The Forgotten Viewpoint: Operational

Eóin Woods
InterTrust Technologies
eoin.woods@intertrust.com

Nicholas Rozanski
Cap Gemini, Ernst and Young
nick.rozanski@capgemini.co.uk

Abstract

Information systems are only useful when they are operational and running well. This implies that the operational concerns of a system should be high on an architect's list of priorities.

Our experience is that architects often don't take operational concerns terribly seriously and those that do often struggle to express their strategies to address these concerns in a consistent way.

Following the viewpoint oriented approach widely used by software architects, we propose a new viewpoint to help address this situation.

We call the viewpoint "operational" and it addresses a number of related concerns around the process of deploying, supporting and managing a system in production.

This paper provides a rationale for the viewpoint and explains its contents.

1. Introduction

Systems are built to run. This seems like an obvious statement to make. However, you wouldn't think that this was the case judging by the amount of consideration of system operation found in most architectural descriptions for information systems.

Further, IEEE standard 1471 [1] states that two of the possible uses of architectural descriptions are "*communications among organisations involved in the development, production, fielding, operation and maintenance of a system*" and "*operational and infra-structure support*".

Both of these facts suggest that any architectural description needs to consider operational concerns seriously.

For all but the simplest systems, installing, managing and operating the system is a complex exercise requiring skilled work from many people. In order for this process to be achievable, it must be considered at system design time. If the process isn't considered, at this point, then there is a high probability that the process of deploying and running the system in production will be a painful one for all concerned.

In order to address the concerns of system operation (in its widest sense) we propose the recognition of what seems to be a forgotten viewpoint: the *Operational Viewpoint*.

In the rest of this short paper, we outline the proposed content for the operational viewpoint. Following IEEE 1471, we define the viewpoint's name (and rationale), the stakeholders who's concerns it addresses, the concerns addressed and the techniques used to create such a view.

2. The Operational Viewpoint

Implementing a complex system is only part of the battle. Having developed a system, it has to be installed and configured; organisational processes have to be migrated to use it; it needs monitored, controlled and operated in order to provide an acceptable level of service to its user community.

Given today's (eminently sensible) move towards incremental system development and delivery, this problem becomes even more acute. Rather than having a few large installation cycles where the whole system is installed and left to run, we now have a situation where deploying software from development to production is a routine part of software development life.

The process of system deployment is a complex one, requiring input from a number of skilled stakeholders including the development team, representatives of the end users, the systems administration team and representatives of specialist groups such as security and IT audit. This process is not going to be successful unless all of these stakeholders believe that their concerns are being met.

In our experience, it is rare to find a coherent and comprehensive discussion of operational concerns in an architectural description. Many architects seem to view it as "someone else's problem" and largely omit discussions of these concerns from their architectural process and the resulting description. This sort of approach is unlikely to result in the enthusiastic cooperation

between stakeholders that is required for successful system deployment.

Where operational concerns are addressed, typically they are scattered throughout an architectural description, reflecting the point at which they occurred to the architect rather than reflecting their relationship to each other.

There are a number of problems that can result from this situation.

- Stakeholders with operational concerns are often not convinced that the architect understands their concerns.
- Often potential operational problems are overlooked because they cannot be viewed and analysed via a single coherent description.
- System deployment is often complicated by misunderstandings between development and administration groups, again because of the lack of an effective communications vehicle.

Our proposed solution to these problems is to define a viewpoint called the "operational viewpoint".

For a typical information system, the primary operational concerns include *installation*, *migration*, *support*, *operational monitoring*, *operational control*, *configuration management* and *performance monitoring*. Therefore, the operational viewpoint that we propose attempts to demonstrate that concerns in all of these areas are understood and addressed.

Although much of the discussion here is primarily in terms of turnkey information systems, implementation and operation is also a concern for product developers who are developing software products intended for use within such systems. Too many software products are initially released in an almost unusable state because of the difficulty of implementing and operating them within their intended deployment environment.

Therefore, while throughout this paper, we focus primarily on turnkey systems rather than the software products we firmly believe that the viewpoint we describe here is equally applicable to software *product* architecture. It is really just a matter of some of the stakeholders being one step removed, as they work in a customer organisation rather than in the development organisation.

