Practical Applications of Context-Aware Computing: A Software Engineering Perspective

Peter Grasch

Under the supervision of Ass.Prof. DI Dr. Gerald Steinbauer

October 31, 2012

While the arrival of the often quoted "post-PC era" is still subject of heated debate, there is an undeniable shift in computing: Many users no longer have one dedicated computer but instead use multiple smart devices that range from personal computers over smart phones and tablets to smart TVs that all easily outperform workstations from merely a decade ago.

However, despite incredible advances in hardware design, software development still mainly focuses on dedicated, individual solutions that provide immediate benefits on the devices themselves. With the wide array of different sensors and other information providers as well as ubiquitous connectivity, much more autonomous and versatile solutions could potentially be developed by focusing on collaboration during the systems design.

In order to allow for such a paradigm shift, communication between not only individual applications and services but also across different devices, needs to be vastly improved. Achieving this while keeping costs in check demands new approaches on efficiently processing information that exceeds the strict confines of individual systems.

Standardized context-awareness middleware frameworks can provide such a communication channel to enable rapid development of such "smart", context-aware solutions as well as retrofitting existing applications with such capabilities, both without significantly increasing development effort.

Several context-awareness systems have been proposed over the years, but until now none ever managed to reach beyond research applications or establish a meaningful developer following.

In this thesis a number of different, promising context-awareness middleware solutions are discussed and compared on key attributes such as versatility, deductive and aggregative capabilities, autonomy, scalability, performance and security as well as their ease of integration in existing systems and their overall stability and development status.

Moreover, a smart video streaming system was designed and implemented with the most promising frameworks to compare their practical applicability.

Contents

1 Introduction

In recent years the ever increasing flood of data has given rise to a flurry of new devices that have since become almost a commodity. In fact, the third quarter of 2011 has seen a record market penetration of smart phones and tablets reaching about 40 % and 11 % of U.S. customers, respectively [1, 2].

As many current smart phones and tablets feature not only a source of constant high-speed Internet access but also high quality webcams, light-, motion- and GPS sensors as well as Bluetooth and NFC connectivity, this leads to an incredible wealth of readily accessible information surrounding us that would have been inconceivable just a decade ago.

Even though smart mobile devices are on the rise, laptops and desktop computers have not been replaced with as much as 76 % of U.S. consumers currently using them [2]. Because of this, a multitude of synchronization protocols such as SyncML [3] and more recently also cloud services like Googles extensive suite of web applications [4] or Apples iCloud [5], have been developed to enable heterogeneous systems to work together.

However, much of the potential of such a ubiquitous network of smart devices stays unexploited.

1.1 Motivation

Despite the popularity of personal computing devices, few systems exist that aim to share and use the vast amount of information that could be extracted from sensors and application logic of interconnected devices.

While this so-called context-aware computing approach has been well studied over more than two decades since Mark Weisers seminal work in 1991 [6], it has found little traction with commercial systems.

In the authors personal opinion this is at least in part due to missing an adequate middleware system.

Just as graphical user interfaces rely on toolkits to allow for a reasonable time to market, consistency and interoperability, a context-aware computing middleware should address issues like flexibility and scalability internally and expose a simple to use, coherent application programming interface to application developers.

This thesis aims to evaluate existing middleware systems that try to enable third party applications to easily expose and take advantage of information about the current situation. Furthermore it tries to determine practical applicability of a selection of solutions with a case study.

1.2 Definition

Before talking about context-aware middleware systems in more detail, some basic vocabulary should be defined for clarity.

Smart environment The term "Smart environment" was coined by Mark Weiser, now considered to be the founder of the concept of ubiquitous computing, and refers to environments that employ a wide array of sensors, actuators, displays and various forms of interconnected computers to provide a natural interaction with its users. A notable, widely used synonym is "Smart space" [7].

Context In general the term "Context" refers to the general situation of the person, device or application.

Over the years this general definition has been broadened and clarified.

In their seminal publication Schilit and Theimer [8] focused on location tracking and spatial relations to realize what has since become known as location-aware computing [9].

A notable definition is that of Dey and Abowd because it is both intuitive and comprehensive [10]:

> Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

However, the definition used in this report is that of Chen and Kotz [11]. They define the abstract term "Context" as a combination of the following components:

Computing context The computing context specifies details about the used hardware like available bandwidth and processing power.

User context This contains information about the persons that are using the device or are simply in its proximity.

Physical context This aspect holds information about the surroundings including properties like temperature, noise- and light levels.

Time context The time context exposes the current time to the context system to allow for time dependent behavior.

It is worth stressing that the term "Context" only refers to the abstract concepts discussed above.

World Model and Knowledge Base The term "World Model" has its root in the field of robotics and usually describes the internal representations of the robots surroundings [12, 13]. A world model can contain any information a system might gather about physical or logical entities (facts) including various sensor data, a set of basic "common sense" knowledge as well as the semantic meaning of symbols [7, 14].

On the other hand, the term "Knowledge base" may also describe any system that maintains an organized set of facts [15] but can also be interpreted even more broadly to extend the concept to representing an "arbitrary logical theory" [16].

Knowledge bases can therefore contain (represent) world models [17, 18] in which case the knowledge base itself may also be referred to as a world model [19–21].

The terms are also closely related to the term "ontology" which usually denotes the structure of a world model or knowledge base but depending on the situation may also be used interchangeably[1] [16, 19].

World models (knowledge bases) can provide ways to store, deduce and represent contextual information (context) [22].

Shared, common world models are often used to establish a common ground between otherwise autonomous systems [23].

Context-aware computing Based on the concept of context outlined above, the term "Context-aware computing" has been established as the practice of incorporating contextual information into services to allow them to react and adapt to their environment [9].

1.3 Related Work

The concept of context-aware computing has been well studied over the last decades.

Notable publication on the topic include what many consider the birth of ubiquitous computing: "The Computer for the 21st Century" by Mark Weiser [6] and two of the first widely publicized articles about the creation of truly context aware systems: "Context-Aware Computing Applications" by William N. Schilit et al. [8] and "The stick-e document: a framework for creating context-aware applications" by Peter J. Brown [24].

Despite the misleading title of the latter publication, all three of these papers talk either only about the theoretical foundation of context aware computing or the practical implementation of a very specific system.

However, just a year after the publication cited above, Schilit proposed "A System Architecture for Context-Aware Mobile Computing" [25], describing a general purpose context-aware computing framework.

Since then, many different context-aware computing frameworks have been presented - often focusing on individual aspects of the very diverse and complex topic.

[1] For a more detailed disambiguation please refer to the cited sources.

2 Concept

In order to evaluate and compare existing frameworks, a list of evaluation criteria was defined.

To accurately assess these qualities of the tested systems, a practical application, presented in Section 2.2 was devised and implemented with the most promising frameworks.

2.1 Evaluation Criteria

Existing systems were scored on the following selection of key characteristics.

Versatility Because of the vast amount of information that might be relevant in a context-aware computing framework, it can not be feasible to define a fixed solution that encompasses all possible use cases. For that reason, a viable framework must make it simple for application developers to provide new sources of context information.

Deduction and Aggregation In a general context-awareness framework information sources can and must not know how the information they provide will be used. Therefore, information dissemination could only use the lowest common denominator (i.e., low level, raw data) to provide information to the system so as not to obscure information that might be interesting for certain applications.

However, in order to fulfill the goal of ease of integration (see below), high level information will usually be requested by end-user applications. For example, a location-aware printer selection dialog as described in [26] should not need to be aware of the actual technical means of tracking the users location.

With built in deduction- (e.g., "User at coordinates x,y" $\rightarrow$ "User is in 'kitchen' ") and aggregation functionality (e.g., triangulation of the users position with multiple uncertain sources) the system can provide access to different degrees of context information without compromising either low level or high level information retrieval.

Autonomy With the rise of mobile devices and wireless networks it has become common practice to frequently move between different computational environments.

A context-aware computing framework must therefore be able to automatically cope with continuously changing computational resources and available services.

Scalability and performance In practice, context-aware systems are only as powerful as the information they provide. In order to become useful enough to warrant the additional complexity, systems need to provide enough incentive for developers. Especially when multiple (potentially low powered) devices need to communicate

over low bus speeds or context inference becomes more complex [27] scalability and overall performance become important aspects to consider.

Ease of integration As mentioned above, market penetration must be one of the core goals of a successful context-aware computing framework. To accomplish that, it needs to be easy and straight forward for developers to integrate context information in their applications.

Security To build a comprehensive context model, situational information must be shared between different devices [14].

However, context data might contain confidential information (e.g., user position). An adequate security concept must therefore ensure that only authorized clients can read protected information and provide a way to prevent malicious clients from publishing spoofed information.

Stability and Development Status Like any other middleware framework context-aware computing solutions need to be stable and actively maintained to be taken into consideration by application developers.

2.2 Experiment

To test the available systems, a context-aware video streaming system was designed that touches on all four context dimensions as defined by Chen and Kotz (see Section 1.2).

2.2.1 Goal

Multiple streaming clients are spread around a smart home environment. A central server provides a video stream that may be displayed by these clients. This basic setup is depicted in Figure 2.1.

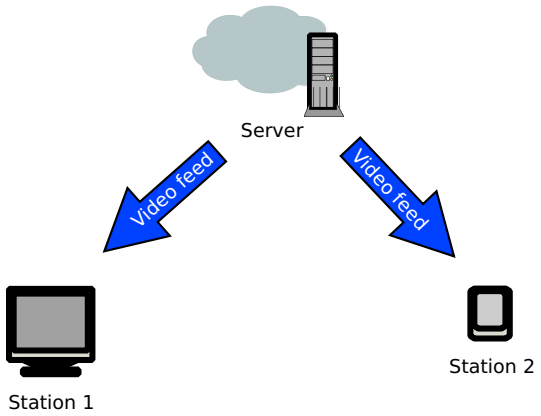

Figure 2.1: System components

The users location is used to activate different screens as the viewer changes location (user context).

The active station monitors background noise with a microphone and adjusts the playback volume automatically (physical context).

The stream quality adjusts itself to the computational capabilities of the active client to allow for high definition playback on clients that support it, without compromising playback performance on more limited platforms (computing context).

The stream itself is started at scheduled times (time context).

2.2.2 Environment

A fictitious smart home with three rooms, shown in Figure 2.2, will serve as the systems environment.

