



US 20030093688A1

(19) **United States**(12) **Patent Application Publication****Helgesen et al.**(10) **Pub. No.: US 2003/0093688 A1**(43) **Pub. Date: May 15, 2003**(54) **METHOD FOR AUTOMATION OF SOFTWARE UPGRADE**(52) **U.S. Cl. 713/200**(76) Inventors: **Jan Ingvard Helgesen, Oslo (NO);
Stefano Angelo Donati, Oslo (NO)**(57) **ABSTRACT**Correspondence Address:
**NIXON & VANDERHYE, PC
1100 N GLEBE ROAD
8TH FLOOR
ARLINGTON, VA 22201-4714 (US)**

The present invention relates to a method for automation of software upgrade of network elements in data and communication networks. In a preferred embodiment of the present invention, an upgrade of network elements in a communication system is controlled from one certain workstation. Before the actual activation of the new software of the network elements takes place, the hardware configuration (5) is validated ensuring that the new software is loadable in the hardware. The software configuration is then also validated ensuring that the new software and the existing software are compatible. A health check is then executed investigating whether the status of the network elements is as required. Then, the certain workstation from which the upgrade is controlled is assigned exclusive network administration access to the network elements. Assuming that the above mentioned is carried through successfully, the actual activation of the new software is executed. Thereafter, the exclusive access which was assigned to the workstation is released, and before cleaning up and terminating, post validation and post health checks are executed.

(21) Appl. No.: **10/312,392**(22) PCT Filed: **Jun. 21, 2001**(86) PCT No.: **PCT/SE01/01417**(30) **Foreign Application Priority Data**

Jun. 21, 2001 (NO) 20003375

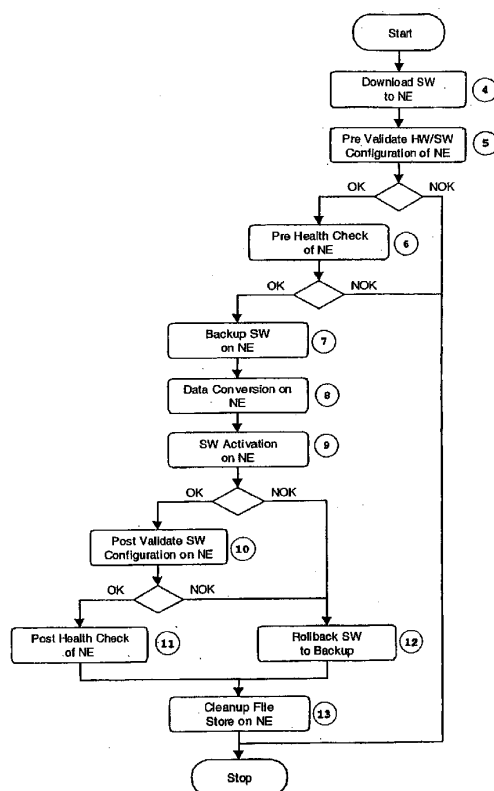
Publication Classification(51) **Int. Cl.⁷ H04L 9/00****Upgrade Process (3)**

