ORYA: A Strategy Oriented Deployment Framework

Pierre-Yves Cunin, Vincent Lestideau, Noëlle Merle

ORYA: A strategy oriented deployment framework

Pierre-Yves Cunin[1], Vincent Lestideau[1], Noëlle Merle[1]

[1] Adèle team, LSR – IMAG
220 Rue de la Chimie, Domaine Universitaire – BP 53, 38041 Grenoble Cedex 9, France
{Pierre-Yves.Cunin, Vincent.Lestideau, Noelle.Merle}@imag.fr
http://www-adele.imag.fr/

Abstract. The current trend consists in deploying, on each machine, a specific version of an application, according to the choices of the enterprise and users, with constraints verified by the target site. To support automated deployment, we propose a model-based deployment framework named ORYA which allows to define and execute deployment strategies. This paper presents and illustrates the concept of deployment strategy supported by the framework.

1 Introduction

Various approaches exist to deploy an application on a set of target machines. One possibility is to create a deployment plan and then to execute it. To produce automatically this plan, we define models which describe *units* to deploy, *target machines* and *enterprise structure* [1]. The *application model* defines the deployment unit (a version of an application) with properties, constraints and dependencies. The *site model* describes the hardware and software configuration of a target machine with properties. The *enterprise model* collects machines into groups and subgroups.
A *property* describes a feature of a unit or a machine. A *constraint*, associated to a deployment unit, expresses a property the target must have. A *strategy*, attached to an enterprise entity (group, machine), expresses a constraint imposed by the enterprise.
Section 2 presents fundamental aspects of our strategy-based approach.Section 3 presents a use case. Section 4 concludes with future works and objectives.

2 Deployment strategies

Large scale deployment is a complex action that cannot be done by hand. Often the deployers use in-house defined deployment strategies to ensure the right quality level of operation (security, homogeneity, standards, …). In some approaches strategies are included (hard coded) within the deployment tools [2]. Therefore a deployer cannot define new ones, better adapted to his needs. Our objective is to help deployers expressing advanced deployment strategies and to provide a framework for piloting strategy-based deployments. An outcome will be a new version of our deployment environment ORYA [3, 4, 5] based also on the GDF experiment [6].

2.1 Approach and algorithm principle

We assume that the strategies are expressed only on sites and groups. Strategies belong to the enterprise and therefore are attached to the entities of the enterprise structure. Each strategy is applied to the current set of deployable units.

A strategy is a 3-uple <*LogicalExpression*, *Activity*, *Choice*> The *Activity* specifies one phase of the deployment. In this paper we consider only the *Initial Deployment* phase. During the *Activity*, the *LogicalExpression* is evaluated for all the current deployable units, i.e. the current *application structure (AS)*. This gives two sub-sets: the "*true* set" and the "*false* set". Then the *Choice*, its associated actions, is applied to these two sets depending on the semantics of the strategy. The result is an *AS* made of the remaining deployable units.

There exist many strategies, for example: enforce the same version on a set of machines, allow replacement of a version by a newer one, favor the deployment of a unit having some characteristic (e.g. choose a unit written in Java instead of the same in C++), deploy the dependencies of a unit before the unit itself, deploy a unit on a group of machines before on another one, roll back during the execution of the plan, due to a change of the environment (e.g. the needed resources are no more available).

The algorithm is a parsing of the *enterprise structure(ES)* with propagation of an *AS* through the whole structure. On each node, strategies are applied in order to prune the *AS*. On a machine node, the constraints of the units are checked.

Strategies can be classified in three main categories: strategies to select units having specific properties, strategies to define the ordering of the plan and strategies used during the execution of the plan (mainly to handle errors).

A strategy is defined by its basic behavior and the following features: 1) the **scope**: a strategy may be attached to a group or a single machine, 2) the **visibility**: a strategy attached to a group may or may not hide - may or may not be overloaded by - any similar strategy expressed on a sub-node, 3) the **propagation**: a strategy attached to a group may impose collecting information about the sub-nodes, 4) the **precedence**: several strategies may have to be applied at the same time on the same node.

2.2 Strategies *VERSION-RIGHT* and *VERSION-SCOPE*

To illustrate some characteristics, we focus on two strategies .

1. Strategy VERSION-RIGHT is attached to a group or a single machine and can be applied without additional information (e.g. from sub-nodes, if any). If Choice is NO, units of the "true set" cannot be deployed on the machine(s of the group) and the resulting *AS* is made of the "false set". If Choice is ONLY, only units of the "true set" can be deployed on the machine(s of the group)and the *AS* is made of the "true set".

2. Strategy VERSION-SCOPE. is a complex strategy used to ensure coherence on versions deployed on all the machines of a group. The semantics of the strategy depends on Choice: a) if ANY, each machine may have a different version and the units of the "false set" are discarded. b) if SAME-TRUE, the units of the "false set" are discarded and one same unit, of the "true set", should be deployed on all machines and should be compatible with the configuration of each machine. c) other values are

possible, for example *SAME-IF-TRUE* means that each machine may have a unit of the "*false* set" or the same unit of the "*true* set".