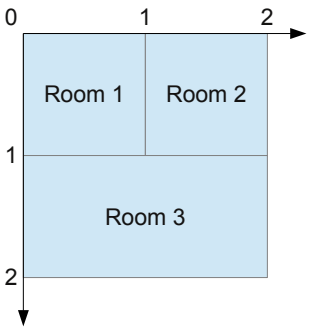

Figure 2.2: Simulated smart home

It is assumed that the environment is equipped with an appropriate sensory network to allow to track the users location. For the purpose of the experiment this is simulated by projecting the whole map in a single, physical room and using a Microsoft Kinect with appropriate software (see Section 2.2.3) to map the users relative coordinates within a simple, two dimensional coordinate space. Figure 2.3 depicts this setup.

Only coordinates are provided to the context-awareness middleware systems to better emulate the output of established indoor positioning solutions.

Two video streaming clients, “Stations”, are virtually placed within the environment. Physically, both are running on the same machine but report different locations and capabilities to the context network as shown in Table 2.1.

Station Number	Location	Computational Class (see Section 2.2.3)
1	Room 1	3: High quality
2	Room 3	1: Low quality

Table 2.1: Demonstration stations

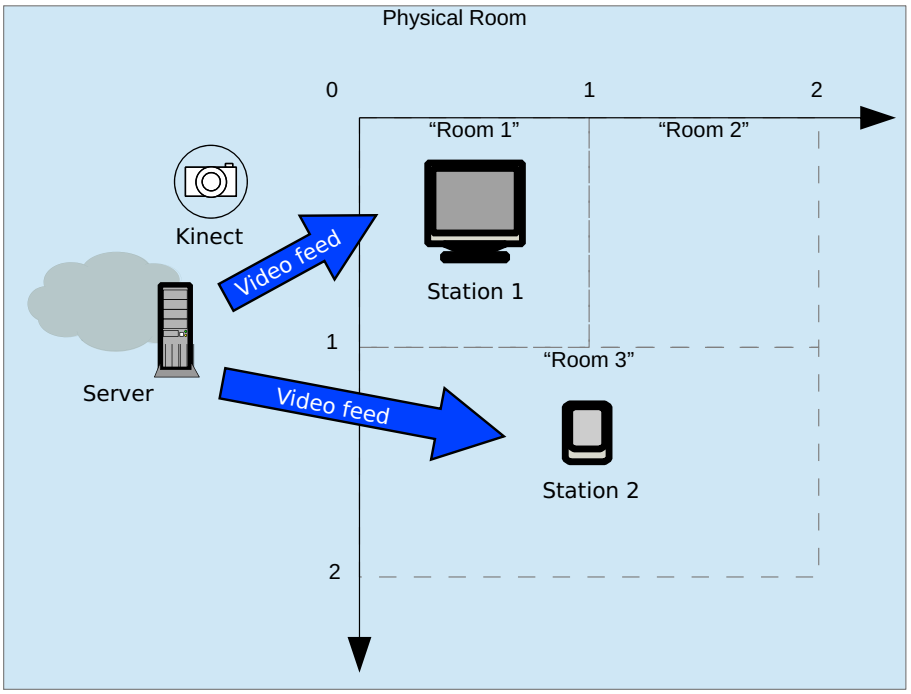

Figure 2.3: Physical experimentation environment

2.2.3 Implementation Details

Independent of the used context-awareness framework, components needed to be developed to act as information sources (sensor abstractions) and actuators.

This section briefly describes these general components of the experiments.

Determining Ambient Noise Volume

In order to automatically adapt the playback volume to compensate for background noise, the ambient noise levels need to be determined first.

To get a rough estimate sufficient for the experiment, the life microphone input is first segmented into one second chunks; for each chunk, the maximum amplitude is determined and scaled to a whole number in the range $[0, 255]$. The resulting value is considered the current level of ambient noise.

$$currentAmbientNoiseLevel := round\left(\frac{max(audioBuffer)}{MAX\_AMPLITUDE} * 255\right)$$

Positioning

To implement location awareness, the users position needs to be known to the system. For that, a Microsoft Kinect was used to track the left hip of any recognized user in front

of it.

The tracking implementation itself is heavily based on the user tracking example of the OpenNI framework [28] and was simplified and slightly adapted to fit the experiments requirements. Both the OpenNI and the PrimeSense NITE framework are used.

The Kinect is anchored at a known position in the projected experimentation environment and the tracking software reports properly scaled coordinates of the user in the virtual environment (both x and y coordinates are in the range $[0, 2]$; see Figure 2.3). It could therefore be argued that the position of the Kinect defines the origin of ordinates of the virtual room even tough the device itself will always be slightly offset the projected environment for obvious technical reasons.

Streaming and Playback

VideoLANs VLC system [29], with the vlcj Java bindings [30] was used both as streaming and playback solution for its maturity and plentiful features.

While VLC does support on-the-fly transcoding of streaming media, this was not used for performance considerations. Instead the video to stream was transcoded offline into three different versions to mirror the three available computational classes laid out in Table 2.2.

Computational Class	Video bitrate (resolution)	Audio bitrate (channels)
1: Low quality	300 kbit/s (320x180)	64 kbit/s (mono)
2: Medium quality	1250 kbit/s (854x480)	112 kbit/s (stereo)
3: High quality	2300 kbit/s (1280x720)	152 kbit/s (stereo)

Table 2.2: Computational classes: Video Quality

The test videos use an H.264 (MPEG-4 Part 10 / AVC, [31]) encoded video stream at 23.97 frames per second and an AAC (Advanced Audio Coding, [32]) encoded audio stream. They are wrapped in an Flash Video container (FLV, [33]) and streamed over HTTP.

Depending on the viewing devices hardware capabilities, the matching file will be streamed. If another computational class was selected before and streaming already started, the new file will resume at the previous position so as to achieve seamless transitions.

Of course, there are much more sophisticated ways of achieving adaptive HTTP streaming such as DASH [34], that would be more suitable for practical deployment, but their complexity would make them significantly harder to integrate with the contextual reasoning with no benefit for the purpose of the experiment.

Streaming Timetable

As noted in Section 2.2.1, the video stream is to be started automatically at certain times.

Practical deployments would most likely want to retrieve the actual timetable from external sources like a database or configuration file but as this has no bearing on the evaluation of the used context framework, the schedule was instead simply hardcoded.

In order to facilitate testing, the stream is started, if it is not already running, once every full minute.

3 Evaluation

After thorough literature research, several promising sounding solutions were chosen to be evaluated.

3.1 GAIA

The first system that was chosen for evaluation was the context layer of GAIA - a wide reaching effort to create an application framework for ubiquitous computing spearheaded by the department of computer science at the University of Illinois.

"GaiaOS" was designed to be a meta-operating system for ubiquitous computing projects. It provides a central abstraction layer to all available devices in an environment and affords means for application developers to both publish data to that network and to display and use provided information through a novel take on the traditional model-view-controller pattern. Mobile code is used to handle varying component life cycles and capabilities which are tracked and managed by a unified bus [35].

Through kernel services GaiaOS also provides services specifically built to support context-aware applications: Message passing capabilities (Event Manager), directory services (Space Repository) and a special file system that supports content adaption and location awareness (Data Object Service) [35].

However, later publications introduce a new, dedicated subsystem: This proposed middleware layer uses CORBA as communication layer and employs an DAML+OIL ontology that describes the properties and structure of context predicates that make up the context model. A central "context provider lookup service" provides discovery [36].

A major focus of the unnamed subsystem is placed on an efficient and powerful reasoning system. In addition to first-order logic, propositional linear-time temporal logic and probabilistic propositional logic providing sophisticated rules of inference, machine learning approaches have been integrated to allow agents to better their understanding of context over time by employing Bayesian methods or reinforcement learning [36].

Because querying the context model is very similar to Prolog, results can range from simple boolean results to explanation on how a specific situation was or could be reached simply leaving different literals unassigned (implicitly providing intelligibility). To react on changes in the context model, agents can also request to be notified of changes [36].

Additionally, agents can define actions that are to be executed whenever a specific context is sensed by the system. This approach is called event-condition-action (ECA) execution model [36].

The context infrastructure consists of "Context Providers" that publish information to the context network, "Context Synthesizers" that infer new context information to be

published to the network, "Context Consumers", also called context-aware applications, one single "Context Provider Lookup Service" to utilize discovery of the context components, one "Context History Service" to store past context information, and one "Ontology Service" to maintain the different ontologies within the network [36].

A reference implementation of the system is described in [36] but has as of yet not been published on GAIAs homepage [37].

There seem to have been no developments after 2005.

An email inquiry to Roy H. Campbell, one of the lead developers, remained unanswered.

Because of that, no experiments could be performed and, despite its promising documentation, the system remains unevaluated.

3.2 SOCAM

SOCAM, short for Service-Oriented Context-Aware Middleware, was originally designed by researchers of the National University of Singapore and Infocomm Research to ease the development and rapid-prototyping of context-aware applications [38].

The system implements the OSGi industry standard to, among other benefits, achieve platform independence and gain access to various security features including digitally signed services and object access control [39].

Similar to the context architecture in GAIA (see Section 3.1), SOCAM also uses OWL context ontologies and represents context, according to the surrounding publications, in first-order predicate calculus[1] [38].

Again, a large focus of the system is a sophisticated reasoning layer providing ontology reasoning (RDF Schema and OWL Lite) and user-defined rule-based reasoning [39].

Architecturally, SOCAM is also very similar to GAIAs context layer. Context networks consist of "Context providers" that retrieve and publish contextual information from various sources, "Context interpreters" that reasons on information from context providers to deduce new information (context interpreters are also context providers), "Context databases" that manage context ontologies and historical context information for one (sub-) domain, "Context-aware services" that adapt their behavior on context information and a "Service locating service" to support discovery of context providers and -interpreters within the network. The communication between those components is implemented using Java RMI [38].

SOCAM also incorporates the notion of "uncertain contexts": Sensed or deduced context information that is likely, but not definitive. This is realized by introducing a probability measure to predicates that can be reasoned about with Bayesian networks [43].

[1] The authors state that the used ontology is made up of "a collection of RDF triples" and the provided example uses RDF to represent domain knowledge directly [38] but RDF is not expressive enough to model first-order logic completely without resorting to special encodings [40].
However, the used semantic web framework, Apache Jenna, does support the use of OWL Full, which even exceeds the constraints of first-order logic [41, 42].
Because the original SOCAM framework was no longer available, the actual expressiveness of the used logic remains unclear.

The lead author of the conceptual papers has also published the source code of the SOCAM system on his University website [44].

The distribution includes demonstration applications of which none properly initialized for reasons ranging from invalid (partly absolute) paths, hardcoded (private) IP addresses and discontinued syntax (the Java code shipped with the distribution is not compatible with Java versions starting at 5.0[2]).

Supporting documents are limited to rudimentary Javadoc documentation.

