Dynamic Service Discovery

A position paper for the W3C Workshop on Web Services for Enterprise Computing, by Kinga Dziembowski of Gestalt-LLC.

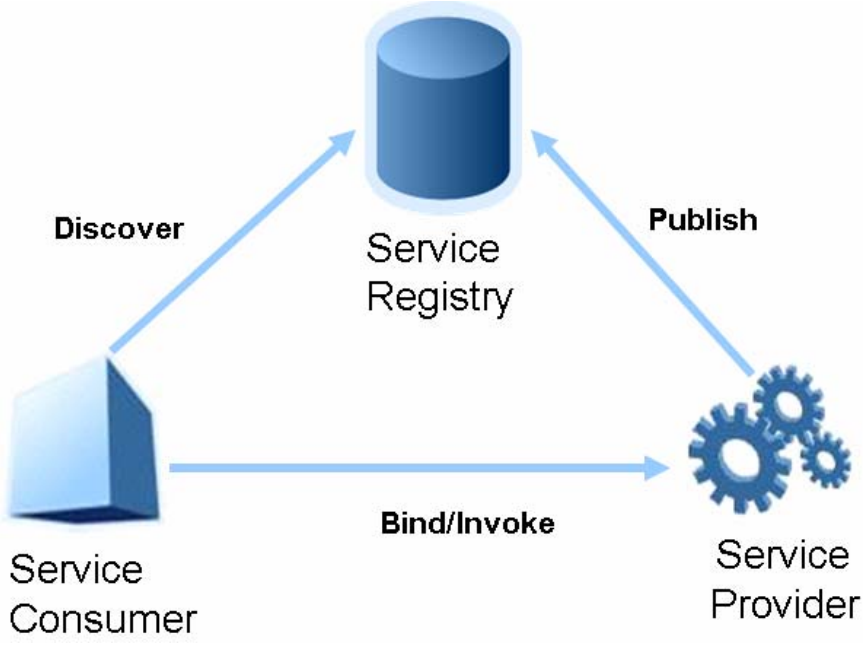
My position

Service Discovery in the dynamic and transient nature of a highly distributed and mobile enterprise, such as the US military, requires a much more dynamic concept of Service Discovery than what is currently handled by most commercial Service-Oriented Architecture implementations.

Problem statement

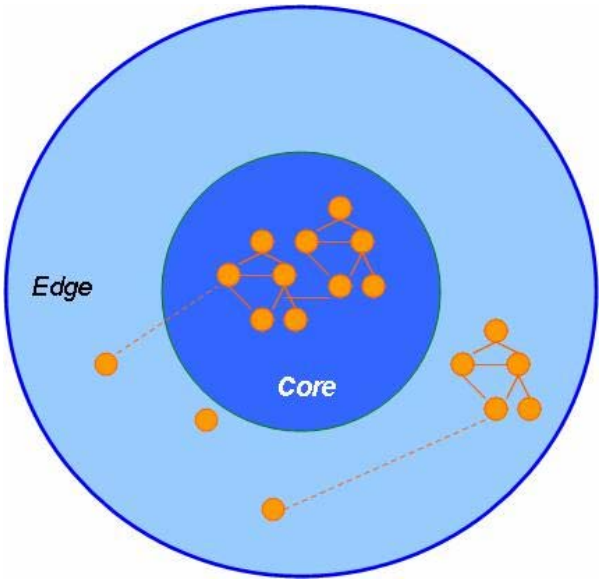
The Web Services architecture, associated specifications and standards make one general assumption: The infrastructure in which the service customers and providers operate is *static*. It is generally assumed that services deployed stay deployed, end point URLs, which point to public services once established stay operational. This picture of static service end points may well represent the current commercial, web-based marketplace, but in a world where services exposed by the enterprise are part of their competitive differentiation and non-operational services consist of business lost for the enterprise, a service discovery architecture that assumes a non-static "fabric" of services distributed across the enterprise may itself be a competitive advantage for a business.

The commercial world comprises the "core" environment in which the well-known Service Discovery paradigm - the triangle between Consumer, Provider, and Registry - works quite well.[1]



[1] There are still many problems associated with the semantic understanding of published information about the service, but this class of problems is outside of the scope of this paper.

