Experiences Using Viewpoints for Information Systems Architecture: An Industrial Experience Report

Eóin Woods

Artechra Limited
Hamilton House, 6th Floor, 111 Marlowes,
Hemel Hempstead, HP1 3HY, UK
eoin.woods@artechra.com

Abstract. There has recently been an increase in interest, among information systems architecture practitioners, in using viewpoints for architectural definition and description. This has been caused by a number of factors including the publication of IEEE standard 1471 and the increasing adoption of RUP (and its "4+1" viewpoint set). This short experience report outlines the experiences that two software architects have had in evaluating and applying a number of viewpoint sets to information systems development. The strengths and weaknesses found with each viewpoint set are described and some general observations on viewpoint set use and definition are presented.

Introduction and Motivation

As a practicing software architect, I am always interested in techniques that can help to manage the complexity of the architectural design process. Most architects would agree that architecture is a many-faceted discipline and developing a successful software architecture involves considering a lot of different system structures simultaneously. This means that using more than one model to capture your architecture is an intuitively appealing approach and many practicing architects do appear to do this informally. Certainly the software architecture research community appears to have decided that representing architectural designs via a number of related models (or "views") is the only way to do it.

Having said this, if we use a number of models to represent our architectural designs then we need some sort of framework to organize the work and its deliverables, so that the approach doesn't become too unwieldy and disorganized to use.

Some time ago, along with another colleague, I came across IEEE standard 1471 [5] (a standard for architectural description), which was then about to be published. It seemed obvious to us that the view and viewpoint based approach defined in the standard had the potential to help us organize the architectural design efforts of our clients and ourselves. We had also previously come across Phillippe Kruchten's well known "4+1" viewpoint set [7] and we started to further investigate its application to our work.

Researching further, we discovered a number of other viewpoint sets including the Siemens set [4], the RM-ODP set [6] and (much more recently) the Garland and Anthony set [3].

We assessed and trialled these viewpoint sets as part of our normal work, considering their application to a number of situations including enterprise and Internet security products, systems integration programmes, and bespoke in-house systems. This process has lead to a number of observations about particular sets of viewpoints and about viewpoint sets in general, and this short paper explains these observations.

The common characteristic across all of our applications of viewpoints is that they are information-oriented systems rather than control systems, and this should be borne in mind when considering our observations.

Our Use of Viewpoints

Views are obviously a useful way of structuring an architectural description (the documentation of the architecture) but in their simplistic form, they are little more than an approach to document organization; ideally an approach based around a set of formal viewpoint definitions should provide us with more than this. In fact, the authors of IEEE 1471 appear to have had more a more ambitious target when they standardized the viewpoint based approach.

From the text of the standard, we find that a viewpoint is *a specification of the conventions for constructing and using a view; a viewpoint acts as a pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis*.

Importantly, this definition makes it clear that a viewpoint is not just the name of a section of a document but is a guide for those who which to create views that comply to the viewpoint, explaining how and why the view is to be created.

We feel that a well-defined set of viewpoints has the potential to be applied as:

- A *description* of a particular approach to software architecture.
- A *store* of experience, advice and best practice for a particular aspect of software architecture.
- A *guide* for the novice architect or the architect who is working in an unfamiliar domain (as happens to us all from time to time), who will in all probability be working alone without an expert architect to guide them.
- An *aide-memoir* for the experienced architect, that they can use to avoid overlooking important aspects of the design process, particularly when considering the areas of the architecture that the architect is not an expert in.

We have attempted to consider all of these possible uses of viewpoints as we have applied and assessed them for our work.

Experiences with the Viewpoint Sets

"4+1" Viewpoint Set

When we first starting using architectural views, we started by using the "4+1" set originally defined by Philippe Kruchten and now forming part of the Rational Unified Process [8]. The viewpoints contained in the original set are briefly described in Table 1.

