# Test of several approaches for the composition of web services in meteorology.

Benoît Gschwind, Lucien Wald, Robert Mahl, François Irigoin, Lionel Ménard

Gschwind Benoît, Lucien Wald, Robert Mahl, François Irigoin, Lionel Ménard, 2007. Test of several approaches for the composition of web services in meteorology. In Proceedings EnviroInfo 2007, O. Hryniewicz, J. Studzinski and M. Romaniuk (Eds), Shaker Verlag, vol. 1, pp. 127-134.

# Test of several approaches for the composition of web services in meteorology

Benoît Gschwind[1], Lucien Wald[1], Robert Mahl[2], Francois Irigoin[2] and Lionel Menard[1]

*Abstract*

*Composition of web services is a powerful means to answer user needs in many domains. This communication focuses on applications in meteorology. This domain presents several particularities which are discussed. We present an overview of three main types of approaches in composition of web services: static plan, IA-plan and theorem proof. We confront these types of approaches to the specific case of meteorology which is not well studied by the research community in web services. We design a test bed that is capable of handling the key issues in meteorology. We have selected three approaches. We adapt them to our case and build three prototypes which are used in the test bed to point out weakness and strength of each approach. We find that current approaches do not fulfill needs in meteorology. We recommend an hybrid approach that combines the three in order to obtain an automatic and adaptative composition.*

## 1. Introduction

This communication deals with web services for applications in meteorology and focuses on the composition of web services. In this domain, like many others, data and applications are geographically dispersed. Weather bureaus are collaborating since long at international level for weather forecast. Other topics in meteorology are also highly demanding for exchanging data and for a better management of the knowledge at an international community level. This is the case of solar radiation, where a need was strongly expressed by various stakeholders (Cros et al. 2004, Dunlop et al. 2006, Stackhouse et al. 2006). The SoDa Service[3] has been created to answer the needs for an harmonized access to information (Gschwind et al. 2005). It invokes web services for the benefit of several tens of thousands users interested in knowing solar radiation for various purposes: energy, building engineering, health, agriculture, meteorology, climate *etc.* (Gschwind et al. 2006). The SoDa Service exploits combination of web services; up to three web services can be chained to offer a new compound web service having a higher added-value (Wald et al. 2002). The composition is defined manually: the plan that describes data flow and how and when to use web services is pre-built and remains unchanged. After four years in operation, the SoDa Service team decided to study solutions for automatic and adaptive composition. Performing such a composition will improve the efficiency of the SoDa Service by allowing to extract the best available resources from the ensemble of data and applications and consequently offer the best possible answer to the user. This communication reports on the first results of this investigation.

Several steps are needed to perform composition: first a query, next descriptions of web services, then build a plan from the descriptions and finally the execution of the so-built plan. A plan is a sequence of

[1]Ecole des Mines de Paris, Center for Energy and Processes, Sophia Antipolis, France
email: benoit.gschwind@ensmp.fr, lucien.wald@ensmp.fr, lionel.menard@ensmp.fr
[2]Ecole des Mines de Paris, Centre de Recherche en Informatique, Sophia Antipolis, France
email : robert.mahl@ensmp.fr, francois.irigoin@ensmp.fr
[3] http://www.soda-is.com

actions that are executed to reach a goal. In web service composition, a plan is composed of several invocations of web services that send and/or receive message from/to web service agents and data flow.

Composition of web services can be divided in three main categories with respect to the degree of automation of the building of the plan: the already discussed manual composition, the composition assisted by the user and the automatic and adaptive composition. The two latter categories offer the advantage of being less constraining for the user of the SoDa Service. The composition assisted by the user represents an advantage compared to the automatic composition because the user knowledge is fully taken into account, even if done implicitly. However, it needs interaction between human and system. Automatic and adaptive composition provides a system that does not need human input. It can handle dynamic changes in web service environment.

