Development Environment Definition

Ensuring a comprehensive consideration of all elements of a development environment
Introduction

A development environment contains everything required by a team to build and deploy software-intensive systems (where software is an essential and indispensable element). So why is having a consistent definition of a development environment important?

Put simply, many organizations are looking to reduce time-to-market, reduce cost and improve quality and all of these business goals are directly influenced by the quality of the environment used to produce their software-intensive systems.

Having a consistent and comprehensive definition of a development environment at hand will ensure that nothing is overlooked when we’re planning an initiative to improve the current environment, defining requirements on the environment, architecting the environment, assessing the environment, ensuring an appropriate return-on-investment when changing the environment and so on. Fundamentally, a development environment definition, the subject of this paper, is a critical input to all of these tasks.

Putting a Development Environment in Context

Before looking at the specific elements that comprise a development environment, we really need to first understand where this environment sits in the grand scheme of things. In Figure 1 we see a Center of Excellence that is responsible for creating and maintaining the development environment. This environment supports development projects that, in turn, create and maintain software-intensive systems (or some other software-related deliverable, such as components or services).

This simple visualization helps clarify the distinction between the role of a Center of Excellence (including their roles, processes and key deliverable – a development environment) and the development projects that use the development environment (and their roles, processes and deliverables).

Figure 1. A Development Environment in Context

Elements of a Development Environment

So what does a development environment actually contain? As shown in Figure 2, IBM Rational considers a development environment to comprise the following six elements, each of which is described in more detail below:

- Method
- Tools
- Enablement
- Organization
- Infrastructure
- Adoption
Readers may be familiar with “people, process and technology” as key ingredients of a successful development project. However, this model is overly-simplistic for our purposes. The elements shown in Figure 2 build on this model where process equates to method, technology is represented by tools and infrastructure, and people is represented by enablement and organization. Adoption is a new (and very important) element that is focused on the introduction of the development environment within an organization, business unit or development project.

Method

A key element of any development environment is the method that is followed, formally or informally, by practitioners. Key method-related elements are:

- Core method elements such as roles, work products, tasks and processes.
- Supplementary method elements such as standards, guidelines, checklists, templates and examples.
- Method deployment topology. Consideration of a deployment topology is needed when, for example, the method is deployed as a website on the company intranet. In this example, a web server is required to host the content, and workstations must have an appropriate web browser installed and connectivity to the web server.

Tools

Development tooling automates aspects of the method being followed. For example, we may use a tool for storing and managing requirements on a development project, use a tool for visually modelling our architectures and designs, use tools for testing our software, and so on. Key tools-related elements are:

- Development tools and their integrations.
- Development tool configurations and install scripts.
- Development tool deployment topology. This includes consideration of software and hardware required both client-side and server-side (including any target platform and emulators when, for example, developing for real-time or embedded devices).

Enablement

Enablement (training and mentoring) of practitioners in the use of the development environment contributes to its successful adoption. An aspect of a development environment is therefore the definition and creation of training and mentoring materials that can be applied. Mature organizations also pay particular interest to the professionalization of their staff and any
alignment with external professional bodies. Key enablement-related elements are:

- Training curriculum and courses. This covers a variety of training needs, from training experienced practitioners in refinements to the development environment, all the way up to a comprehensive training curriculum for a practitioner taking on a new role.
- Mentoring materials.
- Enablement deployment topology. Consideration of a deployment topology is needed when, for example, enablement is provided via web-based training. Again, a web server is required to host the enablement materials, and workstations equipped with a web browser. The deployment topology may also reference any locations and rooms required to deliver classroom training.

**Organization**

Another consideration of a development environment is ensuring that an appropriate organization is in place to define, deploy and manage it. This may include specialists in certain aspects of the development environment (such as method experts, tool specialists, trainers and mentors), personnel to administer and support the environment, personnel with appropriate skills on the company helpdesk and appropriate communities of practice. Key organization-related elements are:

- Definition of organizational roles and units that are part of the development environment.
- Organization deployment topology. The deployment topology will indicate the locations where the various organizational units reside.

**Infrastructure**

A development environment considers infrastructure in terms of both hardware and software. This has already been hinted at above in terms of both development tools (software) and various deployment topologies (software and hardware). There are three reasons for considering infrastructure as a key element in its own right, however.

The first is with regard to consolidation. For example, by looking at the infrastructure needs of the development environment as a whole, we may find that we only need a single web server to support both web-based method content and web-based training. The second is that we may require additional hardware and software to support the development environment itself, such as operating system software, a database management system, or board-level controls and test harnesses if developing for real-time or embedded devices. The third is that we may need not just production infrastructure to support the development environment, but also development and test infrastructure where changes to the development environment are developed and tested before being put in production, just like any other mission critical application.

Key infrastructure-related elements are:

- Locations, nodes and connectivity.
- Supporting software (such as operating systems, database management systems, board-level controls and test harnesses).