Viewpoint	Description
Logical	Logical representation of the system's functional structure. Normally presumed to be a class model (in an object-oriented systems development context).
Process	The concurrency and synchronization aspects of the architecture (process and thread model, synchronization approach etc.)
Development	The design time software structure, identifying modules, subsystems and layers and the concerns directly related to software development.
Physical	The identification of the nodes that the system's software will be executed on and the mapping of other architectural elements to these nodes.

Table 1. 4+1 Viewpoint Catalog

We found the strengths of this viewpoint set to be:

- The set is simple, logical and easy to explain. We found that colleagues, clients and stakeholders understood the set with very little explanation.
- The set is quite generic and seems a suitable base to use for describing information system architectures.
- The viewpoint set is really independent of notation, but is normally used in conjunction with UML, which is widely understood and supported.
- The set of viewpoints defined aligns quite well with existing models that architects build and so it is quite an intuitive set to use straight away.
- This appears to be the oldest viewpoint set and is widely known, discussed and supported (partially due to its inclusion in the RUP "architectural profile").

Problems we found when using this viewpoint set were:

- It is difficult to find a good definition of these viewpoints. The original paper only provides outlines of their expected content and we haven't found other fuller definitions in easily accessible sources. This leads to confusion when the viewpoints are applied, with every architect creating different content for each view. The (proprietary) Rational Unified Process product does provide more information on the views but still doesn't define them in a level of detail that we felt that we needed (for example, the Logical View is defined in about 2 pages).

- We found the names of the viewpoints quite problematic:
  - The term "*process*" tended to suggest business process modeling, which caused confusion when people found that the view contains an operating system concurrency model.
  - The terms "*logical*" and "*physical*" aren't terribly descriptive and different people interpret them in different ways.
- The viewpoint set does not explicitly address data or operational concerns. Both of these aspects of a large information system are important enough to warrant their own view (and more importantly guidance relating to these aspects of developing an architecture needs to be captured somewhere).[1]
- There is no associated set of cross-viewpoint consistency rules defined for the set (at least we were not able to find such a set).

As a basis for system implementation, architectural descriptions based on this viewpoint set have proved to be quite effective. The views can include most of the information needed to support software development and the Development view can act as an effective bridge between architecture and implementation. The main limitation from the software developer's point of view is the lack of a single place to refer to for an understanding of the system's underlying information structure.

In summary, this viewpoint set appeared to be aimed at the area that we were interested in, although the coverage was not as wide as we would have liked. However we found the lack of readily available, thorough, viewpoint definitions to be an obstacle to initial use of the set by our clients and ourselves. That said, we've had a lot of success applying our interpretations of these viewpoints to our information systems architecture problems.

RM-ODP Viewpoint Set

The Reference Model for Open Distributed Processing (RM-ODP) is an ISO standard framework for describing and discussing distributed systems technology [6]. The framework is defined using a set of five viewpoints, briefly described in Table 2.

Viewpoint	Description
Enterprise	Defines the context for the system and allows capture and organization of requirements.
Information	Describes the information required by the system using

[1] Interestingly Rational appear to, at least partially, agree with us, as they have recently added an optional "Data" viewpoint to the set defined in the Rational Unified Process and renamed "Physical" as "Deployment" (as well as renaming "Development" as "Implementation" which we don't think is as important). Other authors (notably Scott Ambler) have also identified the need for operational considerations to be addressed and have extended RUP to meet these needs [1].

	static, invariant and dynamic schemas.
Computational	Contains an object-oriented model of the functional structure of the system, with a particular focus on interfaces and interactions.
Engineering	Describes the systems infrastructure required to implement the desired distribution of the system's elements. This description is performed using a specific reference model.
Technology	Defines the specific technology that will be used to build the system.

Table 2. RM-ODP Viewpoint Catalog

While the RM-ODP approach provides an appealing partitioning of the architectural description, it was actually created to support distributed systems standardization efforts and (as its name suggests) imposes a reference model on the systems being described.