3. Stakeholders

There are three main groups of stakeholders who have concerns that are addressed by this viewpoint.

The first group of such stakeholders are the *system administrators* who are tasked with installing the system, migrating the user organisation to use it, monitoring it and maintaining its operational status. There are many subtypes of "administrator" each with their own specialisation (operating systems, databases, networks, security, auditing etc.). However, for the purposes of these views we group all of these specialisations together as they have similar overall concerns. These stakeholders are concerned with the system's form as a production entity. They have many questions about the system. How will the installation happen? What dependencies exist between parts of the system? What dependencies exist between the system and other technology components? What mechanisms are available for monitoring the system? What mechanisms are available to operate the system? What routine maintenance is required? (and when?)

Another important group are the *software developers* who are developing the system. This group of stakeholders need to make decisions throughout development relating to the operation of the system. Where do they put trace code? How do they collect performance counters? What commands or scripts are required from them? How do they write log messages? What operational constraints exist that constrain their design choices? This view needs to clearly define all of the operational aspects of the system that impact developers to ensure a consistent implementation that can be deployed and operated in the target production environment without rework being required.

The system's *end users* are a large group containing a number of types of user. Some users interact with the system directly (e.g. call centre staff who must use the system to record customer calls and retrieve information to help them answer questions). Others users don't use the system directly, but are affected by its existence and the functions it offers (e.g. executive management who must take the availability of system functions into account when planning their business operations). However, in the context of this view, all of the end users are similar in as far as they are concerned with a small subset of the concerns reflected in this view. Their concerns do not centre on the complex technical aspects of the system that the previous two groups find so fascinating. Rather, this group of stakeholders need to know what functions will be available when and how they make use of them.

4. Concerns Addressed

4.1. Overview

As with any viewpoint, the concerns it addresses relate to the concerns of the stakeholders who the viewpoint is aimed at.

The concerns addressed by this viewpoint are installation, migration, support, operational monitoring, operational control, configuration management and performance monitoring.

The relationship between the concerns addressed and the stakeholders is illustrated in Table 1.

	Systems Admin.	**Developer**	**User**
Installation	X	X	X
Migration	X		X
Support	X		X
Op. Mon.	X	X	
Op. Control	X	X	
Cfg. Mgmt.	X		
Perf. Mon.	X	X	

Table 1 – Primary Stakeholder Concerns

As you might expect, the two technical stakeholder groups are likely to be interested in most or all of the concerns that this viewpoint addresses. Having said this, they have somewhat different perspectives on the concerns. For example, the software developer is concerned with how to *write* log messages to allow his component to be operationally monitored, while the systems administrator is concerned with how she *reads and filters* those messages in order to monitor the component.

As already mentioned, the user stakeholders aren't very interested in the details of monitoring, control etc. They are interested in some key points relating to installation and migration. Namely, what is installed and made available when, how do they use it and what is the impact on them of migration to the new software.

The following subsections discuss these concerns in more detail.

4.2. Installation

The *installation* concerns centre on the process of installing the system on the production machines. This is a task performed by the systems administrators. For this set of concerns, an operational view needs to define:

- The delivery packages that the development team will deliver to the systems administrators.
- The dependencies between the delivered packages and other technology components.
- Volumetrics which can be used to guide administrators when sizing their production machines.
- The installation tasks that the administrators will need to perform to install the software.
- The verification activity that the systems administrators can perform to check the installation.
- The installation tools that the systems administrators can rely upon the development team to deliver.

It is important to note that the view should not be an installation manual and does not contain detailed, step-by-step instructions. If such instructions are needed later, then they can be provided in a separate document. Rather, this view provides enough information to allow the systems administrators to understand the overall form of the system from their perspective to allow them to plan for its installation.

From the user perspective, they are primarily interested in the first of these points – what is being delivered? When?

4.3. Migration

The *migration* concerns are related to how the users of the system start using it. If this is the first release, the migration may be from a manual system or from nothing. If this is a new release, perhaps minor changes to working patterns for end users will be sufficient migration. The migration concerns are primarily of interest to the system administrators and the end users. An operational view needs to define the following aspects of migration:

- What has changed? The key question for administrators!
- Will any form of parallel running be used? If so, how will the system images be kept synchronised?
- Does data need transformed or moved for the migration? If so, where does it come from and how is this done?
- Does the new system have different resource requirements? If so, how will a migration between the two sets of resources be done?
- How will users of the system be trained in its new features (the whole system if it is a new one)?
- What training does the systems administration team need in order to

accept the system from the development team?