**Adoption**

In addition to the elements listed above, we should also be concerned with the adoption of the environment within an organization, a business.
unit or a development project. Key adoption-related elements are:

- Adoption plan. This plan defines the tasks that are normally performed when adopting the environment, such as the acquisition of any hardware and software.
- Techniques for driving the organizational changes required to introduce and embed the development environment into the day-to-day working practices of the affected organizational areas, with an acceptable level of impact.
- Definition of environment metrics. The metrics are used to gauge the effectiveness of the environment.

**Context**

Also shown in Figure 2 is the *Solution Context* (where the development environment is the solution under consideration). The context represents the requirements on the development environment, and can be considered in terms of *functionality, qualities and constraints*.

*Functionality* represents a software engineering practice or discipline to be provided by the development environment. Realizing such requirements leads us to consider all of the elements mentioned previously. For example, and as shown in Figure 3, a requirements management discipline would be supported by:

- A requirements management method.
- Requirements management tools.
- Training and mentoring in requirements management/
- A helpdesk that contains knowledge of the requirements management solution.
- Hardware and software to support the requirements management-related elements.
- Appropriate adoption of the requirements management discipline on projects.

*Figure 3. Required Functionality is Realized by All Elements of a Development Environment*

This thinking can be applied to other capabilities that are part of the development environment, such as architecture or quality management. It can also be applied to specific practices, such as iterative development (that is at the heart of an agile approach to software development and delivery), which also requires us to consider all elements.

*Qualities* represent properties that the development environment should exhibit and this also requires consideration of all elements of a development environment. For example, a scalability quality (the ability to support a varying number of concurrent users, for example) may be accommodated through:
6 Development Environment Definition

- A method that can be customized to fit the size of project.
- Tools that can be configured to support a configurable method.
- Appropriate mechanisms and levels of training for different sizes of project.
- An organization that provides appropriate levels of correctly-skilled resources to support the anticipated number of development projects.
- An infrastructure that can scale to support the anticipated number of concurrent users.
- Appropriate mechanisms to adopt the environment.

**Constraints** that the development environment should accommodate also require consideration of all elements of a development environment. For example, the need to migrate from an existing environment may result in:

- Taking practices from an existing method and incorporating them in a new method.
- Migrating work products from a deprecated toolset to another. Another constraint might be the need to integrate with existing tools that will be retained.
- Providing enablement that acknowledges current understanding and is customized appropriately.
- Providing personnel to ensure a smooth transition from an as-is state to the to-be state.
- Specifying an infrastructure that maximizes reuse of an existing infrastructure (such as reusing existing hardware and software licenses where possible).
- Appropriate adoption mechanisms that acknowledge the migration to be performed.

Another important constraint when an organization considers a change to its existing development environment is the Return-On-Investment (ROI). In order for such an initiative to succeed it must, clearly, deliver positive results that are in line with the business case for the initiative. Each area of a development environment influences the ROI in terms of both cost and benefit.

Although not shown in Figure 2, functionality, qualities and constraints are typically aligned with any business context that has been defined, such as business goals. In this sense, the solution context also embraces business considerations also. This can be especially important when showing how the development environment contributes (directly or indirectly) to achieving business goals.

**Define, Deploy, Manage**

In defining the various elements of a development environment, it has proved useful to consider the following elements of the environment’s lifecycle, as shown in Figure 4, since (in addition to the solution context) they each have specific concerns that influence the definition:

- The definition of the environment.
- The deployment of the environment.
- The management of the environment.

![Solution Lifecycle Diagram]

[Diagram showing the lifecycle of solution contexts, definitions, deployments, and managements]
Before looking at these areas, it is worth explaining why these different areas are linked in a cycle in Figure 4. This figure acknowledges that effective change (in this case, improvements to the development environment) is usually achieved through a series of incremental changes rather than a big bang approach to change (evolution, not revolution) where each increment represents a single pass through the cycle. However, the change that has been implemented in an increment, by definition, changes the context for the next increment (for example, practitioners may now have improved skills, new hardware may be available, new tools may be in place and so on) – hence the cyclic nature shown.

Consideration of each element of a development environment, in conjunction with solution definition, solution deployment and solution management, is given in the sections that follow.

**Solution Definition**

Earlier discussion focused on the key elements to be considered when defining a development environment. That discussion is not revisited here although, for completeness, the various items that are defined are reproduced in Figure 5.

It should also be noted that the definition is typically considered at an organization level and may require a local instantiation to address the needs of a particular business unit or project when it is deployed. This is reflected in the sections below.

---

**Figure 4. The Lifecycle of a Development Environment**

**Figure 5. Solution Definition Considerations**

**Solution Deployment**

Deployment of the development environment introduces specific concerns with respect to each element as shown in Figure 6.
Key method-related elements are:

- Define local configuration. When deploying the method to a business unit or development project, it may require some local configuration to reflect the specific characteristics of the business unit, development project or system (by, for example, providing an appropriate level of ceremony).
- Deploy method. This ensures that the method is available to practitioners.

Key tools-related elements are:

- Perform local configuration. Any local tool configuration is applied to automate the local method configuration.
- Install tools. The tools (and their integrations) are installed and made available to practitioners.
- Migrate local data. It may be necessary to migrate data from an existing toolset to the updated toolset, for example.

