CERN-IT-2000-002
15 February 2000

Integrating Autonomous Problem Resolution Models with Remedy

Miguel Marquina, Raúl Ramos, José Padilla
CERN, IT Division, User Support Group

This paper briefly defines the concept of Problem Resolution Model and shows possible approaches to the issues which may arise when integrating various PRMs to present a consistent view to the end user, despite of the peculiarities of each physical implementation.

Integration refers to various autonomous PRMs having to interact as problems pass from one to another in the resolution flow. This process should be transparent to the user and internally there must be a way to track in which stage of the resolution process any problem is. This means addressing two different issues. On one side PRMs which are to be integrated need to comply with certain interface standards. These standards must ensure that problems exchanged between them can always be traced. On the other side problems owned by different PRMs should be presented to the end user under a homogeneous view. This means having an uniform criteria for automatic notification messages, a single reference point (www) where users can query the status of problems regardless who owns them , etc.

Remedy Action Request System® (ARS) is a specialized development system designed to implement PRMs and it is the current choice of IT Division for such a system. When integrating Remedy ARS based PRMs systems there are some difficulties arising which are intrinsic to Remedy ARS's design. In this paper we describe our assessment of those difficulties and their Remedy ARS specific implementational implications.

1. Basic Concepts.

A *Problem Resolution Model* (PRM) is a defined strategy to handle the *action requests* which arrive to the problem-solving section of an organization. Within IT's helpdesks, **action requests** are problems which may arise regarding the usage of the computing facilities.

A **PRM** defines and states the ways in which an action request is going to be treated all along its lifecycle until its resolution, providing the means to facilitate the flow of information generated by the interventions required to address each action request. It also defines the different layers that are involved in this process (the different *escalating levels*), integration between layers, information required at each stage, integration with other systems, etcetera.

A PRM is often specified using a flow diagram or *workflow* . A **workflow** describes the *states*, *transitions* and *actions* an action request may go through. A **state** is a particular condition (or status) of the action request inside a workflow. In every moment each action request is in one and only one state. A **transition** constitutes a change of state of an action request. An **action** is an event (or set of events) associated to a certain transition which occur when that transition is triggered. Figure 1 shows a simple workflow containing two states (*In Service* and *Closed*), one transition named *Close* and one action happening in that transition (an email notification).

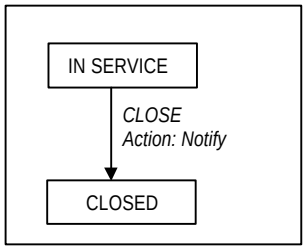

Figure 1: Simple Workflow

Operations following a certain workflow may be performed with the help of a *computer application*. Typically **applications** implementing PRMs consist of a database containing action requests and a set of user interfaces through which anyone involved in the resolution of an action request can trigger transitions, manipulate information concerning action requests, etc. These systems range from simple database manipulation packages to complex architectures to account for different logistics such as security levels for different people solving action requests, integration with other systems, customizable workflows, etc. There are several software products in the market specialized in implementing PRMs. The current choice of CERN's IT Division for such a system is Remedy Corporation's *Action Request System® (ARS).*

Remedy ARS applications are made of a set of custom objects (forms and dependant elements) plus the associate database tables and accessory elements (e-mail gateways, etc.). Typically each Remedy ARS application implements a workflow, but it may also be the case of a Remedy ARS application implementing several workflows. This will be the main scenario of this paper. We denote by **global PRM** a set of PRM acting jointly to perform a homogeneous and defined *service*. In this context a **support service** (or simply *service*) is a support task that is performed by a group of people. Sometimes different support services adopt different PRM which become part of the global PRM of the organization.
Detailed information about these ideas can be found in **[1]** and **[2]**.

2. Interacting Problem Resolution Models.

In complex organizations such as CERN, tasks are usually distributed between independent teams, which are responsible for a certain part of the whole process. Their members are specialized in the tasks at their charge and the overall quality of the service is improved. Every team has its own resources and needs related to their situation, staff available, work methodology, etc. This heterogeneity makes their integration towards achieving a global task a complex issue.

Ideally, a unique global PRM should accommodate every team in a consensuated way. However, we believe true global integration must take into account the intrinsic heterogeneity of independent teams by providing means of interacting between existing PRMs. Sometimes this interaction may just be a transitory step towards global unification of PRMs, other times it will allow truly independent PRMs to participate in a global PRM without loosing their independence.

This context leads to the necessity of envisaging a global workflow to perform a service, interacting eventually with other workflows (such as exchanging action requests, information, etc.) The key idea of interacting workflows is that of an *interface* between them. An **interface** is a transition between workflows. As defined for stand-alone workflows, a transition leads univocally from one state to another, triggering certain actions. Transitions, as interfaces between workflows, take us from one state in a source workflow to a state in a destination workflow, optionally triggering certain actions or carrying certain information between workflows. Figure 2 shows two independent workflows with two interfaces between them. One interface to transfer an action request from workflow A to B, and another interface to transfer the action request back from B to A.

