INSTITUT FÜR INFORMATIK III

DER RHEINISCHEN FRIEDRICH-WILHELMS-UNIVERSITÄT BONN

**The Semantic Shadow
Combining User Interaction
with Context Information
for Semantic Web-Site
Annotation**

Pascal Bihler

IAI-TR-2011-2 Juli 2011

universität**bonn**

INSTITUT FÜR INFORMATIK III

DER RHEINISCHEN FRIEDRICH-WILHELMS-UNIVERSITÄT BONN

**The Semantic Shadow
Combining User Interaction
with Context Information
for Semantic Web-Site
Annotation**

Pascal Bihler

**Forschungsbericht
Technical Report**

The Semantic Shadow

Combining User Interaction with Context Information for Semantic Web-Site Annotation

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Pascal Bihler

aus

Berlin

Bonn, 20. Dezember 2010

Abstract

This thesis develops the concept of the *Semantic Shadow (SemS)*, a model for managing contentual and structural annotations on web page elements and their values. The model supports a contextual weighting of the annotated information, allowing to specify the annotation values in relation to the evaluation context.

A procedure is presented, which allows to manage and process this context-dependent meta information on web page elements using a dedicated programming interface. Two distinct implementations for the model have been developed: One based on Java objects, the other using the Resource Description Framework (RDF) as modeling backend. This RDF-based storage allows to integrate the annotations of the *Semantic Shadow* with other information of the Semantic Web.

To demonstrate the application of the *Semantic Shadow* concept, a procedure to optimize web based user interfaces based on the structural semantics has been developed: Assuming a mobile client, a requested web page is dynamically adapted by a proxy prototype, where the context-awareness of the adaptation can be directly modeled alongside with the structural annotations.

To overcome the drawback of missing annotations for existing web pages, this thesis introduces a concept to derive context-dependent meta-information on the web pages from their usage: From the observation of the users' interaction with a web page, certain context-dependent structural information about the concerned web page elements can be derived and stored in the annotation model of the *Semantic Shadow* concept.

Überblick

In dieser Arbeit wird das Konzept des *Semantic Shadow* (dt. „Semantischer Schatten") entwickelt, ein Programmier-Modell um Webseiten-Elemente mit inhaltsbezogenen und strukturellen Anmerkungen zu versehen. Das Modell unterstützt dabei eine kontextabhängige Gewichtung der Anmerkungen, so dass eine Anmerkung in Bezug zum Auswertungs-Kontext gesetzt werden kann.

Zur Verwaltung und Verarbeitung dieser kontextbezogenen Meta-Informationen für Webseiten-Elemente wurde im Rahmen der Arbeit eine Programmierschnittstelle definiert. Dazu wurden zwei Implementierungen der Schnittstelle entwickelt: Eine basiert ausschließlich auf Java-Objekten, die andere baut auf einem RDF-Modell auf. Die RDF-basierte Persistierung erlaubt eine Integration der *Semantic-Shadow*-Anmerkungen mit anderen Anwendungen des Semantic Webs.

Um die Anwendungsmöglichkeiten des *Semantic-Shadow*-Konzepts darzustellen, wurde eine Vorgehensweise zur Optimierung von webbasierten Benutzerschnittstellen auf Grundlage von semantischen Strukturinformationen entwickelt: Wenn ein mobiler Benutzer eine Webseite anfordert, wird diese dynamisch durch einen Proxy angepasst. Die Kontextabhängigkeit dieser Anpassung wird dabei bereits direkt mit den Struktur-Anmerkungen modelliert.

Für bestehende Webseiten liegen zumeist keine Annotationen vor. Daher wird in dieser Arbeit ein Konzept vorgestellt, kontext-abhängige Meta-Informationen aus der Benutzung der Webseiten zu bestimmen: Durch Beobachtung der Benutzer-Interaktionen mit den Webseiten-Elementen ist es möglich bestimmte kontextabhängige Strukturinformationen abzuleiten und als Anmerkungen im Modell des *Semantic-Shadow*-Konzepts zu persistieren.

Acknowledgements

This thesis could not have been written without the support of many people, to all of whom I would like to express my gratitude, and more explicitly to:

First, I would like to thank Prof. Dr. Armin B. Cremers for his support of my research and his constructive criticism. The inspirational and insightful discussions with Prof. Dr. Albrecht Schmidt positively impacted the course of my research, and I greatly appreciated his persistent inquiries about my progress.

Additionally, my thanks go to my colleagues at the SAM research group, which sustained my excessive expositions, helped my pin down of this thesis' research topic, and excused my office absence while I was writing. Together with the student team of the Agile Lab 2010, they helped to evaluate the prototypical framework.

Finally, I thank Daniel Seidel and Thomas Karcher for reviewing my thesis draft in a tremendously short period of time...

Contents

Figures And Tables

List of Figures

List of Tables

Chapter 1

Introduction

1.1   Motivation

Today's World Wide Web is a massive collection of information, and at the same time the primary platform for communication and commerce. While in the beginning the web's user group was compact, nowadays it's extremely divers. People using a certain web based service[1] not only differ by their social backgrounds, by their favors and dislikes. With the advent of small but high-performance mobile devices like Netbooks, PDAs and Smartphones, coming along with fundamental improvements in wireless Internet coverage, mobile web use is rapidly increasing. More and more tasks are handled on those mobile devices instead of desktop computers. Therefore, the situations in which a web based service is employed are numerous and this usage contexts define another dimension in the service design space.

Web pages and web based services are used in various situations and contexts.

Ideally, a web based service user interface would be optimized for every usage scenario: A teenager ordering at Luigi's Restaurant at midnight on his latest multimedia mobile phone would see another menu than his grandma, using the same restaurant's service at noon from her home TV. The classic way to personalize web information for users and devices is the development of a dedicated, service-specific adaptation process. Taking into account the mentioned increase in user and device variety, this manual adaptation process results in an exponentially growing workload.

Web based services should be optimized for different usage scenarios.

If the web information was annotated with semantic information, revealing the structural and contextual content of a web page to the processing algorithms, these adaptation processes could be, to a certain extent, automated. Herewith, production costs and implementation time of a new service could be reduced while at the same time the knowledge requirements for the service development would be lower. This

Optimization could be automated using semantic meta information.

---

[1]Which might be any single web page or a collection of pages, e. g. a news portal, a shopping site or an online survey.

is one of the arguments supporting the Semantic Web, a vision firstly expressed by Berners-Lee in 2001 [BHL01].

Meta information
on web sites can be
derived from usage.

Unfortunately, the majority of existing web sites and web-applications are not annotated in a way supporting automated context adaptation, neither are most of the information and services published today. But, since the users of a web based service mostly have an idea of the service's semantic structure and relevance of the service's elements in the users' individual situation, the contextual meta information can be derived by observing the use of a web based service, as this research work shows. To use a very visual image, when the light of user interaction shines on a web site, it casts a "semantic shadow".

1.2   Problem Statement

The *Semantic
Shadow* concept
introduces context-
aware annotations.

In this research work, the concept of the *Semantic Shadow* for web sites is elaborated. By attaching context-dependent annotations to web site elements and values, it allows to enrich web based service UIs with structural and contentual meta information. This information can be derived from web site usage and can be then used for diverse dynamic optimization and adaptation processes.

The approach works
for existing and fu-
ture web pages.

Important for the implementation of this concept is its independence from the original content creator of the web based service. Therefore, the concept can be applied to existing web based information and interaction services as well as to newly generated content.

The problem statement is defined by these theses:

Thesis 1: Web based service UIs can be annotated with context-aware semantic and structural meta information.

Thesis 2: Meta information on web based service UIs can be stored and processed independently from the content source.

Thesis 3: With the help of semantic and structural meta information, web based service UIs can be automatically optimized for specific context and usage scenarios.

Thesis 4: By evaluating implicit interaction information, context-dependent structural and semantic information about web based service UIs can be derived.

1.3   Classification of User Groups

Users can be classi-
fied in three groups.

Since the concept of the *Semantic Shadow* is based on user interaction, and the main applications of the concept in the scope of this work are user interface adaptations, the understanding of a target user group is an initial task. When approaching this, Judt names in [Jud04] sociological information as age, gender, body abilities, formation, cultural and ethnic background, training, motivation, goals and personality. This

classification of target user groups can be further simplified. Schneiderman [Sch98] proposes the generic classification of users into three categories:

- First-time and unexperienced users

- Experienced users

- Power users (Experts using the interface frequently)

As the derivation of the Semantic Shadow is based on frequent and numerous use, experienced users and power users can be identified as the target group for implicit data collection. On the other hand, adaptation scenarios based on Semantic Shadow annotations can very likely target first-time and unexperienced users along with the more experienced user groups.

Frequent users are required for data collection, but data use is open for all user groups.

1.4 Current Situation

Currently, user interface adaptation in the web is most frequently implemented by using different web-pages (or, in a more general approach, different visualization styles) and delivering them based on the user's browser type. Various frameworks try to adapt web pages and their content for mobile clients, see [CMZ03, XTL08, MMR⁺10] for examples. While those frameworks take into account the user's device context when adapting the web page, the context awareness is modeled by the adaptation algorithm itself and not embedded into the page meta information.

Current adaptation frameworks do not embed the context into the meta data.

For an overview of related work and technologies, please see Chapter 3.

1.5 Contribution of this Work

In correspondence with the theses of Section 1.2, this work provides the following contributions to implement the concept of the *Semantic Shadow*:

The *Semantic Shadow* concept has been implemented in this work.

Contribution 1: A concept to represent context-dependent semantics and structures of web site elements.

Contribution 2: A procedure to manage and process context-dependent meta information on web UIs.

Contribution 3: A procedure to optimize web based service UIs for specific usage scenarios.

Contribution 4: A concept to derive context-dependent meta-information on web UIs from their usage.

1.5.1   Concept to Represent Context-Dependent Semantics and Structures of Web Site Elements

Structural informa-
tion on elements of
web sites is stored in
annotations.

Automatic context-dependent adaptations of HTML [RHJ99] pages,
e. g. for barrier-free web access [AT00] or for visualization on mobile
devices [CMZ03] can be supported by adding further semantic infor-
mation to web site elements in form of annotations. Two different kinds
of annotations can be distinguished:

1. Annotations describing *the content* of the annotated element.

2. Annotations describing *the structural semantics* of the annotated
   element.

Both kinds of annotations can be connected with context-information
to describe the range of the annotation's validity. Furthermore, this in-
formation can be enriched with a probability value describing the like-
lihood of the trueness of the described annotation in the given context.

Annotations can be
represented in RDF.

To represent semantic information in the area of web, the RDF [KC04]
is an established standard and W3C recommendation. To model web
site elements' semantics in a context-dependent way, this work extends
RDF with a context dimension and uses an extended URI format based
on XPath [DRSC09] to make single web site elements RDF subjects.

1.5.2   Procedure to Manage and Process Context-Dependent Information on Web UIs

Annotations are ma-
naged with the *Se-
mantic Shadow* API.

While information stored in an RDF model can be processed indepen-
dently from an additional access framework, for application purposes
it is more efficient to access the data via an optimized programming
interface (API). This interface represents the *Semantic Shadow* as a pro-
gramming object and encapsulates the management algorithms from
the accessing software, as well as implementation details like the data
representation in memory or in a database. Figure 1.1 shows the gen-
eral architecture, with the external API supporting methods to:

• Annotation retrieval for a given web site element and context

• Annotation retrieval for a given web page and context

• Annotation retrieval for a given URI prefix and context

If the *Semantic Shadow* is mutable, the following actions are in addition
supported by the interface:

• Annotation creation and removal

• Annotation type management
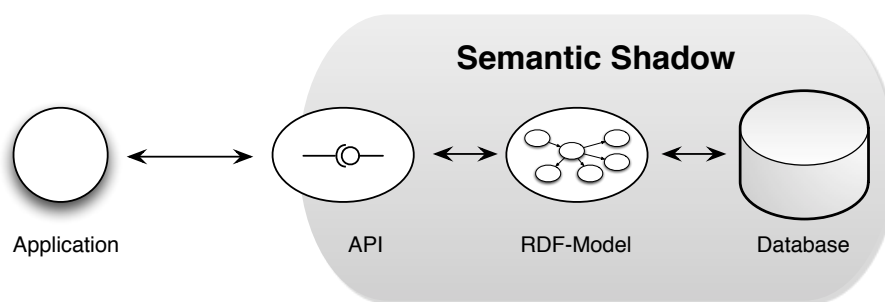
Figure 1.1: The application architecture view on the *Semantic Shadow*: Representation details are hidden from the developer.

1.5.3 Procedure to Optimize Web Based Service UIs for Specific Usage Scenarios

As the information of the *Semantic Shadow* is managed independently from the annotated web data, both information must be put together to optimize user requests. Therefore, an architecture using a web proxy (see Figure 1.2) suggests itself: The web proxy, which might be configured manually by the user or might be installed compulsive by the web service provider, intercepts the clients page request. It fetches the requested page, optimizes it based on the current usage context and information from the *Semantic Shadow* and delivers the result back to the client.

Annotations are used in a web proxy to adapt pages in a context-aware way.

In its easiest way, the current usage context is derived from the HTTP request parameters using local interpretation data or web services (see Section 5.2.3). A modular architecture of the proxy core allows the reuse of the main functionality in different usage scenarios.

An easy context representation can be derived from HTTP request headers.

1.5.4 Concept to Derive Context-Dependent Meta-Information on Web UIs from their Usage

To derive contextual semantics in web documents from their usage, four main levels of data gathering can be distinguished:

Annotations can be derived from web usage data in different granularity.

1. *HTTP request analysis*: Only the information stored in current web server log files is used to deduce web page usage.

2. *HTML form data analysis*: The form data, which is transmitted with a HTTP GET or POST request is used to infer information about the form's HTML elements.

3. JavaScript-*based user tracking*: Before delivering a web page to the client's browser, JavaScript code is injected which enables a logging server to track relevant client-side events. These imply mouse movings, element focus, key presses, scrolling etc.

4. *User tracking with extended hardware support*: By using data collected from camera images or eye tracking hardware, the most precise (but also the most intrusive) data collection is possible.

Figure 1.2: A web proxy intercepts the client's web request and opti-
mizes the server's response based on the *Semantic Shadow* data.

In this work, form
analysis and JAVA-
SCRIPT-based track-
ing is performed.

To extract the information used by the *Semantic Shadow* concept, a
JAVASCRIPT based tracking method in combination with HTML form
data analysis has been found to be sufficient and effective. The thesis'
prototype is based on the USAPROXY [Att08] implementation, primar-
ily developed for unintrusive web site usability studies, but also suit-
able for extended implicit interaction [Sch00] with web sites [AWS06].

Usage data ana-
lysis rises privacy
concerns.

In all cases, privacy issues have to be taken into account. The more de-
tailed information about a user's interaction with a web site is gathered,
the more private information might be exposed. When implementing
a system like presented in this research work on a commercial level,
the agreement of the user with each tracking data usage is obligatorily,
even if the data is later on only used in aggregated form.

1.6   Methodology

Methodology: Scien-
tific Reasoning, Agile
Software Develop-
ment and Empirical
User Observation.

In order to work out concepts and procedures to tackle the challenges
mentioned in the last section, Scientific Reasoning has been used: From
a formal description of the problem statement and a description of the
parameters and data available in the envisioned application environ-
ment, the solution has been deduced following Software Engineering
strategies. The prototypical implementation of the concept's frame-
work and demonstrators have been developed using techniques from
Agile Software Development, namely the Test First approach and a
User-Story based requirements definition. To evaluate the algorithm
inferring contextual weighted structural annotations from tracked user
data an empirical study using online interaction of volunteers has been
performed.

1.7   Thesis structure

A common understanding of some fundamental terms is required to follow the concepts and procedures presented in this work. These basic terms and their current reception in research and practice are examined in Chapter 2. Approaches to utilize the ideas of annotation and adaptation in the scope of web technologies are collected and commented in Chapter 3. This chapter has several focuses: Firstly, Semantic Web fundamentals are refreshed and approaches to include fuzzyness are presented. Secondly, the chapter looks at the ways how to adapt web usage to the user's context. Thirdly, current approaches to infer semantic information via web page analysis will be compared to each other and to this thesis' approach.

In Chapter 4, the concept of the *Semantic Shadow* is introduced by defining its theoretical foundation and a procedure to make the meta information on the web service UI elements accessible using a programming interface. A concept of deriving the meta information by analyzing the service usage is presented in Chapter 5.

To evaluate the concept of the *Semantic Shadow*, several applications have been developed based on a prototypical framework. These use cases are presented in Chapter 6, along with the results of their usage analyses.

In the final Chapter 7, the results of this work are summarized. Also further applications of the concepts and this work's contributions are outlined.

Chapter 2

Definition of Fundamental Terms

The concept of the *Semantic Shadow* is based on a number of other concepts. An interpretation of this research work has to be based on a good understanding of these terms, wherefore they are introduced in this chapter.

2.1 Context

While the notion of "context" has been one of the fundamental terms in software engineering in the last decade, its understanding differs widely among researchers and use cases. This is especially true since the advent of powerful smart-phones, equipped with a variety of sensors to determine position, device acceleration, orientation and so on. Making use of these sensor data is nowadays referred to as making the software "context-aware" (cf. already the definition in [SAW94]) and the summary of available sensors and their values is called "context". While this technology-driven definition is widespread (and also used throughout this thesis, see Section 2.1.2), other definitions are possible and have been proposed in research. The following section aims to summarize the main approaches, focusing on applications in mobile and pervasive computing. For an extensive discussion, including aspects not directly related to software engineering, see [Dou04].

Context can be defined in different ways.

2.1.1 Approaches to Define Context

In general *context* is referred to as a *set of circumstances or facts that surround a particular event, situation, etc.* In computer science a *context is the circumstances under which a device is being used,* e. g. the current occupation of the use.

Context generally is a *set of facts*.

Context as Abstract Notion

Context can be left abstract.

Referring to [Dey01], a couple of researchers like Philip J. Brown, Tom Rodden or Andy Ward just use the word *context* as a synonym for *environment* or *situation*. As this "renaming" is not related to a concrete representation of the context, the use of those definitions is limited.

Definition by Example

Naming examples of context parameters can implicitly define the notion.

In 1994, Bill N. Schilit et al. introduced the term *context-awareness* in their work [SAW94] and observed *context* as the changing parameters of hardware configuration and user properties like location, peer-groups, or changing social situations (see their visualization in Figure 2.1). They focus on "where you are, whom you are with and what resources are nearby". In particular, they name location, lighting, noise level, network connectivity, communication costs, communication bandwidth, and social situation (close-by peers).

Figure 2.1: Context and a context-aware system, as visualized first by Schilit et al. in [SAW94]

Context is defined by associating values with named domains.

This definition of context goes beyond an only location-based view on a system's world by considering "abstract" information which can affect the comportment of a system as well. A similar statement was introduced in 1999 by Albrecht Schmidt et al. when proclaiming "There is more to Context than Location" [SBG99]. In this approach, context is defined as associating specific values with one of the previously named domains (or more specific subdomains of them). The notion of "context" is detached from the individual perception.

States of Interest

Jason Pascoe defines in his work [Pas98] the "context" as a subset of physical and conceptual states of interest, defined by the entity that perceives it. This focus on the context of particular entities leads to the definition of context as a *world of artifacts* maintaining contextual states. These states consist of direct or synthesized *sensor-data*, which in turn is dependent on the associated artifact. To complete the definition of context, the artifacts of the world are interconnected using specified *relationships*. This allows to perform resource discovery by only observing the relationships between the artifacts, given that a complete set of relationships is maintained.

Context can be defined as states of interest.

Context as Shared Environment Information

Also Thomas Strang, Claudia Linnhoff-Popien and Korbinian Frank were targeting an autonomous service collaboration when designing their *Context Ontology Language (CoOL)* [SLF03]. It is built upon their ASC[1]-model where they define "context information" as *any information which can be used to characterize the state of an entity concerning a specific aspect*. The term "context" they define as *the set of all context information characterizing the entities relevant for a specific task in their relevant aspects*, whereas the relevance is based on the possible influences of the task. Finally, they define the term *situation* as *the set of all known context information*.

Context as set of all entity states concerning a specific aspect.

This definition is in line with the one of Anind K. Dey gives in [Dey01]:

Context characterizes the situation of an entity.

> 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.

With this definition Anind Dey tries to cover the context not only for a single entity, because he does not insist on the direct relevance of the information for the entity. Anind Dey covers the context for a complete application scenario: *If a piece of information can be used to characterize the situation of a participant in an interaction, then that information is context.*

Robert Hirschfeld et al. build in [HCN08] upon the definition of Anind Dey, when they treat "context" as *any information which is computationally accessible may form part of the context upon which behavioral variations depend.* While they discriminate between the *actor-, environment-,* and *system-domain* as source for context changes, the authors do not provide a dedicated model for context information, since for their use-case *existing object-oriented abstraction mechanisms are sufficient to model context.*

Any computational object can define a *context*.

---

[1]Aspect-Scale-Context

Context can extend the scope of a single entity.

Also Bill N. Schilit extended his definition of "context" in [Sch95, p. 4] from the entity orientated to a more global view: *Because users interact with a multitude of stationary and mobile systems through the course of a day, mobile systems need to maintain and share a context that is not limited to the boundary of process hierarchy, window manager, or host.*

Context is an abstract description of the real world situation.

In the scope of the concrete development of so called *Smart Appliances* [SL01], Albrecht Schmidt and Kristof van Laerhoven *use the term context in a .. general way to describe the environment, situation, state, surroundings, task, and so on.* In their project, they *define context awareness as knowledge about the user's and device's state, including surroundings, situation, and location.* To discriminate between the real world situation and their abstract description, they define:

- Situation or situational context: These terms describe the real-world situation. [...]
- Context or context knowledge: These terms describe the abstract description of the real-world situation.

This distinction between the "real world" and their abstraction can be interpreted as an answer to the indisposition caused by using "context" just as a synonym of "situation" as proposed by the authors cited above.

Context can be modeled graphically, as database tuples or as abstraction of a situation.

In [HI06] Karen Henricksen and Jadwiga Indulskab found *rich context models that integrate sensed, static, user-supplied (profiled) and derived information to be the most useful.* While especially taking into account the problem of imperfect context information and situations of ambiguity or unknown values, they propose three levels for modeling context data: A graphical model (CML) as extension of Object-Role Modeling (ORM) [HM08], a relational mapping of CML which leads to a representation of context information consisting as a set of facts expressed in the form of database tuples, and a situation abstraction to reuse and combine context information.

Context from a Social Settings Point of View

Context can be seen as a stable and definable form of information, separated from *activity*.

Until now, the definitions of context were mainly focused on the technical point of view. Paul Dourish summarizes this in [Dou04]:

- First, *context is a form of information*. It is something that can be known (and hence encoded and represented much as other information is encoded and represented in software systems)
- Second, *context is definable*. We can, for some set of applications or application requirements, define what counts as the context of activities that the application supports, and do so in advance.

- Third, *context is stable*. Although the precise elements of a context representation might vary from application to application, they do not vary from instance to instance of an activity or an event. The determination of the relevance of any potential contextual element can be made once and for all.
- Fourth, and most importantly, *context and activity are separable*. Activity happens "within" a context. The context describes features of the environment within which the activity takes place, but which are separate from the activity itself. [...]

In contrast to this list, Paul Dourish tries to call the researchers' attention also on the social aspects of context. He states that even though *by incorporating context, system designers have hoped to make their systems more responsive to the different social settings in which they might be used, [...] social and technical ideas often sit uneasily together*. Therefore, Dourish defines his own understanding of context as an interactional rather than a representational problem:

From a social point of view, *context* can be seen as a dynamic, occasioned relation arising from activity.

- First, rather than considering context to be information, he instead argues that *contextuality is a relational property* that holds between objects or activities. It is not simply the case that something is or is not context; rather, it may or may not be contextually relevant to some particular activity.
- Second, rather than considering that context can be delineated and defined in advance, the alternative view argues that *the scope of contextual features is defined dynamically*.
- Third, rather than considering that context is stable, he instead argues that context is particular to each occasion of activity or action. *Context is an occasioned property*, relevant to particular settings, particular instances of action, and particular parties to that action.
- Fourth, rather than taking context and content to be two separable entities, he instead argues that *context arises from the activity*. Context isn't just "there", but is actively produced, maintained and enacted in the course of the activity at hand.

Instead of asking "what is context [and how can it be encoded]?", like Albrecht Schmidt and Kristof van Laerhoven do in [SL01], he wants to know *"how and why, in the course of their interactions, do people achieve and maintain a mutual understanding of the context for their actions?"*. In this model, context *isn't something that describes a setting; it's something that people do. It is an achievement, rather than an observation; an outcome, rather than a premise*. By this, context gets a kind of "shared history" and

furthermore content, so, according to Paul Dourish, it is not possible to distinguish between context and content or "activity".

This notion can support system design rather than automatic adaptation processes.

This understanding of "context" does not lead to a scenario, where a context-aware application can react in a predefined way to the change from one known contextual state to another, but rather lets the system designer point out the question *how can ubiquitous computing support the process by which context is continually manifest, defined, negotiated, and shared?*. Paul Dourish therefore introduces the notion of "practice", the unification of "action" and "meaning", or context as "embodied interaction". *The essential feature of embodied interaction is the idea, as illustrated above, of allowing users to negotiate and evolve systems of practice and meaning in the course of their interaction with information systems. [...] The importance of context is not what it is but what it does in interaction – the role that it plays and the ways in which it is sustained and managed.*

Context in Use

Context can be evaluated focusing on quality or information use.

To evaluate the usage of "context" in context aware applications, Anand Ranganathan et al. propose in [RAB+04, RAB+05] to discriminate between *evaluating the quality of sensed or derived context information* and *evaluating the use of context information*. For this, they identified nine areas of context: *location, time, environmental contexts, informational contexts, user contexts, group contexts, application contexts, system contexts, and physical object contexts* (see Table 2.1 for some examples for these contextual domains)

Contextual domain	Examples
Location	
Time	
Environmental Contexts	temperature, light, sound level
Informational Contexts	stock quotes, sports scores
User Contexts	gaze, orientation, health, mood, schedule, activity
Group Contexts	group activity, social relationships and networks, other people in a room
Application Contexts	email received, websites visited, previous files opened
System Contexts	network traffic, status of printers
Physical Object Contexts	position or orientation of chairs and tables, properties of objects such as temperature and size

Table 2.1: Contextual domains named by [RAB+05]

Together with these contextual domains, Anand Ranganathan defines *context quality* in four dimensions:

Context quality depends on *confidence*, *accuracy*, *freshness* and *resolution*.

Confidence expressed as the probability that the context has been sensed or deduced correctly

Accuracy expressed as an error percentage of the sensed or inferred contexts.

Freshness measured as the average time between readings of a certain kind of context.

Resolution as the area within location information can be narrowed down to (room-level, building-level, etc.)