We found the strengths of this viewpoint set to be:

- The structure of the viewpoint set is logical and reasonably easy to explain (although the "Engineering" viewpoint isn't terribly well named).
- The viewpoint set is aimed at distributed information systems.
- The viewpoint set includes an explicit consideration of data architecture via the "Information" viewpoint.
- The viewpoint set is an ISO standard and so is widely accessible and appears to have been widely discussed in the distributed systems research community.

Concerns we had with regards to this viewpoint set were:

- We couldn't find much evidence of this viewpoint set being used by architecture practitioners.
- A particular set of architectural assumptions appears to have been made when defining the viewpoints. In particular, the "Computational" and "Engineering" viewpoints seem to assume that a distributed object system is being created and specify a particular set of primitives that should be used to describe these views.
- A number of the viewpoints appear to assume that RM-ODP's own modeling notations will be used to describe them (which aren't widely understood or supported by tools).
- The viewpoint set doesn't address operational concerns.
- The definition of the viewpoints is quite daunting to approach.
- There doesn't appear to be a set of cross-viewpoint consistency rules available.

We had quite a few concerns as to how effective an RM-ODP based architectural description would be as a basis for implementation. Implementation isn't mentioned much in the RM-ODP literature and even the more tutorial material we found (such as [9]) seemed rather vague on the subject of how the RM-ODP based description would drive the software development process. There is also the problem that many systems aren't created as distributed object systems that would be compliant with the meta-model that appears to be assumed in RM-ODP's Engineering viewpoint. None of this is to say that RM-ODP architectural descriptions couldn't drive software development effectively, but it isn't clear how to do from the material that we could find to refer to. Of course, if a framework that directly implemented RM-ODP's meta-models were available, this could resolve the problem and make RM-ODP a very attractive approach.

Overall, this viewpoint set initially appeared to be very promising, having an intuitive structure and seemingly being aimed at the kind of systems that we are interested in building. However, further investigation suggested that this viewpoint set is quite specialized and perhaps really aimed at supporting standards efforts rather than mainstream information-systems-architecture definition.

Siemens Viewpoint Set

While working at Siemens Research, Christine Hofmeister, Robert Nord and Dilip Soni developed a set of four architectural viewpoints based upon the way that Siemens' software development teams actually operated. The viewpoints in this set are briefly described in Table 3.

Viewpoint	Description
Conceptual	The conceptual functional structure of the system.
Module	Defines the subsystems and modules that will be realized in the system, the interfaces exposed by the modules, the inter-module dependencies and any layering constraints in the structure.
Execution	The runtime structure of the system in terms of processes, threads, inter-process communication elements and so on along with a mapping of modules to runtime elements.
Code	The design time layout of the system as source code and the binary elements that are created from it.

Table 3. Siemens Viewpoint Catalog

We found the strengths of this viewpoint set to be:

- The viewpoints are clearly defined in the very readable primary reference [4].
- Again, this seems to be a logical viewpoint set that can be explained and remembered easily.
- The viewpoints use UML as their modeling notation, which is widely understood and supported.
- The viewpoints are based directly upon Siemens industrial practice, which gives them some immediate credibility from a practitioner's point of view.
- The viewpoint definitions include tasks required to create them, modeling advice to follow, and common problems ("issues") along with possible solutions to them.

Factors that we found limiting when applying this viewpoint set were:

- The viewpoints are obviously aimed at software engineers working on control systems. The examples and advice in the definitions are control-system rather than information-system centric.
- The deployment, operational and data aspects of the architecture aren't addressed by the viewpoints defined. This makes perfect sense in the control systems environment (as these concerns aren't as relevant or are someone else's problem) but this does limit their application to information systems.
- There is no mention of the applicability of the viewpoints for communication with different stakeholder groups. We suspect that these viewpoints are all aimed at the development team rather than any wider constituency of stakeholders. Again, this may be less of a problem for the architecture of control systems than information systems.
- There is some guidance provided for achieving consistency via traceability rules, but there isn't really a set of clear cross-viewpoint consistency rules.