An **entry point** is the state through which an action request enters the system. Similarly an **exit point** is the last state for every action request in the system. Every action request should finish in an exit point. Typically, an action request enters a workflow through an operator at a helpdesk manually entering the problem in a user interface, a mail gateway automatically processing mails sent to a specific support mail address, etc. Linking workflows through interfaces may result in different combinations of entry and exit points to take into account when agreeing on establishing an interface.

2.1 Multiple entry and exit points.

This is the case linked workflows have independent entry and exit points but they are interacting with each other and have certain transitions that link from one to the other. This is shown in figure 2.

This case typically represents the fact of independent support lines available to users, such as two helpdesks for different kinds of problems, or one email address for the support line of team A and a different email address for the support line of team B. Figure 2 shows that an action request which starts in the entry point of the workflow “A” may finish in the exit point of workflow “B” or vice-versa, depending on the actual definition of the interfaces.

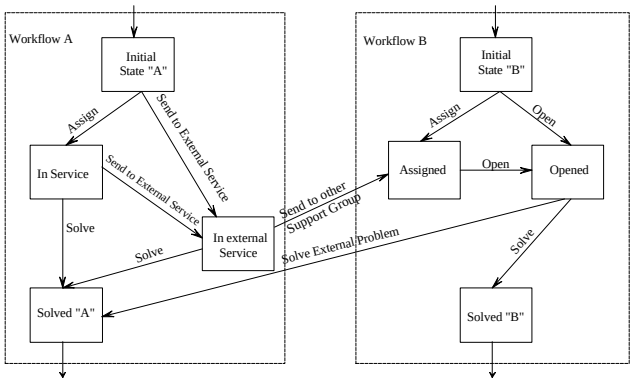

Figure 2

2.2 Single entry and exit points.

In this case there are common entry and exit points, no matter which of the workflows finally handles the request. This is the case of, for instance, team B delegating problem receptioning to team A and providing criteria to distinguish the action requests which must be sent to them.

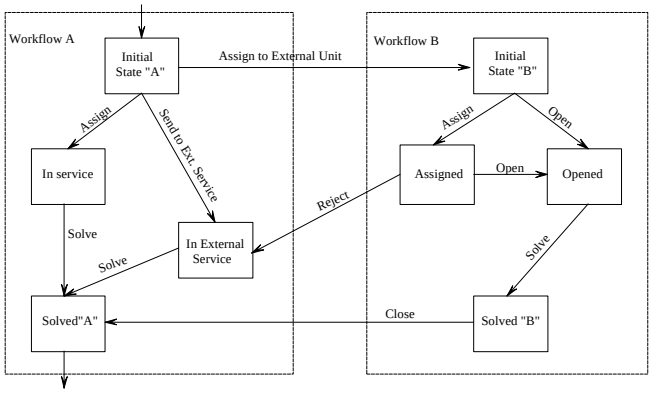

Figure 3

Figure 3 shows an example of this type of interface. The entry point for this system is the state *Initial State “A”* and the exit point is *Solved “A”*. These are the states the action request will always start and finish in, although during the resolution process the action request may pass to the workflow “B” and come back to workflow “A” as many times as needed.

3 Implementational issues for PRM interactions.

As stated above, a PRM defines workflow, which is implemented by an application. Depending on the choice of software certain restrictions apply to the way workflows and interfaces are implemented by applications. Within an organization, the deployment and adoption of workflows and applications implementing them gives rise to different scenarios:

a. Single workflow – single application.
The organization defines a unique workflow accommodating the needs of all support teams. This workflow is implemented in a single application that is also used by all support teams.

b. Multiple workflows – multiple applications.
Different teams work following different independent workflows, and each workflow is implemented by a different application.

c. Multiple workflows – single application.
Different teams work following different independent workflows, but they are all implemented in the same application which accommodates all the logistics required by the workflows.

d. Single workflow – multiple applications.
A workflow is implemented by several applications. It is not very likely to happen in normal cases. Only when special requirements need to be met, such as handling enormous amounts of data, distributed applications, etc. We are not going to consider this case.

Any of these alternatives has pros and cons that everyone involved in the definition and deployment of a (set of) PRMs must bear in mind. Apart from material constrains; the only real requirement that any of these alternatives must follow is to be transparent to the user. This means that the user must not be aware all the internals of the workflows involved in dealing with his problem, in terms of user interfaces, perceived resolution cycles, etc.