2.1.2 A Context Model for the Semantic Shadow

In the *Semantic Shadow* concept, "context" is used as parameter to describe the confidence of an annotation given a concrete user request (see Section 4.1.3). Therefore, "context" is required to be defined in a machine-interpretable way, such that it can be determined at the specific time of a user request, and describes the situation in which the request was performed. This work follows the definitions of [SAW94, SL01, RAB$^+$05] and others by in a first step associating a context as "basic context" to a given set of values:

In this work, *context* must be machine-interpretable.

> **BASIC CONTEXT:**
> A *basic context* $\mathcal{C}$ is a set of sensor-value pairs, (both represented by strings), i.e., $\mathcal{C} = \{(s_1, v_1^1), \ldots, (s_1, v_1^{m_1}), \ldots, (s_n, v_n^0), \ldots, (s_n, v_n^{m_n})\}$. Each sensor can have several values associated.

Definition: *Basic Context*

> **SENSOR:**
> A sensor is one well defined aspect of a situation. It is referenced by a unique name.

Definition: *Sensor*

On the semantic level[2], this definition of context is the same as used by Tobias Rho in his CSLogicAJ language [RSC06]. To give an example of such sensors, a number of them has been predefined in the *Semantic Shadow* concept (see Table 2.2).

SemS predefines some sensor names.

Since the number (and names) of possible sensors should not be limited by the concept, an open world is assumed meaning that more than the given sensor names could be encoded in a concrete context. If a sensor name is not explicitly represented with one or more values in a context, all sensor values are included in the context. This leads to the definition of context inclusion:

For the context sensors, an *open world* is assumed.

---

[2]By replacing the name "sensor" with "field"

Domain	Sensor	Description
Date and Time	`time:hour`	The hour of the current time.
	`time:minute`	The minute of the current times.
	`time:day`	The current day of month.
	`time:month`	The current month.
	`time:year`	The current year.
	`time:dayofweek`	The current day of week.
	`time:zone`	The time zone.
Location	`region`	The region (City, state, nation, ...).
Device	`browser:mobility`	The "mobileness" of the device.
	`browser:screendimension`	The screen dimensions.
	`browser:orientation`	The orientation of the browser's view.
Connection	`connection:speed`	The connection's speed.
User	`user:language`	The user's language.
	`user:age`	The user's age.
	`user:gender`	The user's gender.

Table 2.2: Examples for context sensors.

Definition:
Context Inclusion

> **CONTEXT INCLUSION:**
> A context C_1 is included in a context C_2 ($C_1 \trianglelefteq C_2$), iff for every sensor s in C_1 either all values of s in C_1 are stated in C_2, or no values for s are given in C_2.

Context inclusion $\neq$ subset operation

The open world assumption implies that the definition of context inclusion differs from the definition of the subset operation in set theory. While in set theory, $(A \subseteq B) \wedge (B \subseteq A) \leftrightarrow (A = B)$ for two sets A and B, for two contexts C_1 and C_2 $(C_1 \trianglelefteq C_2) \wedge (C_2 \trianglelefteq C1) \nrightarrow (C_1 = C_2)$.

Definition:
Context Equality

> **CONTEXT EQUALITY:**
> A context C_1 is equal to a context C_2 ($C_1 = C_2$), iff both contexts define the same values for the same sensors. Every sensor defined in C_1 must be defined in C_2 and vice versa.

This implies that $(C_1 = C_2) \rightarrow (C_1 \trianglelefteq C_2) \wedge (C_2 \trianglelefteq C1)$.

Contexts can be combined.

To simplify the definition and re-usability of context definitions, we extend the definition of context by the introduction of *Combined Context*.[3]

Definition:
Combined Context

> **COMBINED CONTEXT:**
> A *combined context* $\mathbb{C}$ is a set of contexts $\{C_1, \ldots, C_k\}$. The elements are called *subcontexts*. No subcontext must be C or contain C (i. e., no circular referencing is allowed).

---

[3]This approach is not as expressive as the situation abstraction of [HI06], but allows simpler implementation and evaluation at runtime.

> **CONTEXT:**
> A *context* is either a *basic context*, or a *combined context*.

Practically, a combined context $\mathbb{C} = \{C_1, \ldots, C_k\}$ can be interpreted as basic context by merging the sensor values of the subcontexts C_i ("flattening").

In some cases, sensor value subsets of a context are contradictory. E. g. for a simple definition of "summer", the location must *either* be in the northern hemisphere and the month in between June and August, *or* the location must point to the southern hemisphere and the month December, January or February. A simple merge ((Location: Northern hemisphere ∨ southern hemisphere) ∧ (Month: January ∨ February ∨ June ∨ July ∨ August ∨ December)) would not be correct. To express this, a combined context can be flagged as being *mutual exclusive*:

> **MUTUAL EXCLUSIVE CONTEXTS:**
> A *mutual exclusive context* $\bar{\mathbb{C}}$ is a combined context, for which the elements must not be merged together during flattening.

When applying the "flattening", this leads to k different variants of a mutual exclusive context $\bar{\mathbb{C}}$, if $\bar{\mathbb{C}}$ consists of k basic contexts as subcontexts. The results of this "flattening" is a set of basic contexts, also referred to as "Flat Contexts". Formally, any context C can be reduced to a set of basic context:[4]

$$flat(C) = \begin{cases} \bigcup_{C_i \in C} flat(C_i) & \text{if } C \text{ is mutual exclusive context} \\ \biguplus_{C_i \in C} flat(C_i) & \text{if } C \text{ is comb. cont., not mut. excl.} \quad (2.1) \\ \{C\} & \text{if } C \text{ is basic context} \end{cases}$$

For flattening combined contexts, $\uplus$ on two sets of basic contexts $C_1 = \{\mathcal{C}_1^1, \ldots, \mathcal{C}_1^k\}$ and $C_2 = \{\mathcal{C}_2^1, \ldots, \mathcal{C}_2^l\}$ is defined as:

$$C_1 \uplus C_2 = \{\mathcal{C}_1^1 \cup \mathcal{C}_2^1, \ldots, \mathcal{C}_1^1 \cup \mathcal{C}_2^l, \ldots, \mathcal{C}_1^k \cup \mathcal{C}_2^1, \ldots, \mathcal{C}_1^k \cup \mathcal{C}_2^l\} \quad (2.2)$$

A possible representation of the context in the *Semantic Shadow* concept is given in Figure 2.2. As the use of "context" in the *Semantic Shadow* concept is reduced to the context inclusion operation and referencing (as described in Section 4.1.3), the substitution of the model presented here by a more powerful one (as provided by the Semantic Space concept [WDC+04] or by one of the other works referenced in the previous section) is possible.

---

[4]For an implementation of the merging algorithm, see Appendix A

Context
sensorValues: Map<String,List<String>> subcontextsMutableExclusive: boolean
inContext(otherContext: Context): boolean equals(otherContext: Context): boolean
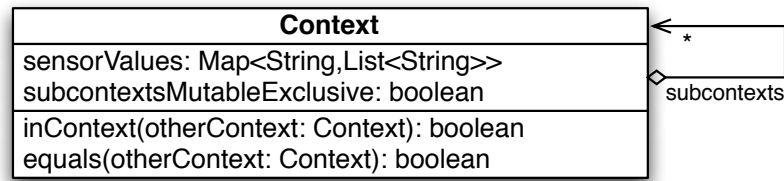
Figure 2.2: Model of the context as used in the *Semantic Shadow* concept.

2.2   User

Users are associated with individuals.

In this work, the "user" is associated with the person using a mobile device to consume information requested by him or her[5] using the device's software. In general, a mobile application providing the information serves more than one user.

Users have a *mental model* of system behavior.

Every user has some expectations of a system and its behavior. These expectations, named the *mental model* of the user, are most of the times not known explicitly by the serving system, but have to be inferred from certain human characteristics. [All97] lists:

Predictions: Users can predict what will happen next in a sequential process and how changes in one part of the system will be reflected in other parts of the system. However the most informative aspect of their predictions are often errors from the model.

Explanations and Diagnosis: Explanations about the causes of an event and diagnoses of the reasons for malfunctions reflect mental models.

Training: People who are trained to perform tasks with a coherent account (i. e., a conceptual model ...) of those tasks complete them better than people who are not trained with the model.

Other: Evidence is also obtained from reaction times for eye movements and answering questions about processes.

Adaptation requires a *user model*.

From these characteristics, a *user model* can be derived, which allows the computer system to distinguish one user from another by varying the model's parameters. The user model can be instantiated explicitly or implicitly [All97]:

Explicit Profiles: The user explicitly creates a profile of interests. However, users might not be able or willing to express their requirements in a compatible way.

Inferences from User Behavior: By collecting model-relevant usage data and by using certain interpretation assumptions, a user model can be inferred. This deduction might lead to wrong conclusions, but often this type of data has considerable value.

---

[5]In the following, pronouns are only used in the main grammatical form, including the other gender's form intentionally.

In the *Semantic Shadow* concept, the user is identified with his current usage context. Therefore, the user is modeled using the context representation. In the case of annotation usage (Section 4.2), explicit profiles are used by interpreting the user's HTTP request headers to construct a context object. When evaluating implicit web site usage data in Chapter 5 in order to infer structural annotations from it, the second approach of profile generation is applied.

Users are modeled by using context representations in *SemS*.

2.3 Adaptation

The capability of an object to adapt itself or being adapted to a changing environment is not only important in computer science, but as well in other scientific disciplines like astronomy [WAL⁺00] or evolution theory in biology. In contradiction to object adaptation in biology where adaptation is a natural process and research aims to understand this process, objects in computer science are synthetic so every adaptation process has to be designed and programmed. This programming can be inspired by nature [Ras86], but this is not the general case.

Adaptation is wide-spread in research.

Adaptation in Software Engineering traverses all stages of application design, from the object interaction level [GHJV95, p. 139] up to the business level [FC96], resulting in adequate structures like design pattern usage and service composition. In every case, adaptability has to be explicitly engineered into the software systems [FC96].

Adaptation is possible on all development levels.

In computer science, especially when designing for mobile environments [Kat94], four domains of adaptation can be distinguished:

Four main adaptation techniques can be distinguished.

- Content adaptation

- Adaptation of the internal object behavior

- Adaptation of object interaction

- Adaptation of the presentation/user interface

2.3.1 Content Adaptation

With content adaptation, the information delivered to the user is transcoded in order to satisfy the user in a given context (e. g. limited resources). If there is an information loss involved (e. g. transforming a video to a lower bitrate), the different requirements with respect to individual definitions of data fidelity [NSP95] have to be taken into account. Different frameworks for adapting a data flow to the contextual limited user resources have been proposed in the last decade, among others [YRP99] and [BBP04]. Figure 2.3 shows a typical architecture for adapting the content (e. g. scaling down, translating etc.) by the platform in accordance to a predefined user profile before delivering the data to the client.

Content adaptation transcodes information in context specific way.
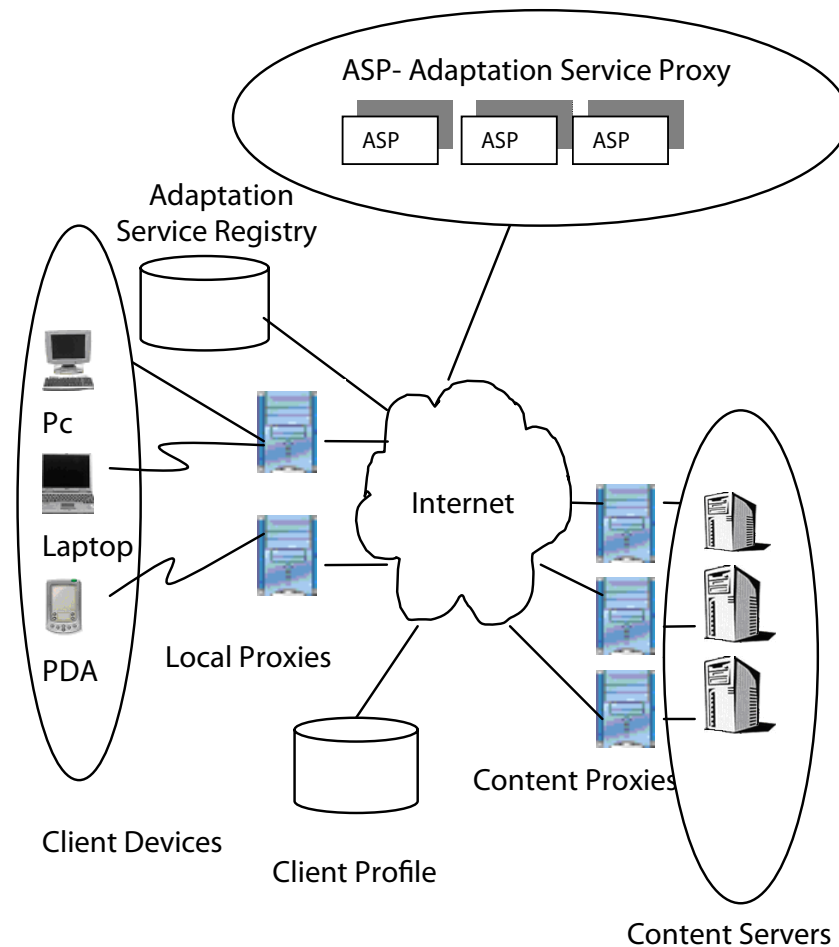
Figure 2.3: Distributed service-based content adaptation from [BBP04]

2.3.2   Adaptation of Internal Object Behavior

Internal object be-
havior has to be
adapted application-
specific.

Adapting the internal object behavior is a very application-specific way
to handle changing context conditions. The modification of a service
request with the current user location can be seen as such an internal
adaptation. Since context parameters might be imprecise, the quality of
a service can be reduced: *The user must be able to tolerate somewhat inac-
curate answers because these requests are inherently imprecise. This is because
the user's location and the current situation (...) are constantly changing, and
even terms like "near" are fuzzy concepts [Kat94].*

Adaptation need not
to be perceivable by
the user.

In other situations, the service and the output provided by the object
may stay the same, but it adapts it's behavior in reacting to e.g. less
power resources by slowing down calculation [FS99], or changing the
communication speed and wireless media consumption in function of
the count of participating radio stations [Wei93].

2.3.3   Adaptation of Object Interaction

Besides adapting the object behavior and its output, the interaction of an object with it's environment or other computing objects can be adapted in function of changing context parameters. Probably the best known example for an adaptation of interaction is the cellular phone handover: A mobile phone has to take the decision, by estimating it's application's context information (field strengths of the available transmitting stations) when to switch into another signal area and by that, when to switch into another interaction channel. While in GSM networks the base stations, and not the cellular itself, are operating the handover, the mobile stations make measurements of the radio link and report the results to their base stations in order to prepare for a handover decision based upon knowledge about the situation at the mobile [Man90].

In some situations the interaction between objects must be adapted.

2.3.4   Adaptation of the Presentation/User Interface

If neither the object's behavior, nor it's external interaction or information output can be adapted to a changing context situation, the user can still be supported by adapting the service's presentation: This ideally leads to a reduction of the number of constraints the user has to specify in his query or to a reduction of the number of actions the user has to take to get the desired information [RAB+05]. The adaptation has to be arranged carefully, so the user is not disturbed in his spatial memory by changing command positions [Gol10].

User Interface adaptation is the most intrusive form of adaptation.

2.3.5   A Taxonomy of Mobile Service Adaptation

In order to classify adaptation processes in software modules in mobile environments, Graham South et al. distinguish in [SLM00] the network element where the decision about adapting the response of a service takes place from the function of the mechanism in the adaptation process (see Figure 2.4).

Adaptation processes can be classified in two dimensions.

Criterion 1: Where decision on adaptation occurs
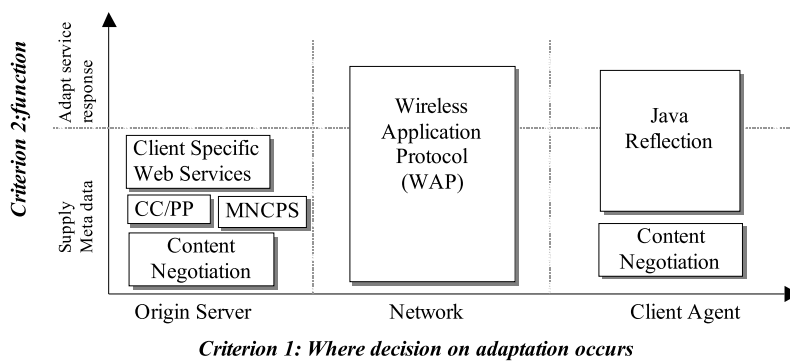
Figure 2.4: Taxonomy for adaptation mechanisms in [SLM00]

2.3.6   Adaptation using the Semantic Shadow

Adaptation in this thesis is focused on the user interface.

The adaptation strategies used in this thesis focus on adapting the output of a web server request, a web page, and thus can be categorized as *Adaptation of the presentation/user interface*. For this, the *SemS* annotations can be classified as *Meta-Data supplier* in the taxonomy of [SLM00] (see Figure 2.5), located either on the origin server or, in most cases, managed in a *Semantic Shadow* database inside the network. In correspondence with Section 4.2.1, the web page adaptation using *SemS* annotations can happen either on the client or in the network.
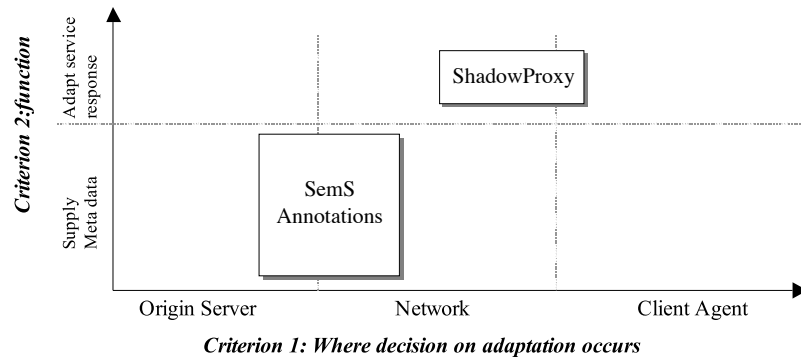
Figure 2.5: *Semantic Shadow* elements in the taxonomy of [SLM00]

Chapter 3

Related Work

In this chapter, selected research work which focuses on topics related to the *Semantic Shadow* concept is listed and discussed. In the first section fundamental definitions from the Semantic Web are refreshed and research approaches to include the notion of fuzziness into its value range are presented. In the second section methods and frameworks to introduce context-awareness and adaptability into the web usage experience are described and compared. Finally, current techniques to deduce semantic information from web page analysis are collated and are set side by side with this thesis' approach.

Here selected works focusing on topics related to *SemS* will be discussed.

3.1 Semantic Web

Tim Berners-Lee, one of the "inventors" of the World Wide Web (WWW), developed in 2001 [BHL01] the vision of the *Semantic Web*, "a new form of Web content that is meaningful to computers [and] will unleash a revolution of new possibilities". Its practical manifestation is built upon the concepts and languages RDF [CK04], RDF Schema [Hay04] and OWL [SWM04].

Tim Berners-Lee envisioned a Web with content meaningful to computers.

The *Resource Description Framework* (RDF) [CK04] is a general model to represent assertions about "subjects" using "objects" and "predicates". A *subject* can be any resource (material or immaterial: persons, files, concepts, ... i. e., anything) which is represented by an Unique Resource Identifier (URI)[1]. The *object* of an RDF statement can also be a URI-identified object, but as well a literal like a number, string or boolean value. The *predicate*, also represented by an URI, describes the kind of connection, which is expressed between the subject and the object.

RDF expresses facts in triples (subject, predicate, object).

Every RDF assertion (also called RDF-triple) is a distinct true statement in the Semantic Web space. Adding any other triples will not change

Triples are distinct in the Semantic Web.

---

[1]While the URIs of RDF subjects look similar to web page URLs, the connection is unidirectional: Every URL is a valid URI, but not every URI points to a valid web resource.

the meaning of the existing triples (i. e., no triple can express the absence of a statement). The *open world* assumption implies that nothing can be said about the truth of a statement not expressed in the model, i. e., if an assertion is not stated, it can not be considered as being false, but is rather undefined.

RDF triples can be visualized as graph.

An RDF model can be visualized as a graph: Every subject and object is represented by a node, while the predicates are illustrated using directed connections. The example in Figure 3.1 shows a triple expressing the fact that for a person "John", "Sleeping" is the preferred action.
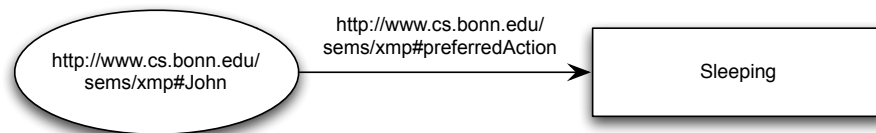
Figure 3.1: An example for the graphical visualization of an RDF triple.

RDF models can be persisted using e. g. RDF/XML or the Notation 3 syntax.

Beside this graphical notation, several textual representations for RDF models have been defined, within *RDF/XML* [Bec04] and the compact *Notation 3* [BL06] syntax. Expressed in RDF/XML, the same statement can be written as:

```xml
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xmp="http://www.cs.bonn.edu/sems/xmp#">
  <xmp:preferredAction
    rdf:about="http://www.cs.bonn.edu/sems/xmp#John">
      Sleeping
  </xmp:preferredAction>
</rdf:RDF>
```

The corresponding triple would be written in Notion 3 syntax like:

```
@prefix xmp: <http://www.cs.bonn.edu/sems/xmp#>.
xmp:John xmp:preferredAction "Sleeping".
```

RDF Schema extends the typing system of RDF.

Since RDF is by default limited in typing, the requirement of introducing an extended typing system suggested itself early and *RDF Schema* (RDFS) [Hay04] was defined. RDFS defines a set-based class model, using the following concepts:

Classes: rdfs:Class,   rdfs:Resource,   rdfs:Literal,   rdfs:Datatype, rdf:XMLLiteral, rdf:Property

Properties: rdfs:domain,   rdfs:range,   rdf:type,   rdfs:subClassOf, rdfs:subPropertyOf,   rdfs:label,   rdfs:comment,   rdfs:seeAlso, rdfs:isDefinedBy

This vocabulary allows the definition of primitive ontologies, i. e., specifications of conceptualization. "The term [ontology] is borrowed from philosophy, where an ontology is a systematic account of Existence. For knowledge-based systems, what 'exists' is exactly that which can be represented" [Gru93]. This representation expresses in computer science the knowledge about concepts, i. e., classes of subjects, their attributes, and their relationships. Every ontology models a limited part of the "knowledge world" and its file representation can be stored independently from other ontologies, which makes the concept flexible and distributable. If two or more ontologies are compatible with each other, i. e., use the same names for the same concepts, they can be merged using a reasoner algorithm. Using inference algorithms, further true statements about the modeled world can be automatically deduced from the given axioms.

An ontology can model concepts with classes, attributes and relationships.

RDF Schema is not expressive enough to model complex interrelationships, for which reason the *Web Ontology Language* (OWL) [SWM04] has been defined upon RDF. As in RDF Schema, classes can be declared in OWL, and these classes can be organized in a subclass hierarchy. Going further, OWL classes can also be specified using logical combinations (intersections, unions, complements). With OWL, the belonging of a given object to a class and its specific property values can be expressed. In its "Full" version, OWL subsumes RDFS, but is not decidable, wherefore two decidable subsets named "OWL DL" and "OWL Lite" have been defined. On the semantic level, OWL DL (and its subset OWL Lite) is based on first order logic, its description logic is named $\mathcal{SHOIN}(D)$.

Ontologies can be modeled in the Semantic Web using OWL.

An extensive introduction to the Semantic Web protocols and technologies is given by Pascal Hitzler et al. in [HHRS08].

3.1.1 Context

The last chapter introduced the term "Context" and presented several definitions for it, among them "Context as Shared Environment Information": Building upon their Aspect-Scale-Context-Model, Thomas Strang et al. introduced in [SLF03] their *Context Ontology Language* (CoOL) as an approach to include the notion of "Context" in the Semantic Web (see Figure 3.2). In order to perform well in the disciplines of knowledge representation and knowledge querying, CoOL has been designed as collection of several fragments: the *CoOL core* (a projection of the context model into OWL and other ontology languages), and the *CoOL Integration* (schema and protocol extensions to include CoOL in Web Services and other frameworks). The latter indicates the orientation of Strang's approach, which is directed towards (web) service interoperability and dynamic composition, thus extending the *ontology of services* DAML-S [ABH+01].

Strang et al. introduced CoOL to include "Context" in the Semantic Web.
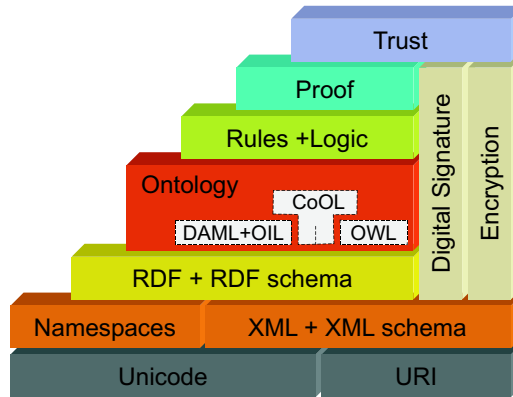
Figure 3.2: Strang et al. embed their CoOL in the Semantic Web [SLF03].
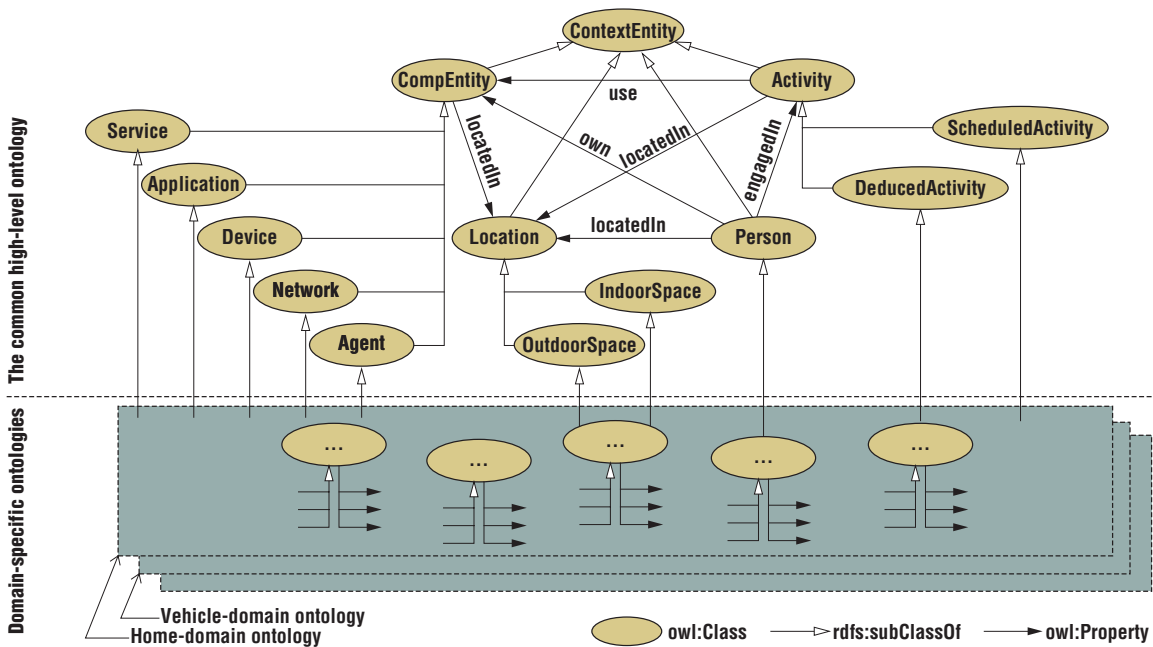
Figure 3.3: Class hierarchy diagram for the context ontologies of [GPZ04].

