

# Patterns for IT Infrastructure Design

Luís Ferreira da Silva, Fernando Brito e Abreu

QUASAR, CITI, Faculty of Sciences and Technology  
Universidade Nova de Lisboa (FCT/UNL)  
2829-516 Caparica, Portugal  
luis.silva@di.fct.unl.pt, fba@di.fct.unl.pt

**Abstract.** The design of large IT infrastructures is a complex problem because is dependent on many variables and must evolve rapidly to cope with business changes. The use of IT infrastructure patterns can improve this design process by allowing to reuse proven solutions to recurrent problems and by facilitating communication among IT design stakeholders. However, known IT infrastructure patterns are mostly like vendor-specific blueprints. As such, they are not very helpful in comparing alternatives and supporting independent design decisions.

In this paper we introduce patterns in the domain of IT infrastructures covering aspects from its rationale to instantiation. This is the first step in the creation of a pattern language that will hopefully leverage the IT infrastructure design process.

**Keywords:** Information Technology, IT Infrastructure, Design Patterns

## 1. Introduction

The concept of IT Infrastructure (ITI) conveys the use of various components of information technology (hardware, software and network infrastructure) upon which IT services are provided [1]. The primary purpose of an ITI is to support and enhance business processes, so they are the foundation upon which the business processes that drive an organization's success are based [2]. ITIs must be quickly adapted to support new technologies (e.g. grid services, web services, internet applications, and application integration) and new types of services (e.g. wireless, broadband media, and voice services), while enforcing stronger access control and auditing policies and keeping high degrees of flexibility and agility.

In such a scenario, one of the major problems faced by ITIs is their increasing size and complexity, that may jeopardize the delivery of real business value [3]. The size and complexity are often the result of ITIs created, designed or adapted by non ITI experts such as business decision makers, consultants, administrators, developers, software engineers, solution architects and other individuals (sometimes conflicting due to their own point of view) without ITI design guidelines and most of the times with the only purpose of responding to the requirements of a particular business application [4]. Designing ITIs for large organizations is a challenge task mainly because it requires knowledge of existing organization processes, the views of different players, and the coordination of technical expertise in three ITI domains (hardware, networking and infrastructure software) that rarely reside in a single individual.

The design of solutions is achieved in most engineering fields by using appropriate abstractions. Although often the devil is on the details, raising the level of abstraction allows practitioners to find, share and apply standardized solutions to recurrent phenomena, by only retaining the information which is relevant for a particular purpose.

In the area of IT infrastructures the application solutions to recurrent problems was caught as a business opportunity by several companies to standardize typical ITI building blocks based on their commercial components. Some of those companies developed methodological approaches to ITI pattern-based design, by proposing design "blueprints" embodying vendor-specific components [5, 6].

In this paper we will report our preliminary effort on building a *pattern language for supplier-independent* ITIs with focus on the *design* based upon proven solutions. Among other aspects, this language is expected to increase agility in the design of ITIs and contribute to their standardization with the use of well-known solutions. This effort is grounded on the hard-won lessons learned by the first author during several years of full-time work designing IT infrastructures for large companies, such as banks, telecoms and big wholesale resellers.

## 2. Pattern Language for ITI Design

The work of the famous architect Christopher Alexander and his colleagues, that created the concept of design patterns, focused not only in individual patterns, but also in the concept of *pattern language* [7]. This term originally was meant to describe a vocabulary of interacting design strategies that can be used to develop human-scale, enjoyable and durable spaces, buildings, landscapes and towns.