Upon closer inspection of the source code it became apparent that the released system does not conform to the system description in [39] and [38]. For example, the proposed Java RMI communication between the systems components has instead been implemented using the Gnutella P2P file sharing network protocol. The published version of SOCAM also relies on an ASP cache to share host information within the context network. Because of that, using the system requires access to an ASP capable server, greatly reducing deployability[3].

These discrepancies to the conceptual works lead to believe that the released version does not reflect the proposed SOCAM architecture, but rather a diverse testbed for the authors varied research portfolio - which includes an ontology-based p2p network for semantic search using a Gnutella-like network [46].

The researcher who published this SOCAM version was contacted and confirmed that the released version does incorporate more recent developments and that the original version is sadly no longer available. He also stated that no further development is planned.

Because of this, no evaluation was performed.

3.3 Context Toolkit

The aptly named Context Toolkit was developed by researchers at the Georgia Institute of Technology and first presented in 1999.

It tries to build on the success of widget toolkits for graphical user interfaces by providing opaque context components called "Context Widgets" [10].

Context widgets are a way to present (potentially high level) attributes and notifications to interested applications (through "Services"). Widgets themselves are merely data management classes. They receive information (attribute values) from "Generators" that encapsulate sensors or other software components and "Enactors" that react on data from other widgets. Generators can be seen as Enactors with no input widget. "Interpreters" can be used to abstract the raw information even before they reach widgets [10].

A central, multi-cast discoverer is used to coordinate the components within the context network [47].

[2] In Java 5.0, "enum" became a reserved identifier [45]

[3] Fortunately, a public instance of the host cache can, at the time of this writing, be found at `http://www.tan-family.com/edmond/socam/lynn_socam.asp`

3.3.1 Experiment

To thoroughly evaluate the context toolkit, the experiment laid out in 2.2 was undertaken using the context toolkit.

This section describes how the Context Toolkit was used to share the information of distributed sources and to encapsulate high level reasoning rules. Please refer to Section 2.2.3 for information about how the information sources themselves were implemented.

To model all the necessary context data and reasoning, a number of new components had to be developed. An overview of the context toolkit components and the data flow between them can be seen in Figure 3.1.

The Context Toolkit uses a special query language to identify components that uses attributes and properties instead of arbitrary names. This allows components to formulate more general links through the context network. Listing 3.2 shows the linking of the `LocationEnactor` to any `Streamer` instance and a specific `Viewpoint` instance.

Location Awareness The `LocationGenerator` provides the users coordinates which are then published through the `LocationWidget`.

The `LocationEnactor` uses a hardcoded map to translate the coordinates to room names.

Additionally, the `FixedLocationGenerator` provides a fixed location to represent the stations (fixed) position. This was done through a dedicated generator instead of a constant value to easily allow for mobile stations in the future by simply linking an equivalent chain of components as used for the user tracking.

The `ViewpointWidget` then aggregates, among other information, both the users location and the viewpoints location.

Finally, the `ViewpointEnactor` sets the `PlayerWidgets shouldPlay` attribute depending on if the users location matches the viewpoints location and calls the function `play(boolean)` in the ViewpointService, passing `shouldPlay` as a parameter.

Ambient Noise Compensation The `AmbientNoiseGenerator` measures loudness and publishes it through the `ambientNoise` attribute in the `AmbientNoiseWidget`. The `AmbientNoiseEnactor` then calculates the playback volume in relation to the ambient volume (both are in the range of 0 to 255, inclusive):

$$volume := min(255, 100 + ambientNoise)$$

The resulting volume is both relayed to the `PlayerWidget` and passed to the `ViewpointService` by invoking `adjustVolume` and providing the volume as parameter.

Adapting to Computation Classes The `PlayerWidget` announces its `computationalClass` which is picked up by the `StreamerQualityEnactor` that adjusts the `quality` attribute of the `StreamerWidget` and alerts the `StreamerService` of the change.

Automatically starting the Stream The `TimeGenerator` sends periodic updates every second publishing the current hour and minute to the `TimeWidget`.

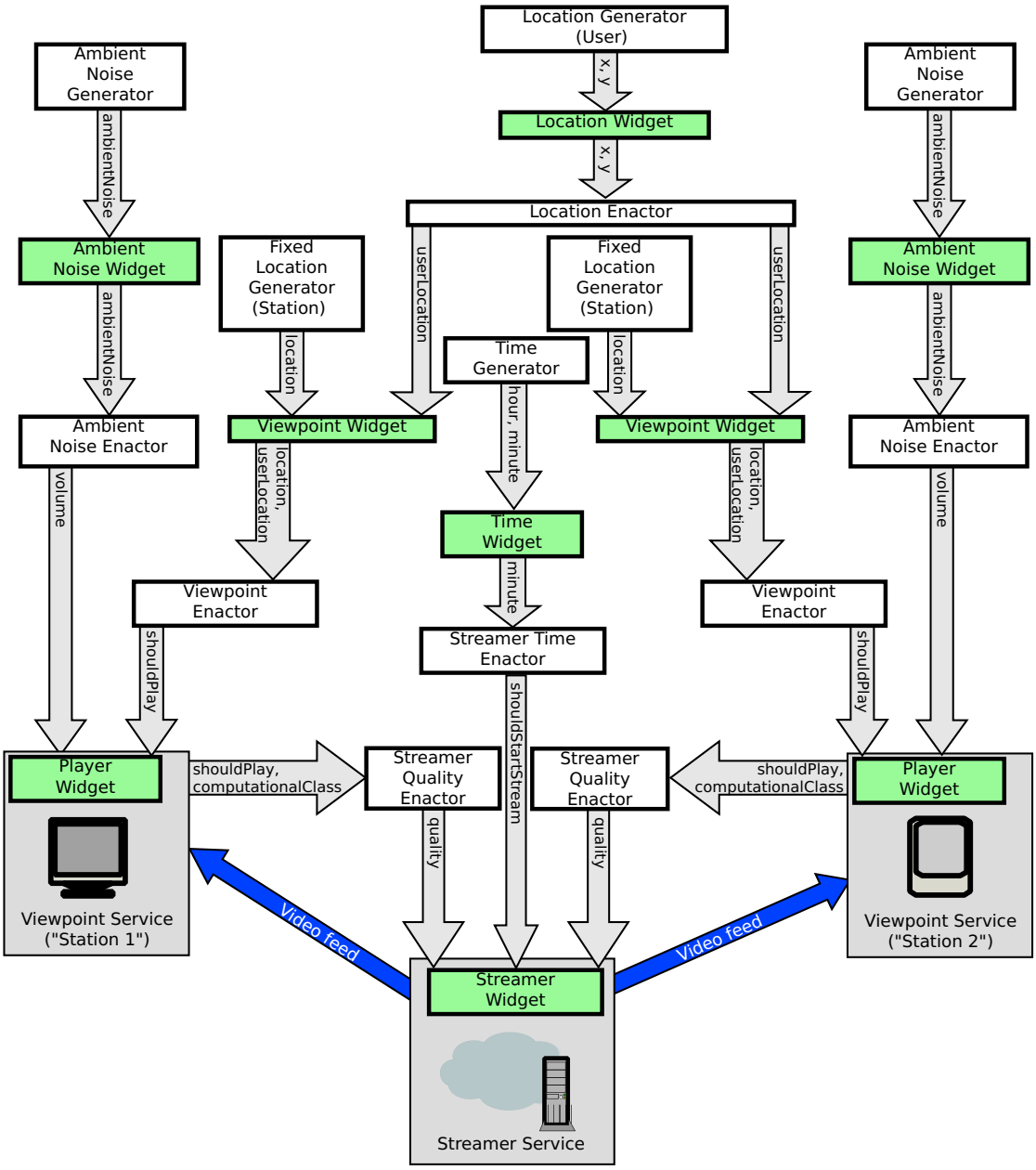

Figure 3.1: Context Toolkit: Conceptual Architecture

`StreamerTimeEnactor` then calls `startStream` of the `StreamerService` when the scheduled start time is reached.

```
AbstractQueryItem<?, ?> streamerQuery =
    RuleQueryItem.instance(new ClassnameElement("StreamerWidget"));
AbstractQueryItem<?,?> viewpointQuery =
    new ANDQueryItem(
      RuleQueryItem.instance(new ConstantAttributeElement(
                             new AttributeNameValue("viewpointId", "Station1"))),
      RuleQueryItem.instance(new ClassnameElement("ViewpointWidget")));
new LocationEnactor(streamerQuery /* subscribe to */,
              viewpointQuery /* output to */, "UserLocationEnactor" /* shortId */,
              "userLocation" /* output name */);
```

Figure 3.2: Context Toolkit: Linking components

3.3.2 Evaluation

Armed with the insights of conducting the experiment as described above, the context toolkit was evaluated using the criteria set out in Section 2.1.

Versatility While only a couple of demo components exist in the current distribution (2.0), widgets, generators and enactors can easily be written and deployed by third party developers.

The framework is therefore not bound to any specific use case.

Deduction and Aggregation For simple aggregation and deduction, "Enactors" provide a natural way to react on changes in widget data.

Complex aggregation setups are supported through the use of "Servers" [10] or "Aggregators" [48].

With the introduction of the intelligibility subsystem, the Context Toolkit can moreover deduce and explain high level information (including "Why?", "Why not?" and a measure of certainty) through the use of rules, decision trees, naïve Bayes and Hidden Markov models [27].

Autonomy As mentioned above, a central discovery server is used to automatically and transparently coordinate running widgets across a distributed context network [47].

Information is propagated through a publish / subscribe mechanism. A special query language is used to subscribe to widgets. Developers do not need to know the architecture of the context network but can instead simply ask for a widget with certain name, type or id (white and yellow page lookups are supported) [47, 49].

Scalability and performance While publications of the authors sometimes reference "Discoverers" (plural) within a context network [47], there is no mention of synchronization / mediation between discoverers [47, 48, 50–52].

A practical experiment showed having multiple discoverers within one network to yield undefined results (critical exceptions, most likely due to race conditions between the discoverers).

Additionally, the latest released version (as of October 2012: 2.0) also does not propagate incremental updates for widget attributes but instead always publishes the complete widget description whenever an attribute changes. If an update could *potentially* change another widgets state (due to, for example, an enactor listening on the changed widget), that widgets full state is published as well - regardless if any attribute actually changed its value or not.

Overall, this leads to a very verbose network that, through the use of a central discoverer, has at least one bottleneck.

No benchmarks were published by the authors and none were conducted for the purpose of this evaluation. However, for the reasons outlined above it could be argued that the framework is most likely best suited for relatively small deployments.