Gu et al. define a formal OWL based context model, which allows reasoning.

In [GPZ04], Tao Gu et al. describe a service-oriented infrastructure for context-aware pervasive applications. In its core, they define a formal OWL based context model, which allows semantic context representation, reasoning, and knowledge sharing. Since the definition of "context" is always, at least partially, application-specific, the authors propose a separation between a common high-level ontology and domain-specific ontologies (see Figure 3.3). In the high-level ontology, general information about the physical world (person, location, computational entity, and activity) is modeled, while the domain-specific ontologies are designed to define the application details. Based on this formal description, a *context interpreter* is able to reason high-level context information from low-level sensor data. Context evaluation and questioning can be performed using OWL or RDFS reasoner.

Paolo Bouquet et al. approach in [BSS05] the challenge of modeling context in an RDF database by preserving standard RDF within a context and adding context semantics and relations between contexts around the standard. Their focus was not to facilitate the interoperability of services, but to persist context information which is required to understand a document. From this point of view context is a second dimension to the information stored in the RDF database since it might be relevant for every given assertion. For this, they do not rely on parameterized statements or extend binary predicates to ternary predicates for identifying the context to which a statement belongs, but apply the principles of Locality and Compatibility [GG98]. The basic idea is to have all statements that belong to a context in a separate named RDF graph, and extend the RDF semantics to enable contexts to appear as standard objects in RDF statements of other contexts. To allow for reasoning across contexts, they envision *Compatibility Relations*, which plug into the system as *semantic attachments*.

Bouquet et al. model context in the RDF in order to persist the document creation context.

3.1.2   Fuzziness

As stated before, expressions in the open world of the Semantic Web classically only define a true statement. When setting statements (in the *SemS* framework annotations) in context, it might happen that the expression is not completely true, but rather something fuzzy in between true and false (see [Zad65]), or it is only true with a certain propability. Several approaches have been made to extend Semantic Web technologies in a way that allows to encode this fuzziness directly and therefore enable a Semantic Web reasoner to handle those fuzzy facts.

Plain RDF cannot define statements as being partly true.

Zhongli Ding and Yun Peng propose in [DP04] to incorporate Bayesian networks into OWL by augmenting the language to allow probabilistic markups. These markups are translated into a directed acyclic graph of a corresponding Bayesian network, which can in turn be used for inference procedures. Mingxia Gao and Chunnian Liu extend in [GL05] this idea to include any kind of uncertain knowledge or vague concept by encoding fuzzy constructors, axioms and constraints. These new fuzzy terms are mapped to a fuzzy description logic. Using a constraint propagation calculus, the extended OWL can directly resolve fuzzy inference questions. Stamou et al. sketch in [SPTH05] the assertion of weighting factors to the statements of SWRL [HPB+04] in order to allow fuzzy statements. In the next step, they include in [SST+05] fuzzy knowledge into the Semantic Web by extending the $\mathcal{SHOIN}$ DL, thus creating a fuzzy OWL. With FiRE [SSSK06], they provide a prototype implementation for a fuzzy reasoning algorithm.

To include fuzzy facts in RDF, new markups might be introduced.

In an approach to provide a standard compliant way to express fuzzy information in the Semantic Web, Fernando Bobillo et al. present in [BS09] a general extension for OWL 2 [GHM+08], building upon their earlier work [BDG06]. In their OWL 2 extension, uncertain information is represented by adding the affected individual instances to an OWL container instance, to which an instance expressing the "uncertainty"

Alternatively, OWL 2 can be extended.

information in form of an uncertainty type and value is added as well[2]. As advantages of their approach, the authors claim that fuzzy OWL ontologies may easily be shared and reused according to the specified encoding; the ontology could easily be extended to include other types of fuzzy OWL 2 statements; current OWL editors can be used to encode a fuzzy ontology; and it can easily be translated into the syntax of other fuzzy DL reasoners (they provide two open-source parsers that map their fuzzy OWL 2 statements to FUZZYDL [BS08] and DELOREAN [BDG08] statements.

The general problem has been discussed in an W3C Incubator Group.

The *W3C Uncertainty Reasoning for the World Wide Web Incubator Group* [W3C08] discusses broadly the question of how to model uncertainty in the context of WWW and the Semantic Web. Their report *identifies and describes situations on the scale of the World Wide Web for which uncertainty reasoning would significantly increase the potential for extracting useful information, identifies methodologies that can be applied to these situations and the fundamentals of a standardized representation that could serve as the basis for information exchange necessary for these methodologies to be effectively used, includes a set of use cases illustrating conditions under which uncertainty reasoning is important, [and] provides an overview and discusses the applicability to the World Wide Web of prominent uncertainty reasoning techniques and the information that needs to be represented for effective un-*

Uncertainty might be inherent in the data or in the process.

certainty reasoning to be possible. In particular, they summarized that two different kinds of uncertainty occur in the context of the Semantic Web: *Uncertainty inherent in the data* and *Uncertainty reasoning for the processing and presentation of information*. For the first case, they propose an inclusion of the uncertainty information into the fact representation, e. g. using an uncertain extension of OWL. The second case turned out to be much more vague, with the prospect to a "very extensive representation task".

More discussion in [LS08].

Thomas Lukasiewicz and Umberto Straccia provide in [LS06, LS08] an overview over different approaches to integrate uncertainty and vagueness into the Semantic Web.

3.2   Context Aware Web Interface Adaptation

3.2.1   Selected Adaptation Approaches

Adaptation of presentation can be distinguished from content-adaptation.

One application of the *Semantic Shadow* framework is the usage of *SemS* annotations for context-aware dynamic web page adaptation. Context-aware adaptation of web UIs can be roughly categorized into three categories:

Adaptation of Presentation   The layout and visualization style of the web page is adapted to the current context, e. g. the limited screen size or visualization capabilities of a mobile device.

---

[2]This is similar to the simplified approach *SemS* is using in Section 4.1.3 which expresses the uncertainty using RDF-triples.

Adaptation of Content The content of the web page is modified in order to meet the requirements of the usage context. This might include a degradation of the content in favor of respecting certain quality of service aspects, e. g. a video might be scaled down to a lower resolution in order to be transferred using a low fidelity wireless link without interruption.

Mixed adaptation The adaptation approaches in this category might modify the presentation as well as the content.

The distinction between "content" and "presentation" is not always very sharp: Changing the font color of a web page to support a visually impaired user might be seen as an adaptation of presentation only, but if the text refers to the color (e. g. "The red words in this sentence are nouns."), the presentation becomes part of the content and its adaptation might be categorized otherwise.

This separation is fuzzy.

Mohd Farhan Md Fudzee and Jemal Abawajy present in [MA08] a more detailed classification of (content) adaptation strategies: They distinguish between where to perform the (content) adaptation *(locality)*, who should perform the adaptation *(strategy)*, why to perform the adaptation *(purpose)* and to what to adapt *(context)*. Focusing on what is to be adapted *(mechanism)*, the question how to adapt *(method)* can be raised (see Figure 3.4). With these categories, the authors span a classification room:

Adaptation can be classified using the categories Locality, Strategy, Purpose, Context, Mechanism, and Method.

Locality Centralized (client side, server side, proxy side), distributed

Strategy Responsibility for adaptation at the underlying system, at the application, at both sides

Purpose General, content type specific

Context Device centric, user centric

Mechanism Appearance adaptation, file format adaptation, characteristic adaptation, encapsulation adaptation

Method Transcoding, layout rearrangement (column layout transformation), distillation, decomposition (text-unit based, block based, unit of information based), fragment generation.

While web page adaptation can be helpful as well in other use cases [AT00], most approaches focus on the use for mobile web access [CS95, LL02, Zha07]. Timothy Bickmore and William Schilit introduce in [BS97] a software called "Digestor" to automatically reauthor arbitrary documents from the world wide web to display appropriately on small screen devices such as PDAs and cellular phones. The authors performed a study to assess the characteristics of typical web pages and to identify candidate reauthoring techniques through the process of reauthoring several web pages by hand. They learned that keeping at least some of the original images is important to maintain the look and feel of the original document. Additionally, images cannot simply be scaled down by a fixed ratio, since important text might turn illegible. Aesthetic images and sidebars have only a minor semantic role, so these can simply be stripped from the pages. Whitespaces and

Most web adaptation scenarios envision a mobile client.

The images are important to identify a web page.
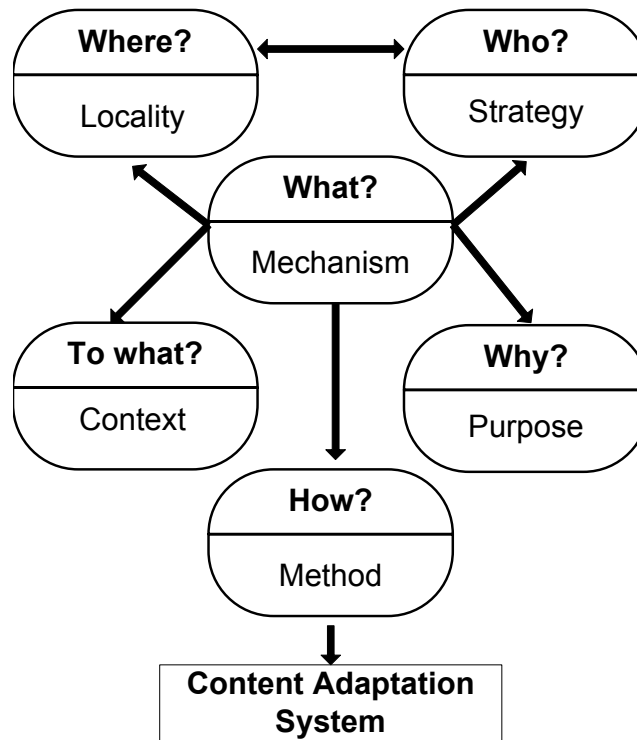
Figure 3.4: Content Adaptation System Classification of [MA08]

Whitespaces, ban-
ners and link lists
might be stripped.

elements from certain categories like *banners* or *link lists* are also candi-
dates for reductive adaptations. Timothy Bickmore and William Schilit
also found that it is not possible to rely on the semantics of HTML tags,
since H-tags for instance are not only used for marking headers (and
therewith structure the document), but commonly also for selecting
different font sizes. In the implementation of their "Digestor" system,
the authors provide a collection of individual reauthoring techniques
(outlining, first sentence elision, image reduction and elision) and an
automated reauthoring selecting the best combination of techniques
for a given document-display size pair. Since the system did perform
well for PDA sized devices but not for the mobile phones of that time,
the researchers introduced in [STHK01] the *m-Links* infrastructure to
adapt web content and services for wireless phones and small Internet

For small mobile
phones, the UI fo-
cuses on link lists.

terminals.   The *m-Links* system focuses especially on web-interaction:
A list-oriented "skeleton view" showing only navigational and "data-
detected" links (phone numbers, addresses, . . .) is presented, providing
"contextual actions" (send by mail, fax, show text) for final documents.

A web page can be
summarized in a
link list to shorten
navigation time.

Orkut Buyukkokten et al. similarly approach in [BGPW00, BGP00] the
problem of bandwidth and I/O limitations of PDAs by focusing on fast
navigation when adapting web pages. They observed that by simple
HTML-downscaling, fundamental differences in user interactions with
small screens are not considered, especially scrolling issues, the lim-
ited view on the page content and the time consuming input modali-
ties. They presented a *PowerBrowser* application, which displays upon

a page visit firstly a distilled view ("summary") of the page, a tree view containing all links present on the web page (using a simple heuristic to create "good" labels for the links). This is meant to shorten navigation time, since the user has to switch to the full text view only at the end of his page navigation. For the text view, all images are stripped from the page, sequences of white spaces are collapsed and most text-attributes are ignored. This is combined with a local site search and a keyword completion feature.

Presenting a web page on a mobile device as close as possible to the desktop view was the goal of Yu Chen et al. when designing their framework [CMZ03]. They proposed an auto-segmentation based on page analysis together with a primary "thumbnail view", allowing to navigate into PDA-optimized subviews of the web page (see Figure 3.5). When it is not possible to extract and reauthor the relevant part of the website, the authors propose an auto-scrolling to the relevant page section while displaying the complete webpage. This is very similar to the web page visualization technique of modern mobile smartphones using Mobile Safari on iOS[3], Chrome Lite on Android[4], or Opera Mobile 9.5[5].

Page layout can be preserved on a mobile client when working with thumbnails and supported zooming resp. scrolling.
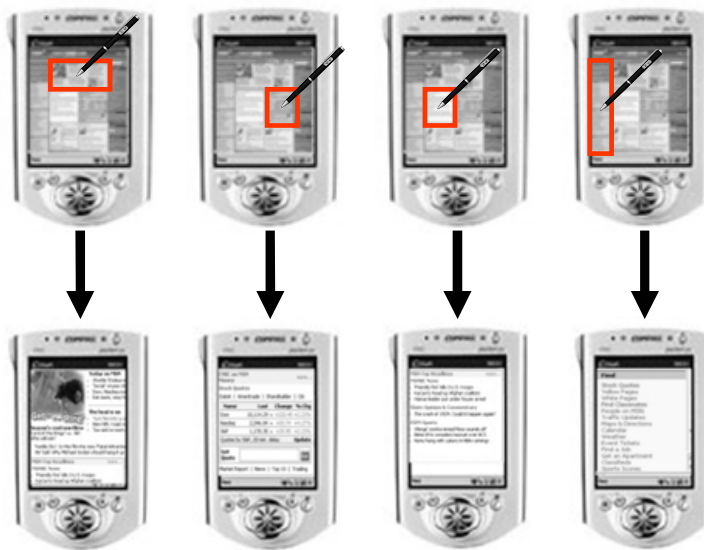
Figure 3.5: A two level navigation as a way two adapt web pages for mobile devices, as proposed by [CMZ03]

Another proxy system, adapting web page layout and content together to serve WWW information for mobile clients, is presented by Timo Laakko and Tapio Hiltunen [LH05]. The design principles for their adaptation algorithm are *functional user experience*, *extendibility*, *decomposition*, *structure preservation*, *performance* and the *delivery context*. For this delivery context, information from user preferences and user agent

Adaptation context can contain user profiles in addition to device classes.

---

[3]http://www.apple.com/iphone/features/safari.html
[4]http://developer.android.com/reference/android/provider/Browser.html
[5]http://www.opera.com/mobile/

profiles (UAProf [Wir01]) as well as general information about the device class are used. As further input, only the source of the web page and no additional meta information is used.

FAÇADE describes adaptation context on device, network and user level.

The framework introduces additional adaptation tags for XHTML.

In [KPG04], Bernd Kurz et al. introduce a "FrAmework for Context-aware content Adaptation and DElivery (FAÇADE)", which is designed to act "as an adaptive shell through which the mobile user interacts with the changing environment, always in an optimal way." The authors consider an adaptation context on three levels: *device characteristics*, *changing network conditions* and *user preferences*. To improve the adaptation process, several additional tags for XHTML are proposed: *Allow adaptation* (yes/no), *content relevance* (mandatory or optional, context specific relevance), *priority level and role* (semantic vs. esthetic), *adaptation unit specification*, *pagination*, *navigation*, *inclusion/exclusion tags*, and *selection tags*. The adaptation itself uses transformations being *semantic* (context-aware content selection, text and media elision, content inclusion/exclusion, and relevance and priority markup), *syntactic* structural and media transformations) and *esthetic* (text layout changes, pagination and navigational structures, and color contrast adjustments). The MIMOSA framework of Delfina Malandrino et al. [MMR+10] extends the idea of FAÇADE by aggregating different context sources, solving conflicts in context interpretation and providing a modular interface for complex adaptation services.

3.2.2   Context-Aware Web Applications

Context awareness can be defined and managed outside a web application.

Michael Hinz et al. present in [HPF07] a component-based extensible framework for modeling context information, that can be effectively used for adapting web applications. The framework supports ubiquitous web scenarios, such as location-aware, device-aware, and personalized web applications. The sensing, processing, and representation of context information is provided by the framework, while the web application providers have to maintain the content and its adaptation.

Sensor data is aggregated using modeling components and persisted in a context model.

The framework itself is split into three layers (see Figure 3.6): The *sensor components* are device based and take care of gathering the sensor data to characterize the user's context. The data is processed by the extensible layer of *context modeling components* in order to discover implicit information and represent the information in a generic way. The *context model* itself persists the data, where a description using RDF vocabulary is considered.

The context model is organized using specific profiles.

The context model consists of individual profiles, namely *a device profile, a location profile, a user profile (identification profile, preferences profile), a session profile*, and *a long term profile*. As proof of concept, the framework has been coupled with the AMACONT adaptation queue [FHMW03] and the generic transcoding tool [FH05] building upon AMACONT.
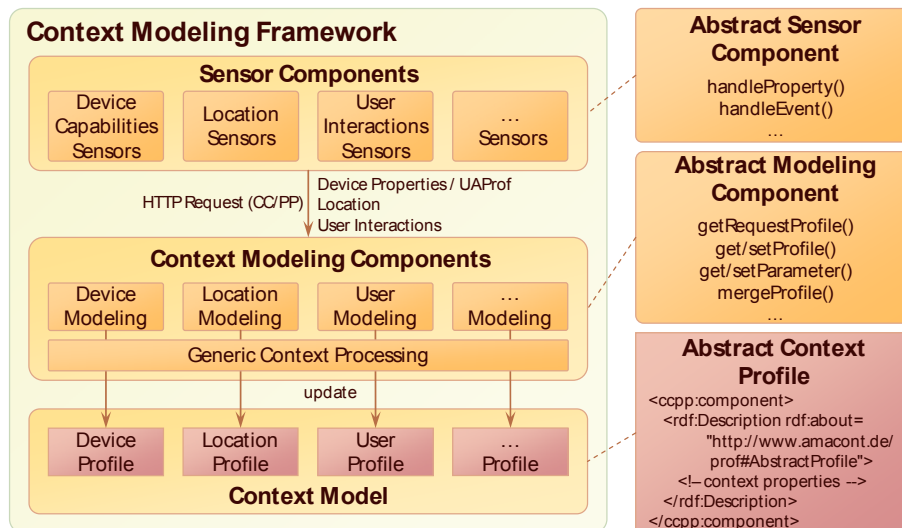
Figure 3.6: Overview of the context modeling framework of [HPF07].

Since a model representation in RDF is considered, referencing of the modeled context objects from other RDF documents is possible. Thus, an extension of the *Semantic Shadow* concept to incorporate this framework seems realizable (see Section 4.1.3), if an implementation of context inclusion quering (Section 2.1.2) is provided.

This framework might be used with the SemS concept.

3.2.3   Annotating HTML documents

When annotating elements of an HTML page, three different kinds of annotations can be distinguished:[6]

1. Annotations as tool for assigning (informal) notes to web pages.

2. Annotations describing *the content* of the annotated element (i.e. some role, meaning etc.).

3. Annotations describing *the structural semantics* of the annotated element (i.e. grouping, priority in comparison to other elements etc.)

An "annotation" on a page element can be informal, related to the content or related to the structure.

Note Assignment

One way of "annotating" a web page is to assign visible remarks to the specific web page (or parts of it). Examples for tools/concepts supporting this kind of annotations are Yawas [DV00] and CritLink [Yee98] (an early vision of a web log ("blog") system).

Visible remarks on a web page are one form of annotations.

The Annotea [Koi05] framework incorporates the idea of interlinking these annotations with the Semantic Web: The framework uses RDF Schema to describe templates to be filled by the author of the annotations, and RDF descriptions can be used as annotation values.

These remarks can be structured using templates.

---

[6]For an overview of semantic annotation approaches, see [Rei06]. For an example of using semantic annotations to display enhanced web pages, see [Sch09].

SemS annotations can be used to manage such remarks.

While the *SemS* framework was not originally designed with the page remark use case in mind, it is easy to imagine a new annotation type extending the current contentual annotation types. With an appropriate user interface, collaborative web page marking, as presented by the named approaches, can be implemented using the *Semantic Shadow* technology.

SHOE

Direct annotation approaches change the web page source by introducing new tags.

The SHOE (Simple HTML Ontology Extensions) language [HHL03] is one of the first languages to design and use ontologies for web pages. For this, SHOE allows to connect parts of a web page to an ontology by marking ("annotating") these parts with special HTML-tags. In this, it is similar to other approaches like Ontobroker [FDES98] (extending HTML-anchor tags), WebKB [ME99] (Conceptual Graphs (CG) embedded in KR-Tags and reuse of the `alt` field in `img` tags), the XHTML additions of FAÇADE [KPG04] and Microformats/RDFa [HH09, BA08, Hau09].

This requires access to the original file.

Since SHOE does not construct a direct relationship between the new tags and the original elements of the web site, SHOE annotations are not annotations of these web page elements in a strict sense. In addition, SHOE does not provide a model for embedding context-awareness into the annotations and requires a modification of the original web page, like the other approaches embedding annotation data directly into the page source.

CREAM

CREAM describes the semantic role of page elements with structured metadata.

Siegfried Handschuh et al. describe in [HSM01] the *CREAM* (CREAting Metadata) framework, a system to annotate existing and newly created web pages with metadata. They focus on using annotations to describe semantically what is being represented by web page parts (referred to in this paper by the term "contentual annotations"). For this, they extend the traditional approach of assigning plain text metadata by the definition of relational metadata, based on a domain ontology. To create and manage the annotation data, a component-based, ontology-driven tool *Ont-o-Mat* was developed.

The metadata is organized using Semantic Web ontologies.

Using an ontology-based approach, the authors aim to provide *consistency*, *proper reference*, *less redundancy*, *mutual relations*, *maintainability*, *ease of use*, and *efficiency* for the annotated data. Using the provided tool, any selectable part of a website can be assigned to an instance of an object from a suitable Semantic Web ontology. To integrate with the existing knowledge of the Semantic Web, a Semantic Web crawler is used. Data integrity is assured using a central storage server.

After one year, the authors extended their approach by adding web page editing features to their annotation creator, thus providing a tool to write web pages and the associated annotations alongside [HS02].

In comparison to the approach presented in this thesis, CREAM and the analogous SMORE framework [KHPG06] do not focus on the creation of structural annotations (whereas they might still be instantiated if an appropriate ontology is used). In addition, the frameworks do not consider the context dimension for the data annotation, thus context-awareness has to be implemented on a controller level. Nevertheless, the idea of going beyond plain text as content of contentual annotations can extend the simple approach which will be employed in Section 4.1.1.

The approach does not consider to describe the page structure or to integrate context.

Hori et al.

In their works [HKO+00, HOAK04], Masahiro Hori and his colleagues from IBM sketch a system designed to use external, RDF-based annotation of web page elements to dynamically optimize web pages for mobile web use. They introduced the referencing of web page elements by using XPath and XPointer expressions in RDF subjects. The annotations are created using a dedicated authoring tool. Using web proxy technology the annotations are interpreted upon mobile request to create an optimized web page. To perform this transformation process, they envisioned annotations to describe *alternatives*, *splitting hints* and *selection criteria*, and published this as W3C note [HMMS99].

Hori et al. introduced the idea of referencing web page elements from RDF using XPointer.

Alternatives are hints for content adaptation and encoding, where the appropriate alternative visualization of the content should be chosen based on the client's capabilities.

They propose annotations to describe alternatives, splitting hints and selection criteria.

Splitting Hints are general information about the *grouping of elements*. In their framework, they are used to split up web pages exceeding the client's visualization capabilities into several appropriate parts.

Selection Criteria are semantic descriptions used in Hori's framework to decide upon an element to display on a mobile device. Besides stating required client capabilities, annotations for describing the element's *role* and its *importance* relatively to other elements of the same page are definied.

The *context* the framework of Hori et al. are taking into account, is primarily defined by the requesting device's capabilities, to which a requested web page is adapted based on the static annotations. These possible capabilities are reflected in the capability description of the *selection criteria* annotation. Further incorporation of other context information is not specified by Hori. The research found its way into the IBM WebSphere Transcoding Publisher product [BBB02].

For the adaptation, they consider device capabilities only.

Nagao et al.

Another approach to construct a semantic super-structure for web page adaptation is presented by Katashi Nagao et al. in [NSS01]. The main idea is to provide any web user the option to extend any element of any web page with *linguistic*, *commentary* or *multimedia annotations*. In a

[NSS01] use external annotations for translations, commentating and video summarization.

similar manner as Hori in [HKO$^+$00] and as the *Semantic Shadow* framework presented in this thesis, the authors refer to web page elements using URLs and XPath pointers, extended by document hash codes. As main adaptation, the presented system supports annotation-based video summarization, language translation, and speech synthesis of documents including images.

For translation, they assign grammar roles to elements of a sentence.

Linguistic annotations indicate a semantic structure of textual elements to an adaptation component (Transcoder). They can be used to develop content-based presentation, retrieval, question-answering, summarization, and translation systems. Examples for such annotations would be sentential unit, noun, noun phrase, verb, adnoun or adverb, and adnominal or adverbial phrase. In addition, binary relations (e. g. agent, patient, recipient) or rhetorical relations (e. g. cause, concession) can be specified.

For commentaries, they allow to model HTML overlays.

