A Situation-centric Approach to Meteorological Services in the SITUMET Platform

Stefan Pfennigschmidt
Fraunhofer Institute for Software and Systems Engineering
10178 Mollstraße 1
Berlin, Germany
stefan.pfennigschmidt@isst.fraunhofer.de

Agnès Voisard
Fraunhofer Institute for Software and Systems Engineering and FU Berlin
10178 Mollstraße 1
Berlin, Germany
agnes.voisard@isst.fraunhofer.de

ABSTRACT

In applications that need meteorological forecast data, for instance for planning purposes (such as construction or environmental applications), each weather service is currently an effort-intensive result of domain-dependent programming. Instead, such applications need weather information platforms that would combine new forecast technologies and new information and communication technologies in a flexible manner. In this context, we introduced SITUMET, a platform for situation-based meteorological services. The core of the platform is based on (i) open sensor integration, (ii) a flexible forecast module, and (iii) situation-based service generation. In this paper we focus on the later, which is based on the notion of users' context that remains constant during a time interval. We introduce a *situation-centric* approach to dynamic weather service provision in push and pull mode. It is based on a common model for current and planned activity sequences and for weather information. We illustrate our approach with a particular up-and-running mobile application defined in the SITUMET framework.

Categories and Subject Descriptors

H.1 [**Models and Systems**]: Miscellaneous; H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Design, Experimentation

Keywords

Mobile Application, Context, Situation

1. INTRODUCTION

Many applications use meteorological forecast data, for instance for planning purposes. We can cite for instance agricultural, construction, transportation, or tourism applications. In such applications, each weather service is currently an effort-intensive result of domain-dependent programming. Meteorological data needs to be collected, interpreted, and incorporated in a program and this is usually carried out in an ad-hoc manner. SITUMET aims at a more general approach to service provisioning in various domains. We focus on applications where planned activities play a central role. Two cases are of interest: (1) given a planned schedule, notify the user about possible disturbances, and (2) help the user to build a schedule based on anticipated weather information (which might be a consequence of case (1) as well in case there is a need for rescheduling).

Our work relies on the concept of *situation* which is a (multi-dimensional) context valid during a time interval. It can then include planned activities. While some applications are concerned with mobile users and user situations, others are concerned with the situation of an other type of entity at a given time (e.g., a mobile device, a freight container, or a road portion). This leads to the concept of *entity situation.*

A schedule is defined as a succession of planned situations. However, such situations are affected due to external, non anticipated events. This can be detected in advance (e.g., we know that it will probably rain at the Brandenburg Gate between 4 pm and 6 pm but also "on the spot"). Detecting situation changes means monitoring spatio-temporal entities for being affected by certain weather conditions.

This poster shows the overall architecture of the SITUMET platform with emphasis on its situation component. A running application example is given by the WIND system.

2. BACKGROUND

2.1 Weather data

Weather and forecast data that considered here are defined in a raster mode. The parameters of interest are 2 m temperature, cloud coverage, 10 m wind, precipitation, relative humidity, dew, soil moisture, and others such as special weather hazards (e.g., thunderstorm, hail). The temporal and spatial granularity of the raster can vary for different parameters, and available sensor and forecast technology: e.g., 10 m wind is available in 3 hour intervals and a geographical scale down to $4 \cdot 4$ km for the next 72 hours (based on a regional forecast model), or thunderstorm/hail in 5 minute intervals and a geographical scale of $1 \cdot 1$ km with a forecast horizon varying from 5 to 50 minutes (based on nowcasting technologies). From the cells of the raster data, points and regions of interest are extracted, corresponding to the situational demand, in terms of parameter, location, and time. Each extracted weather parameter, location, and time interval triple is associated with an event/occurrence probability. Yet, we do not incorporate these probabilities in our alerting and planning services. Instead we introduced a seed that takes only triples with an event probability over 80% into account.

We use taxonomies for meteorology. Attributes include the weather type (e.g., cloudy, fair), the wind speed (coming from instance from the Beaufort scale), the precipitation, the temperature, the humidity-level, and the barometric pressure level. The mapping from physical parameters into the taxonomies is done by mapping rules. In most cases, the mapping is defined by meteorologists (e.g., the Beaufort scale). In addition, we consider user mapping rules in a knowledge base that can override this default mapping taking a user's preferences as well as activities or other situational parameters into account. Specifying the mapping for an attribute *temperature-level* is a good example.

2.2 Situation modeling

We take as a reference the situation model of [3] whose main features are given below.