Ease of integration The context toolkit is written in Java and has dependencies to rather large third party libraries including Hibernate, MySQL, WEKA and JAHMM [53]. All dependencies, as well as the context toolkit itself, are redistributable under OSI approved open source licenses.

There do not appear to be bindings for other languages besides Java but the reliance on standardized components suggests that this could be remedied comparatively easily if necessary.

Integration in existing Java applications is simple.

Context-dependent parts of the software are advertised by creating a new widget containing the required attributes and announcing that to the discoverer. Service subclasses can be used to directly trigger Java method calls upon changes in associated widgets, potentially passing attributes of the widget as parameters. To link up the new widget with the existing context net, enactors are used that can capsule arbitrary logic.

Similarly, generators can be used to publish new data to the context network through widgets.

Widgets and enactors can, in addition to subclassing the appropriate Java classes, also be instantiated from descriptive XML files. To express logic for enactors, a restricted subset of JavaScript may be used. For example, the snippet 3.3 represents the `AmbientNoiseEnactor` (see Section 3.3.1).

While certainly convenient, this feature is, in its current stage, very limiting. For example, all comparison operators (`EQUALS`, `LESS_EQUAL`, etc.) expect one variable to be compared with a constant. Comparing two variables with each other fails silently as the second variable name will simply be treated as a string literal.

Security Both security and privacy concerns are addressed in the publications around the context toolkit: A standard public / private key architecture is suggested, where "owners" of context information can restrict access to the representative widgets [47].

However, this concept has not yet been implemented in the released solution as of version 2.0.

```
<?xml version="1.0" encoding="UTF-8"?>
<Enactor xmlns="http://www.contexttoolkit.org/ctk" name="AmbientNoiseEnactor">
   <InWidget  href="ambient-noise-widget.xml" />
   <OutWidget href="player-widget.xml" />

   <OutcomeName>volume</OutcomeName>

   <!--  Volume := min(255, 100 + ambientNoise) -->
   <Reference name="volumeRef">
     <Query name="volumeQuery">
       (LESS_EQUAL ambientNoise 155)
     </Query>
     <Outcome outAttribute="volume">ambientNoise+100</Outcome>
     <ServiceInput service="ViewpointService" function="adjustVolume">
        <Attribute name="volume" />
     </ServiceInput>
   </Reference>
   <Reference name="volumeRefMax">
     <Query>
       (GREATER ambientNoise 155)
     </Query>
     <Outcome outAttribute="volume">255</Outcome>
     <ServiceInput service="ViewpointService" function="adjustVolume">
        <Attribute name="volume" />
     </ServiceInput>
   </Reference>
</Enactor>
```

Figure 3.3: Context Toolkit: XML enactor definition

Stability and Development Status During the experimentation some critical bugs surfaced. A null pointer exception in the discoverer, for example, would make cross-application subscriptions to widgets almost always fail. A simple patch working around the problem was developed for the purpose of the experiment and the problem has been reported to the developers[4].

Moreover, certain types of queries did not yield expected (documented) results - especially when used in a distributed environment.

The overly verbose updating mechanism discussed above also posed more than performance issues: Would for example widget `A` copy attribute `x` from widget `B` and widget `B` copy attribute `y` from `A`, the system produced an infinite loop of update messages. This seemingly artificial problem presented itself more than once while implementing the demonstration system. One such issue is illustrated in Figure 3.4. Please note that `computationalClass` in the client is explicitly marked as a *constant* attribute of the `PlayerWidget` and yet this still produces an infinite update loop.

The context toolkit source code has been released under the GPLv3 and is available on a public repository.

The last checked in change was logged in December, 2010 but recent publications about the topic from the leading authors suggest continued research interest [54, 55].

[4] "IdElement Queries fail on remote hosts":
`http://code.google.com/p/contexttoolkit/issues/detail?id=2`

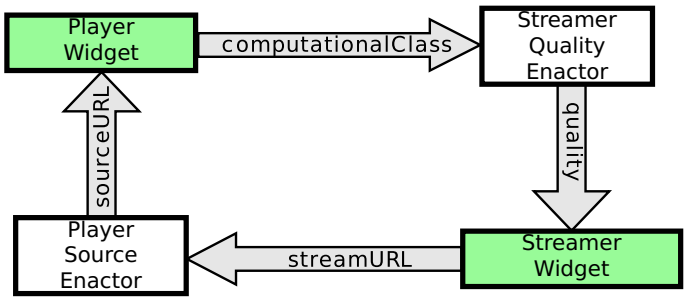

Figure 3.4: Context Toolkit: Infinite loops due to unnecessarily verbose updates

3.4 JCAF

The Java Context Awareness Framework, or JCAF, has been developed by the Center for Pervasive Computing of the University of Aarhus, Denmark and was officially first published through a paper of its lead author Jakob Bardram in 2005 [56]. It had, however, been used for other research projects by the same department before and earlier, no longer available publications suggest that JCAF was under active development as early as 2003 [57, 58].

Despite a focus on health care situations in early applications, the Java Context Awareness Framework, much like the Context Toolkit discussed above, was expressly designed to be a general framework, agnostic of any specific use case [56–58].

Regardless, most of the publicized practical uses of JCAF remain in the medical domain [59, 60].

The framework uses Java RMI as communication channel [56].

JCAF uses "Context Services" to capsule different application domains. Context services can exchange information with other context services through peer to peer connections. A hierarchical structure is mentioned in [56] but does not seem to be implemented in JCAF 1.5.

To represent contextual information within the services, "Context Entities" use "Relationships" to connect to "Context Items" [56]. This way of representing context information can be compared to RDF triples with an entity representing the subject, the relationship the predicate and the item acting as the object.

Context entities are automatically context items as well, making it possible to chain relationships. Items can have no further relationships, but instead act as "leaves" to the JCAF equivalent of RDF graphs [56].

Additionally to these explicit relationships, the entities and items themselves can have (member) attributes as the instances are simply serialized and passed through Java RMI on request, preserving those values. However, no listeners (entity listeners, context actuators; see below) can be used to monitor changes on these private attributes and no mechanism exists to send explicit notifications short of feigning a reflective relationship [61].

In addition to querying the context service, JCAF provides two different types of event subscriptions [56, 61]: "Entity Listeners" can subscribe to entity classes or specific entity

instances to be notified whenever a new relationship with the monitored object at its *root* (RDF: "subject") is registered, modified or removed. "Context Actuators" provide a similar mechanism but are limited to subscriptions to types of context items and provide no way to monitor specific instances. They are triggered whenever a relationship with the watched item as its *target* (RDF: "object") changes.

To source complex contextual information, JCAF also provides "Context Monitors" that listen on requests related to a selected type of entities to interject new context information on-demand (synchronous) or continuously (asynchronous) [56].

"Context Transformers" can be used to aggregate or translate information from one or multiple context items to a single output item [56].

3.4.1 Experiment

Several software components, shown schematically in Figure 3.5 and discussed below, were implemented to realize the experiment described in Section 2.2.

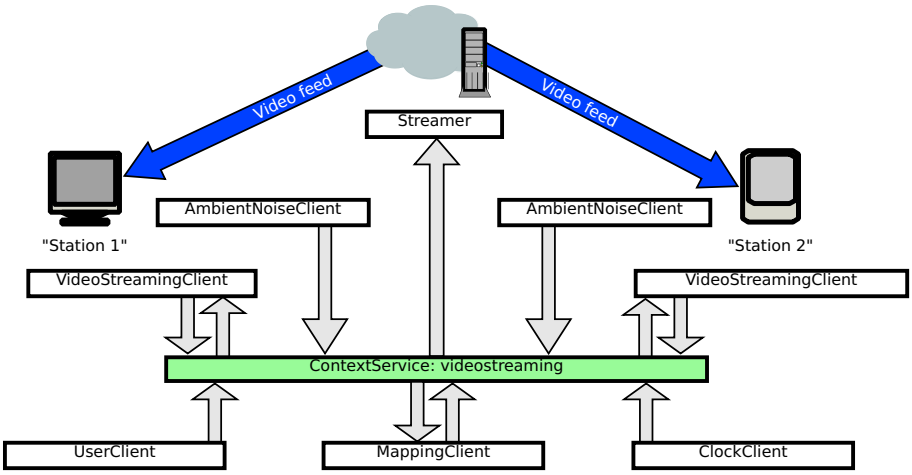

Figure 3.5: JCAF: Conceptual Architecture

Items and Relationships To model the application domain, specific entity, item and relationship classes were designed. An overview can be found in Figure 3.6.

As laid out above, these elements closely resemble the way knowledge is stored in ontology based systems like KnowRob (see Section 3.5) but are in JCAF instead used as actual Java object instances that are transmitted over Java RMI [61].

Location Awareness The `UserClient` context client asserts a detected user by publishing a new `User` entity. Once the users location has been identified, coordinates are published through the `Coordinates` entities `x` and `y` attributes. This entity instance is connected to the `User` object with the `IsAt` relationship.

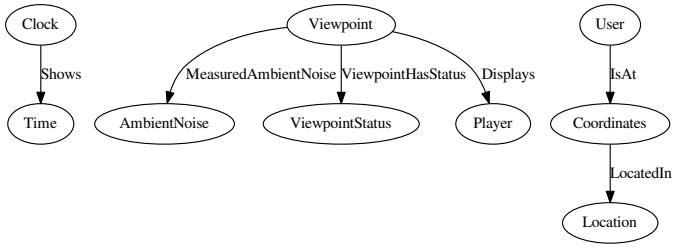

Figure 3.6: JCAF: Entities and Relationships

To model changing positions (tracking), the previous `IsAt` relation (and associated `Coordinates` object) is retracted and a new relationship is set up. This seemingly inefficient solution is required to notify listeners of changes in the users position because, as mentioned above, mere modification of entity attributes can not be monitored [61].

In order to resolve coordinates to location names, another context client, `MappingClient`, monitors the context service for new `Coordinates` entities through a context actuator and asserts `locatedIn` relationships to one of three `Location` items according to the map laid out in Section 2.2.2.

The `VideoStreamingClient` monitors these `Location` items, again through an actuator, and activates playback once the user enters the same "Room" the viewpoint is serving. Additionally, the current state of the video streaming client (active or paused) is published through the `ViewpointStatus` item and the `ViewpointHasStatus` relationship.

Ambient Noise Compensation Similarly to the location tracking, ambient noise information is modeled by continuously asserting and retracting context information.

The current level of ambient noise, represented through the `AmbientNoise` context item, is connected to the respective `Viewpoint` entity with the `MeasuredAmbientNoise` relationship.