The composition is often performed *a priori*. The *a priori* composition handles web services before their invocation. It is based on the description of the web services that can be provided by the means of *e.g.*, WSDL (web service description language). In several cases, an *a posteriori* composition can be performed. It uses the outcomes of a web service, such as result, execution error or partial result, to re-engineer the plan if possible and necessary. In literature we find three main types of approaches for building plans for web services composition : static plan approach, IA-plan approach and theorem proof approach. Static plan approach uses plan model to deduce a plan (Casati et al. 2000, Gschwind et al. 2005, Schuster et al. 2000). A plan model is a model that handles set of plans. An example of plan is the following sequence, “first I take a bus, then fly and go to the hotel”. An example of plan model will be “if the bus is here, I take it, otherwise I take a taxi, then I take the plane, finally if my first choice hotel is not full, I book a room otherwise I find another one”. The difference between these two plans (plan and plan model) is that the first one is only one plan since the second represents several plans, like “I take the bus, then I take the plane, then I sleep in my first choice hotel” or “if necessary I take a taxi, before fly and finally I find another hotel”. In this approach a plan model is pre-defined and never changes. An instance of this approach is BPEL (Business Process Execution Language) which is a model plan description language (Agarwal et al. 2005, Andrade/Fiadeiro 2004, Barros et al. 2005, Khalaf et al. 2006, Pasley 2005, Van Der Aalst et al. 2005) . An improvement is proposed by (Sirin et al. 2002) which creates a model of abstract plan and chooses a web services in real time. This approach relies on a description of ideal web services instead of linking each step of the plan to a specific web service; the system selects the appropriate and available web service in a *just-in-time* mode. The AI-planning approach considers the problem of composition as a planning problem (McDermott 2002, McIlraith/Son 2002, McIlraith et al. 2001, Ponnekanti/Fox 2002, Russell/Norvig 2003, Wu et al. 2003). Web service description is exploited to create pre-condition and post-condition that are fed into a planning engine. The post-condition is often called “effect”. A basic approach uses input as pre-condition like “If I have data that are compatible with expected input I can use this web services”, and output as post-condition like “I get data”. Matching procedures can be used in such cases that can tell if types are compatible or input and output are compatible. Finally the last approach is the theorem proof approach, or rules-based approach (Lämmermann 2002, Medjahed et al. 2003, Rao et al. 2003, Rao et al. 2004, Waldinger 2001) . It considers each web service as axiom or rule, and query as theorem. It combines them to get a proof of the query. A simple example is the following. “A give B” (web service 1), “B give C” (web service 2), “does A can give C?” (query), if yes, the plan can be deduced from the proof.

## 2. Web services particularities in meteorology

The web services in meteorology have several particularities that should be taken into account for designing an efficient solution to composition. These services are only request-response and are synchronized; this keeps us distant from several important issues in composition. Next, we have two types of operations:

one is algorithm implementation and the other is made of operations that give data. Both of them are state-less. A state-less web service or operation does not use stored session information, in opposite to state-full which uses this kind of information. Moreover, our web services are read-only; they never commit data nor use data from web services to create or change database. To say differently, our operations are input/output only for pure algorithm; this kind of web services does not use hidden input or hidden output. Concerning our pure data web services use input/output; they often build their outputs from hidden inputs, which are in fact data originating from databases and that are unseen in the outputs though they affect them. And finally, most of our web services are mixed between algorithm web services and data web services that use input/output and hidden input. These particularities represent a very strong hypothesis. They characterize this domain and offer a simpler context to composition than the very general case of dynamic composition.

However, some complexity arises from the fact that our web services handle data that have a geophysical meaning and that often have mathematical representation. These meaning and representation can be used to handle unit, data transformation, combination, and composition validation. Our web services often handle discrete data in time and/or space domains. For example a typical request is for a time series of daily radiation data at specified geographical locations for specified periods. Such requests need description that allows to describe a list with one or more keys relating to time.

Finally, another source of complexity arises because of the need to assess the quality or accuracy of the outputs of the composition. Like in many domains, providing data is not sufficient if they are not given with quality parameters (e.g., errors in temperature assessment). Providing quality parameters is a condition for a service to be used with confidence. Assessment of data quality has always been an issue in meteorology and will be for long mostly because of the lack of clear reference in most cases (Cros et al. 2004). Assessing quality of outputs of a service is therefore an issue by itself. Web services composition does not simplify this issue because then, the propagation of the errors should be taken into account in a dynamic way. Having these particularities in mind, we analyzed the advantages and limitations of the three approaches discussed in the previous section.