Generally speaking, a pattern language is a practical network of important, related ideas that provides a, as comprehensive as possible, treatment of a subject, using a common vocabulary and understanding. Usually, such languages are the result of accumulated experience and practice [17] and can be used in various situations such as to facilitate communication, sharing of ideas, build complex and heterogeneous solutions, identify recurrent problems, and provide a guided approach to solve those problems [19], therefore improving design quality and efficiency [18]. Since we are concerned with a specific knowledge area, we propose the following definition of a pattern language for ITIs:

*“An interconnected collection of IT infrastructure design patterns that come together to create a secure, reliable, available, performant and manageable IT infrastructure.”*

The use of ITI design patterns can be seen as a process to simplify the ITI design process, while reducing its risk and cost through the use of well-known solutions for recurrent problems. The solutions addressed by design patterns are not intended to be static and final. In fact, they are templates that can be customized and extended. Design patterns help breaking ITI complexity into smaller modules, thus allowing architectural decisions to be taken at a higher abstraction level. Design of infrastructures using this approach makes them more robust, scalable, reliable, and maintainable. Our ITI design patterns have a further advantage – they are supplier independent. A pattern should provide information on how a specific problem can be addressed without focusing on a specific technology or vendor.

Due to space constraints we only include here the description of one ITI design pattern, named *Distributor*, an example from the *Infrastructure Software* domain. The full description of the whole collection of our ITI design patterns will be made available at the QUASAR website (<http://ctp.di.fct.unl.pt/QUASAR/>) as a technical report.

## 3. Distributor Pattern

### Example

Most organizations have thousands of systems (servers, desktops, laptops, mobile devices, etc.) available worldwide and connected through LAN and WAN connections in multiple locations such as the internet, internal networks, perimeter networks, as well as across firewalls and other security equipment that need to be managed and supported centrally, using a systems management tool or solution that has to be designed, adapted and configured in order to address business and technical concerns.

Consider a medium size bank with thousands of branch offices connected to headquarters over 256 Kbps WAN links and managed by a single location. At each branch there are in average 10 systems (e.g. desktop and laptop computers) with a business application called FT (Financial Terminal), which is responsible for performing all financial operations in that branch. In such a scenario the availability and performance of the WAN link is crucial and aspects such as link speed, available bandwidth, number of systems and the amount of network traffic are very important. Depicting the limitations with the WAN link, each of the 10 systems has to be managed (e.g. updates for antivirus, operating system and business applications, solving problems, hardware and software inventory etc.) while ensuring no significant impact in the WAN link. A possible way to manage these systems is to have a technician visiting each branch office or have someone at the branch office to support these activities. A more cost-effective solution consists in performing these activities remotely using a centralized systems management solution. In such a scenario the deployment of 20MB of updates corresponds to 200MB (20MB x 10 systems) which will take almost 2 hours to complete over 256 KB WAN link. Without a proper design, the previously depicted software distribution operation would affect the performance of all branches for a considerable amount of time or even bring the whole infrastructure down.

## Context

Distributed environment with systems deployed across a WAN links and managed by a central location. These systems typically include servers, desktop and laptop computers, among other devices, that need to be managed from a central location across a WAN link, while minimizing the impact of the network and applications at the remote location.

## Problem

In large organizations it can be impractical or even unrealistic to manage distributed systems, in the presence of a reduced communication link. The inadequate design of a system management solution could jeopardize the business of the organization.

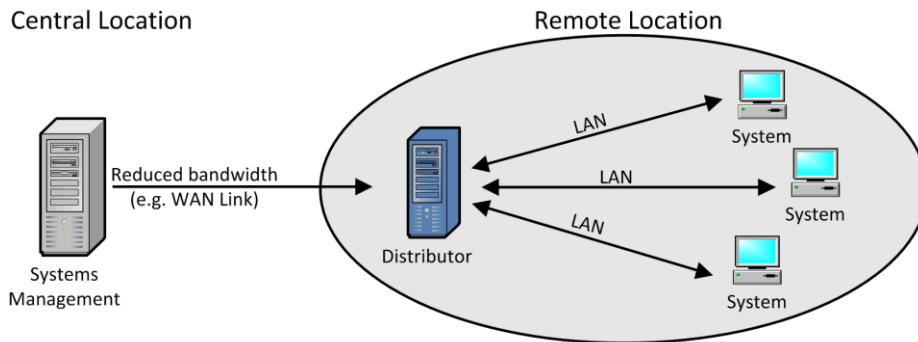
## Forces

Designing solutions to manage distributed systems taking into consideration attributes such as performance, scalability and reliability is a challenge task that often results in poor solution designs with a significant impact to the organization, which sometimes lead organizations to move from one solution to another. In the design of a systems management solution, it is important to consider the following forces that have impact as you consider a solution to the problem:

- The number of systems to manage in a remote location (with reduced bandwidth) influence solution, since may be required additional capacity to support these systems.
- The type and frequency of management activities to perform remotely should be clearly defined. For instance to obtain software and hardware inventory information from remote systems once a month, may not have impact in the efficiency of the network, while the distribution of software packages have a direct impact with reduced bandwidth;
- In the definition of the solution, more important than link speed is the available bandwidth. Often links are relatively high, but bandwidth is low due to multiple applications or applications running over the links;
- The introduction of logical network components in a system management solution should be articulated with the physical network topology;
- To ensure high performance and reliability, the system management solution must be tested across reduced links to validate that the design respond to the requirements;
- Depending upon the activities performed, existing firewalls may have to be configured to allow management activities. The ports to open in a firewall are typical port 80 and port 443, but may vary with the system management solution.

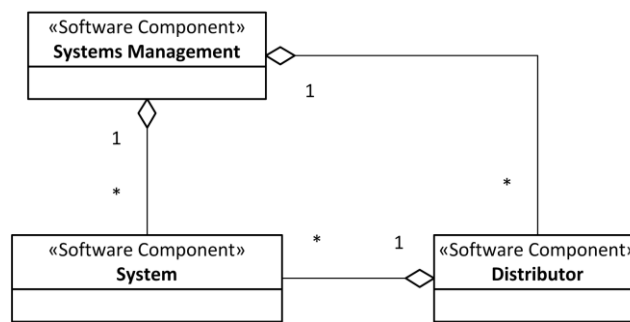
## Solution

The solution consists in the deployment of a server called *distributor* in remote locations acting as a proxy. The *distributor* is connected over a WAN link to the central location where the systems management solution resides and over LAN links to agents residing in remote systems. The use of a *distributor* provides efficient package distribution to remote locations with limited bandwidth. Fig. 1 presents a common layout for a *distributor* in one of the most common management activities (Software Distribution).



**Fig. 1:** *Distributor* (systems pull/push software packages)

With a *distributor*, for instance, packages to be distributed are downloaded only once from the central location to the *distributor* over the WAN link. Without such a *distributor*, each time a remote client system needs some packages, they must be transferred through the slow WAN link. The design structure of a systems management solution to address remote offices connected over slow WAN links is presented in Fig. 2.



**Fig. 2:** *Distributor* Pattern

The solution encompasses the following components:

- **Systems Management:** Represents the main server responsible for managing and coordinating all operations such as provide policies to agents/systems; enable or disable features (e.g. inventory, remote control); provide software repository; receiving and storing agent data among other operations;
- **System:** Each system has a piece of software running, often called *agent* responsible for providing services such as hardware inventory, software inventory and software distribution;
- **Distributor:** Component that separates system management solution from agents or systems. It acts as an intermediate between these two entities to provide efficient use of resources in operations such as software distribution, hardware and software inventory among others. This component often resides in a remote location.

To better describe the solution, we present a sequence diagram for a use case of a software distribution across a WAN link.

### Software Distribution

**Summary:** The administrator intends to distribute a business application update to several systems in a branch office connected over a WAN link.