Architectural descriptions based on this viewpoint set are likely to be a strong basis for system implementation, provided that the system is broadly in the control or real-time systems domain. The Code view forms a good bridge to implementation and all of the aspects of a functionally oriented system, that are important to a developer, can be easily addressed using the other views.

Garland and Anthony Viewpoint Set

Jeff Garland and Richard Anthony are practicing software architects who have recently written a practitioner-oriented guide to software architecture for information systems. In their book they define a viewpoint set aimed at large-scale information systems architecture [3]. Their viewpoints are briefly described in Table 4.

Viewpoint	Description
Analysis Focused	Illustrates how the elements of the system work together in response to a functional usage scenario.
Analysis Interaction	Interaction diagram used during problem analysis.
Analysis Overall	Consolidation of the contents of all of the Analysis Focused view contents into a single model.
Component	Defines the system's architecturally significant components and their connections.
Component Interaction	Illustrates how the components interact in order to make the system work.
Component State	The state model(s) for a component or set of closely related components.
Context	Defines the context that the system exists within, in terms of external actors and their interactions with the system.

Deployment	Shows how software components are mapped to hardware entities in order to be executed.
Layered Subsystem	Illustrates the subsystems to be implemented and the layers in the software design structure.
Logical Data	Logical view of architecturally significant data structure.
Physical Data	Physical view of architecturally significant data structure.
Process	The runtime concurrency structure (operating system processes that the system's components will be packaged into and IPC mechanisms).
Process State	State transition model for the system's processes.
Subsystem Interface Dependency	The dependencies that exist between subsystems and the interfaces of other subsystems.

Table 4. Garland and Anthony Viewpoint Catalog

This viewpoint set was published fairly recently and so we haven't been able to give them as much consideration as we have given the others over time (and we haven't considered their application to a real system). However, this set looks particularly promising for information systems.

We found the strengths of this viewpoint set to be:

- This set of viewpoints is aimed directly at information systems architects and so tries to address their needs directly.
- The viewpoints are all small and focused, with the content and the use of the viewpoint being immediately apparent.
- The viewpoints are quite thoroughly defined, with purpose, applicability, stakeholder interest, models to use, modeling scalability and advice on creating the views all presented. In most cases there is also guidance provided that often includes potential problems to be aware of.
- The viewpoints defined address data explicitly (via the Logical Data and Physical Data viewpoints).
- The viewpoints are all defined using UML as the primary modeling notation, which is widely understood and supported.

Problems we found when using this viewpoint set were:

- There are a lot of viewpoints in the set (14) and so the set can be quite unwieldy to explain and use.
- Many of the viewpoints are relevant to a large or complex system, and so there appears to be a real danger of the architectural description becoming fragmented. We take Garland and Anthony's point that you should only apply the viewpoints relevant to a particular system, but you should do this when applying any viewpoint set, and we feel that for many systems you will end up with quite a few viewpoints when using this set.

- There aren't any consistency rules defined for inter-view consistency. The other viewpoint sets don't tend to have these either but they seem all the more important when you many end up with 14 views.

Architectural descriptions based on this viewpoint set are likely to be a strong basis for information system implementation. Provided that the developers are prepared to understand a number of different views, the information that they require (including logical and physical data structure) can all be represented using views from this set. In addition, the Layered Subsystem view provides a bridge to implementation, which is well defined in the viewpoint definition (in [3]).

Overall, this viewpoint set is probably closest to the ideal set that we were searching for. Having said this, because it is a new set we haven't spent all that much time working with it. However, given that the set is well defined, based on practical experience and aimed at information systems, it appears to have a lot of potential for application to large information systems. The major concerns we have are explaining 14 viewpoints to anyone and creating a coherent, consistent architectural definition with a significant subset of this many parts.

