Qualitatively

The rules in SOAP are important because without these rules, you can’t achieve any level of standardization. REST as an architecture style does not require processing and is naturally more flexible. Both SOAP and REST rely on well-established rules that everyone has agreed to abide by in the interest of exchanging information.

Many developers found SOAP cumbersome and hard to use. For example, working with SOAP in JavaScript means writing a ton of code to perform extremely simple tasks because you must create the required XML structure absolutely every time. one perceived disadvantage is the use of XML because of the verboseness of it and the time it takes to parse.

REST doesn’t have to use XML to provide the response. You can find REST-based Web services that output the data in Command Separated Value (CSV), JavaScript Object Notation (JSON) and Really Simple Syndication (RSS). The point is that you can obtain the output you need in a form that’s easy to parse within the language you need for your application. While this may seem like it adds complexity to REST because you need to handle multiple formats, JSON usually is a better fit for data and parses much faster. REST allows better support for browser clients due to its support for JSON.

SOAP uses WSDL for communication between consumer and provider, whereas REST just uses XML or JSON to send and receive data

SOAP is language, platform, and transport independent SOAP can use almost any transport to send the request, using everything from the afore mentioned to SMTP (Simple Mail Transfer Protocol) and even JMS (Java Messaging Service), but REST requires use of HTTP/HTTPS. Our implementation takes advantage of SOAP so it is also language, platform, and transport independent.

SOAP works well in distributed enterprise environments, but REST assumes direct point-to-point communication and our implementation works well with distributed enterprise environments [CHECK ARTICLES]

SOAP is standardized REST \*\*

SOAP spends a lot of bandwidth communicating metadata so it is not good when your bandwidth is very limited. REST approach uses the standard *GET*, *PUT*, *POST*, and *DELETE* verbs. Again, remember that REST can also use the *XMLHttpRequest* object that most modern browsers support today, which adds an extra bonus of AJAX.

SOAP is hard to implement and is unpopular among Web and mobile developers

SOAP provides significant pre-build extensibility in the form of the WS\* standards

SOAP has automation when used with certain language products. For example if you know WSDL file REST has no automation

REST is easier to use for the most part and is more flexible and no expensive tools require interacting with the Web service. Our implementation does not need any expensive tools \*\*\*

REST is more efficient because SOAP uses XML for all messages, REST can use smaller message formats so it is better to use when information about objects doesn’t need to be communicated to the client. It is beeter to use • When clients need to have access to objects available on servers. But Our implementation uses binary [Article]

REST is faster than SOAP because no extensive processing required. In case of our implementation

if the information can be cached because of the totally stateless operation of the REST approach

When clients need to have access to objects available on servers

SOAP

SOAP (Simple Object Access Protocol), was created in 1998 by Dave Winer et al in collaboration with Microsoft. Developed by a large software company, this protocol addresses the goal of addressing the needs of the enterprise market.

SOAP relies exclusively on XML to provide messaging services. Microsoft originally developed SOAP to take the place of older technologies that don’t work well on the Internet such as the Distributed Component Object Model (DCOM) and Common Object Request Broker Architecture (CORBA). These technologies fail because they rely on binary messaging; the XML messaging that SOAP employs works better over the Internet.

The XML used to make requests and receive responses in SOAP can become extremely complex. In some programming languages, you need to build those requests manually, which becomes problematic because SOAP is intolerant of errors. However, other languages can use shortcuts that SOAP provides; that can help you reduce the effort required to create the request and to parse the response. In fact, when working with .NET languages, you never even see the XML.

Part of the magic is the Web Services Description Language (WSDL). This is another file that’s associated with SOAP. It provides a definition of how the Web service works, so that when you create a reference to it, the IDE can completely automate the process. So, the difficulty of using SOAP depends to a large degree on the language you use.

An interesting SOAP feature is that you don’t necessarily have to use it with the HyperText Transfer Protocol (HTTP) transport. There’s an actual specification for [using SOAP over Simple Mail Transfer Protocol](http://www.pocketsoap.com/specs/smtpbinding/) (SMTP) and there isn’t any reason you can’t use it over other transports. In fact, developers in some languages, such as Python and PHP, are [doing just that](http://www.drdobbs.com/writing-smtp-based-soap-messages-in-php/184416579).

REST

REST (Representational State Transfer) was Created in 2000 by Roy Fielding in UC, Irvine. Developed in an academic environment, this protocol embraces the philosophy of the open Web.

REST provides a lighter weight alternative. Instead of using XML to make a request, REST relies on a simple URL in many cases. In some situations you must provide additional information in special ways, but most Web services using REST rely exclusively on obtaining the needed information using the URL approach. REST can use four different HTTP 1.1 verbs (GET, POST, PUT, and DELETE) to perform tasks.