Figure 1 Remote Upgrade Station

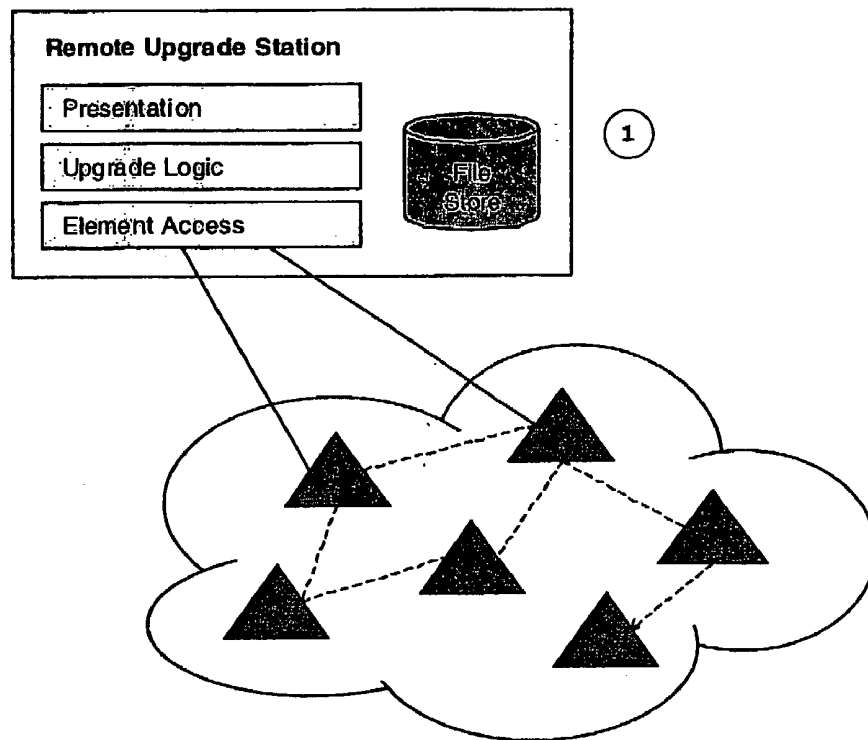


Figure 2 Network Element

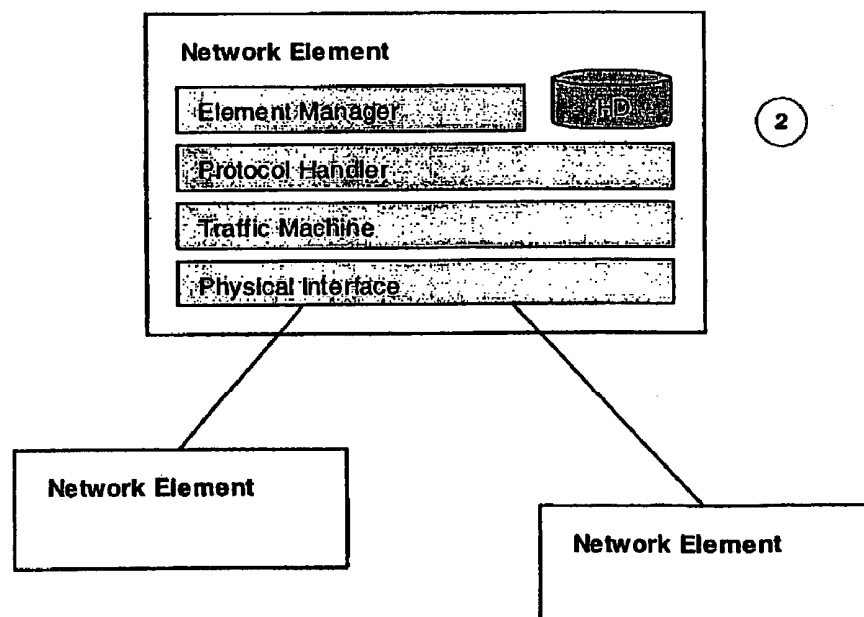
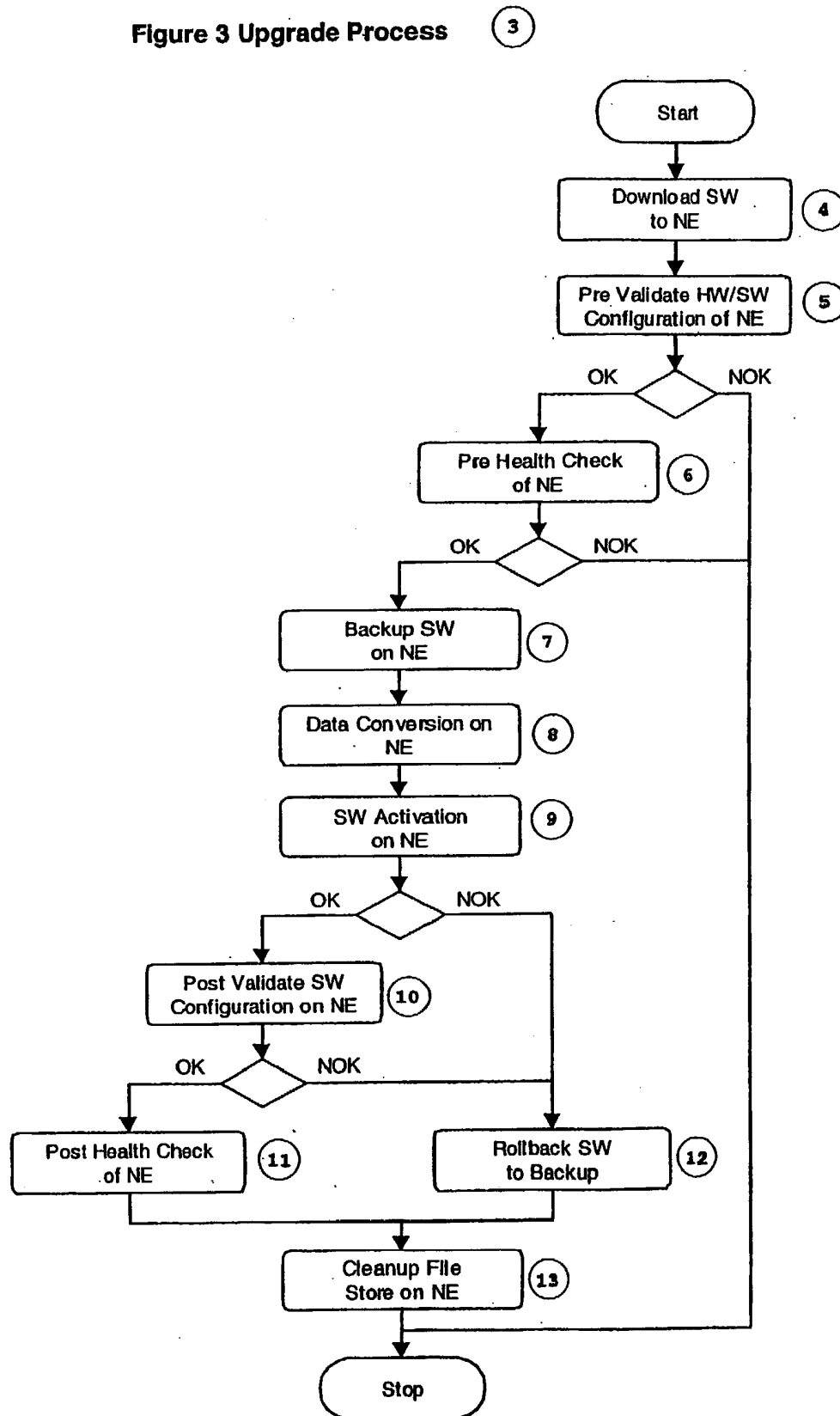


Figure 3 Upgrade Process

METHOD FOR AUTOMATION OF SOFTWARE UPGRADE

FIELD OF THE INVENTION

[0001] The present invention relates to a method for automatic software upgrade of network elements, in particular for use in data and telecommunication network.

BACKGROUND OF THE INVENTION

[0002] A data or telecommunications network consists of a number of network elements interconnected by physical connections. Network elements implement routing, switching and transport functions that are required to establish communication between two or more network end-points.

[0003] In addition, network elements may implement application functions like multimedia telephony or unified messaging, which provide additional value for the end-user beyond basic end-to-end connectivity. In traditional telecommunication networks, the number of different types of network elements has been quite small, because network functions spanning from transport to the application layer have been integrated in a single element.

[0004] However, following the convergence of data and telecommunication technologies, the number of different network element types is increasing. Ericsson and other vendors are dividing the next generation network into connectivity, control and application layers, with a large number of specialized network element types at each layer. The next generation network enables the operator to decrease time-to-market for new services, and differentiate current service offerings more effectively.