Other Viewpoint Sets

Other viewpoint sets exist (such as Dana Bredemeyer's set) that we haven't talked about here, because we haven't spent enough time working with them and considering them to have informed opinions on their utility. We also probably aren't aware of all of the sets that exist.

Dana Bredemeyer's viewpoint set is potentially very relevant to our area of concern, being aimed at enterprise information systems development. It comprises Structural and Behavioral variants of Conceptual, Logical and Execution viewpoints (making 6 viewpoints in all). However, we aren't aware of any publicly available reference source for this viewpoint set (the normal source being Bredemeyer Inc.'s training courses) and this makes it difficult to research the viewpoint set further.

One other interesting set that is worth mentioning, is the set of "view types" introduced in [2]. We are aware of these view types and deliberately don't discuss them as a separate set of viewpoints, because their focus appears to be capturing knowledge and advice related to documenting an architecture rather than actually creating it. We have found the advice in this text to be credible and valuable, but it appears to be relevant irrespective of the viewpoint set in use, rather than being a definition of a new set of viewpoints.

General Observations

Having attempted the application of a number of viewpoint sets to the architectural design of information systems, we have made to a couple of general observations about the approach that are independent of the specifics of a viewpoint set. These observations are summarized below.

- *Viewpoints are an Effective Approach*. We have found viewpoints to be a very effective approach to the problem of organizing the architectural design process and capturing its results (the architectural description). We have found viewpoints to be useful for both novice architects (as a guide) as well as by experts (as a set of aide-memoirs). We have found that a number of viewpoint sets can be very effective for information systems architecture, with the "4+1" and "Garland and Anthony" sets showing particular promise for further refinement and use.
- *Good Viewpoint Sets*. We have found that all of the viewpoint sets we have reviewed are coherent, logical and appear to be well thought out. In reality, it appears that they could all be applied successfully and this bodes well for the general acceptance of the approach.
- *No Standard Viewpoint Definitions*. A constant challenge when trying to understand new viewpoint sets was the fact that there is little standardization between the viewpoint set definitions in the published literature. We found that this meant that it was hard to compare viewpoint sets without a lot of analysis and that starting to use a new viewpoint set can be difficult.
- *General Lack of Cross-Viewpoint Consistency Rules*. In the viewpoint sets that we have reviewed and used, we have generally found a lack of cross-viewpoint consistency rules. Given the inherent fragmentation, that a view based approach to architectural description implies, this seems strange. Our experience is that cross-viewpoint consistency is a significant challenge and that even a simple set of rules helps architects to keep their views consistent, particularly when they are inexperienced with a viewpoint set.
- *Suggested Standard Viewpoint Content*. Based on our experience of trying to apply viewpoint sets, we would suggest that a viewpoint definition should contain:
  - *Concerns* that the view should address.
  - The *Stakeholders* that are likely to be interested in the view (and the reason for their interest) so that the architect can cater to them.
  - The *Activities* to perform to create the view, with guidance on performing each activity.
  - The set of *Models* (or other content) that can appear in the view, with guidance on creating each type.
  - Common *Pitfalls* that the architect should be aware of when creating the view, with pointers to possible solutions for them.

  It is worth noting that this content is compliant with IEEE 1471, as the standard states that a viewpoint definition includes stakeholders; concerns; modeling language and modeling techniques; (and optionally) consistency tests; analysis techniques; and heuristics to guide successful view creation.

  We would also suggest that the definition of a viewpoint set should also include:
  - An *overall model* of how the views are used to represent an architecture (in other words an overview of what goes where and the rationale for this organization).
  - A set of *consistency rules* to allow cross-view consistency to be assessed.
  - A *presentation* that can serve both a novice architect looking for guidance and an experienced architect needing an aide-memoir. (Realistically, this is likely to

imply a hybrid presentation that includes explanation coupled with reference material such as checklists and summary tables.)