## 3. Expected improvements from the composition

Four years of operation has shown the limitations of a static plan. Though the present solution is performing efficiently, the SoDa Service will welcome an advanced solution that would bring the following improvements:

1. *Take into account heterogeneity*: the composition should handle differences between web services, like difference between units of data or difference in temporal sampling or spatial resolution of data.
2. *Catch errors*: when a web service returns an error, the system should be able to choose to replace it by another web service or compose web services for replacement or rollback and choose another plan.
3. *Complete data*: when web services give an incomplete result, the system should complete the result using another web service or composite web services. For several web services it is possible to detect gaps, e.g. in time series, and the system should be able to fill in these gaps by invoking another web service. Of course, the traceability of each data should be ensured.
4. *Increase data quality*: the system may take advantage of the description of the quality given in the service description or in the result and uses it to e.g., select the service providing the best quality in a given set of services. The quality can be used to select which web services can complete or increase quality of current time series.
5. *Data quality estimation*: the system should be able to handle data quality attribute through the composition. Each web service should provide information about data quality.

6. *Take advantage of new web services*: it is an hot-plug approach that allows providers to give a new web service or change a current one without changing anything in the plan except agent description and agent implementation.
7. *Create composite web services*: this kind of web services is different from classic web services, because they result from the chaining of existing web services using an engine that creates composite web services.

These expected improvements can be divided in two groups. The first one comprises improvements for web service composition in general; they are not specific to meteorology: 1, 2, 6, 7. The second group is specific to meteorology: 3, 4, 5.

## 4. Test of approaches

We designed a test bed to handle the issues listed above. It forms an objective means to test and compare different solutions with respect to the expressed concerns. We base our test bed on 15 scenarios that test different aspects of composition. A scenario handles one or more aspects of composition in meteorology, such as units, data quality or spatial/temporal distribution. Scenarios were built in order to demonstrate the relevance of the proposed solution to solve one or more key issues. Examples of scenario are :

**Description:** The web service $S_3$ gives data in unit $U_1$. The web service $S_4$ is a unit converter and gives data in unit $U_2$ from data in unit $U_1$. Combine $S_3$ and $S_4$ web services to get data in unit $U_2$ *a priori*.
**Aim:** Check system capability to compose web services according to user defined unit.
or
**Description:** Using both $S_{36}$ and $S_{37}$ allow using $S_{38}$. Use $S_{36}$ and $S_{37}$, and finally use $S_{38}$
**Aim:** Check system capability to invoke a web service from result of two others.

Each web service ($S_n$) in each scenario is instanced by actual web services that are part of the test bed. The test bed is described by using web services that provide only one operation. For test purpose, operations can be grouped in one or more web services. To perform our test we use two cases of studies in meteorology. The first one use temperature time-series and temperature conversion, whereas the second one is about solar radiation and its corresponding conversion.

We tried three approaches. First, we tested an approach based on static plan definition. For that purpose, we developed a language similar to BPEL (Business Process Execution Language) to describe the plan. Our test shows that this approach solves all issues except one since it fails in taking into account Internet context, because plans are pre-built by human and remain static. The static approach is very interesting in its most advanced versions because it is capable to handle all data flows and plans we could wish. There are drawbacks: it cannot handle easily new web services, because an expert is required to draw all plan models, and it cannot find web services. This understanding and integration process can be error prone, time-consuming, and hard.

The second approach is based on comparison between outputs and inputs. It uses the following rules : “if input contents of web service A are included in output content of web services B then we can use web services B first and use web service A with the outcome of web service B”. This rule allows the building of dependency graph, where nodes represent web service and arcs are dependency between web services. To find new composition the method consists in searching path in graph that bind entry point and exit point. The entry point is found by using requested input contents and the exit point is found by using requested output contents. In our test, this approach cannot handle quality issue or *a posteriori* composition. Quality issue could be solved by using a quality mark for each web service and taking it into account

when searching a path. This approach can handle change in Internet context by refreshing the dependency graph at each web service invocation. This approach cannot perform our second example scenario which uses two web services to use the third web service. This drawback is explained by our weak rule. A method to solve this issue is to change this rule by a more general rule like : "if input contents of web service A are included in union of output contents of web services set, then we can use web service A after having used this set of web service". This rule is more powerful to find composition but is less powerful to build dependency graph, and new graph are more complex since arcs are not 1 to 1 but N to 1. The IA-planning approach is interesting because it gives a fairly easy way to combine web services, since translation rules between inputs/outputs and pre-/post-conditions are simple. The major weakness is that it does not handle web services semantics, *i.e.* it does not give a meaning to web services. For example: a web services which adds $a$ and $b$ and a web service which computes $a$ times $b$ cannot be distinguished since $a$ and $b$ are translated to the same pre-condition and the result of these web services are translated to the same post-condition.