Commentary annotations are meant to help the transcoder with the manipulation of nontextual elements such as images and sounds. These annotations can be compared to notes (see above) and are presented by default as an overlay upon mouse hovering (see Figure 3.7). These overlays are supposed to contain text, images and links.
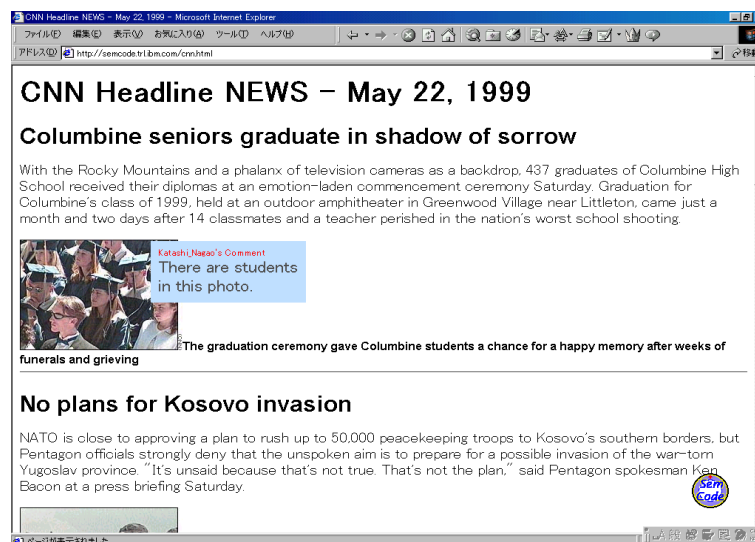

Figure 3.7: Comment annotations of [NSS01] presented as overlay.

To summarize videos, they combine overlays with additional metadata.

Multimedia annotations combine the features of linguistic and commentary annotations. They are mainly used to annotate video sequences with content descriptions, which in a second step can be summarized by the transcoder.

They support access rules and editor tracking for annotations, but design specifically for the given use cases.

While Nagao et al. describe a system to manage external annotations on web pages that is quite complex and provides support for access rules and editor tracking, it aims somehow for different objectives than the *Semantic Shadow* framework. As the framework presented in this thesis provides a general system for context-aware web page element

annotation without specifying dedicated use cases from the bottom up, the system of [NSS01] is clearly designed to support three different "adaptation" processes. For each of these "adaptation" processes a specific annotation type is sketched while a common foundation of all categories is difficult to discover. In addition, the choice not to use a common language for semantic information storage like RDF [KC04] excludes the data from being used with Semantic Web analyzing and reasoning tools [W3C10]. Also, even though the framework aims for a personal adaptation using user profiles, no context dimension is available in the annotation model.

3.3   Inferring Semantic Data from Web Pages

As discussed in the first chapter, creating annotations for existing and newly created web pages is a laborious task, hence for a big number of documents unrealistic. While the idea of this thesis to generate those annotations based on user interaction (see Chapter 5) has only been discussed in the final remarks of [Att08] and not been implemented before, other approaches to automatically create annotations based on a page text or DOM analysis have been presented.

Current approaches to infer semantic data from web pages focus on text and DOM analysis.

In [DEG+03], Stephen Dill et al. present *SemTag*, an application to tag automatically large text inputs. They intend to bootstrap the process of generating data for the Semantic Web by providing 434 million semantic tags generated from the analysis of 264 million web pages. In three passes (Spotting, Learning, Tagging) the algorithm seeks for web page sequences to label, resolves ambiguities using a "Taxonomy-Based Disambiguation (TBD)" and stores them into a tag database. While the analysis is fully automated, only the text of the web sites is used as input for the seeking engine. Going one step further, the approach of Saikat Mukherjee et al. [MYR03] also considers spatial locality and consistence in presentation of the elements in question: A "semantic structure" of the document is derived from its presentational definition and its content is then set into correspondence with domain ontologies and lexical databases. This "semantic partition tree" can then be used to enrich the original document with semantic annotations revealing the observed concepts to external systems. In continuation, the authors refined their approach in [MRS05] to use statistical analysis for automated concept identification. A similar idea was already pursued by Mark Craven et al. in [CDF+98], where the authors trained a system to classify web pages and extracted relations from them using a simple ontology.

SemTag generates meta information by comparing text fragments.

Spatial locality and consistent presentation can be used to identify web page elements with a common role.

Yu Chen et al. focused on small device web site presentation, when developing their algorithm for page structure analysis [CMZ03]. Like in the approach of Saikat Mukherjee, only the grouping of elements is defined as a semantic structure: The page is iteratively segmented into smaller content blocks, until they are suitable for an extracted view on a mobile device.

In order to generate a segmented view of a web page coherent element groups must be identified.

Chapter 4

The Semantic Shadow

4.1 Annotations for Contextual Semantics

The main language to describe information on the web used on current web sites is HTML [RHJ99] (or its XML-compliant variant XHTML [Pem02]). Being a markup language, the goal of the HTML-tags is to describe in a very simple way the semantics of certain parts of a text document. It is up to a displaying application, generally referenced as "browser", to interpret these semantic markups and to display the content appropriately.

The HTML language is designed to describe the semantics of text fragments.

Whereas at the development time of HTML the correct visualization of scientific research texts and, later on, the display of multi-media information on a desktop PC was primarily in focus, the requirements today are by far broader and the web is used in very different ways and on various devices. Especially barrier-free access for impaired users [AT00] and the visualization of web pages on mobile devices with reduced processing power and very limited display capabilities [CMZ03] have been in research focus during the last decade. Furthermore, the increasing mobile use of web resources makes the users aware of the possibilities of context dependent service adaptations, which not only makes sense for location information, but for various contexts on the mobile [SBG99] and beyond (e. g. [Dey98, Dou01]).

The WWW and its usage changed since the introduction of HTML.

To enable automatic adaptation, further semantic information about the web site's content and its structure than given by traditional HTML is required. This information, which can be expressed as an annotation to the web site elements, can either be integrated directly into the HTML page code [W3C07] or stored in a parallel structure [HKO+00, NSS01] (see also Section 3.2.3).

To make the web context-aware, further semantic information must be managed.

4.1.1   Semantic Annotations

Semantic annotations describe the content or the structural semantics.

Regardless of the way web element annotations are stored, two different kinds of semantic annotations can be distinguished:

1. Annotations describing *the content* of the annotated element (i. e., some role, meaning etc.).

2. Annotations describing *the structural semantics* of the annotated element (e. g. grouping, priority in comparison to other elements).

In the following, a base set of annotations, which can be used for various adaptation processes (see Chapter 6) will be sketched.

Contentual Annotations

Contentual annotations assign a *role* to a page element.

The contentual annotations describe the contents of the marked information. These annotations can be interpreted as describing the "role" which the content plays, like the approach of [W3C07] suggests. In the context of the Semantic Web [HHRS08] initiative, the term "annotation" traditionally refers to these contentual annotations [Ins04]. Since there has been research on this topic through the last decade [Rei06], very different roles of web page elements have been proposed (see for example [HMMS99, MRS05, CFB00, DCM08, W3C07, Att08]), within them:

- impress
- license
- address
- login
- phone
- logo
- empty
- icon
- search
- help
- contact

- price
- photo
- author
- main
- seealso
- secondary
- decoration
- ornamental
- category
- hot-deals
- firstName

- lastName
- birthDate
- birthPlace
- related-news
- biographicInfo
- advertisement
- proper-content
- major-headline
- navigation
- navigation-menu
- navigation-bar

Contentual annotations are application-specific.

It is easy to imagine that every specific web site context might trigger other concrete role descriptions. Therefore, we define in the scope of this thesis a general annotation type:

$hasRole(x, R)$

$hasRole(x, R)$:
x defines an element (or a group of elements, see below) of the web page, and R is a string with application-dependent semantics. For complex semantic relations or the interaction of annotations coming from different application domains, web ontologies [HHRS08] can be used to structure the role semantics and to support automatic reasoning on contentual annotations.

Structural Annotations

Structural annotations do not make direct statements about the content of the marked information, but about its function in the page structure or its interaction habits. Therfore, the semantics are less application specific but more directed generally to the page visualization. A base set for structural annotation types, which are used in the scope of this thesis, is introduced in the following:

Structural annotations describe the element's function in a page structure.

$isMemberOf(x, G)$:
The element x is member of a semantic group named G. Elements are by default member of the group `elements`, containing all of the page's elements. In addition, the $isMemberOf()$ predicate can also be applied to semantic groups themselves, forming groups of groups. Analog to elements, groups are by default member of the group `groups`, containing all of the page's groups.

isMemberOf

> **SEMANTIC GROUP:**
> A *semantic group* is a named group of HTML elements on the same page.

Definition:
Semantic Group

$hasPriority(x, P[, G])$:
The element x has a relative priority P (a rational number), compared to other elements of group G or the default group `elements`.

hasPriority

$receivesKeypresses(x)$:
The element x can receive key presses.

receivesKeypresses

$supportsCharset(x, C)$:
The element x receives characters from the given charset C, defined as:

supportsCharset

- `numerical`: 0-9
- `lowercase-alpha`: a-z, ä-ü, ß etc.
- `uppercase-alpha`: A-Z, Ä-Ü etc.
- `whitespaces`: \0x10 \t etc.
- `multiline`: \r \n
- `decimal-separators`: .   ,
- `mail-separator`: @
- `separators`: - ; :   + / etc.
- `any`

$hasValueLength(x, N)$:
The value of element x has a length of N characters.

hasValueLength

$hasAttentionTime(x, T)$:
In average, the element x is for T seconds in the users attention (per page visit, T is a rational number).

hasAttentionTime

$followsFocus(x, y)$:
The focus of the element x follows the focus of y.

followsFocus

$hasFocusFollower$	$hasFocusFollower(x, y)$: The focus of the element x is followed by the focus of y.
$isSummary$	$isSummary(x, Y)$: x is the summary of Y, where Y can be a page element or a semantic group.
$inducedBy$	$inducedBy(v_x, v_y)$: The element x has the value v_x, if y has the value v_y.
$induces$	$induces(v_x, v_y)$: If the element x has the value v_x, the element y has the value v_y.
$dependsOn$	$dependsOn(x, v_y)$: The element x only gets a non-default value if the element y has the non-default value v.
$hasDependent$	$hasDependent(v_x, y)$: If the element x has the non-default value v_x, the element y also gets a non-default value.

4.1.2   Annotation Model

A *Shadow Annotation* has a subject, a type and optional properites.

As sketched in Figure 4.1 and described in the last section, every element of a web site can be enhanced with some semantic annotation *(Shadow Annotation)* a. Therefore, every annotation contains at least the element it relates to, named the *Subject s* of the annotation. Also the *type t* of an annotation is required. Every concrete Shadow Annotation can carry one or several *additional properties* $p_1 \ldots p_k$, as required by the semantics connected with the type of the annotation. These additional properties can be simple scalar values, as an integer representing the number of characters in an annotation of type $hasValueLength$, but they can also reference other web site elements or their values (as in an annotation of type $followsFocus$). To determine the validity range of the annotation, a *Contextual Confidence* Γ can be given, referencing a context object c (see Section 2.1) and a real confidence value $\gamma \in [0..1]$ denoting the probability that the given annotation's statement is valid in the given context. Therefore, a Shadow Annotation is defined as the tuple

A *Contextual Confidence* can define the validity range of a Shadow Annotation.

$$a = (s, t, p_1 \ldots p_k, \Gamma) \tag{4.1}$$

Definition:
Shadow Annotation

with the contextual confidence level Γ being $\varnothing \equiv (1.0, \varnothing)$ or

$$\Gamma = (\gamma, c) \tag{4.2}$$

Shadow Annotations are managed in the *Semantic Shadow*.

The database managing all known Shadow Annotations is called the *Semantic Shadow* and supports querying (and storing) Shadow Annotations using an interface depicted in Figure 4.2.
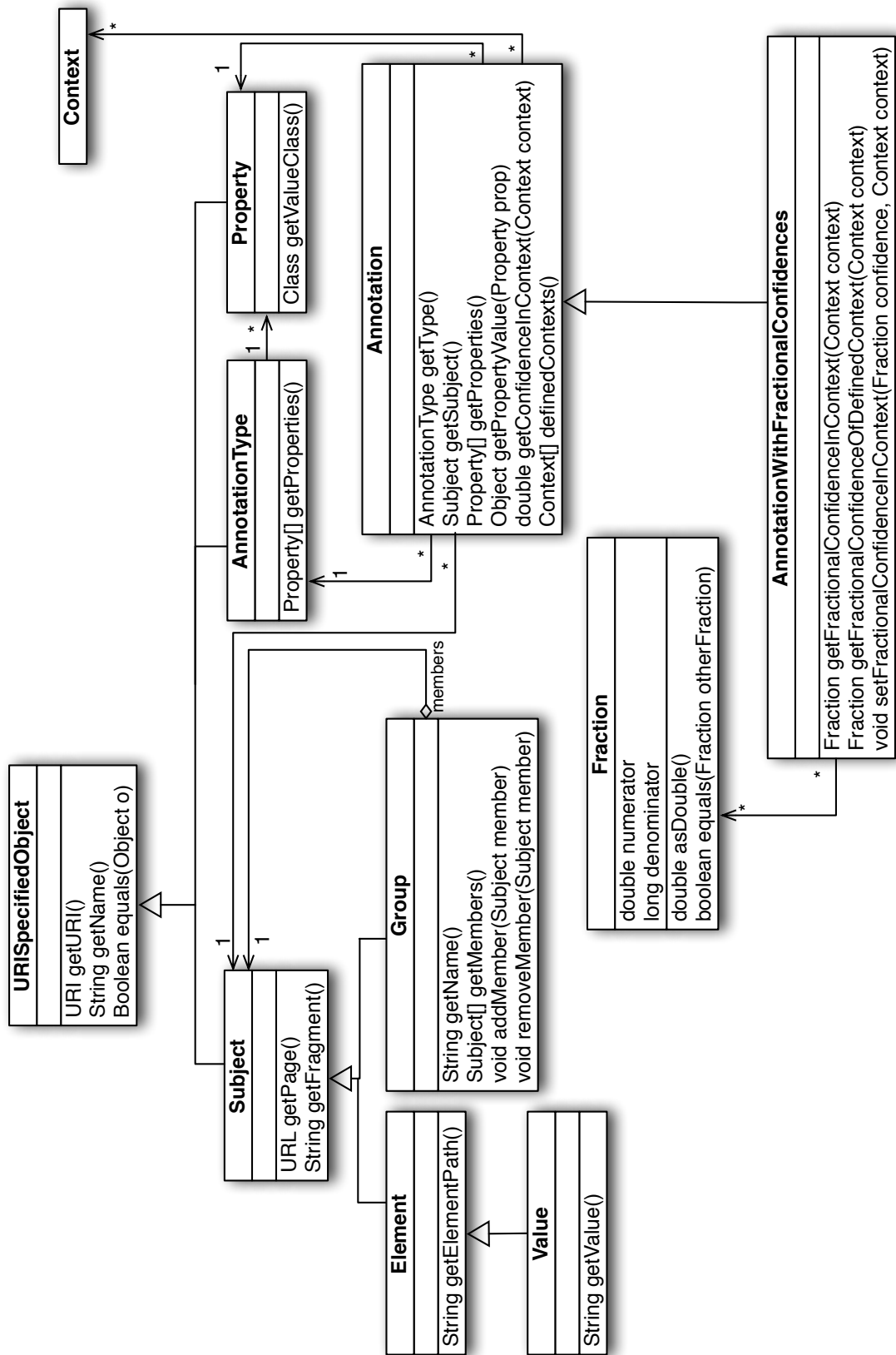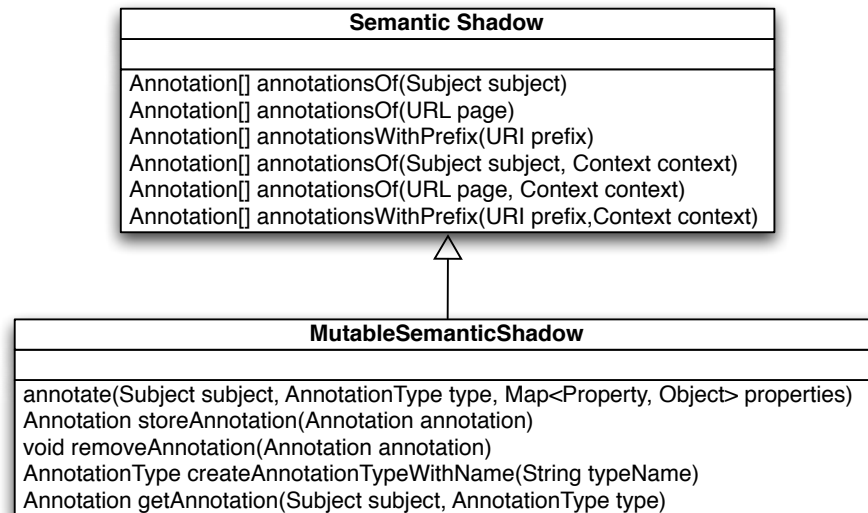
Figure 4.1: The general model of a Shadow Annotation and associated classes.

Figure 4.2: The interface manage annotations of the *Semantic Shadow*.

4.1.3   Model Representation

Shadow Annotations can be mapped to an RDF model.

To interact with the Shadow Annotations, some representation of the model is required. To match with existing Semantic Web technologies, an RDF [KC04] based representation has been developed.

Subject and type can be mapped directly, properties and context confidences with a blank node.

In RDF, every data element consists of the triple *(Subject, Property, Object)*. Whereas the Subject and Property are required to be resources identified by a Unique Resource Identificator (URI), the Object can either be a single value/resource or another RDF triple. The mapping of an annotation $a = (s, t, p_1 \ldots p_k, \Gamma)$ is realized as follows (see graph representation in Figure 4.3):

- s is mapped to the subject of the RDF-triple.

- t is mapped to the property of the RDF-triple.

- As object, an empty node ("Annotation Property Node") is introduced.

- The type specific annotation properties $p_1 \ldots p_k$ are mapped to properties of the Annotation Property Node.

- The contextual confidence Γ is also modeled as independent node connected to the Annotation Property Node with the property `sems:contextual_confidence`.[1]

An annotation can have multiple values for the Contextual Confidence.

If several annotations only differ by their `sems:contextual_confidence` property, they can be represented by a single RDF-triple with several `sems:confidence` properties attached to its Annotation Property Node.

---

[1]The prefix `sems:` is an abbreviation for the URI `http://www.cs.bonn.edu/sems/2010/08/14/`
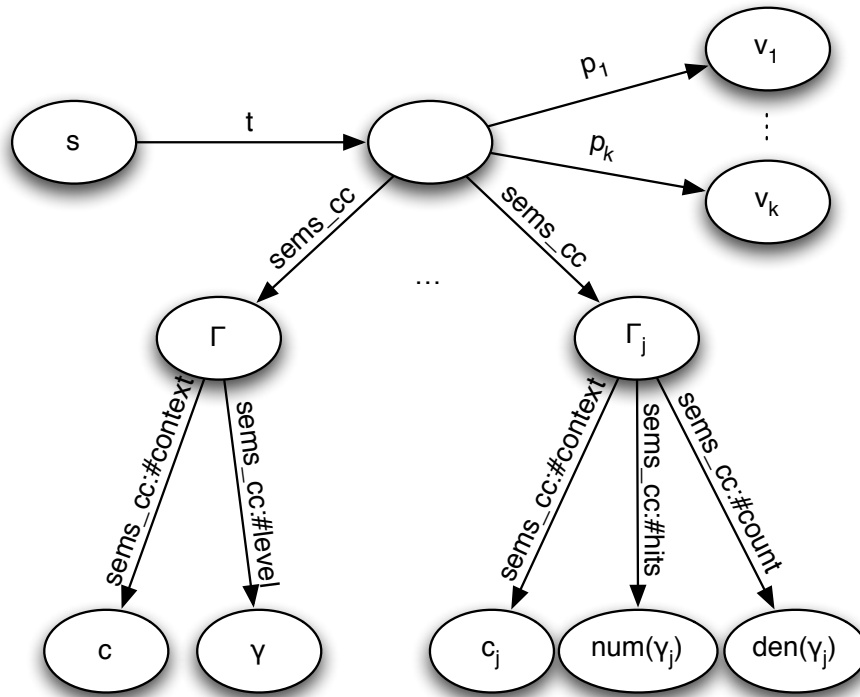
Figure 4.3: An annotation on a web site element can be directly mapped to a representation in RDF.

The Annotation Subject

The annotation subject can either be a web site element, the value of a web site element or a semantic group in a web site. To map to RDF, these resources have to be identified by a URI. In every case this URI is constructed of the related web site's URL (including the location part, if relevant), a # as separator and a fragment part.

Annotation subjects can be elements, values or groups.

In the case of a web site element, the fragment part identifies the web page element using a simplified and URI-compatible XPath 1.0 [CD99] description:

A simplified XPath based URI is used to identify web page elements.

- If the element contains an id, simply the id can be stated.

- Otherwise, the fragment starts with a / followed by the simplest possible path only containing the `child` axes and the functions `position()` and `id()`.

- To shorten the URI length, the following abbreviations are used:
  /element(n) $\equiv$ /child::element[position()=n]

  /element $\equiv$ /child::element[position()=1]

  bla $\equiv$ /id('bla')

The canonical fragment representation of an element is calculated as shown in Algorithm 4.1

Algorithm 4.1 Canonical XPath calculation

1: **function** XPATH(*e*)
2:     **if** *e.id* $\neq \perp$ **:**
3:         **return** *e.id*
4:     **else if** *e.parent* $== \perp$ **:**   *// The root element never has siblings*
5:         **return** "#/" + *e.name*;
6:     **else if** *e.parent.indexOf*(*e*) > 1 **:**
7:         **return** XPATH(*e.parent*) + "/" + *e.name* + "(" + *e.parent.indexOf*(*e*) + ")"
8:     **else**
9:         **return** XPATH(*e.parent*) + "/" + *e.name*

Values and groups are also mapped to URIs.

In case of an element value as annotation subject, the value is added in apostrophes (') to the canonical element fragment. A group is always represented by the fragment `#$groupname`.

The Annotation Type

Annotation type URIs have a dedicated namespace.

The URI of the annotation type, constructing the property of the annotation RDF triple, is generally constructed by using the prefix `sems:atype/` and the common name of the annotation, e. g. `hasValueLength`. For annotations introduced by independent modules, individual prefixes might be used.

The Annotation Properties

Properties are assigned to a blank node called *Annotation Property Node.*

To represent the annotation properties, a blank node called "Annotation Property Node" is used as subject of the annotation RDF-triple. This node is used as RDF subject for triples representing each a type specific property. Their value is represented by the triple object, and the triple property URI is defined as *annotationTypeUri*`#`*propertyName*, e. g. `sems:atype/hasValueLength#length`.

The Contextual Confidence

The Contextual Confidence is also assigned to the Annotation Property Node.

The Contextual Confidence, if applying, is modeled by a blank node connected with the property `sems:confidence` to the Annotation Property Node. It contains two RDF triples named `sems:contextual_confidence#context` pointing to a context description as introduced in Section 2.1 (or compatible) and `sems:contextual_confidence#level` stating the confidence level between $[0 \ldots 1]$. As an alternative, the confidence level can be expressed using the pair `sems:contextual_confidence#hits` and `sems:contextual_confidence#count`. Then, the confidence level can be calculated as the fraction *hits/count*, while at the same time, the fraction's numerator and denominator are maintained for incremental level updates.

Annotation stability

As web sites are updated over time, not only the web site content, but also the presentation structure and within it the elements in the HTML DOM change. If content or new elements have just been added to the web page, the XPath pointers are still valid, and therefore annotations can still be correctly associated [AH03]. In most cases, even if the structure of a web page changes, the element ids are preserved. As the annotation subject identification in the *Semantic Shadow* concept is based on these ids as soon as they are available, the "stability" of these annotations is high. In some cases, if element ids are changed or not available at all, a re-identification of the original annotation subject in the restructured web page is not possible.

To maintain *Semantic Shadow* annotations despite fundamental page structure changes, a time stamp based transformation document is required. This document needs to be maintained by the site publisher, since element correspondences between different page revisions are hard to detect automatically.

4.1.4 Interoperability with Semantic Web Tools

In the last section, a way to store *Semantic Shadow* annotations in an RDF model has been presented. While the way of introducing a blank node to store multiple values of an RDF property (here: the Annotation Property Node) is generally supported by Semantic Web Tools [W3C10], the notion of *contextual confidence* and its semantics are specific to the *Semantic Shadow* concept. To process the data with general RDF tools, the RDF model needs to be reduced depending on the operations which ought to be executed on the model.

Reduction to fuzzy datasets

A number of approaches exist to introduce fuzziness into the binary world of RDF (see Section 3.1.2). Generally, these approaches allow to state the probability of an RDF tuple being true. Depending on the output representation, a specific mapping can be defined, which, given a specific evaluation context c, transforms the Contextual Confidence γ_c into the required probability value on the main annotation tuple. This corresponds to an evaluation of the *Semantic Shadow* in c.

Reduction to plain RDF

If uncertainty has to be eliminated completely from the RDF model, the reduction not only has to take into account a concrete interpretation context c, but a decision baseline γ_{min} as well. Each annotation persisting in the original model, for which $\gamma_c < \gamma_{min}$ will be removed from the model, as well as all `sems:confidence` properties. This corresponds to an evaluation of the *Semantic Shadow* in the concrete context c, while for every annotation a the mapping $\gamma_a(c) \leftarrow \lceil \gamma_a(c) - \gamma_{min} \rceil$ is applied.

4.2   Applications of the Semantic Shadow Concept

SemS data can be used for adaptation and analysis.

The concept of the *Semantic Shadow* extends the currently established web data specification by another, parallel information layer. The information available in the form of contextual semantic annotations can be used for different scenarios, of which at least three groups can be distinguished:

1. Live adaptation of HTML data

2. Off-line adaptation of HTML data and generation of target group variants

3. Static analysis of usage information

Each of these application groups will be explained in detail.

4.2.1   Live Adaptation

Live adaptation does not require special server or client software support.

"Live web page adaptation" is characterized by three vertex points:

1. The web page content is generally stored on the web server in a way that no specific, access-context dependent adaptations are performed by the serving host. This includes static web pages as well as dynamically generated ones. Site adaptations, which are required by the business logic context (e. g. the visualization of a virtual purchase cart) are evidently handled by the web server software.

2. The client is not required to perform adaptations manually, e. g. by defining user side style sheets [Tre08, Bar10] or by navigating to a specific URL triggering a predefined adaptation process.

3. The adaptation work is done on demand, i. e., following a request by the user's browser, the request context is determined and using Semantic Shadow information, the requested web data is adapted.

The adaptation process can take place itself on the *client side* or using a *proxy software* (see [Jan07, Chapter 5.2]).

Client side adaptation is possible, but has drawbacks.

Client side adaptation takes place on the user's machine, e. g. using a browser plug-in or some kind of network layer software. This approach is favorable if privacy concerns prohibit the processing of context data outside the client's machine, or if the *Semantic Shadow* database is user specific and maintained on the client machine. Nevertheless, web page adaptation might consume quite a bit of computing power and memory, depending on the complexity of the adaptation process. If the client's resources are very limited, a dynamic scaling may not possible, neither in the means of CPU, nor in the means of available memory. In addition, the responsiveness of the complete system and with it of other applications running concurrently might be affected. As a further drawback, the user has to install a specific piece of software on his system, which makes the approach more obtrusive. If the *Semantic Shadow*

information has to be requested from a distant server, the querying time adds up to the perceived page loading time and the amount of transmitted data increases. Summarized, client side processing might be an option when used in an environment with specialized requirements and on desktop computers and a future option for mobile phones.