In the highly distributed and mobile world of the military, the space in which Web Services and web technologies operate is much more complex. Part of the military enterprise operates in a similar environment to the commercial world – much like the "core." Other areas operate on the "edge" which has much different characteristics from the "core".



- The "core" is static, stable, has known behaviors, is established, and is always connected. Services are assumed to be operational 24x7.
- The "edge" is dynamic, transient, and sometimes disconnected; you cannot assume that the Service is available at any point of time.

An example of the "edge" are the services provided by mobile units deployed within theaters of operation, which can be temporarily disconnected and then reconnected at some point of time, often to another network segment. You can think about planes which provide image services as part of a reconnaissance operation; planes change their positions, they can be "on" the mission or they can be "off" the mission, yet each plane implements this same Service consumed by consumers.

In such dynamic environments the canonical Services Discovery paradigm is not enough. Additional Dynamic Service Instance Discovery is needed.

It can be argued that this problem should be solved on a per case basis by specific solutions based on specific requirements. **My position is that the problem is applicable to a large enough "population" of communities of interest and requires more global, systematic attention and supporting architectures and standards. At a minimum design patterns, recommendations and best practices for dynamic service discovery should be provided**.
If this problem is not solved at the architecture and standards level, the main benefit of SOA and Web Services – INTEROPERABILITY – will be compromised or lost.

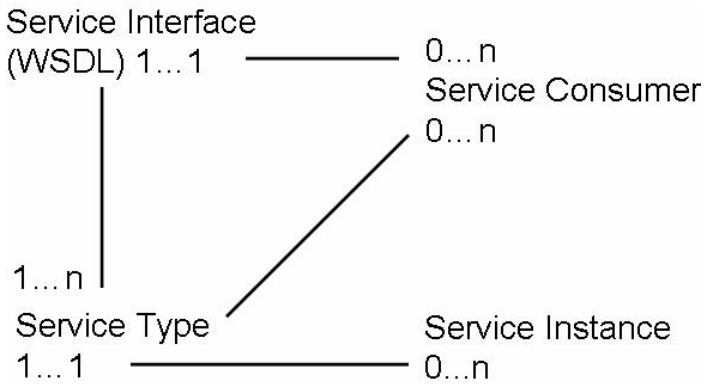
Use cases

For the dynamic and transient nature of military enterprises I see at least four classes of use cases which require additional requirements for Service Discovery.

- Mobile Service Clients – such clients plug into existing systems and need to discover where the needed Services are in the current environment.
- Mobile Services – the Service Instance is moving from point ot point on the network. What is important for this use case is the ability for the client to invoke a specific Service Instance – the state associated with the Service Instance matters.
- Changing Service Instance population – (transient services). The goal for Service Discovery is to find what Service Instances are currently available, it is not important to find a specific Instance, any one instance of a particular service type will satisfy the consumer need.
- Instance federation – In this use case multiple services instances spread resource responsibilities. The population of the Instances is static or slowly changing ("the core"). Service Discovery is concerned with finding the Service Instance with the appropriate context.

What needs to be discovered?

Let's introduce a few definitions and relationships between the defined entities observed in the Service Discovery problem space:



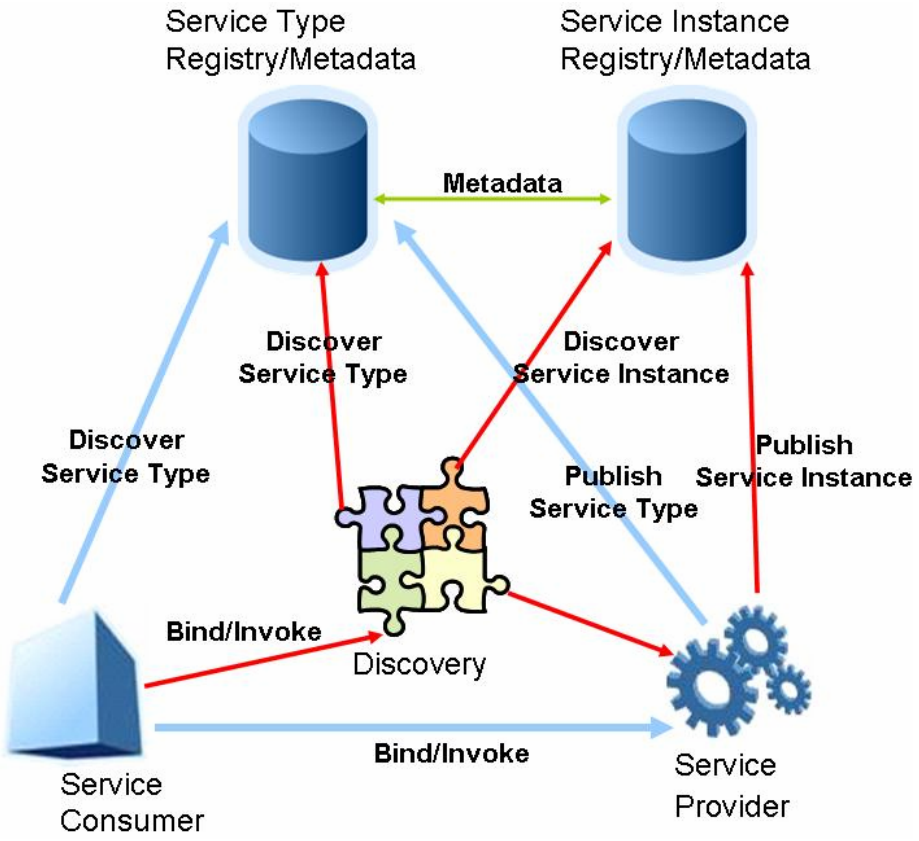
- Service Type – The capability of the Service and the interface for this capability. Service Type contains more than the Web Services Description Language (WSDL), additional metadata describing the capability is part of the Service Type
- Service Instance – The instance of the Service Type capability, it is the implementation of the capability interface and additional metadata identifying the Service Type attributes as well as unique Instance attributes. The Service Instance is managed independently.
- Service End Point – The entry point to the Service Instance. There can be multiple end points for this same Service Instance (for example performance optimizations).

When is Service Discovery performed?

From the Service Client point of view Service Discovery is performed at:

- Design/test-time – The goal of this type of Service Discovery is to answer the question: "does the Service I need exist?" In order to obtain all necessary information to construct the Service Client capable of requesting the Service from the Provider. The Service Type information is the expected result of the successful Discovery process at this lifecycle stage. The requirements for this type of Service Discovery are not time sensitive.
- Deployment-time – At the time when the Service Client is deployed. Service Discovery at this lifecycle stage is concerned with Service Instances and their now real endpoints. At this point the Service Interface definition is not important – the Service Client already adheres to it. The requirement for this type of Discovery is time sensitivity.
- Run-time – The goals for Service Discovery at this stage are to find the appropriate Service Instance satisfying the Consumer's needs. All the complexity of the "edge" use cases need to be addressed here.

The original Service Discovery diagram needs revision to address all aspects of the Dynamic Service Instance Discovery space required for dynamic and transient enterprises (again, such as the military).



The blue arrow based triangle, satisfies only the needs of Design-time/test Discovery. For the Deployment and Run-time Service Discovery stages there is a need to address **Availability** and the **Presence** of Service Instances in a similar way in which an Instant Messaging "buddy-list" maintains the network of entities available for connection. The Service Instance Registry needs to collect real time (or near real-time) information about Service Instance Presence and the changing context of the Instance. The Service Provider/Infrastructure in which the Service Instance lives needs to notify the Service Instance Registry about the changes. The intelligent Service Discovery process will use this information to dynamically find the appropriate Instance satisfying the Consumer need. The Service Type as well Service Instance Registries contains the appropriate metadata for discovery of this service – this metadata needs to be "linked" in order to enable the benefits of discovery.

Conclusion

It will be of tremendous benefit for enterprise communities to have the opportunity during the W3C workshop to discuss and plan the future work needed to provide the architectural patterns and technologies to solve the Dynamic Service Discovery problem for dynamic, transient enterprises in an interoperable way.

- What are the mechanisms, standards, specifications and technologies covering those additional needs?
- Can the Service Type and Service Instance Registry be this same registry? Can UDDI be used for such purposes?
- Should the Registries be replicated or federated?
- What are the criteria for such decisions? What is the role of content search within this problem space? What technologies can be used for each class of use-cases?

These and many more questions need to be answered, and because of the ever changing landscape of enterprises and the never-ending push for mobility and instantaneous reconfiguration of the enterprise technology platform, there is a real need for a cross industry effort to address the problem of dynamic service discovery stated above.