[0005] Network traffic is increasing rapidly in access and carrier networks world-wide, which implies that the number of network elements is growing fast in order to handle the increased network load. This and the fact that network elements are becoming more specialized require more effective operation and maintenance of the network.

[0006] In large operator networks with thousands of network elements, software upgrade is both a time-consuming and complex process. The complexity of the upgrade process makes it difficult to roll out software on several elements in parallel, because the operator needs to manually supervise every step of the process.

[0007] The software upgrade process is also often complicated by dependencies between software modules. There may be dependencies on the version of other software modules that need to be resolved before upgrade can commence. There are also runtime dependencies between software modules, which implies that a software module may not be upgraded while another is running. In addition, the upgrade process is complicated by dependencies between software modules and the underlying hardware that executes the software.

[0008] Exceptional situations may also frequently occur during the upgrade process, which require special attention or even rollback of some or all steps performed up to that point.

[0009] As described above, next generation networks will include a large number of network elements based on different hardware platforms. This further complicates the

software upgrade process, since it requires experience on several different hardware platforms. In the current situation, experience is required only on a small number of different platforms, e.g. the AXE platform for Ericsson's network elements. However, this is about to change and for next generation, Ericsson is introducing Cello, TelORB, Ronja, Erlang OTP and Tigris platforms. The situation is also similar for other vendors.

[0010] Another problem of today's upgrading methods is that education operation and maintenance staff on every new platform is expensive. In addition, software upgrading is in great extent performed manually at the network element site or remotely from an operation and maintenance center. The problem with the manual approach is that experienced operators are required to control the process, and such individuals are often a scarce resource that should not be used for routine like operations. In addition, humans cannot control more than a few upgrades in parallel, whereas computer programs can control a large amount in parallel.

[0011] There are IT applications which produce software packages used to install software components on general-purpose computers, e.g. InstallShield® from InstallShield Software Corporation Inc. These applications produce software packages that contain scripts to interact with the user and copy the software into the system. It is also common to include scripts to remove software from the system. Since a network element is no more than a special-purpose computer, these IT applications should be regarded as known solutions.

[0012] Current script based solutions can be used to install software on a single network element, but there are no known solutions for automating the entire upgrade process. The latter is required to upgrade software on several network elements in parallel. The problem is that any number of exceptional situations may arise during the upgrade process.

[0013] European Patent Application EP 0 945 792 A2, held by Sun Microsystems Inc., describes a technique for implementing a framework for extensible applications. Using the framework, a software module which installs itself within an application can be developed. In this context, one may regard a network element as an "application", thus the patent could be used for software upgrade of network elements.

[0014] However, no known network elements are based on the framework proposed here, and for this reason, this patent proposes no solution to the software upgrade problem of existing network elements. Also, it is not likely that future network elements will implement this framework.

[0015] International Patent Application PCT/SE92/00411, held by ICL Data AB, describes a "method and system for revising data in a distributed data communication system". The following procedure is described in the patent to perform an installation or upgrade of data. In this context, software may be regarded as data.

[0016] 1. Establish a list of destination devices

[0017] 2. Establish a procedure for the revision of data on selected destination devices, and store this as a revision recipe.

[0018] 3. Create a data package with the data to be revised and the revision recipe.

[0019] 4. The data package is distributed to selected destination devices.

[0020] 5. The selected destination devices interpret the information in the data package by means of a special interpreter program installed in each device.

[0021] The above patent concerns only installation or upgrade of workstations in a distributed data communication system. There are different non-functional requirements for upgrade of workstations and network elements. For instance, downtime may be accepted for workstations, while this is unacceptable for network elements, i.e. the upgrade must be performed while traffic is running.

[0022] The object of the present invention is to provide a method eliminating the drawbacks described above.

[0023] More specifically, a main object of the present invention is to simplify the complex and time-consuming upgrade process in a data or telecommunication network including a large number of network elements by introducing automatic software upgrade of the elements.

[0024] Another object of the invention is that it provides the required fault tolerance during the upgrade process to ensure that special attention or even rollback of some or all steps performed up to a certain point takes place. This is achieved by dividing the process in a number of steps, where each step must complete successfully before proceeding with the next.