Publishing an `AmbientNoise` context item activates another actuator in the `VideoStreamingClient` which adjusts its players volume levels accordingly.

Adapting to Computation Classes The `Streamer` context client installs another context actuator to monitor changes in `ViewpointStatus` connections.

When notified of changes, the actuator checks if the changed viewpoint has now become active. If that is the case, the viewpoints associated players (`Player` entities that were connected through the `displays` relationship) are fetched from the context service.

The lowest supported quality profile, stored as a direct attribute (member variable) of `Player`, is selected as the streaming quality.

The implementation of this concept can be found in listing 3.7.

Automatically starting the Stream To start the video stream at specific times, an entity listener monitoring `Clock` entities is registered in the `Streamer`.

```
public Streamer(/*...*/) {
  //...
  RemoteActuatorImpl actuator = new RemoteActuatorImpl();
  actuator.addContextActuator(new StreamerQualityAdapter());
  getContextService().addContextClient(RemoteContextActuator.TYPE_ACTUATOR,
                                       ViewpointStatus.class, actuator);
}

private class StreamerQualityAdapter  implements ContextActuator {
  @Override
  public void contextItemChanged(ContextEvent event) {
    if (event.getRelationship() instanceof ViewpointHasStatus) {
      //viewpoint status has been associated to viewpoint
      Viewpoint viewpoint = (Viewpoint) event.getEntity();
      ViewpointStatus status = (ViewpointStatus) event.getItem();
      if (viewpoint == null || status == null) return;

      //if viewpoint is now active
      if (status.getStatus()) {
        //find viewpoints players
        ContextItem[] players = viewpoint.getContext().
                                    getContextItems(Displays.class);
        // start off with the highest quality profile
        int qualityProfile = 3;
        for (ContextItem player : players) {
          if (!(player instanceof Player))
            continue;

          int compClass = ((Player) player).getComputationalClass();

          //choose lowest preferred profile
          //(ensuring that it is supported by all players)
          qualityProfile = Math.min(qualityProfile, compClass);
        }

        //set profile
        streamingServer.setQualityProfile(qualityProfile);
      }
    }
  }
}
```

Figure 3.7: JCAF: Selecting Streaming Quality Profile

When activated, the listener compares the time the watch entity was connected to through the `Shows` relationship with the internal timetable and launches the streaming process if necessary.

3.4.2 Evaluation

Versatility One of JCAFs stated core design principles is to use "semantic-free modeling abstractions" to represent context information: The framework itself does not autonomously interpret context information but rather provides those capabilities to domain aware applications [56].

This design goal has been consequently implemented in the released version and ensures

a versatile context-awareness platform.

Deduction and Aggregation JCAF supports incorporating context quality into reasoning processes by adding an accuracy measure through the overwritable function `getAccuracy()` of context items [56].

The need to interpret context information through application specified rules has been addressed through context transformers. Transformers can take one or more context items as input, process them and output a single context item to encode the result. Transformers with only a single input item are also called translators [56].

However, in the current JCAF version 1.5 the context transformer subsystem is critically broken: transformers can be neither registered nor managed due to bugs in the implementation. Issues have been reported on the projects tracker[5].

Autonomy JCAF uses context service instances to group entities, items and relationships of a single application domain [56].

Context services need to be started manually and there is no built-in discovery mechanism to determine their addresses. Because JCAF is predominantly based on Java RMI, there are, however, established practices of how to partially work around this limitation [62].

Scalability and performance The RMI basis provides a fast and well tested foundation for JCAF.

As mentioned above, context services partition the whole of contextual information into different application domains [56]. This naturally keeps these information domains smaller and ensures fast and efficient lookup.

To share contextual information across domains, services can connect to each other through peer to peer connections. Entity lookup spanning more than one service can be hop-limited to restrict the search to closely linked services [61].

Additionally, most of the frameworks functionality is exposed through asynchronous interfaces to address the needs of interactive applications and to account for network delays [56, 61].

Ease of integration To start context services, some widely used third party libraries, all of which are released under permissive open source licenses such as the Apache Software License or the Lesser General Public License[6], are required.

To link applications to context services, support for Java RMI is the only requirement next to the JCAF framework itself.

[5] "ContextTransformer can not be added to transformer repository":
`https://sourceforge.net/apps/trac/jcaf/ticket/1`
"TransformerRepositoryImpl can not find previously added Transformers":
`https://sourceforge.net/apps/trac/jcaf/ticket/2`

[6] The used JSON reference implementation contains the following humorous but legally potentially problematic amendment to the standard MIT license: "The Software shall be used for Good, not Evil." [63]

A recent effort to provide a JCAF implementation for Android devices also yielded a partial RESTful[7] implementation of JCAFs API based on the Restlet framework [64, 65]. This work has been integrated into the released JCAF version 1.5 and was used to develop a context aware contacts application for Android phones based on the Context-Phone prototyping platform [64].

The official tutorial, [61], provides excellent developer documentation.

Security JCAF provides a rudimentary security layer based on the Java Security API [56].

Context monitors can authenticate themselves to a secure context service with asymmetric cryptography. All context items published by authenticated context monitors are marked as secure within the context services [56, 61].

No means of exchanging key pairs is provided within JCAF [61].

This functionality does not restrict access to information to authorized parties but is instead meant to prevent malicious clients from publishing wrong information to the context service.

However, as of JCAF version 1.5 this layer is merely a proof of concept. Even without exploiting any possible security issues with the actual implementation, it is easy to publish "secure" context information without ever authenticating with the context service: Unauthorized context services do not strip the secure flag from already registered entities as shown in listing 3.8. The issue has been reported to the developers[8].

```
Clock clock = new Clock(/*...*/);
clock.setSecure(true);
getContextService().addEntity(clock);

//Output: "Secure: false"
System.out.println("Secure:␣" +
                   getContextService().getEntity(clock.getId()).isSecure());

clock.setSecure(true);
getContextService().setEntity(clock);

//Output: "Secure: true"
System.out.println("Secure:␣" +
                   getContextService().getEntity(clock.getId()).isSecure());
```

Figure 3.8: JCAF: Publishing "secure" context information from unauthenticated services

The need to provide access control to sensitive context information is acknowledged but has not yet been addressed by the released version of JCAF [56].

Stability and Development Status Overall, the core of JCAF proved to be stable enough to easily implement the experiment as described in 3.4.1, but some of the more advanced, lesser used functionality like context transformers or the security architecture

[7] Conforming to the architectural style of "Representational State Transfer", or simply "REST".

[8] "setEntity does not reset the "secure" flag":
https://sourceforge.net/apps/trac/jcaf/ticket/3

showed that the framework still has some critical problems that have not yet been addressed.

In August 2009, the source code of JCAF 1.5 was checked into the projects Sourceforge project repository. Since then, a total of just 45 change sets have been committed (September 2012, [66]).

However, the relatively recent work on supporting JCAF on Android devices suggests the possibility of continued research interest [64].

3.5 KnowRob

KnowRob, a knowledge processing and management framework developed by the Technical University Munich, takes a different approach to gathering and distributing contextual information.

Designed for use with the hugely popular robot operating system ROS, the main goal of KnowRob is to enable the development of autonomous robots. The system is based on Prolog and uses OWL ontologies [67].

Through aggregated knowledge, KnowRob can be used to break down complex or unspecific goals (e.g., "Set the table") into easier, concrete subtasks by formulating the goal as a composition of sub-actions grounded in the perception and action system [67].

Such predefined action plans can be augmented through learned actions from humans with the help of machine learning approaches including support vector machines and decision trees [68]. Information can also be shared with other robots through the use of the distributed RoboEarth database [69].

By focusing on the knowledge itself, common-sense knowledge databases can naturally be integrated at all stages of the reasoning process by simply making their data available to KnowRob [70]. Additionally, "computable" classes and properties provide a way to query active information sources at runtime in order to, for example, make live sensor data available to the knowledge base [67].

This knowledge-guided, action driven approach has already led to very promising results in the field of robotics.

However, many of the challenges for a knowledge base to guide autonomous robots addressed by KnowRob are equally relevant in the domain of context-aware computing solutions. Therefore, it was decided to evaluate the framework from the viewpoint of a context-awareness middleware framework.

3.5.1 Experiment

To build the experiment explained in Section 2.2, a custom OWL ontology was designed. Several Java modules were built to assert and maintain diverse information in the knowledge base. Additionally, Prolog computables are used to reason about implicit properties.

The KnowRob database is centrally hosted and made accessible through JSON with the json_prolog ROS node.

A conceptual diagram of how those components fit together can be found in Figure 3.9.

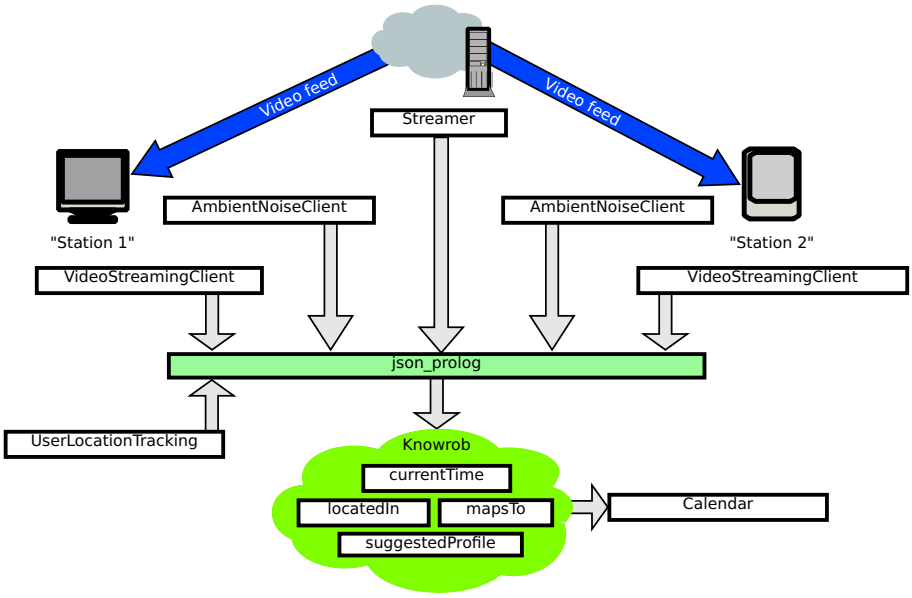

Figure 3.9: KnowRob: Conceptual Architecture

Ontology design The new ontology, shown in Figure 3.10, uses the `http://www.semanticweb.org/at/tugraz/ist/context/videostreaming` namespace and abstracts the simulated environment.