**Actors:** Administrator.

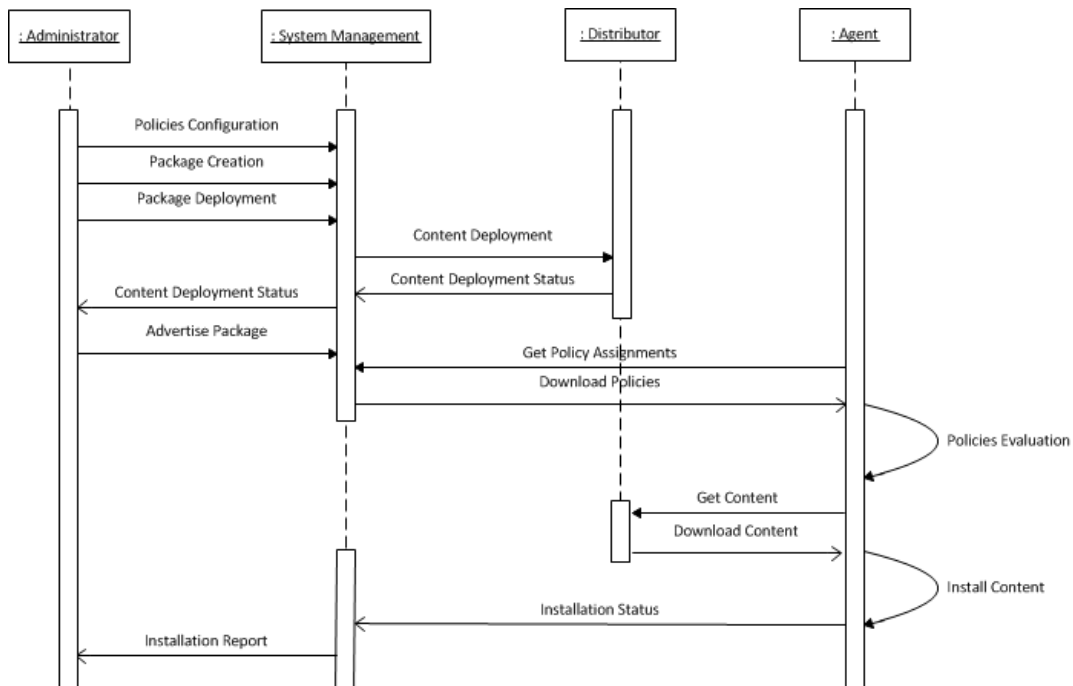
**Precondition:** Connectivity and software availability

#### Description:

- a. The administrator defines policies and creates the software package with the business application updates;

- b. The administrator deploys the software package to distributors;
- c. After successful package deployment to distributors, the package is advertised to systems running systems management agents;
- d. Agents get policies from central management
- e. Agents download content from local distributor and install content;
- f. The status of the installation is reported to Central management and a report is produced for the administrator.

**Postcondition:** A new business application update was successfully deployed to remote systems.



**Fig. 3:** Sequence Diagram for Software Distribution

## Consequences

The *distributor* pattern presents the following advantages:

- More efficiency for features such as software distribution to remote locations;
- Less impact on the network usage, since the bandwidth usage is lower;
- Less time to perform remote operations such as hardware and software inventory and software distributions;
- Less loaded central system management solution, due to the use of distributors.

The *distributor* pattern also has some (possible) liabilities:

- The number of servers deployed can be considerable higher;
- The costs associated with licensing and managing distributed systems may increase;
- Some branch offices may not have conditions for accommodating a distributor due to energy or space constraints.

## Implementation

This section discusses some implementation details regarding the use of *distributors* in system management solutions:

- Depending on the number of systems managed in a remote location the use of a single distributor may not be enough;
- Remote locations with reduced number of systems (e.g. one or two) should not have a distributor. There isn't a exact number of systems to justify the deployment of a distributor, but

most times having more than five systems in a remote location may justify the use of a distributor;

- Remote locations connected over high speed connection links may not require distributors;
- The distributor should have enough disk space to accommodate software distribution packages. That required space depends on the number and size of software packages to distribute;
- The administration of the platform should be performed centrally, but not in the systems management server, to avoid performance impacts in the server. Having administrators, operators or other users working directly on the systems management server in a daily basis, consumes resources that can affect the overall solution performance;
- The number of distributors are dependent on the number of remote locations, numbers of systems in each remote location, WAN or LAN speed;
- The distributor component can be combined with servers performing other tasks.