Key enablement-related elements are:

- Perform local configuration. Enablement materials may be refined to accommodate, for example, the process defined for that business unit or development project.
- Deploy enablement materials. The enablement materials are made available to practitioners, including access to any web-based training.
- Train practitioners. The practitioners are trained and feedback on the training gathered.

Key organization-related elements are:

- Define local configuration. It may be necessary to provide expertise to support the specific needs of a particular business unit or development project.
- Reorganize. Resources are organized appropriately in order to support the development environment.

Key infrastructure-related elements are:

- Define local infrastructure. The infrastructure required by a particular business unit or development project is defined.
- Provision locations, nodes, connectivity. Any hardware required (including any target platform and emulators when developing for real-time or embedded devices) is made available.
- Provision supporting software. Any software that supports the development environment (such as database management systems or test harnesses) is installed.

Key adoption-related elements are:
Define local adoption plan. The adoption plan is refined to reflect the specific needs of the business unit or development project.

Validate the environment. The deployed environment is, essentially, tested in order to ensure that it meets the defined requirements in terms of providing the desired functionality, meeting the stated qualities and functioning within defined constraints.

Solution Management

Management of the development environment, post-deployment, also introduces specific concerns with respect to each element as shown in Figure 7.

<table>
<thead>
<tr>
<th>Key Element</th>
<th>Solution Management Considerations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Gather feedback on method</td>
</tr>
<tr>
<td>Tools</td>
<td>Backup / archive / restore data</td>
</tr>
<tr>
<td></td>
<td>Gather feedback on tools</td>
</tr>
<tr>
<td>Enablement</td>
<td>Mentor practitioners</td>
</tr>
<tr>
<td></td>
<td>Gather feedback on enablement</td>
</tr>
<tr>
<td>Organization</td>
<td>Gather feedback on organization</td>
</tr>
<tr>
<td>Infrastructure</td>
<td>Onboard / retire infrastructure as required</td>
</tr>
<tr>
<td></td>
<td>Gather feedback on infrastructure</td>
</tr>
<tr>
<td>Adoption</td>
<td>Measure environment effectiveness</td>
</tr>
<tr>
<td></td>
<td>Gather feedback on adoption</td>
</tr>
</tbody>
</table>

Figure 7. Solution Management Considerations

Key method-related elements are:

- Gather feedback on the method. A key aspect of managing the development environment is to continually improve. Gathering feedback on each element is therefore a theme that pervades all elements of the development environment. Feedback is typically gathered subjectively using, for example, a questionnaire.

Key tools-related elements are:

- Backup / archive / restore data. Work products created by practitioners are managed appropriately and “good housekeeping” practices applied.
- Gather feedback on tools. Feedback (positive and negative) on the capability and performance of the tooling is gathered.

Key enablement-related elements are:

- Mentor practitioners. Mentors are assigned to projects to mentor practitioners in the use of the environment.
- Gather feedback on enablement. Feedback on any mentoring is gathered.

Key organization-related elements are:

- Gather feedback on organization. Practitioners provide feedback on the support they have been provided in using the development environment (such as the quality of helpdesk support).

Key infrastructure-related elements are:

- Onboard / retire infrastructure as required. As projects start and conclude, the development environment needs to resize accordingly, in order to optimally support the number of practitioners using the environment at any given time.
- Gather feedback on infrastructure. Feedback on both hardware and supporting software is gathered.

Key adoption-related elements are:
• Measure environment effectiveness. A key aspect of adoption is to measure the effectiveness of the environment. For example, a questionnaire may be provided to practitioners to gauge how effective they have been in adopting new practices.
• Gather feedback on adoption. Feedback on the approach to adoption is gathered.

Interdependencies
Finally, it should be pointed out that the various elements of a development environment are not as independent as this paper might imply. An alternative representation of Figure 2 is shown in Figure 8 – a figure that acknowledges that each element has relationships with all other elements.

Figure 8. Interdependencies between Elements
Here are some examples of dependencies between elements:
• The method (method) references available training courses (enablement).
• Tools (tools) automate tasks (method).
• Administration roles (organization) are defined to support the tools (tools).
• Servers (infrastructure) are provisioned to host the toolset (tools).
• Adoption of practices (adoption) is assessed using the defined approach (method).

Summary
This paper has elaborated upon the key elements that comprise a development environment in some detail, and considers the different concerns when defining, deploying and managing such an environment. This paper provides a simple framework for ensuring that all of these aspects are considered when planning an initiative to improve the current environment, defining requirements on the environment, architecting the environment, assessing the environment and so on.
For more information

To find out more about IBM Rational, please contact your local IBM Software Services or Rational Sales Representative.

About the Author

Peter Eeles is an Executive IT Architect and Chief Architect for IT in IBM Rational’s worldwide solution delivery organization, where he helps organizations improve their software development capability. This is often in conjunction with an architecture-centric initiative such as SOA or strategic reuse, where Peter has particular in-depth knowledge. Peter is co-author of "The Process of Software Architecting" (2009), "Building J2EE Applications with the Rational Unified Process" (2002), and "Building Business Objects" (1998). He can be contacted at peter.eeles@uk.ibm.com.