Finally we try another approach based on "theorem proof". We use Prolog to describe web services and give relation between their inputs and their outputs. We define a web service as a rule that say "web service A with I as input and O as output exist if and only if I and O are in certain form and have specific relations". Next we define a composition rule that says "web service AB with I as input and O as output exist if and only if A is a web service and B is a web service and the outputs of A are the inputs to B". To find composition we represent the request by a web meta-service. Next we try to prove that this web service exists, and we deduce the composition from this proof. This approach defines rules for composition of web services which are more expressive than the second prototype because we do not compare the inputs and the outputs only but also how web services transform data. This approach is more effective than others in term of soundness; for example, the second approach may confuse multiplication and addition services while this approach cannot perform such a semantic mistake. However, this approach fails in the second example. The theorem proof approach is an alternative way that can give meaning to web services. Nevertheless this approach is hard to get working because it need to add information about web service that are not currently given by web services providers and that could be hard to provide. Therefore, adopting this approach likely implies changes to current web services approaches.

We also note that all approaches are *a priori* approaches and are not adaptative. Several papers try to improve these approaches by increasing human intervention (Casati et al. 2000, Ponnekanti/Fox 2002) or by selecting web services on call, already discussed in section 1 (Sirin et al. 2002). Today BPEL is the best language for web services composition. In meteorology, the most important need is the capability of adaptation which allows the use of new web services without any human intervention, that is not currently performed except in very few papers. Other issues are assessment of quality of data and data completion which are often omitted.

Presently, solutions for the combination of web services are inappropriate or unsatisfactory in meteorology. This domain is characterized by the importance of the space and time for data and applications. These particularities are not handled properly by the solutions found in literature. This conclusion is supported by the analysis above and by the test of several solutions. Our opinion is that we should combine different approaches. More precisely, the strategy we want to follow is to set up a static plan as a first basis. This would be done by the means of plan model. Next we want to use a mix of IA-planning approach and theorem proof, first to define axiom and secondly to make a proof. We will improve web services description in an abstract program that describes web services behavior, their inputs, their outputs, their hidden inputs, their hidden outputs. The query is a program in the same language that we use to find the appropriate plan model. This strategy is currently under development.

**Bibliography**

Agarwal, V., Chafle, G., Dasgupta, K., Karnik, N., Kumar, A., Mittal, S., and Srivastava, B. (2005) Synthy: *A system for end to end composition of web services*. In: Journal of Web Semantics, pp. 311-339.

Andrade, L. F. and Fiadeiro, J. L. (2004) *Composition contracts for service interaction*. In: Journal of Universal Computer Science, vol. 10, pp. 375-390.

Barros, A., Dumas, M., and ter Hofstede, A. H. M. (2005) *Service interaction patterns*. In: Lecture Notes in Computer Science, no. 3649, pp. 302-318.

Casati, F., Ilnicki, S., Jin, L., Krishnamoorthy, V., and Shan, M.-C. (2000) *Adaptive and dynamic service composition in eflow*. In: Proceedings of 12th International Conference on Advanced Information Systems Engineering, Stockholm, Sweden,. Springer-Verlag.

Cros, S., Mayer, D., and Wald, L. (2004) *The Availability of Irradiation Data*. In: International Energy Agency, Report IEA-PVPS T2-04: 2004.

Dunlop, E., Wald, L., and Suri, M. (2006) *Solar Energy Resource Management for Electricity Generation from Local Level to Global Scale*. In: Nova Science Publishers. ISBN: 1-59454-919-2.

Gschwind, B., Menard, L., Albuisson, M., and Wald, L. (2005) *Three years of experience with the SoDa web service delivering solar radiation information: lessons learned and perspectives*. In : J. R. E. J. Hrebicek, editor, Proceedings of the 19th International Conference on Informatics for Environmental Protection, pages 95-102, Czech Republic. Masaryk University in Brno.

Gschwind, B., Menard, L. , Albuisson, M., and Wald, L. (2006) *Converting a successful research project into a sustainable service: The case of the SoDa web service*. In: Environmental Modelling & Software, 21(11):1555–1561.