[0025] A further object of the invention is that it minimizes the need for expensive, manual intervention by expert operators.

[0026] The present invention also overcomes the drawbacks of the mentioned international Patent Application PCT/SE92/00411, e.g. the scope of the present invention is upgrade of network elements that carry the traffic in the communication system, and no downtime for the elements while upgrading is required.

SUMMARY OF THE INVENTION

[0027] The above objects are achieved by a method characterized in the features defined by the characterizing part of claim 1.

[0028] In particular, the present invention relates to a method for automatic software upgrade of network elements, for use in data and telecommunication network. It describes a recipe that makes it possible to automate the upgrade process, i.e. run the software upgrade without operator intervention. The recipe is required to include the following steps:

[0029] health checks of the upgrade target,

[0030] validation of the hardware/software configuration,

[0031] backup,

[0032] data conversion,

[0033] activation, and

[0034] cleanup.

[0035] These steps are partly executed prior to and after activation of the software on the upgrade target. This in

order to make sure that the required conditions to initiate the activation is met, and to make sure that the activation has indeed been successful.

BRIEF DESCRIPTION OF THE DRAWINGS

[0036] In order that the invention may be more readily understood, the method will in the following be described with reference to the accompanying drawings.

[0037] **FIG. 1** illustrates a sample upgrade station and the communication towards network elements,

[0038] **FIG. 2** illustrates major components in a network element,

[0039] **FIG. 3** is a flow diagram illustrating the steps involved in the upgrade process.

DESCRIPTION OF PREFERRED EMBODIMENT

[0040] With reference to the abovementioned figures, in the following, an example of how a method according to the present invention may be implemented will be described.

[0041] Software Package

[0042] It is assumed that the vendor distributes software upgrades to network operators by means of software packages, which contain everything required to perform the upgrade. This includes at least the following components:

[0043] 1. The software that should be upgraded. The software on the network element may consist of several load modules which are upgraded independently. In this case, the package may include only the modules that are to be upgraded.

[0044] 2. Software configuration. A description of the software modules included in this package and their dependencies on other software modules installed on the network element or included in this package. The configuration also includes a description of the hardware required to install this software. The recipe presented below uses this configuration.

[0045] 3. One or more recipes used to execute each step of the upgrade process (3). These have to be tailored to support upgrade of the particular software delivered with this package. One alternative is to implement the recipe using one script per step of the upgrade process (3), where each script includes the commands required to execute a step. For instance, a script may start by setting up a Telnet session towards the network element, issuing an appropriate sequence of commands, and analyzing the output.

[0046] 4. Reference documentation for the new software. The upgrade process (3) is automated by this invention, hence no documentation is required for this purpose, but troubleshooting information may be necessary. It is also common to include transcripts of the verification test performed by the vendor.

[0047] Remote Upgrade Station (1)

[0048] In most cases, it is preferred to initiate software upgrades from a remote upgrade station (1) to avoid a visit to the network element site, and in order to run several upgrade sessions in parallel. **FIG. 1** illustrates a sample upgrade station and the communication towards network

elements. This communication may be carried together with the voice/data traffic or over a separate management network.

[0049] The figure also indicates the components which would typically be used to implement the upgrade station. This includes presentation, logic used to execute the upgrade, and element access. Presentation and element access is outside the scope of this invention, but included here for completeness. Element access is responsible for communication towards the network element's management interface. SNMP, CORBA or Telnet protocols are commonly used for this purpose.

[0050] The upgrade logic is required to initiate and control the software upgrade process (3) defined later. The upgrade logic will read the recipe and follow the steps defined therein. In order to make the upgrade process (3) run automatically, these steps should conform to the ones defined in this invention. The upgrade logic may be regarded as an interpreter program that performs the upgrade based on information contained in the recipe. This invention defines contents of the recipe that is required to automate the upgrade.