The application of the strategy is different for each value of *Choice*: a) if *ANY*, the strategy is immediately applied at the level of the group node and the new *AS* is equal to the "*true* set". b) if *SAME-TRUE*, the "*true* set" is propagated as *AS*, through a recursive parsing of the *ES*, together with a query about what units of this set can be deployed . When this information is made available at the level of the group node, the *AS* is constructed as the set of the units deployable on every machine. During this recursive parsing local strategies *VERSION-SCOPE* or *VERSION-RIGHTS* on sub-nodes have to be applied before treating the "propagated" query and set of units

3 Use case

The two representations structures are shown in Fig. 1. The *ES* represents the target on which to deploy. The *AS* represents possible units, with their characteristics and dependencies. The deployer wants to deploy the application *U* on the machines of the group *G*. *G* is composed of two groups *G1*, composed of machines *M1* and *M2,* and *G2*. *G2* contains the machine *M3* and the group *G3*, itself composed of machines *M4* and *M5*. The machines have properties specifying operating system (*OS*), memory capacity (*Mem*) and available disk space (*Disk*). Strategies *VS* (*VERSION-SCOPE* strategy) and *VR* (*VERSION-RIGHTS* strategy) are defined, on nodes *G* and *G1*.

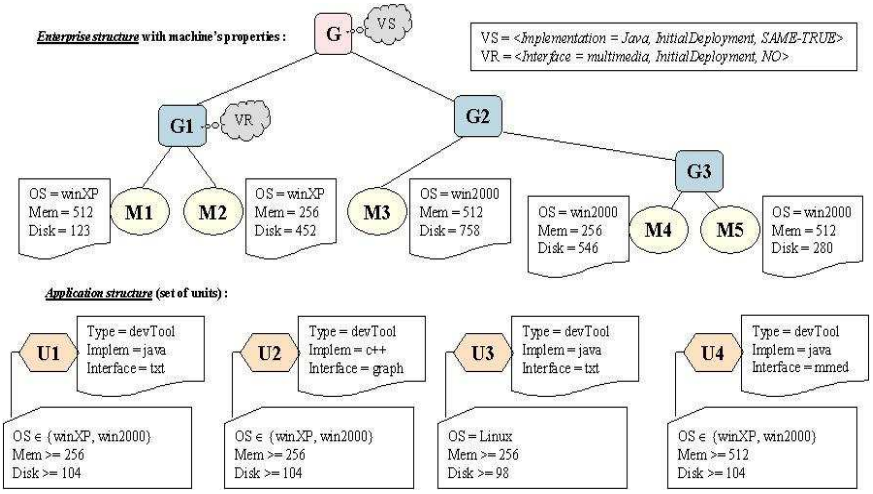

Fig. 1. *Enterprise structure (ES)* and *Application structure (AS)*

The application *U* is available in four versions, described by properties: type, programming language, interface type. Each unit forces constraints : a set of possible operating system, a minimal memory capacity, a minimal available disk space.

On *G*, the application of the strategy *VS* requires information from the sub-nodes. The "*true* set" of units *{U1, U3, U4}* is propagated to *G1* and *G2*. **On *G1***, the strategy VR is applied and the set *{U1, U3}* is propagated to *M1* and *M2*. Then the constraints are checked and *U3* is discarded because it imposes *Linux* as *OS*. So the set *{U1}* for *M1* and *M2* is sent back to *G* through *G1*. **On *G2***, the set *{U1, U3, U4}* is propagated to *M3* and *G3*. The set *{U1, U4}* for *M3* is sent back to *G* through *G2*. **On *G3***, the set *{U1, U3, U4}* is propagated to *M4* and *M5*. The sets *{U1}* for *M4* (*U4* is discarded due to the memory capacity) and *{U1, U4}* for *M5* are sent back to *G* through *G3* and *G2*. **Back to *G***: the strategy VS is finally applied and the *AS* is build as the intersection of the sets of all the machines: *{U1}*. Therefore, in that example, only this unit could be installed on all the machines.

4 Future work and objectives

We have defined and prototyped a design and execution framework. A set of basic strategies has been defined The approach has been validated through real size experiments [7] with simple strategies.

In the example we have not taken into account the dependencies that may exist for each unit. Dependency units are units themselves. Trying to apply strategies to dependencies introduces "meta" strategies, e.g.: should a strategy, applied to a unit, be also applied to its dependencies ? Should we evaluate the *LogicalExpression* of a strategy on (all) the dependencies of a unit ? Should we consider dependencies as being standard units on which apply the strategy algorithm ? The approach we use is an MDE (Model Driven Engineering) compatible one based on three interacting levels: strategy instances, strategy model and strategy metamodel (meta-strategies).

References

1. Merle N., Un méta-modèle pour l'automatisation du déploiement d'applications logicielles. DECOR'04. Grenoble, France. Octobre 2004.
2. Ayed D., Taconet C., Sabri N., Bernard G.: CADeComp : plate-forme de déploiement sensible au contexte des applications à base de composants. 4ème Conférence Française sur les Systèmes d'Exploitation (CFSE'05). Le Croisic, France. 5-8 avril 2005
3. Lestideau V., Belkhatir N., Cunin P.-Y.: Towards automated software component configuration and deployment. PDTSD'02. Orlando, Florida, USA. July 2002.
4. Lestideau V.: Modèles et environnement pour configurer et déployer des systèmes logiciels. PHD Thesis, Université deSavoie, December 2003, http://www-adele.imag.fr/Les.Publications/BD/PHD2003Les.html
5. Merle N., Belkhatir N., Open Architecture for Building Large Scale Deployment Systems The 2004 International Conference on Software Engineering Research and Practice (SERP'04), Las Vegas, Nevada, USA, June 2004
6. On-demand Service Installation and Activation with OSGi. ObjectWebCon05 : Fourth Annual ObjectWeb Conference. January 2005,Lyon, France.
7. Centr'Actoll web site : http://www-adele.imag.fr/Les.Groupes/centractoll/index.html