Alternative **a.** is the simplest. With only one application and one workflow, there is no integration problem and no need to define interfaces. It implies a considerable initial effort to design a consensuated workflow suiting everyone's needs. From the application point of view, it requires a central development team to build and maintain the application, which may be as complex or simple as the compromises implied by the consensuated workflow. A compromise should also be reached in terms of future modifications/updates of the system since it affects every team.

Alternative **b** preserves independence between teams in terms of workflow to adopt and they way to implement it. It requires to define proper interfaces between each interacting team. Changes to a local workflow won't affect to the rest as long as interfaces are respected. On the other hand it requires each team to provide resources to build and maintain their own application. Otherwise, a large enough central development team must take care of everyone's application.

Alternative **c.** combines the aspects of the other two. It is an hybrid solution, where there is a single central application implementing several workflows. It increases the complexity of the application but it still gives different teams the independence to design their own workflow without having to dedicate resources to build and maintain the application that implements it.

The concept that an application can implement several workflows may be better understood with the example of figure 4. In this case we have an application which is implementing two workflows. Workflow 1 is a simple one, action request always enter in the state *New* and from it they can be sent to somebody's queue or be solved directly. From *In Queue* action requests are put *In Service* where they are sent to *In External Service* or they are *Solved*. Workflow 2 starts in the same state *New* but then the action requests can only go to *In Service* from where they can be *Solved* or kept *On Hold*, where they can be *Solved* or *Closed.*

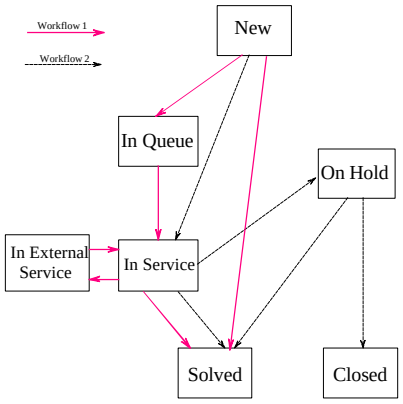

Figure 4

A single application implementing both workflows must provide mechanisms to determine which transitions are available at a certain moment. For instance, when submitting a problem only the transitions to the states *In Queue* and *Solved* should be available to a member of the team using workflow 1, while for a member of the team using workflow 2 only the transition to state *In Service* should be available. The complexity of the single application grows rapidly according to the quantity of workflows it implements and their differences. This implies that the maintainability of this kind of applications is greatly affected with the diversity and quantity of workflows it implements. This limits the practical applicability of this alternative. Also, the facilities that the development environment provides to handle complex projects plays a very important role. Table 1 summarizes the tradeoff between diversity and resources of the different alternatives.

	Applications Complexity	Interface Complexity	Necessary Resources	Versatility / customization
a. One Workflow One Application	*Moderate.*	*Low*	*One relatively small centralized group of developers*	*Low. Every modification affects everybody*
b. Several Workflows Several Applications	*Moderate, according to each team*	*Very High. Necessary agreed interface.*	*Every group need invest resources for developing the tool.*	*Very high. Every team writes its own application according to its needs. Only limited by interfaces*
c. Several Workflows One Application.	*Very High*	*Medium.*	*One relatively large centralized group of developers*	*Medium, depending on the complexity of the application.*

Table 1

These choices always represent a compromise between centralization and independence. These compromises are not always to achieve for different reasons. With the background aim of rationalization in mind, as intermediate choices one may also adopt the following scenarios: ***(1)*** having a single application implementing a single workflow but allowing a limited number of variations over the workflow to different teams. ***(2)*** having one main workflow along with its application for everyone, tolerating teams willing to dedicate resources to create applications as long as they comply with certain interfaces with the main workflow.

3. Requirements for a global PRM at CERN.

3.1 End user requirements.

There are some requirements that have to be achieved no matter which choice is elected for the physical implementation of a global PRM. They influence the complexity of the final system, and they are more easily achieved with some alternatives than with other.

From the user point of view, the requirements can be reduced to uniformity. The user does not have to know about how the any PRM is designed or how many teams are involved in solving the problem (s)he has submitted. From a general point of view, the following non-exhaustive list of requirements must me addressed when designing a global PRM strategy:

Single access point for entering problem. For the average user has to address to a single point to submit problems. This eliminates possible user confusion when confronted to having to choose between different support lines.

Single query point. The user must have a single place where to check the current status of submitted problems.

Single access to FAQ with the most common problems. To avoid submission of problems that users can solve by themselves.

3.2 Problem solving staff requirements.

For the problem solving staff (the persons in charge of solving the problems), the final system must be easy to use and the learning cycle short. Although simple, it must have all the tools that are needed for the problem solving process available in a single environment when possible. These and other desired characteristics are:

Simple interface, easy to use. The interface should be role-based, offering the useful features available for every kind of analyst. (i.e. A Helpdesk operator does not have the same needs that the problem solving staff specialized in solving web problems). For this the system must be customizable enough to be able to show different features according to what the analyst needs, adapting what the problem solving staff see to what they normally use.

