A Proposal for Context Data Markup Language

Mitsuru Minakuchi
National Institute of Information and Communications Technology
3-5, Hikaridai, Seika-cho, Souraku-gun, Kyoto 6190289 Japan
mmina@acm.org

ABSTRACT
We propose Context Data Markup Language (CDML), which is designed to enable context-aware systems to share context data. This paper outlines requirements for CDML, which includes generator, time, person, location, content, assurance and access permission.

Keywords
Context data, frameworks, markup language

INTRODUCTION
Many frameworks for context-aware systems have been proposed. Though they help the development of applications by enabling the reuse of modules, modules are tied to a framework with its own application programming interfaces (APIs). Another problem is that frameworks are occasionally too grand for prototypes and small projects. Consequently, some developers develop their modules individually and port them to a framework after they have finished, but that is troublesome.

Sharing context data among frameworks and modules is a simple idea for achieving framework-independent development. Context Toolkit [1] uses XML for communicating between components. CDML can be regarded as giving a standard format to the data being exchanged.

The requirements for CDML can be overviewed in the manner of W3C. In these requirements, the words "*must*," "*should*," and "*may*" are to be interpreted as follows:

must: The feature is an absolute requirement.
should: There may exist valid reasons to ignore the feature, but we should think carefully before omitting it.
may: The feature will be considered, but further examination is needed.

DESIGN PRINCIPLES
CDML *must* be a representation standard of context data.

Numerous studies have proposed various types of context data. Though there are some common data types such as time, person, and location, we currently cannot define all of the data types. Hence, CDML *should* provide solid specifications for the bases of context data and extensible formats for detailed contents of context data.

MAJOR REQUIREMENTS

Generator
Generators are identifiers of modules that generate context data. CDML *should* provide methods for describing generators for context data. Values of generators *should* be unique identifiers. CDML *may not* define any rule for naming modules. Developers *should* be aware of their uniqueness. Using URIs is a convenient way.

Time
Context data have two important time parameters: timestamps and lifetimes. CDML *must* provide both attributes.

Timestamps are times when context data are generated. They can be expressed with absolute dates or dates relative to other context data. CDML *must* support absolute dates and *may* support relative dates. Every piece of context data *must* have a timestamp.

Lifetimes are periods for which context data are valid. "How" context data are valid will depend on their contents. Lifetimes can be expressed with clock values or the word "indefinite". Every piece of context data *must* have a lifetime or the default value of "indefinite".

The consistency of timestamps is an important issue. If all modules referred to the same time server, there would be no problem. In practice, some systems and devices have their own clocks whose times are not consistent with the global (accurate) time. CDML *may* provide a method for describing timestamp evidences, an additional attribute indicating whether timestamps were obtained from global time servers or local clocks. Describing local consistency, which means assuring the synchronism or order of some context data, will also be useful. CDML *may* provide methods for describing timing and synchronization, like the "par" and "seq" elements in SMIL [2].

Person
The roles of people in context data can be categorized into two: subjects and objects. CDML *should* provide both types of descriptions of people. Context data can include multiple person elements or no person element. Values of people *should* be unique identifiers.

Location

Locations represent places associated with context data. CDML *must* provide methods for describing location.

Locations are not restricted to actual geographic places, but can be virtual locations, for example, somebody is logging onto a system.

Values of locations *should* be unique. Actual places *may* be described with geographic coordinates expressed in latitude and longitude or place names, which are aliases of geographic coordinates. Virtual locations *may* be described with URIs. Locations are optional for context data.

Content

Because the research area of context-aware computing is wide and still growing, we cannot define an ultimate tag set that categorizes all context data perfectly. However, without any classification, the benefit of standardization would be limited. CDML *should* provide general categories and may provide detailed categories.

General categories *should* be independent of any specific data type. We propose a combination of "about what" and "how it relates" types. For the former type, we can define "environment," "user," "system," and "interaction" attributes. Figure 1 shows the relationships among these attributes.

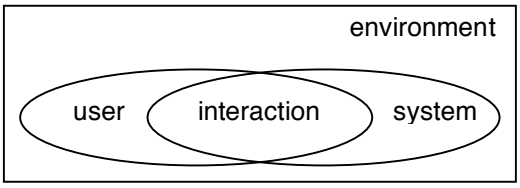

Figure 1: Relationships among "about what" attributes.

For the latter type, we can define "presence," "status," and "action" attributes. Examples of combinations of these types are illustrated in Table 1.

Table 1: Examples of general categories.

	Environment	User	System	Interaction
Presence	(unknown)	Foo is in the room.	There is a sensor.	(unknown)
Status	The temperature is 24 °C.	Foo's heartbeat is 76 bpm.	The sensor is working.	Foo is editing an e-mail.
Action	It starts raining.	Foo starts walking.	The audio player starts playing music.	Foo presses a button.

Detailed data can be represented with sets of a key and a value. CDML *must* provide extensible methods for describing detailed data, but CDML *may* provide some typical keys, for example temperature and heartbeat, as detailed categories.

Assurance

CDML *may* provide assurance to indicate how definite context data are. This parameter will be useful for handling vague context data.

Assurance values can be represented with numbers within [-1, 1]. Values of 1 show that context data are absolutely definite. 0 shows the assurance level is "undefined", which means that context data generators cannot tell how definite they are. Values between 0 and 1 show that context data are indefinite and indicate their degree of definiteness. Negative numbers represent the negation of context data. For example, a value of –1 for "Foo is in the room" means that Foo is absolutely not in the room.

Assurance values *should* represent probabilities, but they are not guaranteed. Generators of context data *should* produce assurance values that are as accurate as possible, or 0 (undefined) if that is impossible. Every piece of context data *must* have an assurance value or the default value 0.

Access Permission

From the viewpoint of privacy, access permission will be useful to prevent unlimited access. CDML *may* provide methods for describing access permission.

Defining permission levels is not difficult, but applying them may be problematic. For example, is a system allowed to access private context data for internal use only? Further investigation will be required.

FUTURE WORK

This work is only just at the starting point. It will be necessary to reflect numerous opinions. We are making draft specifications for CDML. Below is an example fragment of a CDML document meaning that "Foo" and "Bar" are at UbiComp2004 at noon on Sep. 7, 2004:

```
<context generator="IDdetector" time="2004-09-
07T12:00" lifetime="indefinite" category="user"
type="presence" accuracy="1">
    <location>UbiComp2004</location>
    <person>Foo</person>
    <person>Bar</person>
</context>
```

We are also implementing a context database, applications, and sensor modules to investigate the feasibility of CDML.

ACKNOWLEDGEMENTS

This work was funded in part by the exploratory software project 2003 of Information-technology Promotion Agency, Japan.

REFERENCES

1. Salber, D., Dey, A. and Abowd, G. The Context Toolkit: Aiding the Development of Context-Enabled Applications, In *Proceedings of Conference on Human Factors in Computing Systems (CHI'99),* pp. 434-441, 1999.
2. W3C Recommendation: Synchronized Multimedia Integration Language (SMIL 2.0). http://www.w3.org/TR/smil20/