Definition 2.1. A **situation pattern** is a collection of attributes (or characteristics), each defined defined along a hierarchy of concepts (ontology). $SP = \{C_i\}$

In the case of a mobile users, characteristics are for instance location, type-of-connectivity, or activity. The location characteristics in turn can be defined at many levels of abstraction.

Definition 2.2. A **duration-based situation pattern (DSP)** is a collection of characteristics with which is associated a duration. $DSP = (\{C\}, d)$

Definition 2.3. A **situation** is a set of characteristics C_i that hold during a time interval. $S = (\{C_i\}, t_1, t_2)$ where t_1 resp. t_2 are the beginning resp. end of the interval. A situation is then similar to a context that is valid during a time period. This context has a dynamic part, such as location in a mobile application, but also a more static part to encompass user profiles (with a long interval). A situation is considered at a given level of abstraction for each characteristics. If it is defined at level l and if changes concern siblings in the hierarchy, the situation will not change. For instance, if a situation is defined at the city level and if the mobile user changes district in this city (which implies a situation change at level l_{-1}) then the upper situation will not change.

Definition 2.4. A **situation sequence** is a collection of ordered situations that do not overlap.

2.2.1 Situation operators

Operations on situation sequences are of three types: (i) relational operators, (ii) set-based operators, and (iii) temporal operations.

Relational operators include *select* and *project*. The select operator extracts situations from a situation sequence that satisfy a certain condition. The project operator is used to discard irrelevant knowledge from a situation sequence through removing dimensions. **Set-based operators** are used to combine or compare the situational knowledge contained in two situation sequences. Applying the *union* operator results in a situation sequence containing the combined knowledge of the input sequences. An *intersection* operation results in a sequence containing only shared knowledge. The *difference* operator is used to extract knowledge that is contained in either of the input sequences. **Temporal operators** are used to manipulate the time intervals. The *synchronization* operation, for instance, splits the original intervals of two situation sequences such that both sequences are synchronous afterwards. Synchrony means that two sequences are described using exactly the same time intervals. In most cases this results in situation sequences where neighboring situations have identical characteristics. In order to coalesce such situations a *normalization* operation us used.

2.2.2 Identifying situations

Situations are computed in push and pull mode. In a pull mode, this is a straightforward identification of the characteristics values. The push mode is more challenging as situations need to be continuously known by the system. What triggers a situation change is an event that affects a characteristic value — such as a significant change of location. The frequency with which the situations are computed depends on the update frequency of input information, e.g., an trajectory update, the update of the dynamic part of the location or personal profile. This can be done (i) in a cyclic fashion in case the update frequency and times are known or (ii) in an event-based fashion, which means that the processing unit gets informed about relevant updates via an event that triggers the new situation sequence as well as the computation of the delta for its further processing.

2.3 Plans and Activities

When supporting activity planning, two phases are of interest, namely the *planning phase* itself and the *execution phase* (once planned activities have been defined). In the planning phase we would like to support the user by checking a plan against weather data, thus identifying opportunities for carrying it out at a time when weather related requirements are met. In the execution phase, the requirements of a defined plan are checked against the current forecast. The user is alerted if there is a risk of deviation.

A **schedule** describes a plan where all steps or activities have been assigned a certain time interval that is anchored on the time axis. We model a schedule as a sequence of situations $Sc = \langle (SP_1, t_{\mathrm{b}1}, t_{\mathrm{e}1},), \ldots, (SP_n, t_{\mathrm{b}n}, t_{\mathrm{e}n}) \rangle$.

An **activity sequence** describes a plan where only the order of the steps is fixed. Such activity sequences are described as a sequence of situation patterns $As = \langle SP_1, \ldots, SP_n \rangle$. Even tough time is not associated with a single pattern, the anticipated duration of a situation can play a role here $As = \langle (SP_1, d_1), \ldots, (SP_n, d_n) \rangle$.

Setting up a schedule or an activity sequence can be done in different ways: either using planning support systems for use cases in the construction industry or agriculture or using a simple organizer (e.g., MS Outlook) for planning business trips or leisure activities. Another possibility is to specify a route/trajectory (using a routing service, e.g., when biking, walking or driving). In this case the situation sequence, that is, the location-related parts of the schedule, can be derived from this trajectory.

2.4 Situation-centric Service Provision

The idea behind our situation-centric approach is to use the concept of *situations* as a simple interface to provide

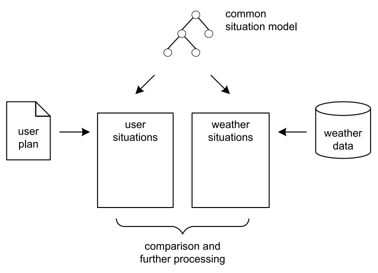

Figure 1: The principle of situation-centric service provisioning.

meteorological services to the user. Situations hence serve as a central model. User requests as well as weather information are mapped onto a common situation model. This includes, on the one hand, abstracting from observed or predicted weather parameters into a weather taxonomy and, on the other hand, mapping the spatial aspects into a location taxonomy. This provides for a simple easily understandable weather model.