- Are special operational procedures or monitoring required during migration?

As with the installation concerns, the architecture does not need to define the migration plan but it does need to convince the stakeholders that such a migration is possible!

4.4. Support

By *support*, we are referring to the overall process of assisting stakeholders with their use of the system. This includes helping end users with routine use of the system (e.g. helpdesk type functions) and assisting systems administrators to resolve queries and problems with the system that they encounter when running it (e.g. technical support type functions).

It is clear that both of these affected groups of stakeholders are going to have requirements for the support available to them.

In order to allow the system to be used and operated successfully, it is important that the architect identify what support is going to be needed, who is going to provide it and how it is going to be provided.

Concerns in this area could include:

- How do end users obtain assistance when they run into problems using the system?
- What is the escalation procedure when a problem is found that the initial support providers cannot resolve?
- How do systems administrators obtain assistance when they encounter problems with operation of the system or components within it? Is the process different for system specific and third party components?
- Who is to provide the support?
- What service level agreement is required for the support provided?

The architectural description should include a clear strategy for providing support to end users and system administrators as well as identifying responsibilities and setting appropriate expectations for the level of support to be provided.

4.5. Operational Monitoring

Operational monitoring concerns focus on how the system will be monitored by its administrators when running in production. Such monitoring is critical for many systems to allow appropriate administrative action in response to critical conditions such as component failures, resource shortages etc. The operational monitoring aspects of the system's architecture are of direct interest to the development team and the administration team. The development team are concerned with the "*how*" – that is, how do they make their system components easy to monitor and what standards should they use when doing this. The administration team are interested in the "*what*" – what is monitored, what should they look for and what should they do in response to certain conditions being recognised.

An operational view of a system's architecture needs to define at least the following aspects:

- An overall system wide strategy for monitoring system components (what sort of events need monitored, what sort of technology will be used to monitor them etc.).
- The classes of event need to be monitored when the system runs in production.
- How specific components will communicate that certain events have occurred (e.g. use of log messages vs. trace messages vs. SNMP traps).
- How events that components cannot report will be recognised (typically component failures).
- Standards for various event communication mechanisms (e.g. log standards relating to log levels, formats etc., standards for "is alive" messages that components must respond to and so on).
- How it will be possible for administration staff to recognise that events have occurred.
- Any requirements for integration into 3rd party monitoring facilities.

In this context, the architect needs to identify appropriate strategies and ensure that suitable standards exist to guide the development process so that all system components can be effectively monitored in the production environment.

4.6. Operational Control

The *operational control* related needs of a system tend to fall into two main groups, routine operational control and control operations relating to failure containment and rectification.

Routine operational control is needed to perform routine maintenance tasks such as starting and stopping system components, managing system resources, performing reconfiguration of components, managing the system security model and so on. Often routine maintenance for a system is primarily

performed using the software utilities supplied with the technology components that the system is built upon (e.g. DBA utilities for databases, administration consoles for application servers and so on).

Failure containment and rectification tools are needed when something has gone wrong. Systems administrators use these tools to diagnose problems, identify a suitable solution and then to rectify the problem by changing the system configuration in some way. Many of these tools are also the utilities supplied by the technology platform suppliers. However, a typical information system will also need its own tools in this category. Given that a system failure is often in reality the violation of one or more internal system invariants, realistically there is no way that generic third party tools can be used to identify and diagnose such conditions. Architects need to make sure that when these conditions occur in their systems that they can be identified and rectified whether by manual procedures or via automated tools.

The system's operational view should include at least the following aspects relating to operational control:

- An overall system strategy for routine operational control.
- An overall system strategy for failure situation operational control.
- An identification of the classes of routine maintenance that will be required and the approach to be used for each.
- An identification of the classes of failure, which will require operational control to rectify them and the approach to be used for each.

Once again, the architecture should concentrate on the system wide strategy, principles and standards to be used for operational control rather than detailed procedures or tool specifications which should be designed by the system administrators and the software designers respectively.

4.7. Configuration Management

The *configuration management* aspects of the operational view relate to managing the configuration of the system in its operational environment.