The rooms one to three and the three available quality profiles have been included in the knowledge base as instances of the `Location` and `Quality` classes, respectively. All other individuals as well as their properties and relationships are built dynamically at runtime.

Location Awareness The `locatedIn` relation is used to reason about the position of locatable objects (individuals of `Viewpoint` or `User`).

Viewpoints, which are assumed to be statically installed in the environment assert their location directly.

The user tracker, on the other hand, does not assert a specific discrete location, but rather provides the sensed information to the knowledge base: Upon detecting a user, a new `User` instance is pushed to the knowledge base. Tracking the user, the determined coordinates (x,y) are asserted as `Coordinate` instance, that is added to the `User` object through the `coordinateOf` relation.

To resolve those coordinates to one of the three rooms in the simulated environment, a Prolog computable (`mapsTo`) is used. Additionally, a `locatedIn` computable accurately reflects such implicit `Location` relations.

As an example, the definition of the `mapsTo` computable, part of the ontology, can be found in listing 3.11. Its Prolog implementations is to be found in listing 3.12.

Ambient Noise Compensation The ambient noise detection asserts and updates the `ambientNoise` data property of the `Viewpoint` instance that is running the measurement.

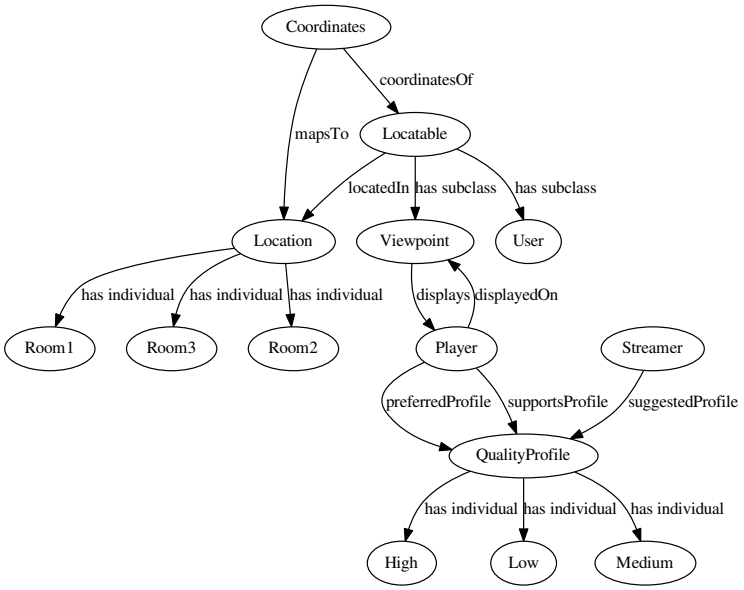

Figure 3.10: KnowRob: Video streaming ontology

This property is then parsed by the player to update the playback volume.

Adapting to Computation Classes The preferred computational class is asserted by the `Player` instance upon creation.

The streaming server adapts the streaming quality based on the lowest `QualityProfile` of all players that are displayed in a `Location` where there is currently a `User` present. This logic is abstracted in another Prolog computable and exposed as the `suggestedProfile` attribute of the `Streamer` instance. The computables implementation can be seen in listing 3.13.

Automatically starting the Stream To automatically start the video stream at certain times, a Prolog predicate is used that, mainly to demonstrate the capabilities of computable knowledge within KnowRob, creates a call to Javas Calendar object utilizing the JPL Prolog bindings. The predicate is shown in full in listing 3.14.

3.5.2 Evaluation

Versatility While there are some ROS modules available that provide KnowRob integration (e.g., comp_cop, tf_prolog), most of them are firmly rooted in the robotics domain and likely of lesser interest to software developers writing end-user applications.

However, writing new computables is fairly easy and well documented.

Prolog computables and various language bindings for SWI Prolog can be used to interact with most popular programming languages.

```
<ObjectProperty rdf:about="&videostreaming;mapsTo">
    <rdfs:domain rdf:resource="&videostreaming;Coordinates"/>
    <rdfs:range rdf:resource="&videostreaming;Location"/>
    <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
</ObjectProperty>

<computable:PrologProperty rdf:about="#computeMapsTo">
  <computable:command rdf:datatype="&xsd;string">comp_mapTo</computable:command>
  <computable:cache rdf:datatype="&xsd;string">cache</computable:cache>
  <computable:visible rdf:datatype="&xsd;string">unvisible</computable:visible>
  <computable:target rdf:resource="&videostreaming;mapsTo" />
</computable:PrologProperty>
```

Figure 3.11: KnowRob Prolog computables: Definition

Additionally, SQL computables can expose database information to the knowledge base.

Deduction and Aggregation The Prolog foundation affords for very powerful deductive reasoning. Additional predicates and ontologies (and therefore also additional computables) can be defined at runtime by all clients.

For advanced reasoning techniques, one can either implement the desired algorithm in Prolog or use one of the numerous language bindings to use external components (see Section 3.5.1 for an example employing JPL to call Java code).

An example implementation of decision trees inference for KnowRob can be found in [68]. The code is available through the ROS module comp_orgprinciples.

Autonomy In principle, ROS is a peer-to-peer system: individual ROS nodes communicate directly. However, a central name service called "roscore" (or sometimes simply "master") is used to discover available nodes in a network [71].

Normally, the network address of roscore must be known and set manually.

However, a ROS zeroconf module is available[9] that publishes information about the master node through Avahi. A very simple Bash script that correctly exports the `ROS_MASTER_URI` environment variable on Linux systems with the address of the first available roscore server on the local network can be found in listing 3.15.

Scalability and performance While SWI-Prolog and its semantic web extensions for handling OWL ontologies performed very well for the experiment outlined above, the current KnowRob system contains some fundamental scalability issues.

There is currently no method of subscribing to be notified of changes in the knowledge base.

Because KnowRob does not keep a complete database of all reasoned outcomes but instead performs reasoning only on demand (queries), adding a publish / subscribe

[9] Documentation: `http://www.ros.org/wiki/zeroconf`

```
objectMatchesClass(Id, Class) :-
  owl_has(Id, 'http://www.w3.org/1999/02/22-rdf-syntax-ns#type', Class).
objectMatchesClass(Id, Class) :-
  owl_has(X, 'http://www.w3.org/2000/01/rdf-schema#subClassOf', Class),
  objectMatchesClass(Id, X).

comp_mapTo(Coordinates, Location) :- objectMatchesClass(Coordinates,
  'http://www.semanticweb.org/at/tugraz/ist/context/videostreaming#Coordinates'),
  objectMatchesClass(Location,
  'http://www.semanticweb.org/at/tugraz/ist/context/videostreaming#Location'),
  % extract x and y coordinates:
  owl_has(Coordinates, videostreaming:'x', literal(X)),
  owl_has(Coordinates, videostreaming:'y', literal(Y)),
  ( % map coordinates to one of three possible locations
    (Location =
       'http://www.semanticweb.org/at/tugraz/ist/context/videostreaming#Room1',
     X >= 0, Y >= 0, X < 1, Y < 1) ;
    (Location =
       'http://www.semanticweb.org/at/tugraz/ist/context/videostreaming#Room2',
     X < 2, Y >= 0, X >= 1, Y < 1) ;
    (Location =
       'http://www.semanticweb.org/at/tugraz/ist/context/videostreaming#Room3',
     X < 2, Y < 2, X >= 0, Y >= 1)
  ).
```

Figure 3.12: KnowRob Prolog computables: Implementation

interface would not be trivial[10]. According to a private mail of Moritz Tenorth, one of the lead developers of KnowRob, no such system is planned at the moment.

To keep track of changes in the sensed context, application developers are therefore required to use polling, putting constant stress on the infrastructure.

Moreover, KnowRob does not provide information replication between different instances. The only way to share information between knowledge bases is through explicit ex- and imports - possibly to a third, shared KnowRob based database like RoboEarth [69].

Ease of integration To integrate KnowRob with their application, developers need to look up the roscore name service and then use that to find the KnowRob (json_prolog) node. Service calls (queries) are then sent through simple XML-RPC messages. Results returned by json_prolog are JSON encoded.

In addition to using open, well documented protocols where helper libraries exist for most major programming languages, higher level ROS abstractions are also available for many languages including C++, Python, Octave and LISP [71].

All of ROS itself, the entire KnowRob stack and SWI-Prolog itself are licensed under the terms of either the BSD license, the GPL or the LGPL.

[10] As a partial workaround, SWI-Prologs broadcast library can be used to provide application level events that are broadcast explicitly. However, this only triggers events within the knowledge base. A separate application layer message passing system would be required to act on subscribed clients.

```
%pick lowest preferred profile of all playing players
comp_suggestedProfile(Streamer, PreferredProfile) :-
  objectMatchesClass(Streamer,
     'http://www.semanticweb.org/at/tugraz/ist/context/videostreaming#Streamer'),
  objectMatchesClass(A,
     'http://www.semanticweb.org/at/tugraz/ist/context/videostreaming#User'),
  objectMatchesClass(UserLocation,
     'http://www.semanticweb.org/at/tugraz/ist/context/videostreaming#Location'),
  objectMatchesClass(Player,
     'http://www.semanticweb.org/at/tugraz/ist/context/videostreaming#Player'),
  objectMatchesClass(PreferredProfile,
     'http://www.semanticweb.org/at/tugraz/ist/context/videostreaming#
                                                          QualityProfile'),
  objectMatchesClass(Viewpoint,
     'http://www.semanticweb.org/at/tugraz/ist/context/videostreaming#Viewpoint'),
  ( % sort ascending on quality profiles
    PreferredProfile =
        'http://www.semanticweb.org/at/tugraz/ist/context/videostreaming#Low';
    PreferredProfile =
        'http://www.semanticweb.org/at/tugraz/ist/context/videostreaming#Medium';
    PreferredProfile =
        'http://www.semanticweb.org/at/tugraz/ist/context/videostreaming#High'
  ),
  comp_locatedIn(A, UserLocation),
  comp_locatedIn(Viewpoint, UserLocation),
  owl_has(Player, videostreaming:'displayedOn', Viewpoint),
  owl_has(Player, videostreaming:'preferredProfile', PreferredProfile).
```

Figure 3.13: KnowRob Prolog computable: Adjusting stream quality

```
currentTime(Hour, Minute, Second) :-
      jpl_call('java.util.Calendar', 'getInstance', [], Cal),
      jpl_call(Cal, 'getTime', [], Time), jpl_call(Time, 'getHours', [], Hour),
      jpl_call(Time, 'getMinutes', [], Minute),
      jpl_call(Time, 'getSeconds', [], Second).
```

Figure 3.14: KnowRob Prolog predicate: JPL call to Java module

Security Neither ROS itself nor KnowRob addresses security aspects. No access limitations can be defined.