A *proxy software* hooks into the data flow and works between the client's browser and the web server software (see the visualization of USAPROXY in Figure 5.1). The technique is also called "third-party adaptation" or "dynamic reauthoring" and is, according to Kurz et al. [KPG04], one of the most widely accepted approaches for dynamic adaptation processes. Kurz names minimal maintenance effort and single-source authoring as reasons, but this goes together with other advantages like the option for transparent setup, minimization of data transferred to the client's device over the "last mile" and externalization of resource usage. In their paper from 2004, Kurz et al. mention nine approaches of adaptation proxies in research studies dating back to 1987, so the approach can be seen as being established. There are several deployment options (see Figure 4.4) which rank from a personal proxy installed on the user's system (similar to the approach sketched in the last paragraph) over a transparent proxy system running on a provider's gateway computer, an external service which is provided as a general solution and has to be configured in the user OS/browser's network connection settings, up to acting as the web server software, which transparently forwards the request to the actual, hidden web server. In the user's perception, the approach is similar to the now popular "computation in the cloud", where a calculation is not performed on the user's device, but on distant machines [AFG+10].

Using an adaptation proxy is the preferred deployment scenario.

4.2.2   Offline Variant Generation

While the live adaptation outlined in the last section is a very flexible approach, it is at the same time very ineffective regarding CPU and memory resources. At the instance the user requests a web page, the adaptation process has to be triggered and performed without saddling the user with a long waiting time until the requested web page is displayed in his browser. This limits the runtime of the adaptation algorithm and therewith its complexity. In addition, if several users are requesting an adaptation process concurrently, enough resources have to be provided to perform the adaptation calculation in parallel.

Live adaptation is a resource-expensive approach.

One approach to reduce the adaptation resource demands is the precalculation of adapted web pages and the persistent storage (caching) of those variants. This approach works especially well if the content to be adapted is rather static and the relevant request context can be simplified into a few standard context classes (see *User Categorization* in [All97]). The definition of these context classes and the mapping of a specific request class to a specific context class as a step of simplification is highly scenario dependent and might require further domain knowledge (see Figure 4.5). For a detailed discussion, see Section 6.1.4.

Pre-calculation and caching of a limited variant set solves the resource problem.
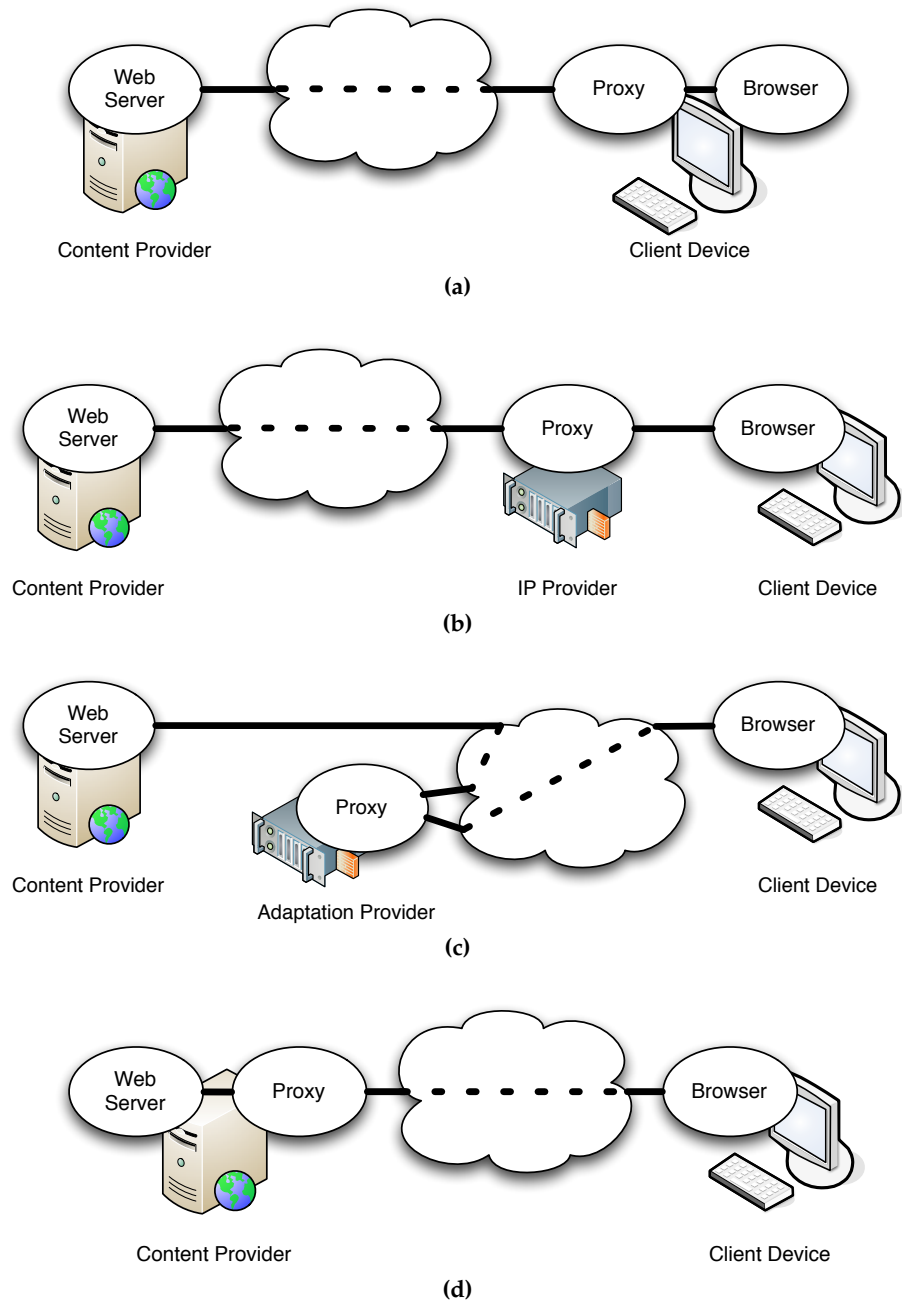
Figure 4.4: Proxy adaptation can be installed on the user's system (a), at the provider's gateway machine (b), as an independent service (c) or on the content provider's side in front of the content server (d).

Both approaches can be combined to optimize the tradeoff between variant calculation and system responsiveness.

Practically, the off-line variant generation can be combined with the live adaptation process presented in the last section: Upon an incoming client request, the request context is mapped to a variant context class and the variant cache is checked for an appropriate recent page variant. If there is no variant available or the pre-cached variant is outdated, a live adaptation using a representative context description of the required context class as request context is performed. The result is returned to the client as well as cached on the adaptation server
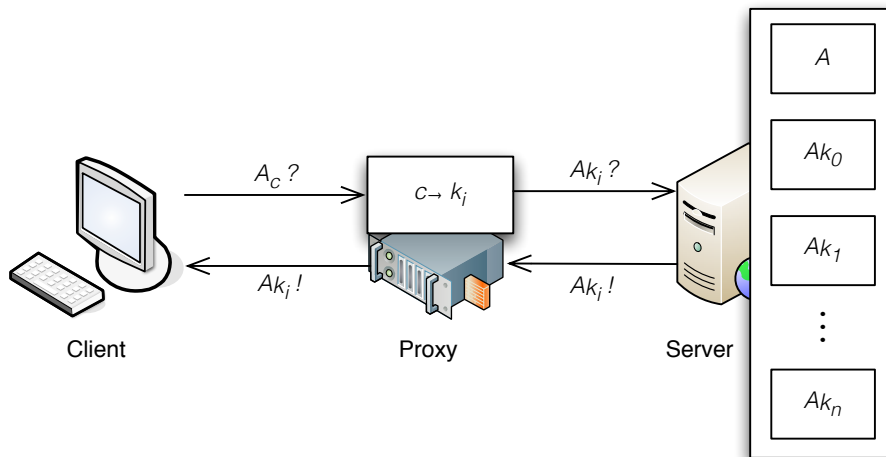
Figure 4.5: Upon request A_c, the request context c is mapped to a representative context class from k_i and the web page variant pre-calculated for the given context class A_{k_i} is transmitted to the client.

for further use (see Figure 4.6). This combined approach still suffers from complexity limitations implied by the response time limits, but on the other hand minimizes response time for further requests in context mapped to the same context class. In addition, the variant management is simplified and optimized, since no pre-generation of variants practically never requested is done. Using a simple request statistics (or an analysis process as sketched in the next section), proactive pre-generation of the most probably requested variants can be performed while at the same time requests with contexts mapped to uncommon classes can still be served.

4.2.3 Static Analysis

The third category of applications for the *Semantic Shadow* concept presented here does not have a direct impact on the user's page visualization: The static analysis of annotations. An application from this category uses the annotations of the *Semantic Shadow* as input data to infer further knowledge about the web page, its usage (if the annotation data was generated based on web site usage, see Chapter 5) and its structural semantics. The result of such an algorithm might be the definition of representative context classes to partition the set of possible request contexts (see the previous section). Another result might be a human readable report indicating the typical use of the web site by the clients in varying contexts, so that content or presentation optimizations can be manually performed (as done with classic web mining approaches, see [Spi00]). A third option is to generate new *Semantic Shadow* annotations, based on further interpretation knowledge embedded into the analysis algorithm (see "Knowledge Discovery" in [NSS01]). An example for such an algorithm is presented by Jasmin Kreutz in [Kre10], where configurable patterns (e. g. address form, gallery, login form,

Static analysis can infer further knowledge from annotation interpretation.

Pattern-matching on annotations is one analysis example.
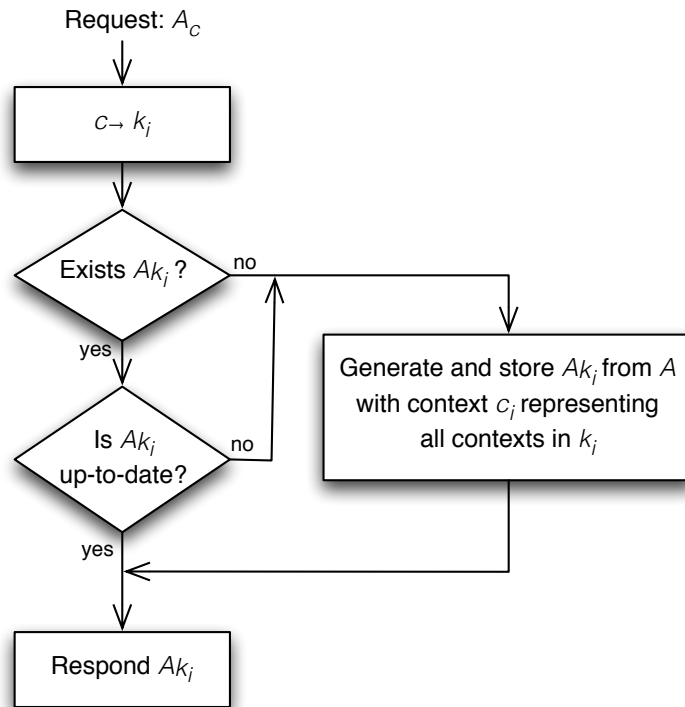
Figure 4.6: Combining the live adaptation strategy with variant generation allows to perform the trade-off in between responsiveness and efficiency.

date selector, shopping cart) embedded in a web site are identified. For this identification, the web page HTML code is analyzed in conjunction with the corresponding *Semantic Shadow* annotations. The identified patterns are in turn persisted in the *Semantic Shadow* using group and role annotations, so they can be used subsequently by adaptation algorithms.

Chapter 5

Deriving Web UI Meta Information from their Usage

To extend web sites with context-dependent semantic annotations as described and used in the previous chapters, several methods can be applied: The most basic way is to construct the semantic annotations together with the rest of the web site while creating the page. This way is frequently used for adding Semantic Web [HHRS08] annotations to web pages or marking certain roles of web site elements in XHTML [W3C07]. The second possibility is to use dedicated annotation tools (see for example [HKO$^+$00, Koi05] and for comparison [Rei06]).

In a simple approach annotations are generated manually.

A third way, proposed in this thesis, is the derivation of contextual semantics by unobtrusively studying frequent user interactions, as given by the regular use of the web pages. This method requires several steps: Firstly, the usage of the web resource in question has to be tracked. Secondly, this tracking data has to be summarized for privacy and manageability issues. Thirdly, meaningful semantic annotations have to be extracted from this aggregated usage data and linked back to the related web site element.

Annotations can also be generated from usage data analysis.

5.1   User Tracking Methods

Several methods can be applied to look over the user's shoulder while he or she is surfing the web and inspecting its pages. These methods can roughly be separated into four categories, each improving the quality (and amount) of tracked user data, but at the same time requiring more complicated software and client-server interaction.

Four levels of web user tracking can be distinguished.

5.1.1   Inspecting HTTP Request Logs (Level 1)

In order to display a web page in the user's browser using the the HTTP protocol [FGM$^+$99] a GET or POST request is send to the server hosting the web page in question. Besides the requested web resource,

First level usage data is based on HTTP access logs.

the request contains meta information such as the browser's identification string ("User Agent") and the web site which was viewed before the browser issued the request ("Referrer"). Traditionally, web servers maintain log files were they add a line to for every handled request ("Access Log"). In general, the logged dataset contains information about the user's IP address, the request time, the requested resource, the user identifier (if the resource was password protected), the user agent and the request's referrer. The wide-spread Apache HTTP server defines this in its documentation by the "Common" and the "Combined" log format [The09]. Similar information is also stored by other web servers like the Microsoft IIS [Mic10] using the W3C Extended Log File Format [HB99]. While this log information does not reveal details about the user's interaction with the web site's elements per se[1], at least the request context (see Section 5.2.3) and basic page related usage data (e. g. viewing time) can be inferred. To generate annotations modeling the interaction of a given page with other pages of the same site, level 1 (L1) logging information is often sufficient [Spi00].

5.1.2   Inspecting HTTP Request Parameters (Level 2)

On the second level, form values submitted using HTTP request parameters are evaluated.

If a user fills out a form on a web page, the corresponding data is transmitted together with the next HTTP request. In case of a GET request, the data is attached to the request URL, separated from the page URL by a question mark sign (?), in case of a POST request, the data is transmitted in the request body. In both cases, not the input's field DOM position or ID is used as value identifier, but a dedicated `name` attribute of the corresponding form element. Using the page HTML data, the originating input element can be identified using the `name` attribute, but unfortunately, this association is not guaranteed to be unique. In addition, only the final value of the user input can be inspected using this method, intermediate values or web side interactions not resulting in a form value change can not be tracked.

5.1.3   Inspecting Client Events (Level 3)

For more detailed tracking, client side code execution is required.

The first two methods explained in this section can be implemented completely on the server side of a web surfing experience, since no more information is gathered and taken into account than required anyway to deliver the service "Web". If more detailed information about the web site usage is desired (as for example in Usability research), the client computer has to be advised to cooperate and capture client events like mouse movements, key presses and focus shifts.

User side event capturing can be done by embedding JAVASCRIPT event handlers into the requested page.

Classically, dedicated software [GS00, GF03] or browser plug-ins [SLP99, Sri10] need to be installed to capture user events. A more unobtrusive way to record events triggered by the client browser is the usage of embedded JAVASCRIPT, like done by [Wnu05, PSK09].

---

[1]Except for the case where form data was transmitted using a GET request, see next section.

A filed application to record these events remotely is the USAPROXY [Wnu05, Att08], which this thesis' user tracking prototype is based on.
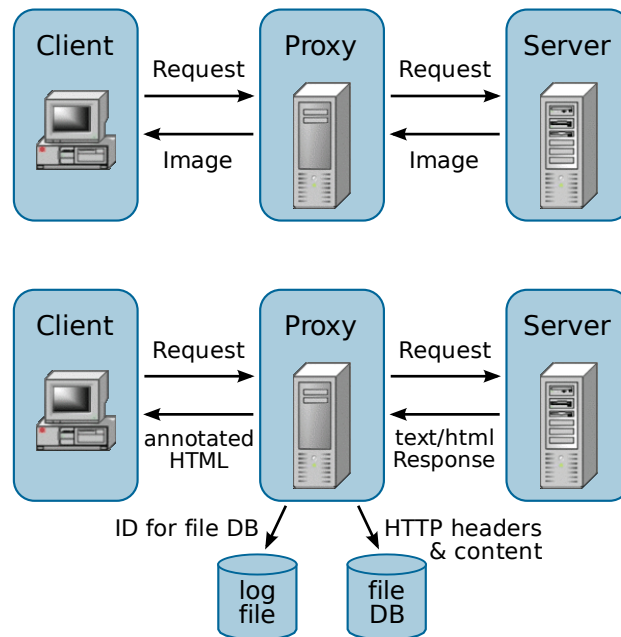
Figure 5.1: The main architecture of USAPROXY: While HTML responses are modified for user tracking, any other server response is simply forwarded to the client. [Att08]

As the name suggests, the USAPROXY technically acts as a web proxy, receiving HTTP requests from a browser and redirecting them to the requested web server. The proxy receives the response from the web server and forwards it to the web client (see Figure 5.1). If the server's response indicates that a HTML document is delivered, the USAPROXY adds a special `script` tag to the page's header, which points to the browser event tracking code. Every time the user triggers a relevant event, the script communicates in the background with the proxy software to log this event on the proxy server (Figure 5.2).

The USAPROXY framework implements such a tracking algorithm.

In order to record the data required for usability testing of web applications, the following events are tracked by the embedded JAVASCRIPT and forwarded to the proxy server writing a log file ([Wnu05, Att06, Att08]):

All browser events related to user interaction are captured.

onClick : The user clicks on a DOM element. Together with the event, the source element and the x and y offset of the mouse position from the top left corner of the window are recorded.

onMousemove : The mouse is moved over an element. Together with the event, the x and y offset of the mouse position is recorded.

onKeyPress : The user presses a key on his keyboard. Together with the event, the key that was pressed is recorded.

onLoad : A page is loaded. Together with the event, the width and height of the browser window is recorded.
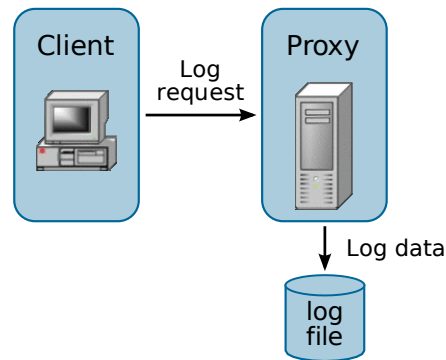
Figure 5.2: The logging procedure of USAPROXY: Browser events are captured on the client side and forwarded asynchronously to the browser. [Att08]

onFocus : An element or the document is focused.

onBlur : An element or the document looses focus.

onUnload : A page is unloaded.

onResize : A browser window is resized. Together with the event, the new width and height of the browser window is recorded.

onScroll : The document was scrolled.

onValueChange : Changes are made to the value of any field in a form, including radio buttons, drop-down menus, checkboxes and text fields.

onSelection : Text is selected using the mouse or keyboard, either inside text fields or anywhere else on the page.

For every event, the related HTML element is recorded, identified either by its `id` or `name` value, or by its relative position in the DOM tree. As *onMousemove* and *onScroll* events are triggered very frequently, these events are captured for logging only in constant intervals triggered by a timer.

Minor modifications on the USAPROXY software were required for this thesis.

Based on the open source of USAPROXY 2.0 [WA06], for the purpose of this thesis, minor modifications have been made to the application:

- Instead of recording an abstract identification string for elements without defined id (see [Att08, Section 7.4.3]), an XPath pointer is calculated on the client side and transferred to the logging system.

- Small fixes related to the correct persistence of time information have been added.

5.1.4   Using Dedicated Hardware (Level 4)

The most sophisticated way of tracking the user's attention when scanning a web page is to follow his or her eyes while they slide over the browser's page visualization. Using eye trackers as computer input has long been under research [WM87][Duc07, Chapter 20], often for the evaluation of web site interaction [GJG04]. In experimental settings, an eye tracking hardware like the *Tobii X120* [Tob08] can be used to determine the user's points of fixation and scanpaths on the web page [SG00, JK03, IB04, PB05]. Concluding from the point of focus to the focus of interest is valid, as John Hansen argues in [Gri10], but might not be all the time correlated [SFYW98]. In addition, to create a critical data mass the mining of usage data for web site annotations needs to go beyond experiments, which requires an eye tracking being part of the standard web usage process. Eye tracking hardware, which can be integrated into standard computer equipment [Tob09] might be a future option. A low fidelity approximation is the use of (integrated) webcams [HMHN04, HP05, LBP06, SSM+10], as this eye tracking method is also usable for mobile devices [MWC10].

On the fourth and most detailed level, eye tracking hardware is used.

5.1.5   Comparison of Tracking Methods

The quality of the tracked information increases from one tracking level to the next one. At the same time, the amount of data generated by the tracking algorithm increases, as does the time and processing power which are required to analyze the data. With respect to the given application requirement, the minimally required tracking method is to be selected.

A tradeoff has to be made between data quality and amount.

The advantage of eye tracking usage to determine user focus time on web elements is its higher precision compared to an approximation by interpreting the mouse cursor as point of interest (cf. the discussion in [AWS06, Section 6.2]). On the other hand, measurements might still be imprecise, since eye tracking algorithms are very sensitive to external disorders like caused by the instability of the eyes, blinking and external factors as contact lenses and eyeglasses (especially bifocals), long eyelashes, heavy eye makeup or "droopy" eyelids [Duc07]. Distortions resulting from the user's UI interaction manners, which might be corrected in experimental settings [PN09], would lead to bad algorithm performance in a real web usage scenario. Additionally, eye tracking is a procedure the average user is not used to, so especially in the case where video data is evaluated, privacy concerns might make the user feel uncomfortable with the usage tracking method.

Eye tracking algorithms are sensitive to distortions.

For the structural annotations, which are generated using the procedures and concepts presented in this thesis, a detailed view on the user's interaction with the web page is required. On the other hand, in order to generate resilient context-dependent annotations, interactions from as many subjects as possible have to be aggregated, so the inhibition threshold of participating in the project should be low. This

For generating SemS annotations, tracking data from level 2 and 3 is used.

excludes the use of dedicated hardware and leaves the options of analyzing server-based and JAVASCRIPT generated tracking data.

5.1.6 User Tracking and Privacy

Usage data collection has to be minimal and transparent to the user.

In all cases, privacy issues have to be taken into account. The more detailed information about a user's interaction with a web site is gathered, the more private information might be exposed. For the JAVASCRIPT based tracking process, [Att08, p.178] gives implementation hints, which can be summarized as: Do not collect information without informing the user which data is collected and how the data is used, and process information only if the user agreed for this specific process. On the server side, data must be aggregated in an early stage and every piece of information, which might allow a backtracking to an individual user has to be access protected, and deleted as soon as the aggregation process is finished.

5.2 Deriving Contextual Annotations from Usage Data

This section sketches inference algorithms for the annotation base set.

In Chapter 4, the concept of the *Semantic Shadow* has been introduced, based on the idea of providing further annotations to web page elements (Section 4.1). This section shows how the structural annotations from the base set introduced in Section 4.1.1[2] can be derived from usage data, tracked as described in Section 5.1.

5.2.1 Offline Derivation

Usage data of page views is persisted in event lists (log files).

The information about the usage of web pages is normally stored in sequential lists of events, traditionally referred to as *logfiles*. In the following we assume that for every user u viewing a page p in a context c, a dedicated event list $E_v = (e_v^0, e_v^1, \ldots, e_v^{n_v-1})$ for this view $v = V(u, p, c)$ is available. A page view v is defined as the process requesting the web page and interacting with it's visualization, until requesting the next web page or canceling the web browsing session. The total number of events tracked during the page view v is referenced as $n_v = |E_v|$.

Log-lines are time stamped and temporarily ordered.

Every event e_v consists of a timestamp t_{e_v}, an event name n_{e_v}, the event scope s_{e_v} (i. e., the web page element x the event relates to) and additional, event specific parameters π_{e_v}. The events of an event list E_v are ordered chronologically, that means by definition

$$\forall e_v^i, e_v^j \in E_v : (t_{e_v^i} < t_{e_v^j}) \leftrightarrow (i < j) \tag{5.1}$$

Since timestamps do not have an arbitrary precision when stored on a computing system, it can happen that several events seem to occur "at once", i. e., have the same timestamp. Therefore we only assume the

---

[2]Except for $isMemberOf(x, G)$ and $isSummary(x, Y)$, which can better be generated using HTML analysis, see Section 3.3.

following, weaker conclusions:

$$\forall i,j \in \{0,1,\ldots,n_v-1\}, i < j:$$

$$(t_{e_v^i} < t_{e_v^j}) \to (i < j) \tag{5.2}$$

$$(i < j) \to (t_{e_v^i} \le t_{e_v^j}) \tag{5.3}$$

$$(t_{e_v^i} = t_{e_v^j}) \to \forall k \in \{i,..,j\} : t_{e_v^k} = t_{e_v^i} \tag{5.4}$$

All elements of a page p form the set A_p. In addition, the following set of children of an element $x \in A_p$ is defined:

$$O(x) = \{y \in A_p | (y = x) \vee (x \leftarrow y)\} \subseteq A_p \tag{5.5}$$

$O(x)$ contains the element x and all direct or transitive children of x.

where $(x \leftarrow y)$ symbolizes that the element x is the direct or transitive parent of y. In the following, the tracked web page usage information is referenced as L1, L2, L3 and L4, depending on the different ways of data collection described in Section 5.1. If not stated otherwise, the event list E_v of a single view v is treated as input, and reasonable aggregations over all tracked page views (average, weighted average, global extreme values etc.) have to be made to calculate the overall valid predicates for all $x \in A_p$.

The list E_v contains all events of a page view.

$hasPriority(x, P[, G])$:   *hasPriority*
When deriving the priority from usage data, a simple approach is to assume that for active elements (e. g. buttons) the importance of an element is directly correlated with the activation frequency, which can be calculated observing the L3 `'onfocus'` or `'onclick'` events[3]. For passive elements (e. g. paragraphs), the importance of an element can in a first approach be seen as correlated with the time the user spends his attention on it and calculated by normalizing this time (as expressed by $hasAttentionTime(x,T)$) by the amount of information the element carries (e. g. number of characters).

The absolute values then have to be compared with those of the other elements in G, i. e., all members associated to the semantic group G using the $isMemberOf(G)$ annotation type, to calculate relative priority values. If $x \notin G$, then P is 0.

$receivesKeypresses(x)$:   *receivesKeypresses*
With L3 tracking information, this predicate can be set for every x, where an event e exists with $s_e = x$ and $n_e = $ `onKeyPress`.