A modern information system is a complex environment typically containing a number of inter-related components, each of which needs configured to meet the needs of the system. To make matters worse, the configuration of different parts of the system is often interdependent, complicating this situation still further.

The software architecture must address how the system will be amenable to production configuration management.

It is often necessary for a system to have a number of routine configurations to support different operational states (batch runs vs. online processing being something of a clichéd example). These configurations need to be stored in some way so that they can be applied to a system with the minimum amount of effort and disruption.

As well as routine configuration changes, systems often require configuration changes over their lifecycle in order to reflect changing requirements, changing operational environments and similar pressures. Such configuration changes are often much more wide ranging and are required less often than routine changes. However, it must still always be possible to reverse such a configuration change in the case of it having unplanned adverse consequences.

The management of these routine and exceptional configuration changes is the process of operational configuration management.

The details of operational configuration management are the domain of the systems administration staff. However, the architect needs to ensure that the system they are designing is amenable to the configuration management needs of this group of stakeholders.

The operational view needs to include the following aspects relating to configuration management:

- A clear identification of the groups of configuration information ("configuration groups") within the system.
- For each configuration group identified, how it is controlled (via a file, as a configuration set in a database, by storing it in an operating system structure, …).
- For each configuration group, how it can be managed as part of a larger, system-wide configuration set.
- A list of planned operational configuration profiles with an analysis of the groups of configuration information that will need to be changed to apply the profile to the system.
- For each operational profile, how the system-wide set of configuration groups would be applied and the impact on the users of the system (e.g. how configuration files get replaced and whether a restart is required for the components affected).

The result of this process should be a "configuration architecture" that allows the

configuration of the system to be managed in production. This set of guidelines and strategies should be carefully reviewed with the system administrators to ensure that they can implement configuration management procedures for the proposed system using the approach that the architect has proposed.

4.8. Performance Monitoring

Almost by definition, a system is too slow. No matter how carefully the performance aspects of a system are worked out, designed and tested, once it is running in a production environment pressures mount for it to run "faster".

Frequently, these pressures actually come from success. For example, if the system becomes more popular and more users use it, then response times often increase as the workload on the system rises. Similarly, good information systems often stimulate their own use by making their users more productive and so enthusiastic users who create more workload. This can particularly be the case with web connected systems.

In order to manage the inevitable performance problem, it is important that systems are built with performance monitoring in mind.

There tend to be at least three aspects to monitoring performance in a production system.

Firstly, there is the task of measuring *performance from the perspective of the system's users*. This typically takes the form of test programs that simulate user operations and measure the time taken for the system to complete their requests.

Secondly, there is the requirement to measure *resource usage*. How much CPU time is used by system components? How much memory is in use? What volume of network and disk I/O is being performed? In each case, it is useful to know which system components are using which resources.

Finally, there is the measurement of *performance from the system's point of view*. How long does it take the system to process a transaction from its arrival to the response being sent? How many times have certain performance related events occurred? (E.g., total client requests, I/O waits, queue overflows.) How long are certain key system internal operations taking? (E.g., response times from databases.)

These aspects of performance monitoring are obviously related and monitoring them on a routine basis is important to form the basis for performance engineering for the system. The aim should be to monitor these values on a regular basis during different operational states so that a performance profile of the system can be built up. Such a profile allows trends to be identified before they become crises, provide a basis for performance modelling and tuning and provide a context for the inevitable discussions with the user community about why the system is slower this week than last week.

Performance problems are rarely neatly confined to a single component. Given the number of components and the complex inter-relationships in a typical information system, it is important that a system wide view is taken. This allows performance monitoring to relate information from different parts of the system when tracking down system wide problems.

The operational view of a system's architecture needs to clearly define the following aspects relating to performance monitoring:

- What kinds of performance information are valuable for this system?
- Which components can be used as a source of performance information?
- How the required performance information will be extracted from system components.
- What relationships exist between the various performance information groups.
- How the performance information would be used to investigate a system wide performance problem.

Once again, the system's administrators are key stakeholders in this area. While the documentation resulting from this architectural area will guide the work of the development team, the final outputs (i.e. the performance measures) and strategies identified here need to reflect the needs of the administrators who will perform the monitoring of the system.