Stability and Development Status Both KnowRob and ROS itself are actively developed. Packages for Ubuntu are available and there are experimental packages for Windows, Mac OS X and other Linux distributions.

KnowRob is meant to be compiled from source with the ROS build system.

```
export ROS_MASTER_URI=‘avahi-browse -t -r _ros-master._tcp |tr ’\n’ ’␣’ |
                       awk -F ’[\\\\[\\\\]]’ ’{␣print␣"http://"$4":"$6␣}’‘
```

Figure 3.15: ROS: Improvised discovery of roscore node

For the purpose of the experiment, a KnowRob installation was completed under a current version of Gentoo revealing countless problems - at least some of them certainly not limited to the platform. Multiple issues were reported[11].

ROS developers, contacted through the official IRC chat room mentioned that the current *source* version, when installed as per instructions on the official documentation is “probably not very stable right now”.

An installation on Ubuntu 12.04 using the provided ROS fuerte *binary* packages posed no problems.

Both ROS and KnowRob are free software and follow an open development methodology [72].

[11] “rospack overwrites PKG_CONFIG_PATH incorrectly”:
`https://code.ros.org/trac/ros/ticket/4034`
“json_prolog requires simplejson but does not depend on it”:
`https://code.ros.org/trac/ros/ticket/4040`
“rosjava_jni does not log java exceptions”:
`https://code.ros.org/trac/ros/ticket/4041`
“json_prolog: Hardcoded paths to Java libraries”:
`https://code.ros.org/trac/ros/ticket/4042`
“rosjava: Sourced maven repository no longer available”:
`https://code.ros.org/trac/ros/ticket/4044`

4 Findings

This publication presents a software engineering take on currently available context-awareness middleware systems.

Three of the most promising solutions were tested thoroughly by implementing a context-aware video streaming solution that was designed to involve all four major aspects of context-awareness: time, user, physical and computing context.

4.1 Reasons for context-awareness middleware systems

To provide and use contextual information there are many challenges that arise naturally and are not bound to any specific application domain.

These challenges include context acquisition (how to retrieve context from a multitude of sources), context representation (how to codify acquired context information), context dissemination (how to transfer context information across different devices) as well as context deduction, aggregation and reasoning in general (how to draw conclusions from reasoned context).

Next to these functional requirements, context-aware systems also usually operate in changing, potentially very large networks, have to handle uncertain or even conflicting knowledge and need to provide ways to secure access to contextual information.

Context-awareness middleware systems can present a sensible approach to solve these and other problems at an infrastructure level and provide an efficient base for third party application developers to build on.

4.2 State of the art

A lot of context-awareness middleware frameworks are available but evaluating all of them would obviously go beyond the scope of this thesis. Instead, a selection of three highly regarded solutions was compared.

All three tested frameworks proved to be helpful by providing ways to exchange information with other applications and services both locally and over a network connection.

However, the evaluation process also showed significant differences between the individual solutions, highlighting the diversity of the field.

4.2.1 Context Toolkit

Out of the three evaluated here, the Context Toolkit will likely feel the most natural framework to use for developers coming from a traditional UI programming background.

In contrast to both JCAF and KnowRob, the Context Toolkit does not build a network of shared context information but instead focuses on communicating components.

This more imperative approach results in a "widget" metaphor that fits well with how most applications are currently build. Integration with "legacy" applications is therefore comparatively easy.

The released implementation (version 2.0) is fairly easy to set up and relatively lightweight but limited and, in parts, simply not reliable enough.

4.2.2 JCAF

JCAF wants to establish itself as a standard Java API for context-aware systems [56]. Through the use of standard technology like Java RMI, the system will certainly feel familiar to existing Java developers.

The JCAF framework provides a reasonable object oriented approach to ontology based systems, taking advantage of the expressiveness of explicitly modeled relationships while keeping overhead low by using automatically serialized Java objects to represent information.

Of the three tested solutions, it features the easiest setup and fewest dependencies but is also the least powerful, providing no real reasoning capabilities, a dangerously flawed security layer and no automatic discovery of peers.

4.2.3 KnowRob

KnowRob, the only tested framework that was not explicitly designed to aid development of context-aware solutions, worked surprisingly well doing exactly that in practice.

Its information centered approach based on OWL ontologies and other semantic web technologies force application developers to strictly formalize logical objects and relationships of their application domains.

However, this significant time investment is rewarded with superior interoperability simply by sharing ontologies between context-aware applications and a far greater reasoning potential within the knowledge base which is easily exploitable through application defined "computables".

Sadly, the lack of any form of push notifications on changes in the knowledge base, the significant dependencies introduced by ROS and limiting official support to only Ubuntu Linux all greatly diminish KnowRobs practical usefulness at the time of this writing.

4.2.4 Summary

The Context Toolkit, JCAF and KnowRob were all able to support the context network needed for the experiment described in Section 2.2.

The tested systems have all released their source code under open source licenses but none, with the notable exception of JCAF which received a significant contribution from another team of researchers for a short period of time [64], managed to gain real momentum.

As the original core developer teams of researchers moved on to other projects, each codebase became neglected and often de facto unmaintained.

In their current state, none of the three tested solutions can be recommended to application developers for anything other than research purposes.

4.3 Future Prospects

Context-awareness remains an actively researched topic.

While extensive literary research was undertaken before selecting the context-awareness frameworks evaluated here, many other promising solutions like SOLAR [73], PACE [74], SALES [75] or even the commercial Ideate Framework [76] remain unexplored due to the limited scope of this thesis.

Regardless, it can be noted that context-awareness middleware solutions need to reach a certain adoption rate to achieve their full potential outside of carefully crafted use cases. Therefore, the main challenge ultimately lies not in open research problems but in attracting a vibrant developer community.

While the recent rise of smart mobile devices, which stand to benefit greatly from context-aware computing, and the ever present curiosity in open source communities both present opportunities for context-awareness middleware platforms, it remains to be seen if any standardized solution manages to reach practical significance in the near future.

List of Figures

Bibliography

[1] The Nielsen Company: *The Mobile Media Report.* December 2011.

[2] Ipsos MediaCT Germany: *Mobile Internet & Smartphone Adoption.* Google Mobile Ads Blog, January 2012.

[3] Hansmann, Uwe, Peter Thompson, Riku M. Mettala, and Apratim Purakayastha: *Syncml: Synchronizing Your Mobile Data.* Prentice Hall Professional Technical Reference, 2002, ISBN 0130093696.

[4] *Google: Product catalog.* `http://www.google.com/about/products`.

[5] *Apples iCloud: Official Homepage.* `https://www.icloud.com`.

[6] Weiser, M.: *The computer for the 21st century.* Scientific American, 265(3):94–104, 1991.

[7] Cook, Diane and Sajal Das: *Smart Environments: Technology, Protocols and Applications (Wiley Series on Parallel and Distributed Computing).* Wiley-Interscience, 2004, ISBN 0471544485.

[8] Schilit, Bill and M. Theimer: *Disseminating Active Map Information to Mobile Hosts.* IEEE Network, 8(5):22–32, 1994.

[9] BELLAVISTA, P., A. CORRADI, M. FANELLI, and L. FOSCHINI: *A Survey of Context Data Distribution for Mobile Ubiquitous Systems.* ACM Computing Surveys, 45(1):1–49, 2013.

[10] Abowd, Gregory D., Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles: *Towards a Better Understanding of Context and Context-Awareness.* In *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, HUC '99, pages 304–307, London, UK, UK, 1999. Springer-Verlag, ISBN 3-540-66550-1.

[11] Chen, Guanling and Kotz, David: *A Survey of Context-Aware Mobile Computing Research.* Technical report, Hanover, NH, USA, 2000.

[12] Khelil, Abdelmajid, Faisal Karim Shaikh, Brahim Ayari, and Neeraj Suri: *MWM: a map-based world model for wireless sensor networks.* In *Proceedings of the 2nd International Conference on Autonomic Computing and Communication Systems*, Autonomics '08, pages 5:1–5:10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ISBN 978-963-9799-34-9.

[13] Konrad, Kułakowskim and Wąsm Jarosław: *World Model for Autonomous Mobile Robot – Formal Approach.* In *Intelligent Information Systems*, pages 37–45, Al. Mickiewicza 30, 30-059 Cracow, Poland, 2010. ISBN 978-83-7051-580-5.

[14] Vagts, Hauke, Erik Krempel, and Yvonne Fischer: *Access controls for privacy protection in pervasive environments.* In *Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '11, pages 52:1–52:8, New York, NY, USA, 2011. ACM, ISBN 978-1-4503-0772-7.

[15] Calangiu, G.A., I. Sarkany, M. Stoica, and F. Sisak: *Modeling a knowledge base for generating new trajectories for a robot arm located in a cell of a flexible fabrication line.* In *Optimization of Electrical and Electronic Equipment (OPTIM), 2010 12th International Conference on*, pages 813 –818, may 2010.

[16] Guarino, N. and P. Giaretta: *Ontologies and Knowledge Bases: Towards a Terminological Clarification.* In Mars, N. (editor): *Towards very large knowledge bases.* Amsterdam: IOS Press, 1995.

[17] Barakonyi, István and Dieter Schmalstieg: *Augmented reality agents for user interface adaptation.* Comput. Animat. Virtual Worlds, 19(1):23–35, February 2008, ISSN 1546-4261. `http://dx.doi.org/10.1002/cav.v19:1`.

[18] Müller, Jörg P. and Markus Pischel: *The agent architecture inteRRaP: Concept and application.* Technical report, German Research Center for Artificial Intelligence, 1993. `http://jmvidal.cse.sc.edu/library/muller93a.pdf`.

[19] Knight, Kevin and Steve K. Luk: *Building a Large-Scale Knowledge Base for Machine Translation.* In *National Conference on Aritificial Intelligence, AAAI*, pages 773–778, 1994, ISBN 0-262-51112-6.

[20] Izumida, Yoshio, Hiroshi Ishikawa, Toshiaki Yoshino, Tadashi Hoshiai, and Akifumi Makinouchi: *A natural language interface using a world model.* In *EACL*, pages 205–212, 1985.

[21] Maurelli, Francesco, Zeyn Saigol, Carlos C. Insaurralde, Yvan R. Petillot, and David M. Lane: *Marine world representation and acoustic communication: challenges for multi-robot collaboration.* In *Proceedings of IEEE AUV2012, Southampton, UK*, 2012.

