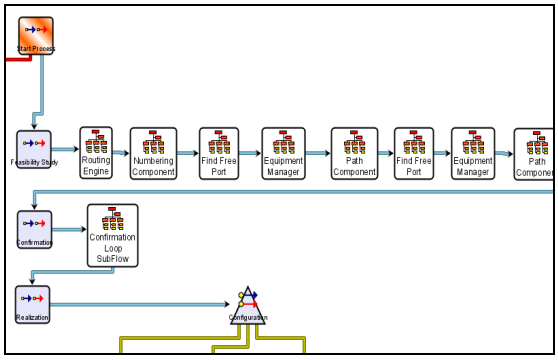

Figure 6: POTS+ADSL+IPTV Activation – start of process

During the feasibility study phase Expediter calls PCRCs that reserve resources in the inventory and prepare data for the execution of tasks. The overall result yielded by the PCRCs could be a task list with most of the data defined. Alternatively, the results may consist of some output attributes that point to an Expediter *Technical Solution* process, a COMPASS feature designed for handling complex service scenarios. A Technical Solution process is capable of capturing the work-flow needs of any type of service. Just like solutions communicated from DCAT, a Technical Solution will yield one or more task lists for execution by Expediter.

The Realization phase is comprised of:

- Configuration (configuration of active elements)
- Infrastructure (infrastructure tasks, mostly connections and disconnections)
- Activation (configuration of active elements when services are activated at the end)
- Finalization (post execution tasks like quality check)

Each of the tasks in the Realization phase is a sub-flow (see Task Control Sub-Flow) that handles complex dispositions returned by executed tasks.

The task *IN Configuration*, for instance, is executed when the process should handle LNP or add the IOCB value-added service. There are two possibilities for how to deal with this check in the process. It can be used as a switch activity or as condition script. In the case of IN Configuration it is a condition script that is used. In the second part of the realization there is a VMS check (switch activity) that in case of an existing VMS triggers a T-Mobile request task.

Usually all tasks wait for a response from the targeted system. Sometimes, however, the process shouldn't wait for a response; like in case of *QualityCheck* (this implements the functionality of WWMS where full black circle is used to identify the end of the task). Tasks using this functionality have "no response" text in the name of the activity.

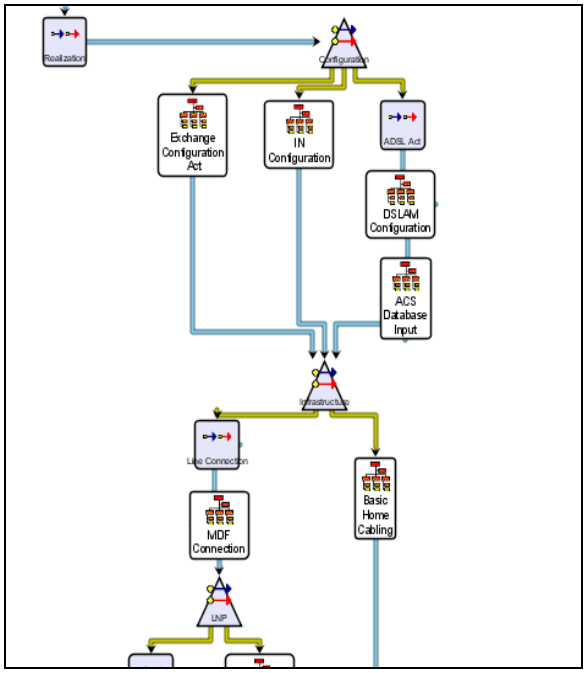

Figure 7: POTS+ADSL+IPTV Activation – first part of realization

In the second part of the realization there are two exceptions thrown. This is an example of dispositions that are not generic. When a "not-generic" disposition is received it can be handled in the main flow, as illustrated by this case. After the AN Connection task there is a check that handles uncommon dispositions, like "MaxTV is not possible" or "MaxADSL is not possible".