In addition, querying such data in an ad hoc way (pull services) or registering a persistent query with a service to be alerted later (push services, continuous queries) is done by specifying a situation or a sequence of them. The services use the situations as their underlying "domain model". They interpret, combine, and compare the sequences in an application-dependent way to deliver their information.

In our example of *planned activity support* the users' schedules and activity sequences are formulated as situation sequences or situation pattern sequences. Matching between user plans and potentially conflicting weather conditions is done by using operations defined on situations and situation sequences. The following general actions are necessary to provide such services:

- Generate weather-related situations (weather situations) from meteorological data.
- Manipulate situations and situation patterns (select, project, combine, find overlaps, find discrepancies, and so on (see Section 2.2.1)).
- Manipulate situation sequences and pattern sequences.
- Monitor situation sequences for change.
- Search situation sequences or pattern sequences for occurrence of search patterns.

3. THE SITUMET PLATFORM

3.1 Architecture

The architecture of the SITUMET platform consists of three subsystems, namely, the *forecast backend*, the *meteorological product engine*, and the *application services*. A conceptual presentation of the architecture is given on Figure 2.

3.2 Forecast backend

The forecast backend subsystem is responsible for providing weather forecast information that can be queried by application services. It consists of (i) a set of data sources, (ii) a data acquisition layer, and (iii) the actual forecast system.

The forecast information is collected in the forecast database, which serves as a logically centralized data container. The database decouples the forecast computation from handling the service requests. Because most of the forecasting algorithms are time consuming, we decided against the possibility of a direct and synchronous initiation of a calculation by a client. Services can either query existing information from the database, or in case they have special requirements that are not covered, they can place a calculation request (calculation on demand). Such a request can be specified to be recurring or can be parameterized, e.g., with a timeout. An "on-the-fly" computation can then be realized by an asynchronous two-step process of first placing the demand and later querying the result from the database.

3.3 Meteorological product engine

The task of the meteorological product engine is to receive and interpret weather information requests from client services, to decompose it, to select the relevant weather information from the forecast database, and to aggregate that information according to a client's request. The product engine fulfills this task using the situation-centric approach described in this paper, thus providing the interface of the SITUMET platform for service developers. It provides processing of ad hoc queries (pull mode) or continuous queries (push mode). Querying weather information in an ad hoc manner is done by the service using user situations to specify the request. In the case of continuous queries, there are two types of events to be monitored: (i) the change of a user situation (triggered by the service) (ii) the change of a weather situation (initiated by the forecast monitor).

The situation generator is responsible for mapping weather forecast data into the weather- and location-related situation dimensions. Additionally, demand-oriented calculation of forecast information and derivation of weather situations is supported through interfaces that provide means to place a respective request with the forecast system and/or the situation generator.

3.4 Services

Using the interfaces of the product engine, the meteorological services implement the application-specific business logic. The services are responsible for presentation of the weather information received and for the further application-specific processing of the information. The services use the functionality of the situation framework to capture the characteristics of the user's context over time and for detecting situational changes. Situational information (that is, historic, current, and anticipated situations) is supplied as implicit parameters to requests, which allows for intelligent and pro-active weather information services.

4. APPLICATION

4.1 Application Description

WIND is an implementation of a weather alerting system developed at Fraunhofer ISST in collaboration with meteomedia a private European weather service and several insurance companies in Germany and Austria. The name is short for *Weather Information on Demand*. WIND is operational since 2004 with approximately 350.000 users (as of writing), a number that is continuously growing. The meteorological backend service of the system provides area-based informa-