[0051] Network Element (2)

[0052] FIG. 2 illustrates major components in a network element (2), but these may vary depending on the role of the network element (2). What is important with regards to this invention, is that the network element (2) must implement some sort of management interface, which may be realized using available standards (e.g. SNMP, CORBA, FTP and Telnet) or vendor specific protocols.

[0053] Commands are invoked on the management interface to perform the software upgrade from a remote location, e.g. there may be a set of commands for downloading the software to the network element (2) and another set for activating the software.

[0054] Naturally, the granularity of the management interface may vary from one network element (2) to another, and the trend is towards more high-level support implemented in the network element (2). This means that different network elements provide different levels of support for software upgrades, which again affects the level of control that the remote upgrade station (1) needs to enforce on the upgrade process (3). For older network elements, the station may need to issue a number of low-level commands for every step in the process (3), while newer elements may support a single high-level command for each step.

[0055] Upgrade Process (3)

[0056] The present invention assumes that software packages should include a recipe to automate the upgrade process (3). This recipe is embedded in a software upgrade package distributed to customers. The recipe should be considered as the foundation for building tools to automate the upgrade process (3), and the customer need not be concerned about the nature of the recipe.

[0057] The present invention specifies a number of steps that need to be covered in order to perform a successful upgrade of a network element (2), and every step needs to be covered by the recipe embedded in the software package. The steps involved in the upgrade process (3) are illustrated in FIG. 3.

[0058] The recipe is assumed to handle exceptional situations during every step of the upgrade process (3), but of course, it is practically impossible to cover every possible situation. In this case, a step will exit with a NOK result code as indicated in the figure, which aborts the upgrade process (3) and initiates rollback and cleanup activities if required. The latter depends on how far the upgrade has progressed before a critical incident occurs that the script cannot catch.

[0059] The steps required to automate the software upgrade process (3) are further described below. However, detailed execution of each step is outside the scope of this invention, since it largely depends on the management interface towards the network element (2).

[0060] Download Software to Network Element (4):

[0061] The download (4) of software to the network element does not necessarily need to be covered by the recipe to automate the upgrade, but the step is included here for completeness. The remote upgrade tool is responsible for implementing this step, most likely by using an available file transfer mechanism like FTP, NFS, or the like. The step itself includes transferring the software that are to be upgraded from the file server connected to the remote upgrade station (1) to the file system of the network element (2).

[0062] Software packages are likely to be distributed in ZIP or TAR format. The download step (4) needs to inflate the package in some designated directory on the remote upgrade station (1), and to initiate transfer of the software files to the network element (2).

[0063] By default, a delta of the software should be transferred to the node to optimize the bandwidth usage. Delta transfer means that components already existing in the node file system are not transferred. As an option, the operator may choose to transfer the complete set of software to the node. All components will then be transferred, even if they already exist in the node file system.

[0064] Pre Validate Hardware Configuration of Network Element (5):

[0065] There may be dependencies between the software modules to install and the hardware configuration of the network element (2) to upgrade. E.g. the software may require that a special I/O processor is installed or a higher stepping of the main processor.

[0066] Software/hardware dependencies should be checked prior to initiating any further upgrade activities, since they will ultimately force the upgrade to backtrack. A step of the recipe is designed to check these software/hardware dependencies for the software modules included in the package distributed.

[0067] The check is performed to assure that the new software is loadable in the current hardware configuration. The software configuration states the HW compatibility window for the delivered software. The compatibility window will be compared with the installed HW revision.

[0068] The check function will return OK/NOK status. If the check fails, the recipe will not continue unless the operator overrides the result. The operator is warned by the system that overruling of the negative result may lead to traffic disturbances.

[0069] Pre Validate Software Configuration of Network Element (5):

[0070] The step of the recipe is performed to assure that that the software in the package and the current executing software in the network element (2) are compatible, i.e. it is possible to perform a data conversion. This is done by comparing the software revision compatible window given by the configuration with the software revision of the running system.

[0071] In the situations where not all software modules on the network element (2) are upgraded, it is essential to check that the revision of the modules that are left untouched will work together with the upgraded modules. This is referred to as software-software dependencies.