If only L2 tracking information is available, the predicate can be deduced from the presence of an event with $s_e = x, n_e = $ `value`, $\pi_{e_v} \neq \emptyset$. Unfortunately, this method can just be applied for text input fields in submitted forms and is only a sufficient criterion[4] (Due to the fact the user might have entered something and removed it before submitting the form). With L1 information, the predicate can not be deduced.

---

[3]If only L2 tracking information is available, the `'onclick'` event can be "simulated" by analyzing which values of button input elements have been submitted.

[4]Given that the text field is not disabled, which requires further processing of the input HTML.

supportsCharset

$supportsCharset(x, C)$:
This predicate can be determined using the same techniques as described for $receivesKeypresses(x)$, extended with the concrete analysis of the input value. As soon as a key from a certain charset is detected, this charset can be flagged as "supported".

When using L3 tracking information (events 'onkeypress' and 'onvaluechanged'), attention has to be paid to analyze only the final result of an input event set (or at least to value this higher), because most probably the final input reflects the intended element input semantics better than intermediate typing errors.

hasValueLength

$hasValueLength(x, N)$:
To get the length of an input value, input can be tracked as in $supportsCharset(x, C)$, and the length of the input can be evaluated. In case of E_v being L2 tracking information, the simple Algorithm 5.1 can be applied. For L3 tracking information, the same measures as in $supportsCharset(x, C)$ are required.

Algorithm 5.1 ValueLength calculation

1: **function** VALUELENGTH(x)
2:      **for all** $e_v \in E_v$ **:**
3:          **if** $(s_{e_v} \in O(x)) \wedge (n_{e_v} = \texttt{value})$ **:**
4:              **return** STRLEN(π_{e_v})
5:      **return** $\perp$

---

hasAttentionTime

$hasAttentionTime(x, T)$:
The attention time of a certain element x can be tracked using L4 log data by analyzing the visual focus of the user (see Algorithm 5.2).

Algorithm 5.2 Attention time calculation

1: **function** ATTENTIONTIME(x)
2:      $\tau_x \leftarrow 0$
3:      $t_{last} \leftarrow \perp$
4:      **for** $i \leftarrow 0, n_v - 1$ **:**
5:          **if** $n_{e_v^i} = \texttt{'eyefocus'}$ **:**
6:              **if** $s_{e_v^i} \in O(x)$ **:**   *// element (or child) x is focused*
7:                  **if** $t_{last} = \perp$ **:**
8:                      $t_{last} \leftarrow t_{e_v^i}$
9:              **else**   *// another element is focused*
10:                  **if** $t_{last} \neq \perp$ **:**
11:                      $\tau_x \leftarrow \tau_x + (t_{e_v^i} - t_{last})$
12:                      $t_{last} \leftarrow \perp$
13:          **if** $t_{last} \neq \perp$ **:**   *// if last focused element was x*
14:              $\tau_x \leftarrow \tau_x + (t_{e_v^{n_v - 1}} - t_{last})$ *// approximation only*
15:      **return** τ_x

In this approach we assume that the user does only focus on web site

elements (and does not move the focus from the browser). In a real implementation, the block beginning in line 9 has to be executed upon other events, which indicates that the user completely withdraws his attention from the website, too.

If only L3 tracking information is available, the Algorithm 5.2 might nevertheless be applied by replacing the event's name in line 5 with `'onmousemove'`. This seems to be a good approximation as the discussion in [AWS06, Section 6.2] shows.

The predicate can only be derived very roughly using L2 or L1 information by assigning the total viewtime of the page p to every page element.

$hasFocusFollower(x, y)$ / $followsFocus(x, y)$: $hasFocusFollower$
In order to derive these predicates from usage data, L3 tracking infor- $followsFocus$
mation is required. As shown in Algorithm 5.3, the annotations can
easily be extracted by observing the `'onfocus'` event (for the focus
successor definition, the subject and property element of the annotation are exchanged).

Algorithm 5.3 Focus predecessor calculation

1: **function** FOCUSPREDECESSORS(x)
2:      $e_{last} \leftarrow \perp$
3:      $P_x \leftarrow \varnothing$
4:      **for** $i \leftarrow 0, n_v - 1$ **:**
5:          **if** $n_{e_v^i} = $ `'onfocus'` **:**
6:              **if** $s_{e_v^i} = x$ **:**   *// element x is focused*
7:                  **if** $e_{last} \neq \perp$ **:**
8:                      $P_x \leftarrow P_x \cup \{e_{last}\}$
9:                  $e_{last} \leftarrow \perp$
10:             **else**   *// another element is focused*
11:                 $e_{last} \leftarrow s_{e_v^i}$
12:     **return** P_x

$induces(v_y, v_x)$ / $inducedBy(v_x, v_y)$: $induces$
These predicates can only be reasonably derived if a multitude of page $inducedBy$
views is analyzed. From the L2 tracking information (and from L3,
when measures like in $supportsCharset(x, C)$ are taken into account),
all submitted value tuples can be derived. To reduce the complexity, only the subset of inductions based on a direct focus switch are
considered, filtering the tuples to those element pairs (x, y), for which
$followsFocus(x, y)$ has been declared. A more sophisticated approach
would be the application of machine learning algorithms [Mit97]: The
value tuples can be converted into decision trees deciding upon v_x, and
based on these trees decision rules can be inferred. The rules can then
in turn be persisted as $inducedBy(v_x, v_y)$ predicates.

hasDependent
dependsOn

$hasDependent(v_y, x)$ / $dependsOn(x, v_y)$:

Using the correspondent $inducedBy(v_x, v_y)$ predicate set, all v_y for which the $dependsOn(x, v_y)$ statement applies can be calculated as

$$\bigcup_{v_x}\{v_y \in inducedBy(v_x, v_y)|v_y \neq \text{DEFAULTVALUE}(y)\} \qquad (5.6)$$

In the same way, $hasDependent(v_y, x)$ can be derived from the $induces(v_y, v_x)$ predicate set.

The computations must be triggered regularly.

In practise, these algorithms are executed at a scheduled interval, e. g. once a day, to update the *Semantic Shadow* annotation database. The availability of all required log files is assumed. The two predicates $isMemberOf(x, G)$ and $isSummary(x, Y)$ are more easily extracted from the page's HTML structure, e. g. by observing `div` nesting or CSS classname consistency (cf. the identification of content blocks in [CMZ03]).

The Contextual Confidence value can be calculated by observing the frequency of an annotation in a certain context.

To calculate the Contextual Confidence value, a reference value (e. g. the page view count in the page view's context) is stored together with the number of times a concrete annotation is valid in the request context. The Contextual Confidence γ of an annotation a in the context c is then expressed by

$$\gamma = \frac{\sum_{v \in \mathcal{V}_c} t(a, v)}{|\mathcal{V}_c|} \qquad (5.7)$$

where $\mathcal{V}_c$ denotes all page views in a context c and $t(a, v)$ is 1 iff a is valid for the page view v and 0 otherwise.

All events of a page view must be known at analysis time.

The algorithms sketched in this section are designed under the assumption that E_v for all page views v of all elements of a page are known when creating the annotations of the *Semantic Shadow*. In a linear implementation all events are processed in chronological order and the annotations derived from a single page view are finally calculated and persisted as soon as the last event of a page view has been processed. An example for such an analysis has been implemented for this thesis in the *SemSAnalyzer* project.

5.2.2 Online Derivation

Instant derivation of annotations has advantages and disadvantages.

While the previous section described how to derive concrete Shadow Annotations from web site usage data at dedicated points in time, this section approaches the idea to generate Shadow Annotations on-the-fly, i. e., update the annotation database with every new chunk of usage data. Generating the data in this way promises a couple of advantages:

- The annotations are instantly available.
- CPU usage can be scattered over time.
- Logging information, which might contain private data, need not to be stored over a longer period of time when evaluated instantly.

On the other hand, there are also disadvantages:

- Not all required information is available to conclude the correct semantics at every point in time.

- It might be difficult to filter unintended usage.

A good compromise is the steady event processing at the point in time when a user finishes his page view: All required information to infer page-view-based annotations (as described in the previous section) are available as well as the page view's usage context. Page views, which trigger then the *Semantic Shadow* update tend to range within minutes[5].

Generating annotations at the end of a page view is a good compromise.

The basis for an online event processing is an algorithm design, where page events are processed in chronological order (as soon as they have been tracked). During a page view, temporal information required to derive annotation data might be stored in memory or at another suitable data storage until the annotations are persisted at the end of the page view. Data required to derive the current usage context must be available as soon as the first events of the page view get analyzed. To assure a high responsibility of the event-tracking web proxy despite possibly slow annotation derivation calculus, an implementation using a separate thread and a circular buffer for event data is recommend. An adaptation of USAPROXY (see Section 5.1.3) generating annotation data on-the-fly was developed in the scope of this thesis.

Context information must be available when starting to generate annotations.

5.2.3   Context Information

Confidence level calculations for annotations in the *Semantic Shadow* concept are context related. Therefore, when generating annotations based on tracked usage information, the context of the corresponding page view has to be constructed and persisted as well. If no extension of the user's browser in the form of plug-ins is arranged, only the HTTP transfer data can be examined to build up the page view context, i. e., the connection information (particularly the user's IP address) and the exchanged HTTP headers.

Basic context information must be inferred from the HTTP request headers.

Using a Web Service as the one provided by IPInfoDB [Car10], the IP address can be used to guess the user's location when issuing the page request.[6] For more precise data, the W3C Geolocation API [Pop09] can be used to query the request location from the user using embedded JAVASCRIPT while collecting tracking information. With a reverse geocoding Web Service as provided by GeoNames [Wic10], the context data can be filled with region and country information.

Web Services can derive the user's location and time-zone from the request's IP address.

---

[5]Danaher et al. note in [DME06] for the "Top 50 Websites" a maximum average page view time per website of 3,8 minutes, with median 52 seconds.

[6]This data is usually quite imprecise, since IP addresses from a locateable address block might be distributed over a wide area. Also Internet network technologies like virtual private networks (VPN) or network address translation (NAT) can lead to wrong location guesses.

Based on the calculated geographic origin of the request, the user's time zone can be calculated and time stamp information can be shifted to match the user's time experience during the page view.

Language context is directly encoded in the HTTP header.

Another important regional aspect of the request context is the user's language. Browsers submit the `Accepted-languages` header, containing an ordered set of language identifiers the user has chosen. To get information about the user's browsing device, the submitted browser identification string (`User-Agent`) can be matched against a database of known devices, like Wurfl [Pas10].

Chapter 6

Application and Evaluation

The evaluation of the *Semantic Shadow* concept is twofold: Firstly, the usability of the web page annotation design is shown with the implementation of a core proxy system, providing dynamic context objects and *Semantic Shadow* access at specific extension points. Three different extensions covering individual usage scenarios are provided as examples. Secondly, the generation of *Semantic Shadow* annotations based on usage data analysis has been evaluated using a dedicated web application. During a one week study, usage data has been collected and the structural annotations generated based on this usage data has been compared to expert knowledge.

The usage of *SemS* annotations and their derivation from usage tracking data is examined.

6.1 Annotation Usage

In this section, three examples of annotation usage are introduced and their implementation on the basis of a *core proxy framework* is presented to show the applicability of the *Semantic Shadow* framework implementation.

For annotation usage evaluation, three scenarios are given as examples.

Scenario 1: A web designer specified different page format rules in a CSS file for different context scenarios (e. g. phone use, tablet view). To take advantage of the *Semantic Shadow* framework, annotations with a confidence value > 0.7 in the current context are transformed to CSS classnames, which are attached to the corresponding web site elements. These classnames are in the predefined CSS file connected to a certain view style, so the browser visualizes the web-page (using these styles) in a context-aware way.

Scenario 2: A restaurant offers a web page for online purchase. In the version provided by the restaurant manager, the selectable items are placed in the same order as they are on his printed menu. Using contextual information provided by the *Semantic Shadow,* the items are dynamically rearranged to match best the prospected client wishes in the current context.

Scenario 3: Due to hardware limitations, many web pages cannot be displayed the same way on a mobile phone as they are on a PC based browser. In order to provide his clients the best web experience, a mobile service provider installs a web proxy which dynamically adapts the requested web pages to the needs of the client device using annotations from the *Semantic Shadow*.

The Shadow Proxy

For adaptation implementation evaluation, an extendable proxy framework has been implemented.

All live adaptations have been implemented using the proxy approach (see Section 4.2.1). A simple Java-based proxy application has been developed, based on the open source USAPROXY implementation [WA06]. By default, this *Shadow Proxy* simply forwards incoming requests to the correspondent web server and sends the server's response back to the client application. To carry out a concrete adaptation scenario, the developer can extend the proxy using REQUESTHANDLERs and RESPONSEHANDLERs.

RequestHandlers can adapt the client request to the web server.

All REQUESTHANDLERs are called upon reception of a client HTTP request. The called handler can read and modify the request URL and the HTTP headers. In addition, the *ShadowProxy* provides an API to access the *Semantic Shadow* database connected to the proxy. For context-aware adaptation, the request's `context` object can be requested, which is ardorned using time, user, location and device information by inferring them from the HTTP headers.

ResponseHandlers can adapt the web server's response.

If the web server responses with a status code other than 200 or with a file, which does not contain HTML (i. e., is not marked as having the MIME type text/html [CM00] or text/xhtml), the response is simply redirected to the client. Otherwise, the defined RESPONSEHANDLERs are called. Every handler is notified about the HTTP header of the client request and of the server response. The handler also has access to the context object and the *Semantic Shadow* database. To perform context-dependent adaptations, the handler receives the returned HTML page either as raw byte stream, or as XML document (supporting the Simple API for XML (SAX) [Meg98], the W3C DOM API [NCH$^+$04], and the JDom API [Hun09]). This XML-view on the HTML page is provided using the HTML to XML transformation library *TagSoup* [Cow09]. The transformation is rather tricky, due to the fact that browsers interpret HTML code rather forgivingly and page editors tend not to care too much about specifications. Finally, the handler is required to output an HTML document, which can either be the same as its input or an adaptation of it.

6.1.1   CSS Class Annotation

Shadow Annotations can be transformed to CSS classnames.

Regarding single web site elements, a direct way to hand over annotation information from the Semantic Shadow to the user's browser is the generation of canonical CSS classes. These classes are then directly attached to the corresponding elements by the proxy. If a web site sup-

ports those dynamically generated classes, it includes specialized CSS style descriptions controlling the formatting of the annotated elements (see Figure 6.1).
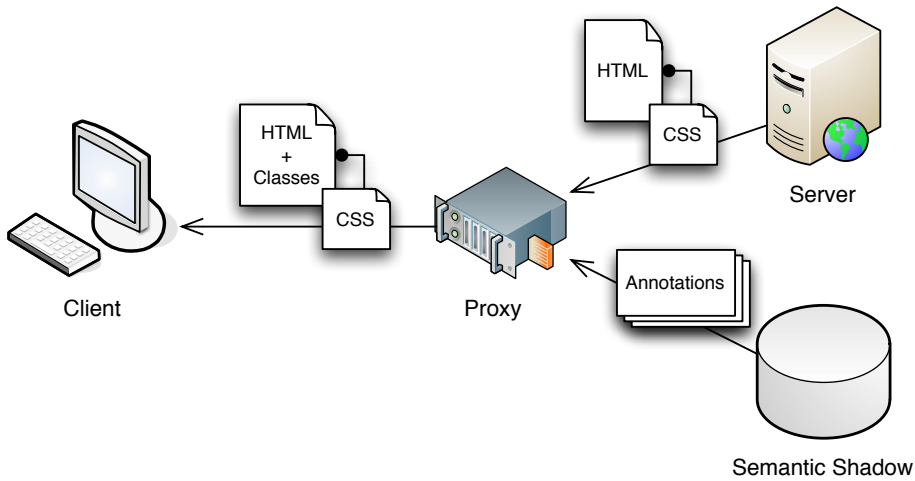
Figure 6.1: A proxy application attaches CSS classnames based on annotations on the web site elements.

To generate a "canonical classname" (CCN) which can be referenced from the CSS file, any prefix is stripped from the annotation type's URI and the string representations of the annotation's properties are, separated by an hyphen ("-") character, concatenated to the name in their defined order. Any character invalid for a CSS class name, including "-" and "_" are escaped using urlencode, while the "%" is replaced by an underline character.

This transformation is character-based and strips URI-prefixes.

If an annotation with an element subject s has, in a given request context c, a confidence value $\gamma_c > \delta$ for a configurable threshold δ (e.g. $\delta = 0.75$), the canonical classname <ccn> is attached to s in the responded HTML code. Additionally, a classname of the form <ccn>-g is attached to s, where g is defined as $g = \lfloor 5 * \gamma_c + 0.5 \rfloor \in 0, 1, \ldots 5$. This simplifies the confidence value space to six partitions. Some examples for generated classnames are given in Table 6.1.

Classnames are generated based on a value threshold, or by mapping the confidence value.

Annotation	γ_c	Canonical Classname	Additional Classname
sems:receivesKeypresses	0.9	receivesKeypresses	receivesKeypresses-5
sems:hasValueLength(5)	0.8	hasValueLength-5	hasValueLength-5-4
sems:hasPriority(7)	0.2	–	hasPriority-7-1

Table 6.1: Mapping to CSS classnames for a request context c with threshold $\delta = 0.75$.

For practical reasons, the set of annotations which are mapped to CSS classnames are restricted in the proxy's configuration, as well as the generation of the additional classnames is optional (might as well be triggered by a second threshold) to avoid "classname congestion".

Not all annotations are transformed for not to overload the HTML file.

6.1.2   Dynamic Restaurant Menu

Using user profiles, a restaurant menu can be sorted by relevance.

If a restaurant web page is dynamically generated from a database which contains information about the clients and their recent purchases, the page rendering software is able to generate a client-specific menu page. On this page, the most probable choice can for example be rendered first to reduce the navigation time and speed up the order process. By matching different user profiles this adaptation might abstract from the specific client and also allow adaptation for first time visitors. If no such database exists (e. g. for privacy reasons) or the adaptation should be performed by an intermediate subject (e. g. the cell network provider), the adaptation has to be applied without access to purchase data.

Such profiles can be replaced by *SemS* annotations.

Using L2 or L3 tracking information (see Section 5.1) of a static restaurant menu web page (Figure 6.2a), the users' preferences with respect to a given context can be determined, e. g. using frequency analysis. These preferences can be expressed using the `sems:hasPriority` annotation, assigning a rational priority value $p \in [0, 1]$.

Sortable lists can be identified in source using a simple element pattern.

The algorithm implemented to carry out this annotation based adaptation analyzes web pages for the subsequent occurrence of the same HTML tag (e. g. `<p>`, `<li>`) on the same DOM level.[1] For these elements, the `sems:hasPriority` annotation is queried from the *Semantic Shadow* with respect to the current request context. The priority property value is then multiplied with the annotation's contextual confidence value γ to reflect a contextual priority value. All subsequent page elements are finally reordered using the contextual priority value (Figure 6.2b). For any element which is not annotated, a default priority of 0.5 is assumed.

6.1.3   Adaptation for Mobile Web Access

Driven by *SemS* annotations, several adaptations for mobile web usage can be implemented.

Many web page adaptations for mobile web usage have been discussed, taking into account the limited visualization capabilities of mobile devices and their restrictions with respect to user input (see Section 3.2.1). Besides the already presented adaptation to simplify menu ordering by item rearrangement, several other adaptation examples have been implemented driven by *Semantic Shadow* adaptations:

Mobile Browser Detection Evaluating the device sensor values of the context object provided by the *ShadowProxy* framework, mobile page access can be distinguished from classical web browsing. Based on this discrimination, further filter options can be dynamically enabled or disabled.

Input Field Length Optimization This adaptation maps the value of the `hasValueLength` annotation with the highest contextual confidence in the evaluation context to the `size` attribute of the

---

[1] Alternatively, an $isMember$ annotation could be evaluated, if present.
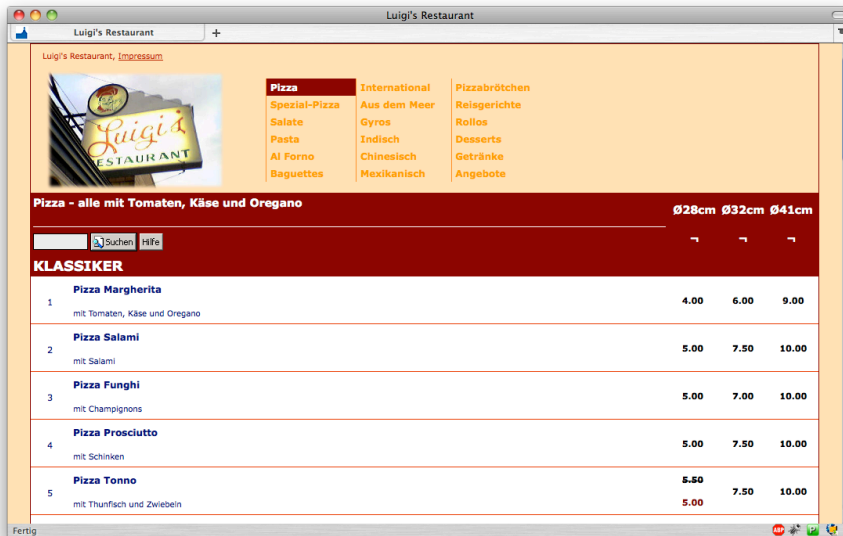
(a) The restaurant menu page as specified by the page editor.
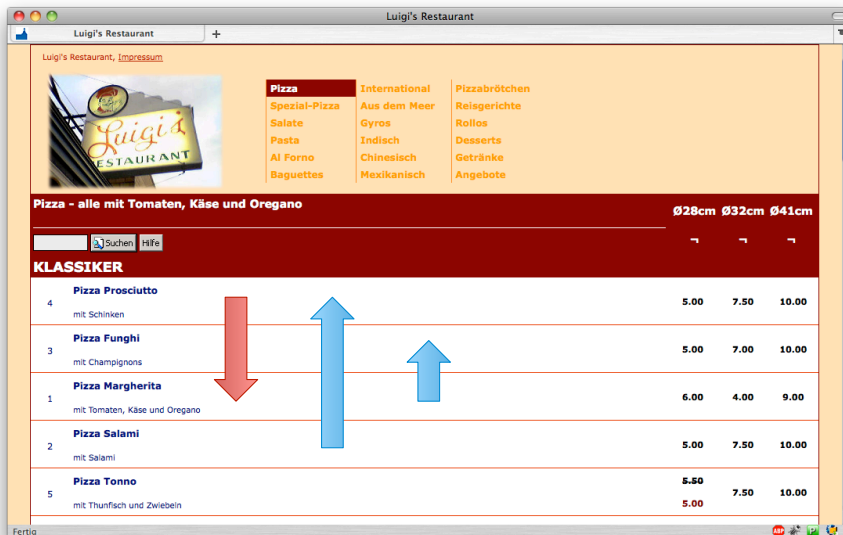
(b) The adapted menu page respecting context-dependent priority annotations.

Figure 6.2: A static restaurant menu page (a) is dynamically adapted to the request context by considering usage derived *SemS* annotations (b).

corresponding input element. This adapts the visual size of an input field in a context-aware manner to the most probable length, giving the user unobtrusive assistance in filling out the web form.

Tab Order Adaptation The `tabindex` attribute of HTML allows the definition of an order, in which input fields are entered when the user presses the tabulator key. Evaluating the `hasFocusFollower` annotations of the elements on a page, a context-aware tabulator order is derived. In the case of cycles, the cycle element occurring first in the web page DOM is selected as starting point.

Priority Based Filtering To reduce the information overload on a small devices, it makes sense to hide on a first page view elements which are unlikely to be relevant to the user in his current context [STHK01]. With *SemS* annotations, this adaptation can be implemented by evaluating the elements' `hasPriority` annotations in combination with their contextual confidences in the request context. Only elements with a defined priority that exceeds a given threshold will be rendered on the page, while the remaining elements are filtered out.

Content Folding While filtering is one option to reduce information overload, another approach is to hide details only and to show them upon request. This technique is also known as *folding*: A small element acts as representative for a longer text passage, which is folded out upon explicit user request. The `isSummaryOf` annotation allows to identify such representatives (usually a headline, but also any other element like an image can be assigned this annotation). During the adaptation, JAVASCRIPT code is inserted into the web page which dynamically hides and shows the summarized page elements, showing one detail view at a time.

Paginate Using the `isMemberOf` annotation, it is possible to separate a web page in logical subgroups. This adaptation scans the page for such groups (e. g. separated news articles), which are marked using the `hasRole` annotation as `content`. To reduce the amount of elements presented to the mobile user, only one of those groups is shown at a time, providing simple navigation buttons above and below the groups for navigation.

Auto Fill Many browsers assist the user by providing selectable options on free text input fields or pre-fill such fields with values used before on this page. Since this auto-fill algorithm only works on a per-user basis, the user cannot take advantage of values entered by other users in a similar context. In addition, most pre-filling algorithms do not evaluate the values already filled in by the user on the same page. Assuming such values have been persisted using the `induces` annotations, the adaptation dynamically extends the user's input to the most probably value, using AJAX techniques.

Annotation Embedding To prepare further client-based adaptations, this "adaptation" introduces new attributes on annotated elements in the HTML page. Only those annotations exceeding a predefined threshold in the contextual confidence value for the request context are included in the embedding, performing the context awareness on the proxy side while leaving the concrete page adaptation to a client-based algorithm. The new element attributes are named `sems-`*TypeName* referencing the value of the annotations' first property. If there are several annotations creating the same attribute name, the annotation with the highest contextual confidence is used. In case of `hasPriority` with specified group, the attribute name is extended by `-in-`*groupName*.

6.1.4 Offline Adaptation

In Section 4.2.2, the idea of generating web page variants based on *Semantic Shadow* annotations has been introduced. This approach is accompanied by a mapping of the request context to predefined context classes. Using a representative of these context classes as request context, the adaptation algorithms from the last sections can be executed in order to generate the corresponding page variant. These algorithms can be more complex in the case of an off-line variant generation, as execution time limits are somehow more relaxed than in the live adaptation case.

Offline adaptation is similar to live adaptation, if executed in the scope of a context class representative.

For the mapping of concrete request contexts to a generic context class, no universal recipe can by given, since the nature of the web site determines possible request partitions. In general, the reason for mapping context to classes is the context simplification for the variant generation process. Less variants are generated considering that adaptations might not be as precise as in the individual adaptation case. The reduction of the context representation to the value of just one, relevant sensor and the segmentation of the value sensor range into a finite number of segments defines a simple mapping algorithm as well as an easy choice for a canonical (representative) concrete context for each class. For an example, see Figure 6.3. If several sensors have been identified to be relevant for the the adaptation process, the cross product of the corresponding simple context classes can be used as request-context-space partition.