## Known Uses

The use of the *distributor pattern* in the conception of system management solutions can be configured in some commercial products, such as the Microsoft System Center Configuration Manager [8] or Tivoli Configuration Manager [9].

## Example Resolved

The use of a distributor in each bank branch office will significantly decrease bandwidth usage required to manage systems. Deploying 20MB for each system will be performed in approximately 10 minutes.

## 4. Related Work

As mentioned in section 1, some companies have proposed customized ITI design patterns, known as “blueprints”, with the obvious purpose of helping their customers to select the most adequate ITI configurations based upon their product and service offerings. Two examples that deserved our attention were the ones of Sun Microsystems [6] and Microsoft Corporation [5], which were the only ones we found with comprehensive related documentation available in the web.

Sun promotes the *SDN (Service Delivery Network)* approach to design service optimized network architectures for customer and in-house implementations. This approach consists of basic network building blocks, common network design patterns, integrated network components, and industry best practices that together are carefully blended in response to a customer's business and technical goals. *SDN* provides a set of network connectivity, routing, load balancing, and security mechanisms that, when applied in combination, result, according to [6], in “flexible network infrastructure designs that provide high performance, scalability, availability, security, flexibility, and manageability”. As for the patterns themselves, Sun proposes in the same document a set of so-called “common *SDN* patterns”, highlighting which key forces differentiate each pattern from the others. These patterns are based on a set of building blocks and include: the *Single Service Module Pattern*, the *Multi-Service Module Pattern*, the *Single Service Module With Integration Security Module Pattern*, the *Single Service Module With Domain Security Module Pattern*, the *Single Service Module With Integration Security Module and Domain Security Module Pattern*, the *Multi-Service Module With Integration Security Module Pattern*, the *Multi-Service Module With Service Security Module Pattern*, and the *Multi-Service Module With Integration Security Module and Service Security Module Pattern*.

Meanwhile, Microsoft promotes a related approach, named *IPD (Infrastructure Planning and Design)*, based on a set of guides that provide architectural guidance for Microsoft infrastructure. The current *IPD* documentation [10] focuses on helping the reader to plan and design the implementation of several proprietary technologies. According to the authors, the *IPD* guides are supposed to assist the architect in planning for complex scenarios requiring multiple infrastructure technologies. Those guides complement product documentation by focusing on infrastructure design options and share a common structure, that includes: (i) defining the technical decision flow through the planning process, (ii) listing the decisions to be made and the commonly available options and considerations, (iii) relating decisions and options to the business in terms of cost, complexity, and other characteristics, and (iv) framing decisions in terms of additional questions to the business to ensure a comprehensive alignment with the appropriate business

landscape. IPD highlights when service and infrastructure goals should be validated with the organization and provides additional questions that should be asked of service stakeholders and decision makers. Regarding design patterns, Microsoft organizes its approach in a set of design clusters including: *Web Presentation Patterns, Deployment Patterns, Distributed Systems Patterns, Performance and Reliability Patterns* and *Services Patterns* [5].

Both approaches provide methodological guidance along with ITI design patterns customized with proprietary products. Although their structure and detail are varied, both approaches can be seen as *proprietary pattern languages for ITI design*. We believe that a *non-proprietary pattern language for ITI design* like ours may play an important role in the design of ITIs comprising multiple source technologies.

## 5. Conclusions and Future Work

IT infrastructures (ITIs) are increasingly important for organizations, due to their impact in the business. Besides being complex, those ITIs most frequently use heterogeneous (multiple source) components. To increase the agility in ITI design, we have introduced a supplier-independent pattern language for ITI design. Currently we are consolidating this language, by detailing the collection of ITI design patterns presented in the annex. In parallel we are devising heuristics for ITI patterns instantiation and composition.