[0072] The check function will return OK/NOK status. If the check fails, the recipe will not continue unless the operator overrides the result. The operator is warned by the system that overruling of the negative result may lead to traffic disturbances.

[0073] Pre Health Checks of Network Element (6):

[0074] This part is very specific for each network element (2) type, but generally includes checking the current status of the network element (2), and possibly taking the appropriate countermeasures to ensure that status is as required for the upgrade to progress.

[0075] The health check step normally includes checking the alarms pending for this network element (2) and any records logged in the syslog. It is also common to check the current load of the network element (2) to see if it is feasible to progress. At this point, the upgrade process (3) has not affected the network element (2), and consequently there is no need for rollback if the step returns a NOK exit status.

[0076] Backup Existing Software on Network Element (7):

[0077] This step includes backup of the software that is to be upgraded. A backup may be performed by saving the HW and SW configuration settings, or by dumping the entire memory. Naturally, the latter option consumes far more space in the network element (2) file system.

[0078] The existing software may either be stored on the network element (2) file system, or the remote upgrade station's (1) file server. Storing the backup on the remote upgrade station (1) provides additional safety against data loss during the upgrade, since the network element (2) may become unavailable. However, remote storage also comes at a cost, since the transferring of the backup over a remote connection will consume bandwidth and may take hours to complete over a slow connection.

[0079] The normal procedure to handle failures during the upgrade procedure is that a return to the backup is made by the network element (2), by reloading the backup. There can be occasions when it is desirable not to return to the backup and instead make a forced reload with the new software.

[0080] Obtain Exclusive Access to Network Element:

[0081] This step includes actions required to obtain exclusive network management access to the network element (2) that is target for the upgrade. Only the remote upgrade station (1) should be allowed to manage the network element

(2) during the upgrade. For instance, all statistics collection should be turned off during the upgrade.

[0082] If the software management application does not have exclusive access during the upgrade, it is practically impossible to automate the process, since other applications may execute management operations on the network element (2) that is not included in the recipe. External applications will introduce non-deterministic behavior into the upgrade that will be almost impossible to catch in the recipe.

[0083] Data Conversion on Network Element (8):

[0084] Before loading the new software, data conversions may have to be made. Data conversion may differ in complexity depending on the support provided by the network element (2). Necessary data conversion instructions for conversion of data structure and format are provided in a separate step in the recipe.

[0085] Old values will remain where appropriate. If a parameter value has to be recalculated, the conversion step handles this. In case of a new parameter (due to structure conversion), the parameter will get a default value from a default value in the software configuration. The operator should be able to change the supplied default values to site-specific values.

[0086] Activate the New Software on Network Element (9):

[0087] Activating the new software modules may differ in complexity depending on the support provided by the network element (2). A separate step in the recipe should supervise the loading of modules from the network element (2) file system onto main and special processors. Note that some elements may also have additional processor types.

[0088] There are typically a number of different exceptional situations that may arise during activation, simply because it is a complex task to upgrade network elements (2) that is in operation. Problems or deficiencies that have not been discovered earlier, will often surface during activation. For this reason, the activation step needs to focus on exception handling, and support rollback of activations that could not be completed.

[0089] The order in which the load modules shall be loaded depends on interface compatibility rules. The step will define instructions to keep the interface compatibility. The step also contains instructions of where, i.e. which target processors the load modules belongs to, and the order in which the load modules shall be activated.

[0090] The step could provide checksum control during software loading.

[0091] Release Exclusive Access to Network Element:

[0092] This step includes releasing the exclusive access obtained earlier, and restarting any network management activities that was running prior to data conversion and activation of the network element (2). This may be implemented by having the "obtain exclusive access" step of the recipe save the activities that were running in some temporary storage, and then re-invoke the saved activities in this step.

[0093] Post Validate Software Configuration of Network Element (10):

[0094] A step is included to check that the software configuration performed after the upgrade complies with expected values. This is to ensure that the correct software revisions have actually been installed. The step is similar to the software validation performed prior to activation, but if the validation fails, the activation should be rolled back. For this reason, rollback functions needs to be provided in this step.