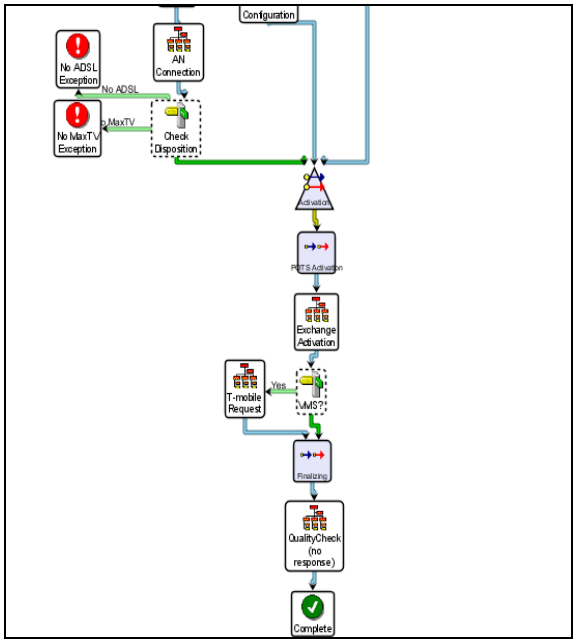

Figure 8: POTS+ADSL+IPTV Activation – second part of realization

5. Summary and Conclusions

In previous chapters we have shown that it is possible to employ a unified approach to the fulfillment process by building a finite number of modular and flexible "atomic" functional modules that will provide elementary functions for all operations necessary within the fulfillment domain. These concepts are being successfully applied within Croatian Telekom. A key question, however, for future development is how these concepts can be extended to inter–company domains. The key idea here is ability for customers (in the CMR paradigm mentioned already) to create product/service bundles that are crossing the boundaries between cooperating companies (Figure 9). Not considering economic and competitive concerns for a moment, it is obvious that a single service catalog for all providers is unlikely to exist, since all partners are likely to insist on their own solution. Alternative solution is to use *Semantic Web* inspired approach since this provides key elements for the semantic architecture of Web Services. In general, by applying concepts such as OWL-S (Web Ontology Language for Services) every Web Service (including PRCS if exposed to external world) can be described with OWL defined ontology, a semantic mark-up language for publishing and sharing ontologies on the WWW.

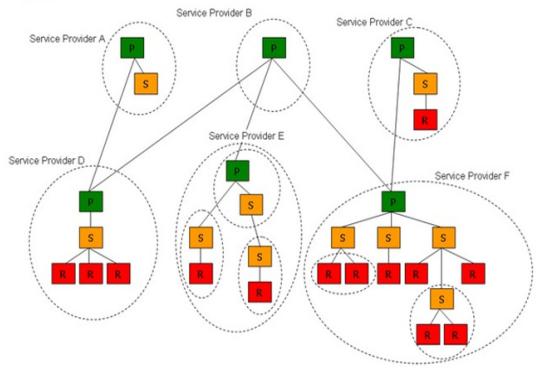

Figure 9: Inter-Company relationships

Ontologies provide a shared vocabulary to represent the meaning of entities, while knowledge representation provides structured collections of information and inference rules for automated reasoning. As a result, intelligent software agents can interpret and exchange semantically enriched knowledge for users.

The development of PRCS components will evolve in the suggested direction with creation of appropriate ontologies.

Finally, by way of a COMPASS specific conclusion the three pillars of that project:

1. Unified Service and Resource Inventory
2. Dynamic Service Catalog [11]
3. Provisioning Control Resource Components

have enabled a small team of professionals from Croatian Telecom and GDi GISDATA to achieve substantial and valuable results in short time frame.

Built on a foundation of resource components representing fundamental operations, the over-arching concepts of Catalog Driven Order Management been verified. The resulting COMPASS solution has been committed to production since July 2010 and has already proven to justify the investment.

The concepts underlying COMPASS were presented during Management World 2010 as part of the Catalyst program [10] and were also brought to the attention of a wider audience by a Gartner Group report as a first implementation of this new concept in Service Catalog domain [9].

References

[1.] TM Forum GB921: eTOM Concepts and Principles 9.10

[2.] TM Forum GB921-D: Process Decompositions and Descriptions 9.10

[3.] TM Forum GB921-E: End-to-End Business Flows9.10

[4.] TM Forum GB921-F: eTOM Process Flow Examples 7.50

[5.] TM Forum GB921-P: An eTOM Primer 8.10

[6.] TM Forum GB921-U: User Guidelines for eTOM 8.10

[7.] TM Forum GB929: The TAM - The BSS/OSS Systems Landscape 2.10

[8.] TM Forum GB929: The TAM - The BSS/OSS Systems Landscape 3.00

[9.] Gartner Competitive Landscape Catalog Sept 2010

[10.] TM Forum - COMPASS Catalyst Results V1.1 Final

[11.] Medved, Damir; Tutman, Hrvoje; Periškić, Dominik; Jagau, August-Wilhelm: "Introducing Advanced Service Catalog Capabilities in Croatian Telekom NGOSS Infrastructure", proceedings of the 2011Mipro Conference Opatija, Croatia