Context class definition is application-specific.
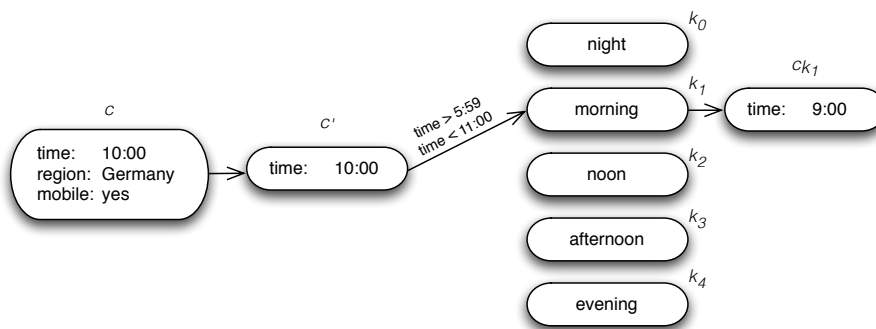
Figure 6.3: A concrete context c is reduced to a simplified version c'. It is then assigned to a sensor value range from which one value has been selected to represent the context class k_1. For web page variant generation, this value is used to construct a representative context c_{k_1}.

If the complete web page variant generation is triggered regularly, only a suitable context class identifier is required to distinguish generated variants: The generation can be performed by simulating a page request using a representative context for every context class and storing the result as HTML file (or some kind of template in the case of dynamic web content), including the class identifier in the filename.

For caching, every context class needs an identifier.

If the generation process is incremental, modifications of the web page and the *Semantic Shadow* must trigger variant updates.

In the case where variants are generated with a lazy technique (i. e., only upon the first request mapped to the corresponding context class, see Figure 4.6) or the variant generation should be optimized by avoiding the regeneration of the same variant, further meta information for each variant has to be stored: A time-stamp, representing the current state of the adapted web page and a time stamp representing the current state of the *Semantic Shadow* with respect to the adapted page. If the web page itself or the annotation data extending the web page changes beyond a certain limit, a variant must be marked as "outdated" and a variant regeneration has to be performed. For this, a time-stamp technique must be supported by the underlying resource system and the *Semantic Shadow*.

6.2   Usage Data Analysis

Usage data has been analyzed in a web based study.

To evaluate the derivation of semantic annotations from tracked usage data using the strategies presented in Section 5.2, a web based study has been conducted.

Usage data has been tracked while participating in an online survey.

During this study, the participants were invited to participate in a web survey about a subject different from this thesis' one [Blu10]. Apart from the survey's explicit intension, the implicit interaction with the survey was recorded using the L2 and L3 tracking techniques presented in Section 5.1. The structural annotations derived from the collected usage data were evaluated and visualized on the rendering of the corresponding web page to allow a correspondence verification with the semantical structure of the page (See Appendix B). The evaluation process is documented in the following sections.

6.2.1   Survey Design

On the first page of the survey, the user provided context information and agreed to the data collection.

The survey consisted of four pages: On the introductory page, the user was introduced into the survey's topic and informed about the usage data tracking. In a form, the user entered "statistical data", i. e., his age, and gender[2], confirmed that he agreed with the data collection for scientific purposes and entered the survey form by clicking a JAVASCRIPT button. This technically enforced the user to have JAVASCRIPT enabled in his browser, since this is essential for the following usage data tracking.

On the other pages, the actual usage data was collected.

The other pages contained the actual survey, carried out as a classical HTML page, where every page element had a unique id. At the end, the user had the option to leave his email address for participation in a small gift drawing. The usage of the pages was continuously tracked and logged on the server. After submitting every page to the server, the survey data is stored in a database to allow statistical evaluation. As a result of the last form transmission, the user received a "Thank you" page.

---

[2]Data representing context information.

In order to evaluate the strategies presented in Section 5.2, the survey intentionally included several specific structures:

- Some questions only had to be filled out if on a precedent question a specific value was selected.

- Some input values induced other input values intentionally.

- Some fields required specific input value formats (e. g. email address, numbers).

6.2.2   Analysis Results

The user tracking phase was conducted from February 1st, 2010 to February 7th, 2010. During this time, 999 page visits from 353 individuals have been captured and 414,353 user events have been recorded using USAPROXY [Att08]. Using the algorithms presented in Section 5.2, 13,178 annotations have been generated (8,462 for the first page, 4,562 for the second page, and 154 for the third page[3] of the survey). The resulting annotations exceeding a contextual confidence level of 0.3 for an unrestricted context and based on more than ten user interaction observations have been visualized on the corresponding web pages (see Appendix B for selected visualizations on the first page of the survey). These visualizations have been compared to the semantical content structure of the page and to the results of the survey [Blu10][Chapter 5.7]. This comparison showed that the derived annotations fitted into the perceived semantical structure of the web pages and therefore reflected well the inherent structure of the page. These annotations could now be used to implement a smart view of the survey, pre-filling default choices in function of earlier input or hiding choices irrelevant due to the request context.

In summary, deriving *Semantic Shadow* annotations from web page usage data has been shown to be accomplishable. Nevertheless, the inclusion of the derivation and analysis algorithms into a production environment would most probably fail with current implementations due to their long execution times. Thus faster execution environments and a further optimization of the algorithms are then required.

---

[3]Which only included the option to enter an email address to win a small gift.

Chapter 7

Summary and Further Work

7.1 Summary and Contributions

In this thesis, the general idea of annotating web page elements and their values in a context-aware way has been elaborated. This concept has been called *Semantic Shadow*, since the context-dependent annotations can be generated by evaluating the users' interaction with the corresponding web page elements and thus they can be said to persist, metaphorically, the shadow, users drop on a web page while interacting with it. In order to work out the *Semantic Shadow* concept and evaluate its general feasibility, the following contributions have been made during this research work:

1. *A concept to represent context-dependent semantics and structures of web site elements.* This concept describes the general structure of *SemS* annotations, models context-dependency and describes the possible representation of the annotations in RDF. Furthermore, a base set of contentual and structural annotation types has been worked out.

2. *A procedure to manage and process context-dependent meta information on web UIs.* This procedure is realized in providing a programming interface (API) for annotation creation, annotation retrieval, annotation manipulation and context definition. For this interface, two implementations have been developed, of which one is relying solely on Java objects and the other is using the defined RDF representation with the Jena framework [Jen10] as persistence backend.

3. *A procedure to optimize web based service UIs for specific usage scenarios.* In approaching a *SemS*-based mobile page adaptation, a proxy implementation has been developed. This proxy dynamically generates context representations based on the delivered HTTP header information and is tightly coupled with the *SemS*-API. This API can be used by request and response-handlers, which perform the concrete adaptation work. As examples for

such handlers, several mobile oriented adaptation scenarios have been implemented: mobile browser detection, input field length optimization, tab order adaptation, item rearrangement, priority based filtering, content folding, pagination, auto-fill, CSS class generation, and annotation embedding.

4. *A concept to derive context-dependent meta-information on web UIs from their usage.* The method proposed in this thesis uses JAVA-SCRIPT based user tracking information and HTTP request headers to infer structural semantics of web page elements. These structures are then persisted using *SemS* annotations. In an empirical study, the concept has been shown to be realizable.

7.2   Further Work

Further development concentrates on *Context*, *Calculation*, *Adaptation* and *Swarm Intelligence*.

This research work introduced the concept of modeling context-awareness alongside structural and contentual annotations on web page elements, and it presented a concept to generate these annotations based on the evaluation of user tracking. For both concepts, the feasibility has been shown on a basic level, but further work is cogitable in order to improve the procedure implementations and to further develop the idea of the *Semantic Shadow*. These continuations can be categorized with the terms *Context*, *Calculation*, *Adaptation* and *Swarm Intelligence*.

Context

The context model can be further extended or replaced by a dedicated management framework.

The context definition used in the current implementation of the concept is basic (see Section 2.1.2) in that it assumes a context to be describable in terms of simple key-value-pairs. In a first approach, this model can be extended to incorporate more standardized context data, as for example by supporting an extended version of UAProf [Wir01] descriptions, like realized by [KPG04]. In a second step, the complete management of context descriptions could be externalized by using a dedicated framework like the one implemented by Michael Hinz et al. in [HPF07] (see Section 3.2.2). If such an external context management supports the referencing of concrete "context instances" using URIs, and an context inclusion (see page 15) querying is provided, the integration in the *Semantic Shadow* concept is realizable.

Calculation

Calculation of the contextual confidence for contexts without predefined confidence value can be improved.

The current prototype is able to infer intermediate values for the contextual confidence on request contexts which were not explicitly defined: All explicit contexts are checked for including the request context and a weighted average of the corresponding confidence values is calculated. All including contexts are equally considered – in a further refinement, context inclusion might not be defined in a binary way only, but may allow to express the "extent" of inclusion. This can be combined with further confidence factor combination approaches, as presented in [Mer89].

Another shortcoming of the current prototypical implementation is the linear approach when analyzing user tracking records. Here, the complexity of the used algorithms can be further optimized, e. g. by using machine learning algorithms [Mit97].

Derivation algorithms can be optimized.

Adaptation

The Section 6.1 defined a basic set of mobile web page adaptation strategies, which can be implemented using the knowledge represented in structural and contentual annotations. Their current implementations show the general feasibility, but can be further developed to optimize the visualization of the content on mobile devices. Also adaptations for other scenarios than the mobile usage (e. g. accessibility support, parental controls) can be realized, even if not in the focus of this work's evaluative implementations. To address these aspects, the experience from previous work (see Section 3.2) must be taken into concern.

Adaptation algorithms can improved for visually more appealing output.

Swarm Intelligence

The approach to evaluate implicit interaction with web sites, in order to draw knowledge about the structure of the web site from it, receives its validity from the number of individuals interacting with the web site in question. Generally spoken, the more independent probes have been taken about the page, the more you can trust in conclusions drawn from frequent phenomena occurrence. This is related to the topic of "'Swarm Intelligence"' [LP00, BM08], where simple interactions of individuals produce further knowledge when combined with other individual's interactions.

Evaluating large user inputs is related to the topic of "Swarm intelligence".

In the current stage of development, the web page usage analysis is sketched unidirectionally, i. e., the usage of a web page is analyzed to extract structural information about it and adapt the same page at some other time (see Section 6.1.3). If this adaptation was be applied to the general view of the web page on which in the following new usage data would be acquired, then the usage analysis would affect itself. This self-interaction (which would be distributed over all individuals using the page) might trigger further effects, for which the knowledge from swarm intelligence research could provide explanations.

Applying knowledge from "Swarm Intelligence" research would require a direct feedback loop.

Appendix A

Prototype Implementation

To demonstrate the applicability of the *Semantic Shadow* concept, a prototypical implementation of the annotation management framework, as well as framework applications for annotation generation and usage have been implemented in the course of this thesis. This appendix introduces shortly into the available projects and highlights, after some general remarks, interesting aspects of each implementation.

Prototypes to generate, manage and use SemS annotations have been developed.

A.1   General Remarks

The prototypical implementation was designed in order to understand the *Semantic Shadow* concept and to demonstrate a way of representing *Semantic Shadow* annotations and their processing. Due to this academic approach, only very few optimizations have been applied to the framework code so as to simplify the processing algorithms and their understanding. As implementation language, the academic quasi-standard Java 6 [Ora10b] was chosen and an open license (GPL2 [Fre91]) assigned.

The implementations focus on the academic aspect.

The code was developed completely using the agile test-first approach [Bec03], supported by a Hudson [Ora10a] based continuous integration server. In total, a 100% test coverage on non GUI-class level (93% statement coverage, 85% branch coverage) could be achieved, resulting in 1706 JUnit 4 [BG98] cases (136 classes, 16,795 LOC[1]) testing 161 framework classes (11,776 LOC). An imbalance in between writing "productive" code and testing code could be recognized while implementing, as a perceived 75% of the development time was spend in developing the tests.

The code was developed using the test-first approach.

---

[1]lines of code

A.2   SemS

The SEMS project implements the management framework for *Semantic Shadow* annotations and context representations.

The complete framework is described using interfaces, while for the *Semantic Shadow* itself, a read-only variant and a mutable interface are defined. An extension using fractions of separated numerator and denominator instead of a rational number for Contextual Confidences is defined (see Section 4.1.3).

Two implementations of the framework interface are provided. The first one manages the complete *Semantic Shadow* state in serializable Java objects. This approach on the one hand limits the access to the shadow to a single application (holding the shadow state), but on the other hand is quite fast in its runtime behavior. The second implementation maps the annotation and context structure to an RDF model using the Jena [Jen10] RDF framework (see Section 4.1.3). This allows not only to manage the shadow in memory, but also to persist the shadow state in different interchangeable file formats (RDF/XML [Bec04], Notation 3 [BL06]) and to share the shadow at runtime using a relational database system as persistency level. Unfortunately, the extended flexibility results in a longer run time. As both implementations can be used interchangeablly (and combined), depending on the actual application requirements a suitable implementation has to be chosen.

In the implementation of the context representation (which combines the representation of basic contexts and combined contexts in one class), the definition of equality based on sensor values and the calculation of the $\trianglelefteq$ operator are central (see Section 2.1). Both methods rely on the calculation of the so called *flat context*: A context representation containing only sensor-value pairs and no further sub contexts. Due to the fact that sub contexts can be flagged as being mutual exclusive, there is more than one *flat* representation of a given context. This results in the recursive implementation of getFlatContext().

```java
public List<Context> getFlatContexts() {

  Context[] subContexts = getSubContexts();
  List<Context> flatContexts = new ArrayList<Context>();

  if ((subContexts == null)||(subContexts.length == 0)) {
    flatContexts.add(newContext(getId()));
  } else if (isExclusiveSubContexts()) {
    for(Context subContext: subContexts) {
      flatContexts.addAll(subContext.getFlatContexts());
    }
  } else  {

    List<Context>[] subSubContextLists =
                        new List[subContexts.length];
```

```java
    int totalContexts = 1;
    for(int i = 0; i < subContexts.length; i++) {
      subSubContextLists[i] = subContexts[i]
                                  .getFlatContexts();
      totalContexts *= subSubContextLists[i].size();
    }

    // cross product
    boolean useCounter = (totalContexts > 1);
    for (int i = 0; i < totalContexts; i++) {
      ContextCommons context =
          newContext((id + (useCounter ? "-" + i : "")));
      int divisor = 1;
      for (int j=0; j < subSubContextLists.length; j++) {
        int currentSubContextListSize =
                          subSubContextLists[j].size();
        int index = (i / divisor)
                          % currentSubContextListSize;
        mergeSensorValues(
              (Context) subSubContextLists[j].get(index),
              context);
        divisor *= currentSubContextListSize;
      }
      flatContexts.add(context);

    }

  }

  // merge own sensor values into contexts
  int count = 0;
  for(Context flatContext: flatContexts) {
    mergeSensorValues(this,flatContext);
    count++;
  }

  return flatContexts;
}

private void mergeSensorValues(Context sourceContext,
                               Context targetContext) {
  for(String sensor: sourceContext.getSensors()) {
    for (String value: sourceContext.getValues(sensor)) {
      targetContext.addValue(sensor, value);
    }
  }
}
```

Using a modification counter, a cache for the generated flat representations can be used to optimize runtime behavior.

To check equality of two contexts, the flat contexts' sensor names and values can be concatenated to a String and by this, equality comparison can be reduced to String comparison.

```java
protected static String
    getNormalizedStringRepresentation(Context context) {
  if (context == null)
    return null;

  StringBuilder sb = new StringBuilder();
  Context[] flatContexts = context.getFlatContexts();
  if (flatContexts != null) {
    String[] flatContextsContents =
                        new String[flatContexts.length];

    for(int i = 0; i < flatContexts.length; i++) {
      flatContextsContents[i] =
                        simpleContents(flatContexts[i]);
    }
    Arrays.sort(flatContextsContents);

    for(String s: flatContextsContents) {
      sb.append(s);
    }
  }
  return sb.toString();
}
```

To implement the context inclusion test operation, for all sensors where a value is defined in the given context, the sensor values of the other context have to be compared:

```java
public boolean inContext(Context otherContext) {
  if (otherContext == null)
    // empty context is all the world
    return true;

  Context[] otherFlatContexts =
                        otherContext.getFlatContexts();
  Context[] thisFlatContexts = getFlatContexts();

  int inclusionCount = 0;
  THIS_FLAT_CONTEXT_TEST:
   for (Context thisFlatContext: thisFlatContexts) {
     // my flat context must be within the other Context
     OTHER_FLAT_CONTEXT_TEST:
     for (Context otherFlatContext: otherFlatContexts) {
       //check inclusion of all sensors:
        CHECK_SENSORS:
       for(String sensor: otherFlatContext.getSensors()) {
         String[] thisValues =
                        thisFlatContext.getValues(sensor);
         if ((thisValues == null) ||
             (thisValues.length == 0))
           continue OTHER_FLAT_CONTEXT_TEST;
           // The current otherFlatContext is too specific
```

```java
        //check if one of thisFlatContext sensor values
        // is in the otherFlatContext
        List<String> otherValuesList = Arrays.asList(
                    otherFlatContext.getValues(sensor));
        for (String value: thisValues) {
          if (otherValuesList.contains(value))
            continue CHECK_SENSORS;
        }
        continue OTHER_FLAT_CONTEXT_TEST;
        // No values for this sensor in the other context
      }
      // thisFlatContext is in the other context!
      inclusionCount++;

      // => Check next thisFlatContext
      continue THIS_FLAT_CONTEXT_TEST;
    }
  }
  return inclusionCount == thisFlatContexts.length;
}
```

A.3 SemSAnnotator

The SEMSANNOTATOR is a basic utility to create *Semantic Shadow* annotations using a graphical user interface (see screenshot in Figure A.1). The system is based on the Eclipse Standard Window Toolkit [Ecl10], but adds another layer of abstraction to adhere to the classic Model-View-Controller pattern [KP88]. This allows a good testability while at the same time a state-of-the-art UI framework is supported. At the current stage of development, it is possible to add and remove annotations by loading the corresponding web page into the application's browser and navigate in the corresponding simplified XPath view to the element in question.

The SEMSANNOTA-TOR provides a GUI for manual annotation creation.
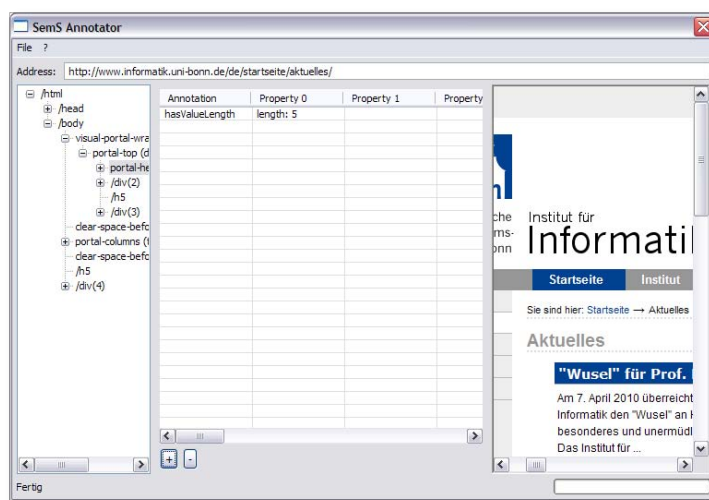
Figure A.1: Annotation creation with the SEMSANNOTATOR tool.

A.4   ShadowProxy

The SHADOWPROXY provides a base proxy implementation for *SemS* based context-aware adaptation scenarious.

The general purpose of the SHADOWPROXY has been introduced in Section 6.1. Technically, the implementation is originally based on the US-APROXY project code [WA06]. From this codebase, all usability tracking code has been removed and an API to access a *Semantic Shadow* instance has been added.

To allow extensions of the proxy the interception of HTTP requests and responses, a `RequestFilter` and a `ResponseFilter` interface have been defined. While the request filter simply allows the inspection and modification of HTTP headers and the request URL, the response filter allows furthermore processing of the responded HTML code. Depending on which methods provide an appropriate implementation, a concrete request filter can inspect the HTML stream either in raw (i.e., character oriented) format, or interpreted as an XML file. For this interpretation, the Tagsoup HTML parser [Cow09] has been integrated. The concrete request filter accesses documents either linearly using a SAX parser implementation, or with random element access using a Java W3C DOM [NCH+04] or a JDOM [Hun09] representation of the document.

Request and response filters are sequentially executed for a given request.

All filters are called in sequence, so the modifications of one filter may effect the outcome of the next filter. To allow for parallel processing with a multi-threaded architecture, the `CircularByteBuffer` implementation of Stephen Ostermiller [Ost08] has been chosen for data exchange, since the JDK's *PipedStreams* are not thread save.

A request context representation based on the HTTP header is provided.

To supply the filters with the current request context, the HTTP connection parameters are analyzed and, using Web Services from `geonames.org` [Wic10] and the Wurfl database [Pas10], a context object is constructed.

A.5   SemSAnalyzer

The SEMSANALYZER implements offline derivation of *SemS* annotations from usage data.

To perform the derivation of *Semantic Shadow* annotations from user data as introduced in Chapter 5, the SEMSANALYZER has been developed. In a linear process, the logging information of UsaProxy [WA06] is evaluated. Using the request context construction from the last section, the algorithms introduced in Section 5.2.1 are executed and generate annotations and calculate the according confidence level.

Based on this implementation, also an online derivation was realized.

With a modified version of USAPROXY, usage data analysis is possible online, where the SEMSANALYZER code is executed independently for each request. This minimizes the privacy impacts, since the usage logging information can be deleted shortly after the request itself and does not have to be stored until the central analyzing is triggered. Only derived data is persisted.

Appendix B

Derived Annotations from the Online Survey

This appendix visualizes the annotations generated from usage tracking of the exemplary online survey (cf. Section 6.2). The derived annotations with a significant contextual confidence value and usage frequency are visualized for one page of the survey, the other pages induced similar results.

Typical derived annotations from user tracking data analysis are visualized.

B.1   Commentaries to the Graphics

In the visualizations, the following abbreviations are used:

p denotes the annotation's contextual confidence value in a non-restricted context

n denotes the number of individual usage recordings which contributed to the annotation's confidence value calculation

v denotes the value of the subject of the annotation with the denoted confidence value

Type: receivesKeypresses, supportsCharset

The confidence level of these annotations has been calculated with respect to the number of page visits. Therefore input fields which received keypresses only seldom are not marked with an annotation in the visualization.

These events have been considered per page visit.

Type: hasValueLength

To all radio buttons and checkboxes of the survey, a 1-character digit has been assigned as value. Therefore, the analyzing algorithm reported 1-character value length annotations for all selected radio-buttons and checkboxes and only values for textareas are visualized in the attached graphics.

Input length of radio buttons and checkboxes has not been considered.

Type: hasDependent/dependsOn, induces/inducedBy

Dependency was filtered by focus paths.

As described in Section 5.2.1, the annotations are for complexity reasons filtered on those, which are based on the user's focus path (i. e., the order in which the user focused the page elements). Only if a user focused the subject in question, a contribution to the corresponding annotation is recorded. For the calculation of the confidence level, the frequency of the annotation with the given property object is set in relation with the frequency of annotations of the same type and subject but different property objects.

Value induction did not reveal further structural information on the elements.

The annotations with type `induces` and `inducedBy` are subsets of the annotations with type `hasDependent` and `dependsOn`, extended by the concrete element values. Since for the most part of the web page elements of the survey (all elements, except free text inputs) the value was bound to the element itself, the graphical presentation of the derived annotations does in this case not communicate new structural information and has therefore been omitted here.

Type: hasFocusFollower/followsFocus

Also focus paths with low significance are visualized.

The significance level for the contextual confidence has been chosen as 0.3 for the attached visualization to show possible meanderings from the main navigation path.

Type: hasAttentionTime

The granularity for attention time tracking was to coarse for web surveys.

A visualization for the annotation Type `hasAttentionTime` has been omitted, since all annotations with a denoted attention time $> 1sec.$ are based on less than ten user observations.

B.2   Visualization of Annotations

Arrows represent annotations in the visualisation.

To visualize the annotations derived from the usage data, arrows are drawn in the following over a facsimile of the corresponding web page rendering. The arrow itself points from the subject of the annotation to the first property object and is annotated with the confidence value in a non-restricted context. If the annotation references to one web page element only, a bubble is used as visualization.