[0095] Post Health Checks of Network Element (11):

[0096] The post upgrade health checks are similar to the checks performed prior to the upgrade, but in this case, there may also be a need to rollback the activation if discovered that the network element (2) is not working properly.

[0097] Cleanup Temporary Files in Network Element File System (13):

[0098] During upgrade, a number of files may be temporarily stored in the network element (2) file system, and to resolve this situation, a cleanup step should be included in the recipe to finalize the upgrade process (3).

[0099] The present invention is intended for software upgrade of network elements, which may be seen as a special-purpose computer for use in communication networks. However, there is nothing stopping it from being applied for software upgrade of general-purpose computers. In fact, network elements that realize application functions like unified messaging in next generation networks will be general-purpose computers.

[0100] Abbreviations

- [0101] CORBA Common Object Request Broker Architecture
- [0102] FTP File Transfer Protocol
- [0103] HW Hardware
- [0104] NE Network Element
- [0105] NFS Network File System
- [0106] O&M Operation and Maintenance
- [0107] SNMP Simple Network Management Protocol
- [0108] SW Software
- [0109] TAR Tape Archive
- [0110] ZIP ZIP is a common archive format

1. Method for installation of new software or upgrade of existing software by new software in workstations or network elements in a distributed data or telecommunication system, characterized by an automated process (3), controlled by a certain workstation (1), including the following sequential steps:

validation of hardware configuration (5) of said workstations or network elements (2), where said validation is performed to assure that said new software is loadable in said hardware configuration, finishing said process (3) if the validation (5) is not carried through successfully,

first validation of software configuration (5) of said workstations or network elements (2) prior to activation of new software, where said validation is performed to assure that said new software and said existing software are compatible, finishing said process (3) if the validation (5) is not carried through successfully,

first condition check (6) of said workstations or network elements (2) prior to activation of new software, including necessary checks and measurements to ensure that status is as required for the upgrade process (3), finishing said process (3) if the condition check (6) not is carried through successfully,

carrying out actions providing the certain workstation with exclusive network management access to said workstations or network elements (2),

activating new software (9) on said workstations or network elements, returning an OK value if carried through successfully and a NOK value if not.

releasing the exclusive network management access of said certain workstation

second validation (10) of software configuration of said workstations or network elements (2) after activation of new software, where said validation is performed to assure that said new software correctly has been installed, returning an OK value if carried through successfully and a NOK value if not

second condition check (11) of said workstations or network elements (2) after activation of new software, including necessary checks and measurements to ensure that status is as desired in a normal state,

cleaning up (13) temporary files that may have been temporary stored during installation or upgrade.

2. Method as defined in claim 1, characterized in an additional step (4) prior to the step concerning validation of hardware configuration (5), where said new software which is to be installed or used for updating is transferred from said remote workstation (1) to the workstations or network elements (2) of the distributed data communication system.

3. Method as defined in claim 1 or 2, characterized in an additional step (7) prior to the step concerning actions required to obtain exclusive network management access and after the step concerning condition check (6), where backup of said existing software and configuration data are accomplished and stored either on the network elements/workstations or on the remote workstation (1).

4. Method as defined in any of the preceding claims, characterized in an additional step (8) prior to the step concerning release of the obtained exclusive network management access and after the step concerning actions required to obtain exclusive network management access, where required data conversion including recalculation of values and introduction of new parameters in the existing software is accomplished.

5. Method as defined in any of the preceding claims, characterized in that the step concerning the second condition check (11) includes checking the current load, the alarms, or any logged records pending for each of the workstations or network elements (2) in the distributed data or telecommunication system.

6. Method as defined in any of the preceding claims, characterized in an additional step (12) prior to the step

concerning upcleaning **(13)** initiated only by a NOK value from the step concerning activation of the new software **(9)** or from the step concerning second validation **(10)** of software configuration, where the process **(3)** is rolled back,

which includes undoing all changes to the network elements **(2)** or the workstations up to this point in the process **(3)**.

* * * * *