Khalaf, R., Keller, A., and Leymann, F. (2006) *Business processes for web services: Principles and applications*. In: IBM Systems Journal, vol. 45, pp. 425-446.

Lämmermann, S. (2002) *Runtime service composition via logicbased program synthesis*. In: Department of Microelectronics and information Technology, Royal Institute of Technology, Stockholm, Sweden.

McDermott, D. (2002) *Estimated-regression planning for interactions with web services*. In: Proceedings of the 6th International Conference on IA Planning and Scheduling, Toulouse, France. AAAI Press.

McIlraith, S. A. and Son, T. C. (2002) *Adapting golog for composition of semantic web services*. In: Proceedings 8th International Conference on Knowledge Representation and Reasonning, Toulouse, France.

McIlraith, S. A., Son, T. C., and Zeng, H. Semantic webservices. IEEE Intelligent Systems, 3(16):46–53, 2001.

Medjahed, B., Bouguettaya, A., and Elmagarmid, A. K. (2003) *Composing web services on the semantic web*. In: The VLDB Journal, vol. 12, pp. 333-351.

Pasley, J. (2005) *How BPEL and SOA are changing web services development*. IEEE Internet Computing, vol. 9, pp. 60-67.

Ponnekanti S. R., Fox A. (2002) *SWORD: A Developer Toolkit for Web Service Composition*. In Proceedings of The Eleventh World Wide Web Conference (Web Engineering Track), Honolulu, Hawaii, USA, May 7-11, pp. 83-107, 2002.

Rao, J., Küngas, P., and Matskin, M. (2003) *Application of linear logic to web service composition*. In: Proceedings of 1st International Conference on Web Services, Las Vegas, USA.

Rao, J., Küngas, P., and Matskin, M. (2004) Logic-based web services composition: from service description to process model. In: Proceedings of the 2004 International Conference on Web Services, San Diego, USA. IEEE.

Russell, S. and Norvig, P. (2003) *Artificial Intelligence A Modern Approach*. Pearson Education, Inc. Second Edition.

Schuster, H., Georgakopoulos, D., Cichocki, A., and Baker, D. (2000) *Modeling and composing service-based and reference process-based multi-enterprise processes*. In: Proceedings of 12th International Conference on Advanced Information System Engineering, Stockholm, Sweden. Springer-Verlag.

Sirin, E., Hendler, J., and Parsia, B. (2002) *Semi-automatic composition of web services using semantic descriptions*. In: Proceedings of Web Services : Modeling, Architecture and Infrastructure Workshop in conjonction with ICEI2003.

Stackhouse, P., Renne, D., Perez, R., Meyer, R., Wald, L., and Suri, M. (2006) *Towards designing an integrated earth observation system for the provision of solar energy resource and assessment*. In: Proceedings of IEEE IGARSS Symposium 2006, Washington, DC, USA, 2006. IEEE. Denver, Colorado, USA, 31 July 4 August 2006.

Van Der Aalst, W. M. M., Dumas, M., ter Hofstede, A. H. M., Russell, N., Verbeek, H. M. W., and Wohed, P. (2005) *Life after BPEL?* In: Lecture Notes in Computer Science, 3670:35–50.

Wald, L., Albuisson, M., Best, C., Delamare, C., Dumortier, D., Gaboardi, E., Hammer, A., Heinemann, D., Kift, R., Kunz, S., Lefevre, M., Leroy, S., Martinoli, M., Menard, L., Page, J., Prager, T., Ratto, C., Reise, C., Remund, J., Rimoczi-Paal, A., der Goot, E. V., Vanroy, F., and Webb, A. (2002) *SoDa: a project for the integration and exploitation of networked solar radiationdatabases*. In: Environmental Communication in the Information Society, pp. 713–720, Vienna, Austria, 2002. the International Society for Environmental Protection. W. Pillmann, K. Tochtermann Eds.

Waldinger, R. J. (2001) *Web agents cooperating deductively*. In FAABS '00: Proceedings of the First International Workshop on Formal Approaches to Agent-Based Systems-Revised Papers, pp. 250–262, London, UK, 2001. Springer-Verlag.

Wu, D., Sirin, E., Hendler, J., Nau, D., and Parsia, B. (2003) *Automatic web services composition using SHOP2*. In: Workshop on Planning for Web Services, Trento, Italy.