[22] Fuchs, S.: *A Comprehensive Knowledge Base for Context-aware Tactical Driver Assistance Systems.* Smart system technologies. Shaker, 2008, ISBN 9783832280789.

[23] Soylu, A., P.D. Causmaecker, and P. Desmet: *Context and adaptivity in pervasive computing environments: Links with software engineering and ontological engineering.* Journal of Software, 4(9):992–1013, 2009.

[24] Brown, P.J.: *The stick-e document: a framework for creating context-aware applications.* ELECTRONIC PUBLISHING-CHICHESTER-, 8:259–272, 1995.

[25] Schilit, W.N.: *A system architecture for context-aware mobile computing.* PhD thesis, Columbia University, 1995.

[26] Schilit, B., N. Adams, and R. Want: *Context-Aware Computing Applications.* In *IEEE Workshop on Mobile Computing Systems and Applications*, pages 85–90, 1994.

[27] Lim, B.Y. and A.K. Dey: *Toolkit to support intelligibility in context-aware applications.* In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pages 13–22. ACM, 2010.

[28] Ltd., PrimeSense: *User Tracker - sample program (Java)*, 2011.

[29] *VideoLANs VLC media player; Official Homepage.* `http://www.videolan.org`.

[30] *vlc - Java Framework for the vlc Media Player.* `http://code.google.com/p/vlcj/`.

[31] *ITU-T Recommendation H.264 : Advanced video coding for generic audiovisual services*, November 2007. `http://www.itu.int/rec/T-REC-H.264-200711-I/en`.

[32] *Information technology — Coding of audio-visual objects — Part 3: Audio*, September 2009. `http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=53943`.

[33] *Adobe Flash Video File Format Specification*, 2010. `http://www.adobe.com/devnet/f4v.html`.

[34] Lohmar, T., T. Einarsson, P. Frojdh, F. Gabin, and M. Kampmann: *Dynamic adaptive HTTP streaming of live content.* In *Proceedings of the 2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, WOWMOM '11, pages 1–8, Washington, DC, USA, 2011. IEEE Computer Society, ISBN 978-1-4577-0352-2. `http://dx.doi.org/10.1109/WoWMoM.2011.5986186`.

[35] Cerqueira, R., C. K. Hess, M. Roman, and R.H. Campbell: *Gaia: A Development Infrastructure for Active Spaces.* In *Workshop on Application Models and Programming Tools for Ubiquitous Computing*, September 2001.

[36] Ranganathan, A. and R.H. Campbell: *A middleware for context-aware agents in ubiquitous computing environments.* In *Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*, pages 143–161. Springer-Verlag New York, Inc., 2003.

[37] *GAIA: Official Homepage.* `http://gaia.cs.uiuc.edu/`.

[38] Gu, T., H.K. Pung, and D.Q. Zhang: *A service-oriented middleware for building context-aware services.* Journal of Network and Computer Applications, 28(1):1–18, 2005.

[39] Gu, Tao, Hung Keng Pung, and Da Qing Zhang: *Toward an osgi-based infrastructure for context-aware applications.* IEEE Pervasive Computing, 3(4):66–74, October 2004, ISSN 1536-1268. `http://dx.doi.org/10.1109/MPRV.2004.19`.

[40] Mcdermott, Drew and Dejing Dou: *Representing disjunction and quantifiers in rdf.* In *In Proceedings of International Semantic Web Conference 2002*, pages 250–263, 2002.

[41] McGuinness, Deborah L. and Frank van Harmelen: *Owl web ontology language overview.* Technical Report REC-owl-features-20040210, W3C, 2004.

[42] *Apache Jenna: Homepage.* `http://jena.apache.org/index.html`.

[43] Gu, Tao, H. K. Pung, D. Q. Zhang, Hung Keng Pung, and Da Qing Zhang: *A bayesian approach for dealing with uncertain contexts.* 2004.

[44] *Homepage of Tao Gu at the University of Southern Denmark.* `http://www.imada.sdu.dk/~gu`.

[45] *J2SE 5.0 in a Nutshell.* `http://www.oracle.com/technetwork/articles/javase/j2se15-141062.html`.

[46] Gu, Tao, Daqing Zhang, and Hung Keng Pung: *An ontology-based p2p network for semantic search.* IJGHPC, 1(4):26–39, 2009. `http://dblp.uni-trier.de/db/journals/ijghpc/ijghpc1.html#GuZP09`.

[47] Dey, A.K., G.D. Abowd, and D. Salber: *A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications.* Human-Computer Interaction, 16(2-4):97–166, 2001.

[48] Dey, A.K.: *Enabling the use of context in interactive applications.* In *CHI'00 extended abstracts on Human factors in computing systems*, pages 79–80. ACM, 2000.

[49] *Context Toolkit: API Documentation.* `http://contexttoolkit.googlecode.com/svn-history/trunk/docs/apidoc/index.html`.

[50] Dey, A.K. and G.D. Abowd: *The context toolkit: Aiding the development of context-aware applications.* In *Workshop on Software Engineering for wearable and pervasive computing*, pages 431–441, 2000.

[51] Dey, Anind K.: *Understanding and using context.* Personal Ubiquitous Comput., 5(1):4–7, January 2001, ISSN 1617-4909. `http://dx.doi.org/10.1007/s007790170019`.

[52] Dey, Anind K. and Jennifer Mankoff: *Designing mediation for context-aware applications.* ACM Trans. Comput.-Hum. Interact., 12(1):53–80, March 2005, ISSN 1073-0516. `http://doi.acm.org/10.1145/1057237.1057241`.

[53] *Context Toolkit: Installation instructions.* `http://www.contexttoolkit.org/?page_id=4`.

[54] Lim, Brian Y. and Anind K. Dey: *Investigating intelligibility for uncertain context-aware applications.* In *Proceedings of the 13th international conference on Ubiquitous computing*, UbiComp '11, pages 415–424, New York, NY, USA, 2011. ACM, ISBN 978-1-4503-0630-0. http://doi.acm.org/10.1145/2030112.2030168.

[55] Lim, Brian Y. and Anind K. Dey: *Design of an intelligible mobile context-aware application.* In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, pages 157–166, New York, NY, USA, 2011. ACM, ISBN 978-1-4503-0541-9. http://doi.acm.org/10.1145/2037373.2037399.

[56] Bardram, J.: *The Java Context Awareness Framework (JCAF)–a service infrastructure and programming framework for context-aware applications.* Pervasive Computing, pages 98–115, 2005.

[57] Bardram, Jakob E. and Thomas R. Hansen: *The aware architecture: supporting context-mediated social awareness in mobile cooperation.* In *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, CSCW '04, pages 192–201, New York, NY, USA, 2004. ACM, ISBN 1-58113-810-5. http://doi.acm.org/10.1145/1031607.1031639.

[58] Bardram, Jakob E.: *Applications of context-aware computing in hospital work: examples and design principles.* In *Proceedings of the 2004 ACM symposium on Applied computing*, SAC '04, pages 1574–1579, New York, NY, USA, 2004. ACM, ISBN 1-58113-812-1. http://doi.acm.org/10.1145/967900.968215.

[59] Bardram, Jakob E. and Niels Nørskov: *A context-aware patient safety system for the operating room.* In *Proceedings of the 10th international conference on Ubiquitous computing*, UbiComp '08, pages 272–281, New York, NY, USA, 2008. ACM, ISBN 978-1-60558-136-1. http://doi.acm.org/10.1145/1409635.1409672.

[60] Bardram, Jakob E.: *Activity-based computing for medical work in hospitals.* ACM Trans. Comput.-Hum. Interact., 16(2):10:1–10:36, June 2009, ISSN 1073-0516. http://doi.acm.org/10.1145/1534903.1534907.

[61] Bardram, J.: *Tutorial for the Java Context Awareness Framework (JCAF), version 1.5.* 2005.

[62] Bishop, Philip and Nigel Warren: *Jini-like discovery for RMI*, November 2001. http://www.javaworld.com/javaworld/jw-11-2001/jw-1121-jinirmi.html.

[63] *JSON Reference Implementation: License.* http://www.json.org/license.html.

[64] Szanto, Karoly: *Extending the Java Context Awareness Framework for Android.* 2010.

[65] *Restlet: Homepage.* http://www.restlet.org/about/introduction.

[66] *JCAF: SVN Repository statistics.* https://sourceforge.net/projects/jcaf/stats/scm?repo=SVNRepository&dates=2005-07-18+to+2012-09-17.

[67] Tenorth, Moritz and Michael Beetz: *KNOWROB: knowledge processing for autonomous personal robots.* In *Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems*, IROS'09, pages 4261–4266, Piscataway, NJ, USA, 2009. IEEE Press, ISBN 978-1-4244-3803-7. `http://dl.acm.org/citation.cfm?id=1732643.1732745`.

[68] Schuster, Martin, Dominik Jain, Moritz Tenorth, and Michael Beetz: *Learning organizational principles in human environments.* In *IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, MN, USA, May 14–18 2012.

[69] Waibel, M., M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J.M.M. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, and R. van de Molengraft: *Roboearth.* Robotics Automation Magazine, IEEE, 18(2):69–82, 2011, ISSN 1070-9932.

[70] Kunze, Lars, Moritz Tenorth, and Michael Beetz: *Putting People's Common Sense into Knowledge Bases of Household Robots.* In *33rd Annual German Conference on Artificial Intelligence (KI 2010)*, pages 151–159, Karlsruhe, Germany, September 21-24 2010. Springer.

[71] Quigley, Morgan, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng: *Ros: an open-source robot operating system.* In *ICRA Workshop on Open Source Software*, 2009.

[72] Cousins, S., B. Gerkey, K. Conley, and W. Garage: *Sharing software with ros [ros topics].* Robotics Automation Magazine, IEEE, 17(2):12 –14, june 2010, ISSN 1070-9932.

[73] Chen, G. and D. Kotz: *Solar: An open platform for context-aware mobile applications.* Technical report, DTIC Document, 2005.

[74] Henricksen, K. and J. Indulska: *A software engineering framework for context-aware pervasive computing.* In *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*, pages 77–86. IEEE, 2004.

[75] Corradi, A., M. Fanelli, and L. Foschini: *Implementing a scalable context-aware middleware.* In *Computers and Communications, 2009. ISCC 2009. IEEE Symposium on*, pages 868 –874, july 2009.

[76] Duggal, Dave and William Malyk: *A resource oriented framework for context-aware enterprise applications.* In *Proceedings of the Second International Workshop on RESTful Design*, WS-REST '11, pages 33–38, New York, NY, USA, 2011. ACM, ISBN 978-1-4503-0623-2. `http://doi.acm.org/10.1145/1967428.1967438`.