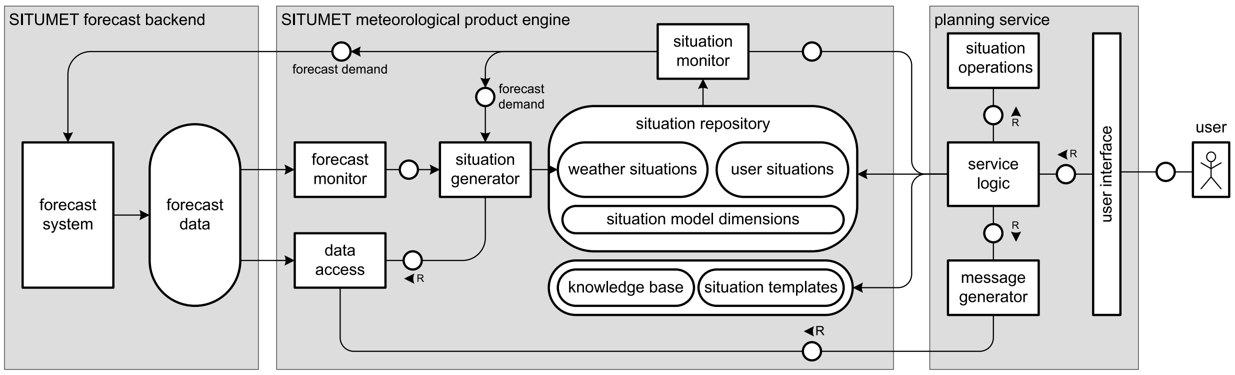

Figure 2: Conceptual presentation of the architecture of the SITUMET platform.

tion about severe weather warnings. There are two types of warnings supported.

- *Region-based warnings* use predefined natural regions (represented as sets of polygons) and are produced by the meteorologists of the *Unwetterzentrale* (*thunderstorm alert center*) of meteomedia or they are being generated from radar input. Nature regions are small areas defined under meteorological aspects along weather divides. Region-based warnings are stored as situations encompassing location- and weather-related dimensions (e.g., weather type, severity, wind speed). Additionally, urgency and probability/certainty levels are given.
- *Cell-based warnings* are being generated from radar-based systems that automatically detect location, severity, and movement of thunderstorm cells. We further distinguish static cells from dynamic cells. Static cells are represented by a circle describing location and size of the cell. Dynamic cells are modeled using one circle for the origin (the place where the cell was actually detected) and a second circle for the anticipated destination describing drifting direction and change in size. The trapezoid that is defined by the intersection points of the outer tangents of both circles describes the drifting course of the cell. The whole shape, that is, the union of the circles and the trapezoid is called a *club*.

WIND is a subscription-based system. In their simplest form, the subscriptions specify the location of interest and a profile (e.g., car driver, house owner). This profile is interpreted as a permanent situation (although it is possible to change the location sending an SMS containing, e.g., a zipcode). The profile definitions form the knowledge base and tell which kind of weather situation is potentially dangerous (the defaults are set together by experts from the weather service and insurance company). They can be changed by users to meet their individual needs. In its extended form, a subscription specifies a location along with ranges for the different meteorological parameters that the user wants to be alerted about.

WIND mobile adds dynamic to this alerting service. It is an application for the current generation of smart mobile phones that informs users about currently active weather alerts according to their current and anticipated location and their situation-dependent profile (e.g., outdoor, in car). The application supports two user interface modes: an overview mode and an alert mode. In overview mode the user can use map-based views or a list view in order to inform him- or herself about the current weather situation.

4.2 Technical Details

WIND mobile is written in Java and runs in an J2ME environment on the client (currently Nokia's GPS-enabled N95 and 6110) and in an J2SE environment on a Debian Linux server. We rely on OpenStreetMap data to support the map-based view. The map interface is a hybrid view using pre-calculated bitmap tiles for the map and Scalable Vector Graphics (SVG) for the presentation of weather data. We use GPS-based positioning to retrieve the current position of the user. In addition, we are experimenting and testing also with vague positioning using simulated cell-based (GSM/UMTS) positions. In order to comply with standards set in the early warning arena we use the Common Alerting Protocol (CAP) [1] and the EDXL Distribution Element [2] as the interface between the application and the request handling service of the server backend. Both standards are XML-based languages specified by the OASIS consortium.

The application uses both pull and push mode. This means that it uses ad hoc situational requests to update warning information on demand. Furthermore, a monitor is established, which is updated using current location and anticipated situational information in case the client application is passive in the background.

5. ACKNOWLEDGMENTS

This work was partly supported by the (*Zentrum für Information und Technologie (ZIT)*), Vienna, Austria under the Vienna Spot of Excellence Program.

6. REFERENCES

[1] Common alerting protocol (CAP), v1.1, oasis, url = http://www.oasis-open.org, 2005.

[2] OASIS, emergency data exchange language (EDXL) distribution element, v. 1.0, committee specification, url = http://www.oasis-open.org, 2006.

[3] U. Meissen, S. Pfennigschmidt, A. Voisard, and T. Wahnfried. Context- and situation-awareness in information logistics. In *Current Trends in Database Technology – EDBT Workshops*, volume 3268 of *LNCS*, pages 335–344, Berlin/Heidelberg/New York, 2004. Springer Verlag.