Towards Transparent Architectural Decisions for Software Deployment

Balthasar Weitzel
Fraunhofer IESE
Fraunhofer-Platz 1, 67663 Kaiserslautern
Email: balthasar.weitzel@iese.fhg.de

Abstract:
The operation of large scale information systems requires investment into hardware infrastructure and causes running cost for keeping it in a productive state. This especially applies in an enterprise environment where also expenses for software licenses costs or penalties for downtime occur. The deployment of software influences these costs both in their amount and their composition. In order to optimize them a transparent view on these costs and the deployment is mandatory. In this paper, we present a conceptual model of deployment. The model is populated by reverse engineering of deployment descriptors but as well uses runtime traces and usage profiles. We envision – having both made explicit on an architectural level – a comprehensive decision making and optimization of software deployment.

1. Introduction

The development of software system (i.e., writing high-quality source code) is a challenge – but only one side of the coin. The other side of the coin – and nonetheless challenging – is the operation of the software system.
Operating a software system causes cost of different types. There are fixed costs like procurement or replacement of infrastructure. Additionally running costs for the operation as such in terms of energy, staff and software license cost, but also penalties for downtime may apply if certain quality levels of service are not achieved.
Different deployment alternatives influence both the total cost but also individual matters of expense. For example, the decision about the allocation of computation effort in a distributed information system can significantly raise or lower the investment costs for a central server. Considering cloud services, the running costs can be varied significant.
Such decisions are made on an architectural level in conjunction with various others architectural concerns. Architecture defines the boundary for possible deployment alternatives and enables to reason about their impact. For achieving such a holistic reasoning it is necessary to connect the deployment view of the systems architecture to other views, such as structural or behavioral ones.
When optimizing existing systems, deployment decisions already made have to be handled and balanced with potential migration effort. Historic data regarding operation characteristics and usage profile are available.
We envision that having an integrated view on the relevant aspects of deployment facilitates conscious architectural decision making. The planning of migration or change effort is supposed to be more reliable.
Our experiences in industry show a limited awareness of the influence of deployment on operation cost. Nevertheless, nearly all industry partners are able to tell a story where such a decision caused high cost (e.g., replacing a third party component to reduce license cost due to usage on multiple servers or enhancing network connections to locations to handle the traffic a new feature introduced). Decisions are typically made implicitly or have been deliberated once (at the time of the first release) and remained unchallenged since then (although the system evolved).

2. Related Work

Deployment on architectural level is represented sparse in literature. An example is presented in [1], where the distribution of the software is in focus. Analysis of cost factors is not considered, the overall goal is to present a specific technology.
The need to capture the complex aspects of building a software system is mentioned in [4], with a focus on the linkage of development and deployment view.
An overview of deployment technologies is presented in [2], where again the infrastructures used to realize specific deployment alternatives are considered.

3. Conceptual model of deployment

We defined a conceptual model of deployment on an architectural level that is very close to the UML notation of deployment aspects. The core elements of the model and their relationship are depicted in Figure 1. In general deployment is about mapping

the *software elements* onto *hardware elements*. This distinction is represented by the two abstract software and hardware model elements which are further refined. On the hardware side there are *nodes* which are *physically linked* to each other or *grouped* together. The part of the model that represents the software is divided into *software execution environments* (like operating systems, application servers, database management servers, etc.) and the *executables* that are handled by them. An executable contains one or more modules and can thus also have *logical links* to others.

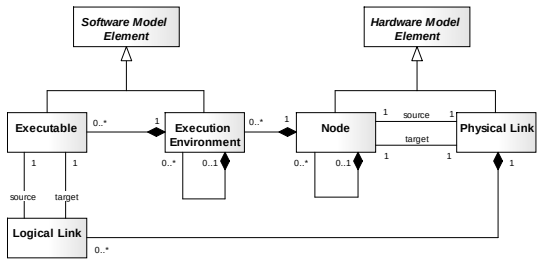

Figure 1: Core Conceptual Deployment Model

The connection between hardware and software is on the one hand represented by having the execution environment deployed on a physical node. On the other hand there are logical links between executables that are mapped to physical links between nodes.

Having this conceptual model cost factors can be introduced with the concept of profiles for expressing characteristics of a specific element. We identified four types of *profiles* that can hold expected or measured data:

- *Module Resource Utilization*: Load that a component causes
- *Executable Resource Utilization*: Aggregation of load caused by modules of an executable
- *Resource Capability*: Capacity of a hardware resource
- *Software Usage*: Typical usage behavior of a part of the software system by a group of users

4. Envisioned support for architects

Our goal is to support decision making on basis of an explicit deployment of a software system. In case of optimizing existing systems we can analyze historical usage data of the system. This enables us to provide the software usage profiles with heuristics about the typical usage behavior.

Having also runtime traces at hand will give the possibility to populate the three other types of profiles with data (module and executable resource utilization, resource capability). This highly depends on the granularity of these traces.

Finally analyzing deployment descriptors and thus reengineering the current deployment of the system allows giving an architectural view on the deployment which is enhanced by sound information gained from the past experience.

It is not mandatory to reveal these facts by automated reverse engineering tools, a manual modeling based on expert knowledge should be possible, too.

For optimizing a deployment an architect has the possibility to develop new deployment alternatives by analyzing the existing one. If the reduction of cost is in focus main cost drivers can be identified easily and their cause could be traced back to an architectural decision.

Alternative deployment options can be defined based on the model that represents the current deployment of the system. Such new options can consist of changes in the software itself, in the hardware landscape or in the mapping between them. The characteristics of such an alternative can be evaluated based on the different profiles that are available in the model and that are based on historic experience. Such characteristics are on the one hand related to resource demands but the main focus is on cost behavior.

If the model and the profiles are tool supported an easy explorative way of finding and automatically analyzing new deployment alternatives can be chosen. A virtual “monitoring” of the intended deployment option gives direct feedback on any changes that are performed.

5. Research Agenda

The conceptual model we presented needs to be evaluated by applying it in industrial case studies to ensure that it is expressive enough to represent multiple technologies that are used but at the same time is simple enough to be accepted in practice. Especially the concept of profiles is at an early stage and needs further refinements. To facilitate the application of the approach tool support is needed, currently only a prototypical realization of reverse engineering of deployment descriptors has been realized.

We received promising feedback when presenting our vision to industrial stakeholders, which motivates us continuing this line of research.

References

[1] Malek, S. et al.: An Extensible Framework for Improving a Distributed Software System's Deployment Architecture; Software Engineering, IEEE Transactions on; 2012

[2] Carzaniga, A. et al: A Characterization Framework for Software Deployment Technologies; Technical Report; Dept. of Computer Science, University of Colorado; 1998

[3] Object Management Group: UML 2.0 Specification; 2005; http://www.omg.org/spec/UML/2.0/

[4] Tu, Q.; Godfrey, M.W.; The build-time software architecture view; Proceedings of ICSM; 2001