Further Work

After initial work with the available viewpoint sets (at that time, “4+1”, Siemens and RM-ODP) we decided that we needed a fully defined, information system centric, viewpoint set to use ourselves and with our clients. We felt that such a set would be useful for teaching, consultancy, mentoring and during practice.

In response to our need, we designed such a set as a clear evolution of the “4+1” set, which appeared to provide the best basis to work from. The aim of this paper is to compare other people’s viewpoint sets rather than to introduce another, but the brief descriptions in Table 5 give a flavour of the content of the set.

Viewpoint	Description
Functional	Describes the system’s runtime functional elements, their responsibilities, interfaces and primary interactions
Information	Describes the way that the architecture stores, manipulates, manages, and distributes information (including content, structure, ownership, latency, references, and data migration).
Concurrency	Describes the concurrency structure of the system, and maps functional elements to concurrency units to clearly identify the parts of the system that can execute concurrently and how this is coordinated and controlled.
Development	Describes the constraints that the architecture places on the software development process.
Deployment	Describes the environment into which the system will be deployed, capturing the hardware environment, the technical environment requirements for each element and the mapping of the software elements to the runtime environment that will execute them.
Operational	Describes how the system will be operated, administered and supported when it is running in its production environment. For all but the smallest simplest systems, installing, managing and operating the system is a significant task that must be considered and planned at design time.

Table 5. Proposed Information Systems Viewpoint Catalog

We also decided to follow our own advice and define some consistency rules that can be used to help check a view set for consistency. Some example consistency checks between the Functional and Development views are shown in Table 6.

1	Does the code module structure include all of the functional elements that need to be developed?
2	Does the Development View specify a development environment for each of the technologies used by the Functional View?
3	If the Functional View specifies the use of a particular architectural style, does the Development View include sufficient guidelines and constraints to ensure correct implementation of the style?
4	Where common processing is specified, can it be implemented in a straightforward manner over all of the elements defined in the Functional View?
5	Where reusable functional elements can be identified from the Functional View, are these modeled as libraries or similar features in the Development View?
6	If a test environment has been specified, does it meet the functional needs and priorities of the elements defined in the Functional View?
7	Can the functional structure described in the Functional View be built, tested and released reliably using the codeline described in the Development View?

Table 6. Example Consistency Checks

While these checks aren't complex or terribly sophisticated, they do reflect the sorts of mistakes that we all make when creating architectural descriptions and aim to help architects – particularly those inexperienced with the viewpoint set – to avoid the most common mistakes. The rules also help those using the viewpoint set to improve their understanding of it and help to resolve confusions about the role of each viewpoint.

We are currently completing the development of this viewpoint set (along with further work to help architects address quality properties for their systems) and hope to publish this work during 2004.

Acknowledgements

The work described here has been performed with Nick Rozanski, an IT architect with the French-Thornton Partnership in London, UK.

References

[1] Ambler S., Constantine L.: The Unified Process Transition and Production Phases, CMP Books (2001). See also http://www.enterpriseunifiedprocess.info.
[2] Clements P., Bachmann F., Bass L., Garlan D., Ivers J., Little R., Nord R., Stafford J.: Documenting Software Architectures, Addison Wesley (2003).
[3] Garland J., Anthony R.: Large Scale Software Architecture, John Wiley (2003).
[4] Hofmeister, C., Nord, R., and Soni, D.: Applied Software Architecture, Addison Wesley (1999).

[5] Recommended Practice for Architectural Description: IEEE Std-1471-2000, IEEE Computer Society (2000).
[6] Reference Model for Open Distributed Processing: International Standard 10746-1, ITU Recommendation X.901, International Standards Organization (1996).
[7] Kruchten P.: Architectural Blueprints – The 4+1 View Model of Software Architecture, Software, Vol. 12, Number 6, IEEE (1995).
[8] Kruchten P.: The Rational Unified Process: An Introduction, Second Edition, Addison Wesley (2000).
[9] Putman J.: Architecting with RM-ODP, Prentice Hall PTR (2001).