5. Techniques

Many proposed architectural viewpoints are described as some sort of system structure model (for example, the logical view introduced in [2] is typically a UML static class model, the module view introduced in [3] is a structural software module and so on).

However, system structure models are only one set of outputs of architecture. The other outputs should include strategies, guidelines and standards.

This viewpoint relates more to strategies, guidelines and standards than system structure models. Therefore the techniques relevant to this viewpoint are somewhat different to those

used for viewpoints relating to system structure.

We are still developing our ideas in this area and so we don't feel we have yet developed a definitive set of techniques to be used in developing and communicating this viewpoint. However, our experience of using this viewpoint suggests that a couple of modelling techniques described below are often useful when working in this area.

- *Information Modelling* – an information model is often necessary for migration planning if data migration is necessary. It can also be useful for describing configuration information and may be useful for illustrating relationships between sources of performance information.
- *Use Case Model* – a use case model can be useful to clarify the process and relationships within a support strategy.
- *Component/Layer Model* – a software component and/or layer model can illustrate the standard software components to be used for logging, tracing, performance counters and configuration sets. (Alternatively, this could be part of the development view and referenced from here). A component model can also be useful to describe the packaging of and relationships between system components for installation (if this is different to the packaging for development and deployment).

In addition to these software-like aspects of this viewpoint, the viewpoint should contain extensive textual description (or alternatively references to documents containing such description). The textual description should be used to describe strategies, guidelines and standards relating to operational concerns.

A suggested set of outputs from this viewpoint would be:

- *Installation Information*: A textual description of the installation approach that the system requires (see 4.2). Component model of installation components if this is useful.
- *Migration Information*: An information model for data migration if needed. A textual description of the migration strategy (see 4.3).
- *Support strategy and Responsibilities*: An explanation of the support strategy and a clear allocation of responsibilities for implementation (see 4.4). A use case model of the support process if this is useful.
- *Operational Monitoring Strategy*: An explanation of the strategy and system wide standards for operational monitoring (see 4.5). A component and/or layer model for the development team to follow when implementing this in components.
- *Operational Control Strategy*: An explanation of the strategy for operational control. A definition of the tools (or at least types of tools) that development will provide the systems administrators to assist with operational control. (See 4.6.)
- *Configuration Management Strategy*: An information model illustrating the configuration information within the system and its inter-relationships. An explanation of the strategy to employ for both routine and exceptional configuration reconfiguration activities (see 4.7).
- *Performance Monitoring Strategy*: A textual description of the strategy for collecting a coherent set of performance metrics across the system. An information model illustrating this if the complexity justifies it. A software component and/or layer model showing how software to collect performance metrics will be integrated with the rest of the software.
- *Integration Design*: If integration into third party management software is required, a component model showing how this will be achieved.

The result of this set of techniques should be an operational viewpoint that will communicate effectively with the affected stakeholders and help ensure their cooperation and that their needs are met.

6. Conclusions

To return to our original point; systems are created so that they can run in production and provide useful facilities for their users.

Implementation of a modern information system is a complex and intricate process involving extensive planning and skilled work by a number of people.

Yet, much software architecture work today appears to neglect the operational aspects of a system's lifecycle and similarly neglects the needs of an important group of stakeholders: the system's administrators and operators.

We would suggest that a possible remedy to this situation is the addition of an operational viewpoint to the library of viewpoints used by architects. Such a viewpoint can be used as the basis for a system view which addresses the operational needs of the system's stakeholders.

An operational view needs to address installation, migration, support, operational monitoring, operational control, configuration management and performance monitoring concerns.

The operational view is less focused on system structure than many of the other views, but rather is a rich source of strategies, guidelines and standards to drive the deployment and operation of the system.

We would suggest that without an operational view, an architectural description cannot be considered complete.

7. References

[1]. Institute of Electrical and Electronics Engineers, *IEEE Std 1471-2000, IEEE Recommended Practice for Architectural Description of Software Intensive Systems*, IEEE, New York, 9th October 2000.

[2]. Philippe Krutchen, *The 4+1 View Model of Architecture*, IEEE Software, 1995; 12(6).

[3]. Christine Hofmeister, Robert Nord and Dilip Soni, Applied Software Architecture, Addison-Wesley, 1999.