Regarding future work, there are several avenues we plan to explore. After language consolidation we intend to validate it through experts' opinion. We will use the corresponding feedback to improve the patterns themselves and to develop the methodological part of the pattern language, in order to enable patterns-driven ITI design. With a higher number of patterns in our language, we intend to build complete solutions using our pattern language. To support the application of our pattern language we plan to develop a computational environment with all ITI design patterns to facilitate the selection of the best patterns to address a specific problem. We will also plan to develop ITI pattern mining algorithms, supported by a tool, both to help documenting existing legacy ITIs and to detect which are the patterns that are more used in combination.















## Acknowledgment

This work was partially funded by the FCT/MCTES and the CITI research center.

## References

1. Sirkemaa, S.: IT infrastructure management and standards. Proceedings of the International Conference on Information Technology: Coding and Computing. IEEE Computer Society, Las Vegas (2002) 201
2. Gunasekaran, A., Williams, H.J., McGaughey, R.E.: Performance measurement and costing system in new enterprise. *Technovation* (2005) 523-533
3. Sessions, R.: *Simple Architectures for Complex Enterprises*. Microsoft Press, Redmond (2008)
4. Perry, R., Gillen, A.: *Demonstrating Business Value: Selling to Your C-Level Executives*. (2007) 15
5. Trowbridge, D., Mancini, D., Quick, D., Hohpe, G.: Enterprise solution patterns using Microsoft. NET version 2.0. *Patterns & Practices* (2003) 342
6. Lofstrand, M., Carolan, J.: *Sun's Pattern-Based Design Framework: The Service Delivery Network*. Sun Microsystems, Santa Clara, CA, USA (2005)
7. Alexander, C., Ishikawa, S., Silverstein, M.: *Pattern Languages*. Center for Environmental Structure, Vol. Vol. 2. Oxford University Press, New York (1977)
8. Mosby, C., Crumbaker, R.D., Urban, C.W.: *Mastering System Center Configuration Manager 2007 R2*. SYBEX Inc. (2009)
9. Redbooks, I.: *IBM Tivoli Configuration Manager and Tivoli Provisioning Manager for Software Coexistence and Migration Considerations*. International Technical Support Organization (2007)
10. Dunham, L., Rytkonen, P.: *Infrastructure Planning and Design*. Microsoft (2009)

## Annex – Draft of a pattern language for ITI design

Pattern / Domain	Description
 Failover Cluster Server	A set of cooperating servers that act as a failover cluster to support one or more critical applications, which should be available even with the failure of an entire server.
 Load Balancing Server	A set of cooperating servers that provide load balancing to support applications that require high availability and large number of users.
 Fault Tolerant Server	The characterization of a fault tolerant server capable of supporting services and applications with high availability requirements.
 Simple Server	The characterization of servers capable of supporting services, application or solutions that do not require high availability requirements.
 Border Network	The set of intertwined physical and network devices that support the communications from the perimeter to the internet and from the internet to the perimeter network.
 Perimeter Network	The set of intertwined physical and network devices that support the communications from the border to the perimeter and from the perimeter to the border network.
 Internal Network	The set of intertwined physical and network devices that support the communications from the perimeter network to the internal network and from the internal to the perimeter network.
 Perimeter Firewall	Requirements for the firewall placed in the perimeter network between the border and internal network.
 Internal Firewall	Requirements for the internal firewall placed in the internal network and acting as bridge between internal network and perimeter network.
 ITI Messaging	Messaging system taking in consideration specific Service Level Agreements such as availability, reliability, and scalability.
 ITI Directory	Central repository for authentication and authorization, which is able to store, organize and provide access to information.
 Perimeter Web	The characterization of web servers to be deployed in the perimeter network.
 Perimeter Database	The characterization of a database system to be deployed in the perimeter network.
 Internal Proxy	The configuration that provides internal users a secure and efficient access to the internet, using a proxy.

*Note - the icons denote the domain dominance of each pattern, as follows:*

 - Hardware

 - Network

 - Software