Uniform look & feel of the application. Despite the peculiarities that every role needs, there are some common elements, which should be uniformly distributed for all the roles. This way we reduce the adaptation time when a member of the problem solving staff changes his role in the organization when only a few singular characteristics differentiate each role. This uniform presentation should be present independently from the different implementations that it may hide.

Homogeneous and simple searching and manipulation. When using the interface, the problem solving staff are constantly manipulating the different tools that they have available. This daily process should be comfortable and simple for them.

Fast access (consult / insertion) to a knowledge base. (Q&As database). Apart from the access to the usual problems, the problem solving staff should have access to "added value" Questions and Answers knowledge bases with high quality and very detailed explanations. They should be responsible for promoting typical problems to the status of questions and answers by enhancing the answer and making the information available to others.

Access to the current status/details of a problem. There should be a simple way of access to every action request (except the confidential ones) in the global PRM, in which the problem solving staff should have access to all the information (including the comments and internal stuff) available for a given action request.

4 Constraints of Remedy's ARS.

As described in **[3],** the main constraint of Remedy's ARS is its limitations to handle in a maintainable way complex

applications. This affects options **b.** and **c.** The practical consequence of this is that it becomes necessary to develop several coexistent applications as soon as a certain level of complexity is reached (for handling the diversity of the different stages and escalations in the resolution process). Once we have several applications interacting in a single PRM we have to design coherent interfaces allowing action requests to pass from one application to another.

If application developers follow certain rules, Remedy allows an easy design of interfaces between applications keeping the traceability of the action requests. One important aspect to fulfill the requirements described in last section is keeping an unique identifier of the action request, no matter in which application (or workflow) it is. Remedy does not provide built-in facilities to guarantee uniform action request numbering across applications, so this must be ensured by the interfaces between application and requires every application to conform to those interfaces. Also, applications must exchange or feed data to keep a central consult point for checking the current status. This process may be more or less simple depending on the complexity of the different applications and the central consulting system.

Summarizing, Remedy's particular characteristics constraint in a significant way choices for integration and heterogeneity. However Remedy's conceptualizations and tools still provide a rapid development environment for well established workflows.

5 A proposal for centralization.

Bearing in mind all the previous considerations we think that the best alternative in the long term is a global common application implementing a consensuated global workflow and establish certain interfaces and rules of interaction with other applications. Those teams willing to invest the resources to implement their singularities are to be encouraged to use these rules to still be part of the global architecture and maintain uniformity towards the end user.

Those interfaces and rules should be aimed at ensuring the requirements of section 3 and should be tightly integrated with the global application design. Well known procedures must be established for applications to join and define responsibilities for the maintenance and evolution of the overall system.

Other solutions can be adopted on a global basis. *Searching of action requests* in the PRM, no matter who is handling them may be achieved by web based solutions, avoiding the dependency from the Remedy clients specially for the end user. Extraction of data from Remedy's database to web engines may be done using regular scripts with the Remedy API or SQL queries triggered at regular intervals. This unifies the search of information coming from Remedy with the one that is not in Remedy such as documentation, Frequently Asked Questions etc. which may be searchable from the same place. *Backups of data and applications* can also be implemented in a centralized way as a service for Remedy administrators. *Periodic reporting facilities* both batch and interactive improve usability and add value to problem solving staff and end users. Centralized version control mechanisms and statistical facilities are also valuable assets to any application developer and problem solving staff.

In general, a sound architecture will grow confidence on those teams confronted with the choice of joining the global workflow or creating their own. This benefits in the long term the homogenization of the different problem solving processes existing within the organization, avoiding in the present and in the future, cumbersome decisional processes to teams for whom problem solving constitutes a means to reach their goals.

6 Conclusions.

In this paper we defined the concept of Problem Resolution Model and showed approaches to the issues which may arise when having various PRM interacting as action requests pass from one to another in the resolution flow.

We also discussed the specific difficulties that appear when various PRM that interact are implemented in Remedy AR System.

Finally we presented an assessment to overcome both the general and Remedy AR System specific drawbacks which intrinsically derive from integration.

7 References.

[1] M. Marquina, R. Ramos and J. Padilla "Implementing Problem Resolution Models in Remedy " IT Report CERN-IT-2000-001. February 2000.

[2] Remedy Corporation, "Action Request System 4.0 Concepts Guide". December 1998

[3] A. Ballaminut and R. Ramos, "Evaluation of Remedy's Problem Report Management System", Technical Report CERN-IT-US-1998-05. July 1998.