Figure B.1: Derived annotations of type `receivesKeypresses` with $p > 0.3$ and $n > 10$

Figure B.2: Derived annotations of type `supportsCharset` with $p > 0.3$ and $n > 10$

Figure B.3: Derived annotations of type `hasValueLength` with $p > 0.3$ and $n > 10$
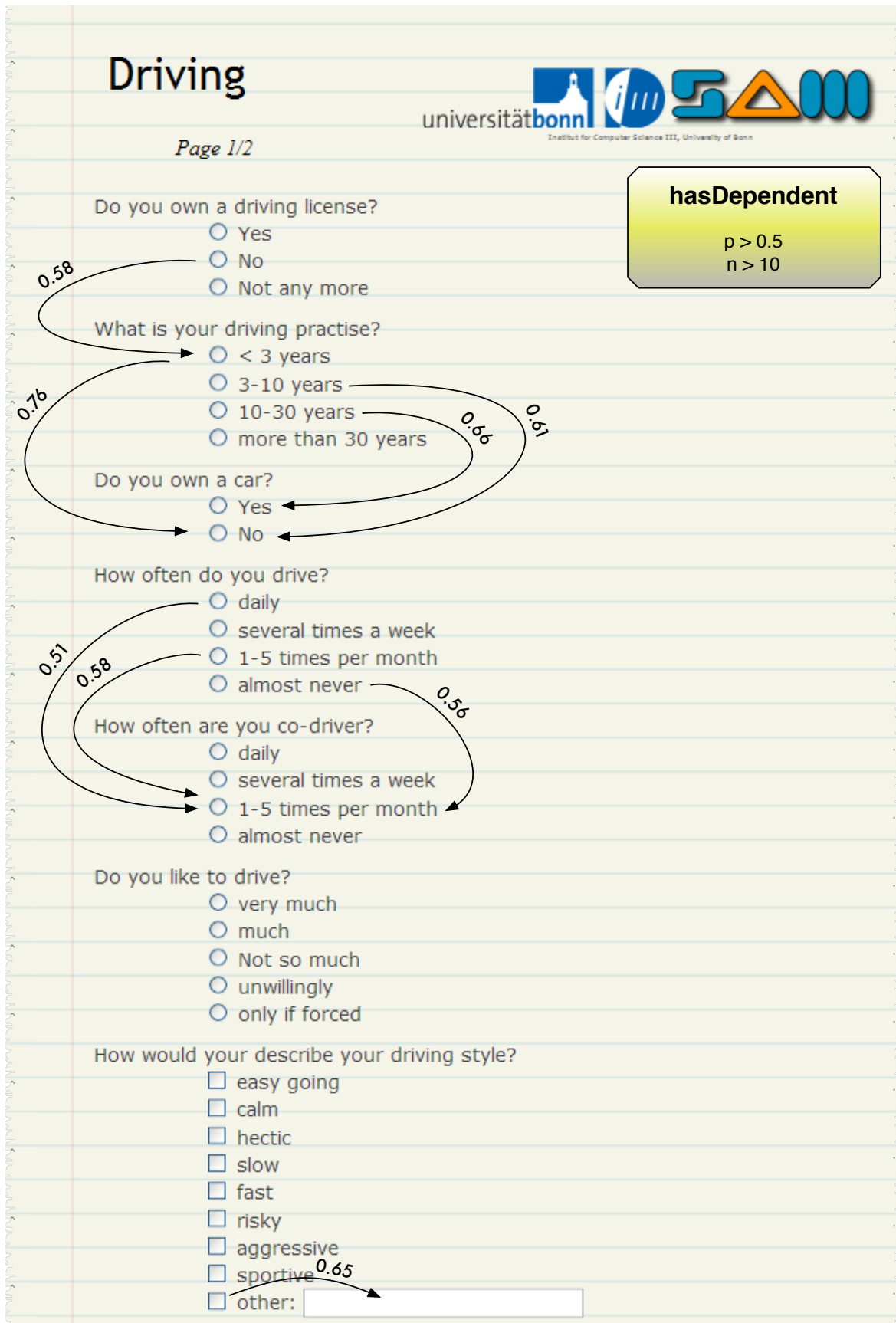
Figure B.4: Derived annotations of type `hasDependent` with $p > 0.5$ and $n > 10$ (Part 1/2)

Figure B.5: Derived annotations of type `hasDependent` with $p > 0.5$ and $n > 10$ (Part 2/2)
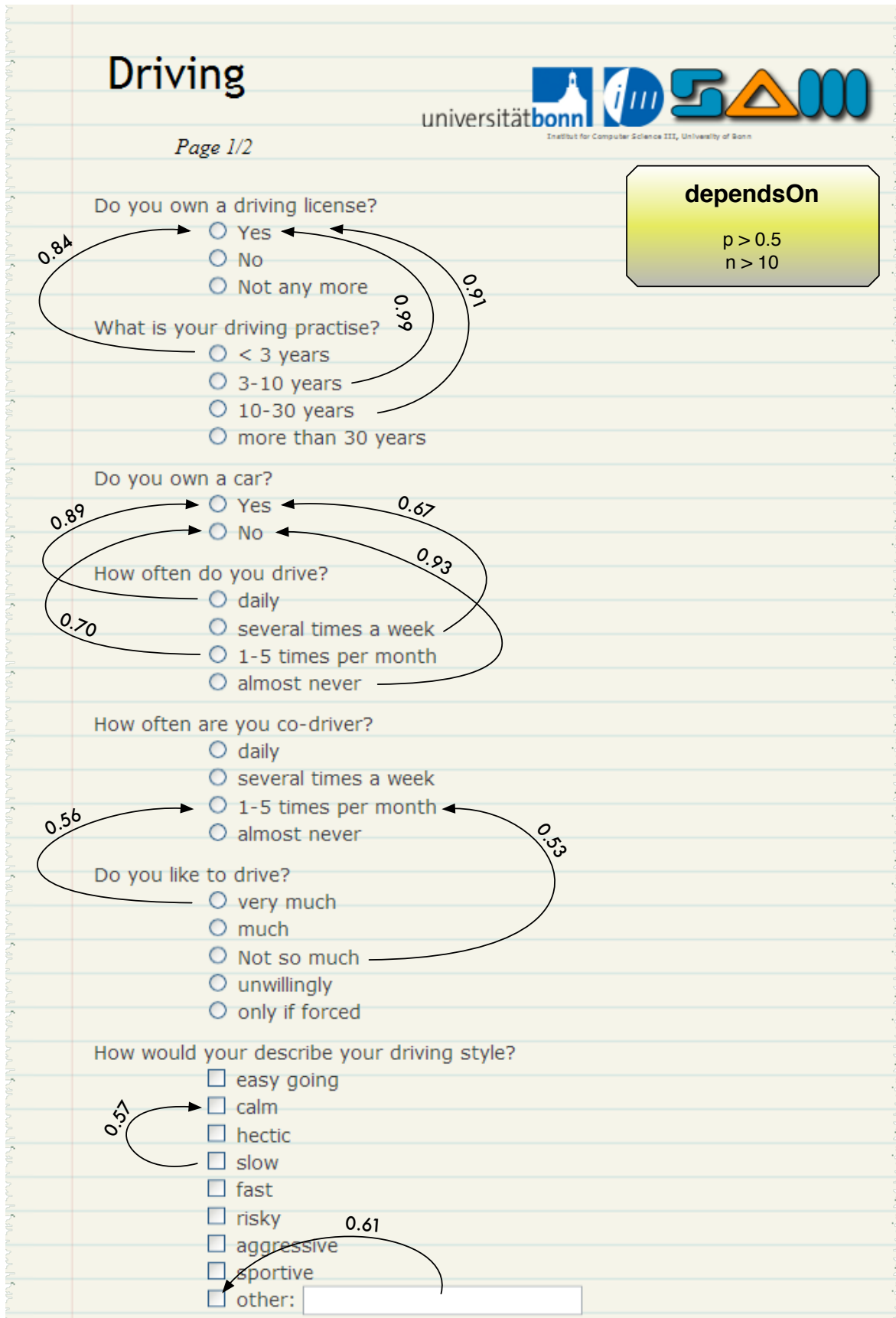
Figure B.6: Derived annotations of type `dependsOn` with $p > 0.5$ and $n > 10$ (Part 1/2)

Figure B.7: Derived annotations of type `dependsOn` with $p > 0.5$ and $n > 10$ (Part 2/2)
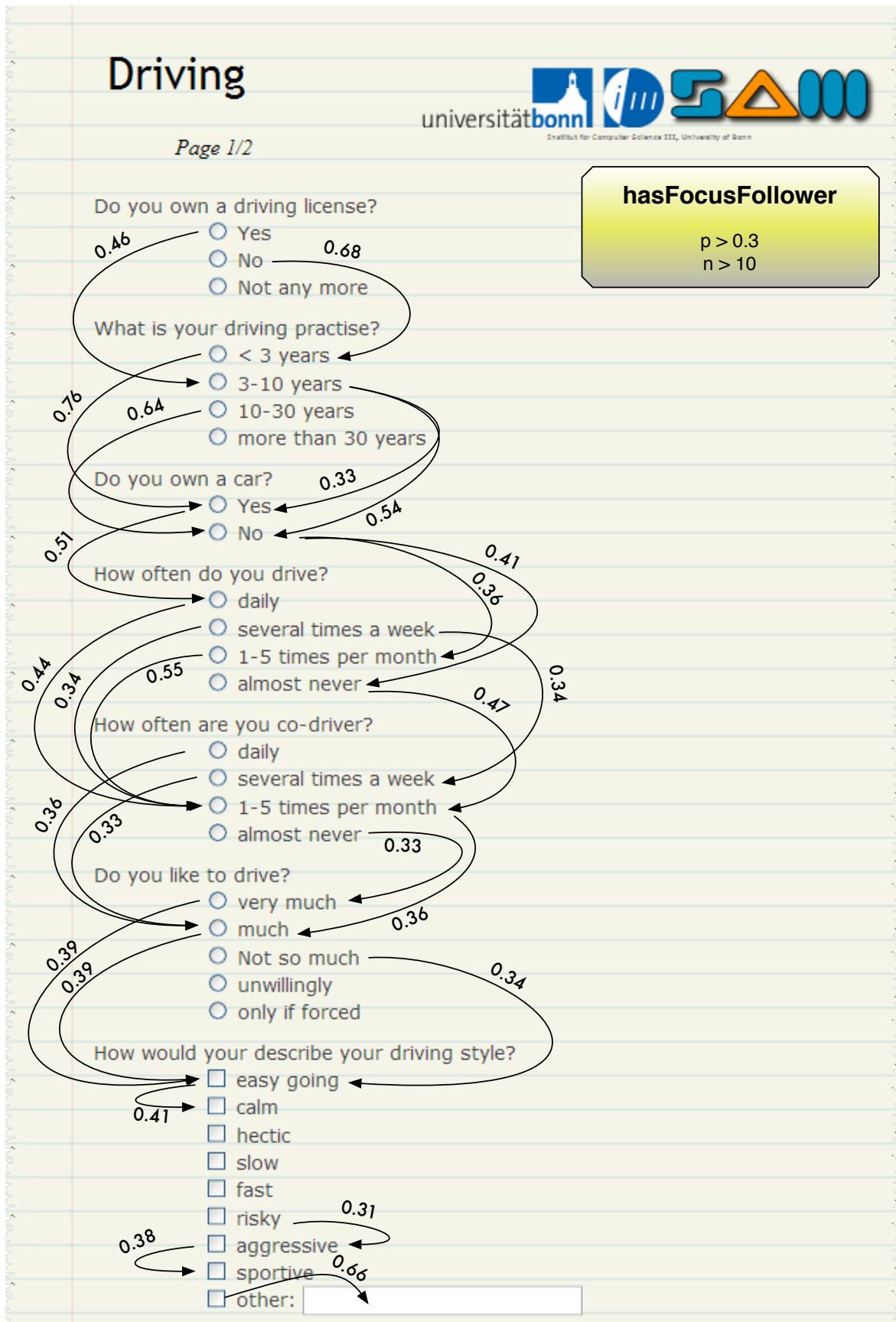
Figure B.8: Derived annotations of type `hasFocusFollower` with $p > 0.3$ and $n > 10$ (1/2)
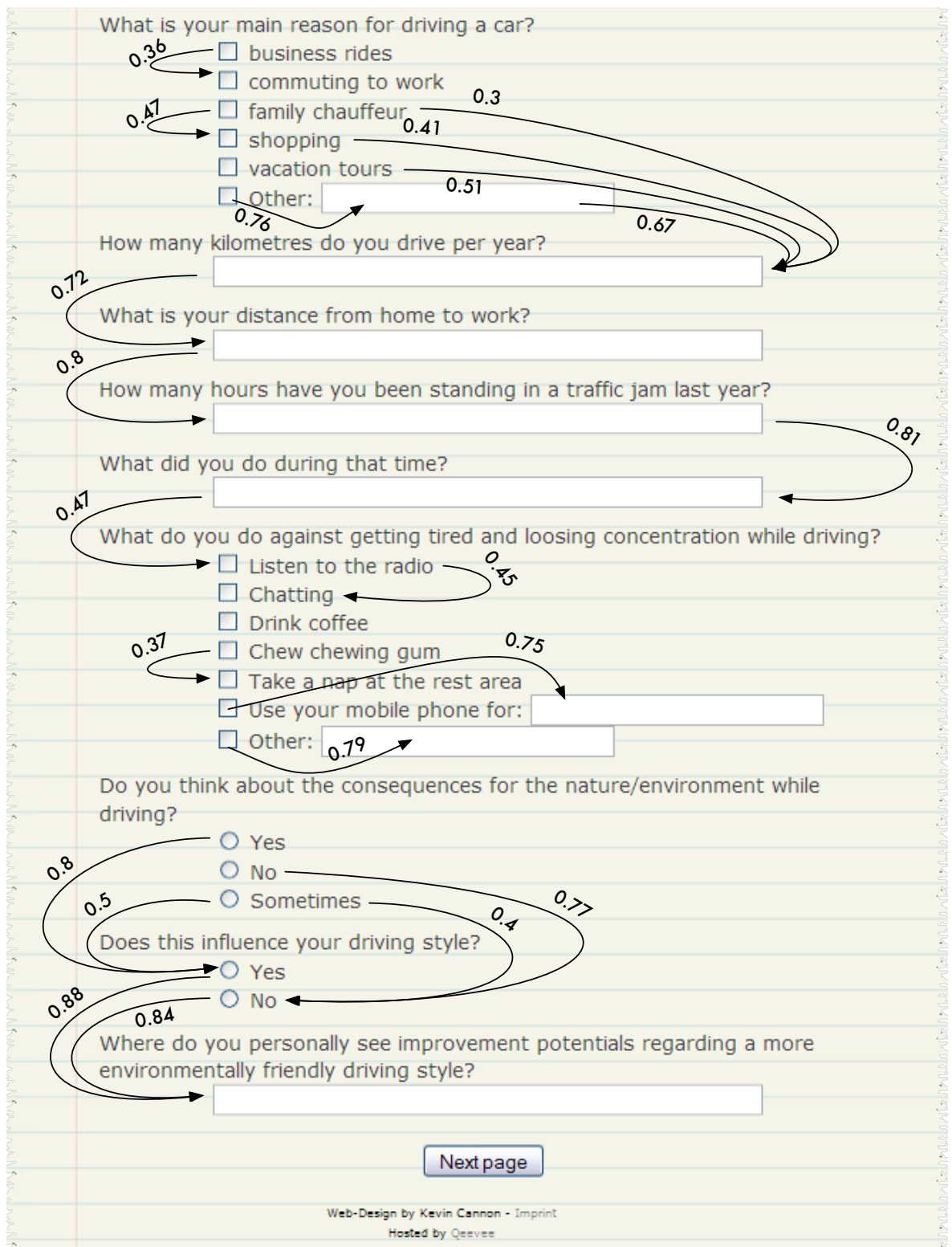
Figure B.9: Derived annotations of type `hasFocusFollower` with $p > 0.3$ and $n > 10$ (2/2)
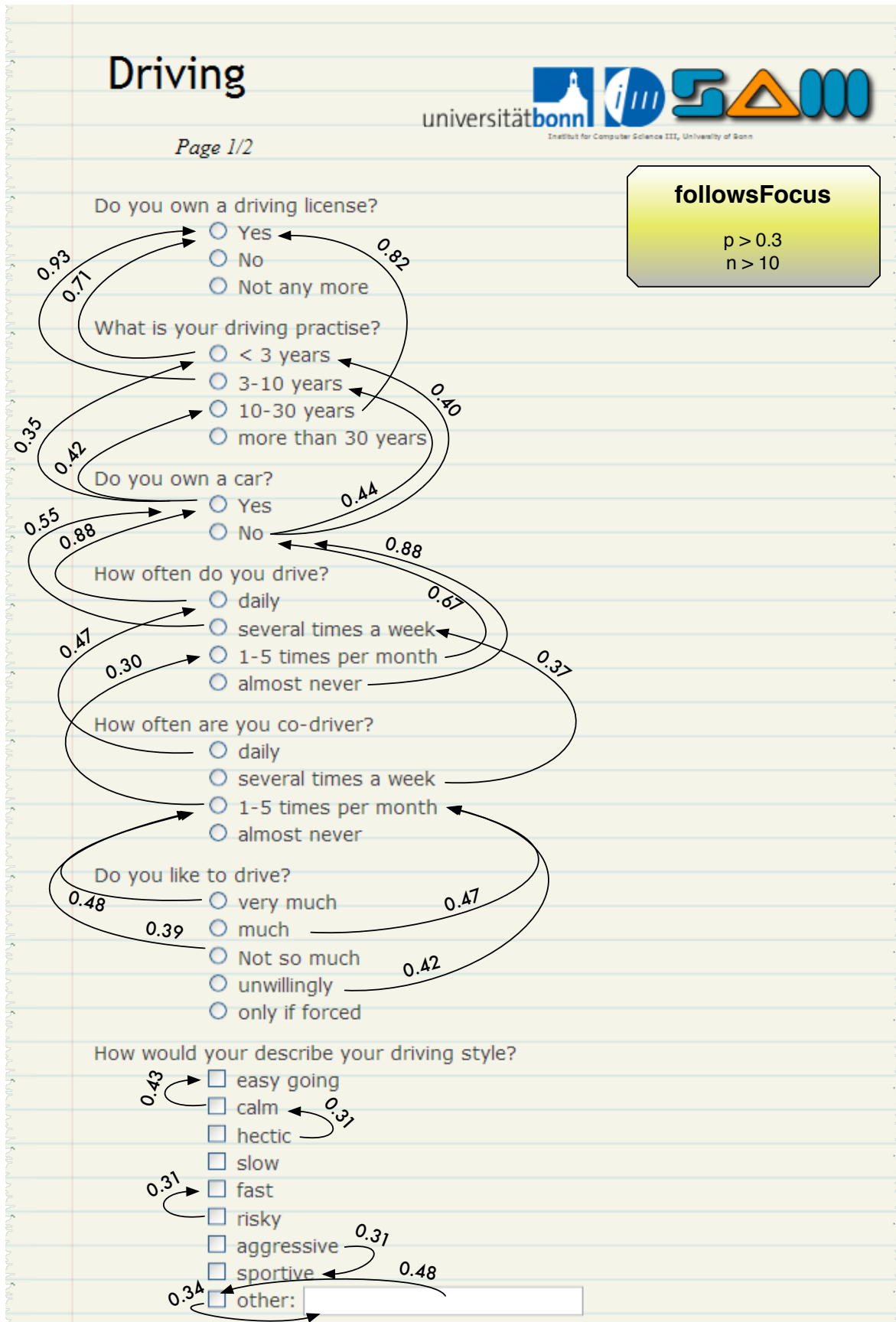
Figure B.10: Derived annotations of type `followsFocus` with $p > 0.3$ and $n > 10$ (1/2)
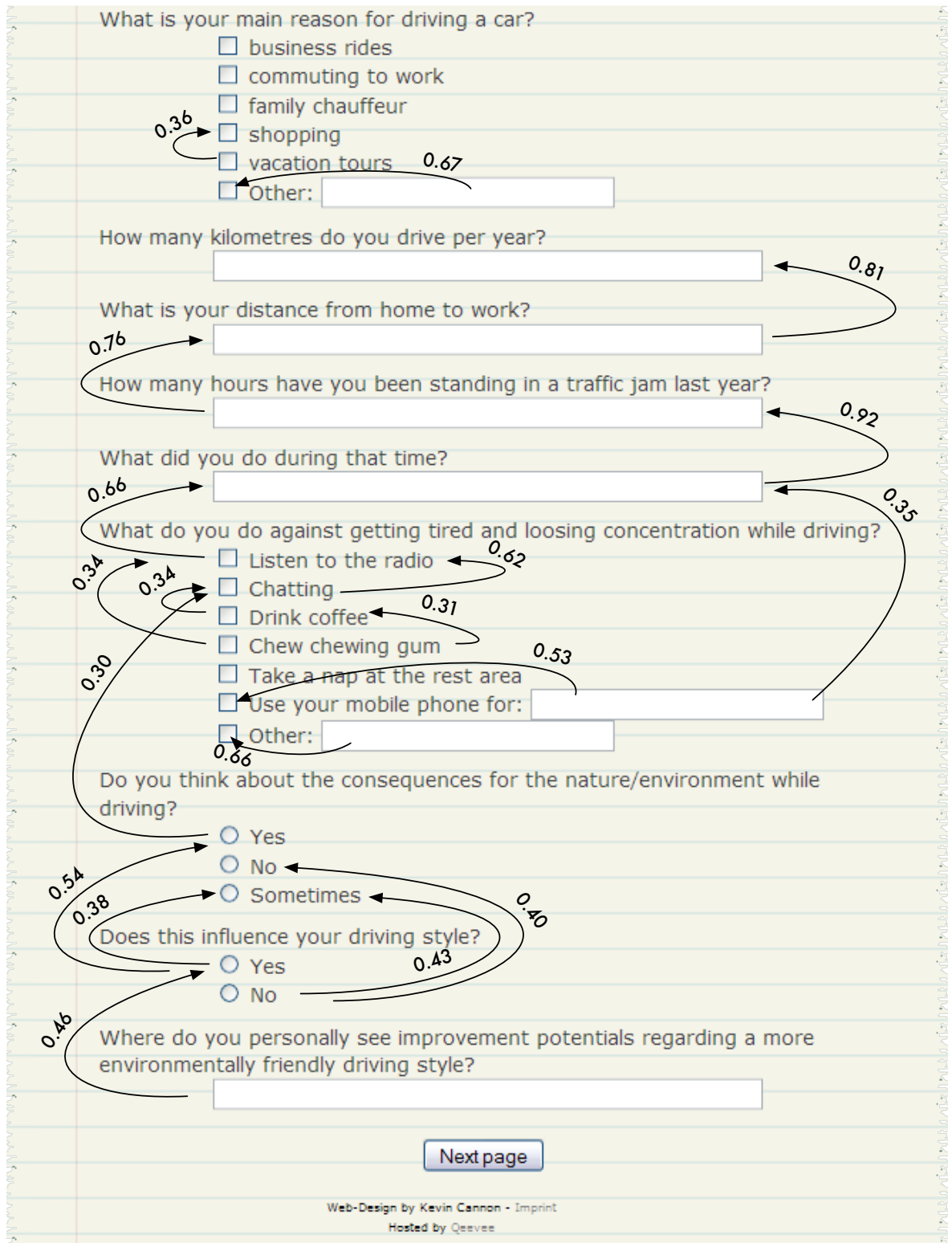
Figure B.11: Derived annotations of type `followsFocus` with $p > 0.3$ and $n > 10$ (2/2)

Bibliography

[ABH+01]   A. Ankolekar, M. H. Burstein, J. R. Hobbs, O. Lassila, D. L. Martin, S. A. McIlraith, S. Narayanan, M. Paolucci, T. R. Payne, K. P. Sycara, H. Zeng. DAML-S: Semantic Markup for Web Services. In Cruz et al. (eds.), *The Emerging Semantic Web*. Frontiers in Artificial Intelligence and Applications 75. IOS press, 2001.

[AFG+10]   M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia. A View of Cloud Computing. *Commun. ACM* 53(4):50–58, 2010. doi:10.1145/1721654.1721672

[AH03]   M. Abe, M. Hori. Robust Pointing by XPath Language: Authoring Support and Empirical Evaluation. *Applications and the Internet, IEEE/IPSJ International Symposium on* 0:156, 2003. doi:10.1109/SAINT.2003.1183044

[All97]   R. B. Allen. Mental Models and User Models. In Helander et al. (eds.), *Handbook of Human-Computer Interaction*. Pp. 49–63. Elsevier Science, 1997.

[AT00]   C. Asakawa, H. Takagi. Annotation-Based Transcoding for Nonvisual Web Access. In *Assets '00: Proceedings of the fourth international ACM conference on Assistive technologies*. Pp. 172–179. ACM, New York, NY, USA, 2000. doi:10.1145/354324.354588

[Att06]   R. Atterer. Logging Usage of AJAX Applications With the "UsaProxy" HTTP Proxy. In *Proceedings of the WWW 2006 Workshop on Logging Traces of Web Activity: The Mechanics of Data Collection*. Edinburgh, Scotland, May 2006.

[Att08]   R. Atterer. *Usability Tool Support for Model-Based Web Development*. PhD thesis, Faculty of Mathematics, Computer Science and Statistics, University of Munich, 2008.

[AWS06]   R. Atterer, M. Wnuk, A. Schmidt. Knowing the User's Every Move - User Activity Tracking for Website Usability Evaluation and Implicit Interaction. In *Proceedings of the 15th International World Wide Web Conference (WWW2006)*. Edinburgh, Scotland, May 2006.

[BA08]   M. Birbeck, B. Adida. RDFa Primer. W3C note, W3C, October 2008. accessed: 2010-01-19. http://www.w3.org/TR/2008/NOTE-xhtml-rdfa-primer-20081014/

[Bar10]   J. Barnabe. Restyle the web with Stylish! 2010. accessed: 2010-11-15. http://userstyles.org/

[BBB02]   B. Baker, T. Bohannon, T. Barbee. WebSphere Transcoding Publisher. online, April 2002. accessed: 2010-11-03. http://www.ibm.com / developerworks / websphere / library / techarticles/0204_baker/0204_baker_integrating.html

[BBP04]   G. Berhe, L. Brunie, J.-M. Pierson. Modeling Service-Based Multimedia Content Adaptation in Pervasive Computing. In *CF'04: Proceedings of the first conference on Computing frontiers*. Pp. 60–69. ACM Press, New York, NY, USA, 2004. doi:10.1145/977091.977102

[BDG06]   F. Bobillo, M. Delgado, J. Gómez-Romero. A Crisp Representation for Fuzzy SHOIN with Fuzzy Nominals and General Concept Inclusions. In *In Proc. of the 2nd International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 06)*. 2006.

[BDG08]   F. Bobillo, M. Delgado, J. Gómez-Romero. DeLorean: A Reasoner for Fuzzy OWL 1.1. In Bobillo et al. (eds.), *URSW*. CEUR Workshop Proceedings 423. CEUR-WS.org, 2008. http://ceur-ws.org/Vol-423/paper2.pdf

[Bec03]   K. Beck. *Test Driven Development: By Example*. Addison-Wesley, 2003.

[Bec04]   D. Beckett. RDF/XML Syntax Specification (Revised), W3C recommendation. Technical report, World Wide Web Consortium, 2004. http://www.w3.org/TR/rdf-syntax-grammar/

[BL06]   T. Berners-Lee. Notation3 (N3) A readable RDF syntax. online, March 2006. accessed: 2010-11-13. http://www.w3.org/DesignIssues/Notation3.html

[BG98]   K. Beck, E. Gamma. Test Infected: Programmers Love Writing Tests. *Java Report* 3(7):51–56, 1998. http://members.pingnet.ch/gamma/junit.htm

[BGP00]   O. Buyukkokten, H. Garcia-Molina, A. Paepcke. Focused Web Searching with PDAs. *Computer Networks* 33(1–6):213–230, 2000. doi:10.1016/S1389-1286(00)00060-8

[BGPW00] O. Buyukkokten, H. Garcia-Molina, A. Paepcke, T. Winograd. Power Browser: Efficient Web Browsing for PDAs. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. CHI '00, pp. 430–437. ACM, New York, NY, USA, 2000. doi:10.1145/332040.332470

[BHL01]   T. Berners-Lee, J. Hendler, O. Lassila. The Semantic Web. *Scientific American* 284(5):34–43, 2001. http://www.scientificamerican.com/article.cfm?id=the-semantic-web

[Blu10]   D. Blumenau. *Entwicklung mobiler, orts- und kontextbasierter Spiele für Fahrer auf Basis der Fahrzeugsensorik*. Diplomarbeit, Institut für Informatik, Universität Bonn, Römerstr. 164, July 2010.

[BM08]   C. Blum, D. Merkle (eds.). *Swarm Intelligence: Introduction and Applications*. Natural Computing Series. Springer, 2008. doi:10.1007/978-3-540-74089-6

[BPM+08]   T. Bray, J. Paoli, E. Maler, F. Yergeau, C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0 (Fifth Edition). W3C recommendation, W3C, November 2008. accessed: 2010-01-18. http://www.w3.org/TR/2008/REC-xml-20081126/

[BS97]   T. W. Bickmore, B. N. Schilit. Digestor: device-independent access to the World Wide Web. *Computer Networks and ISDN Systems* 29(8-13):1075–1082, 1997. Papers from the Sixth International World Wide Web Conference. doi:10.1016/S0169-7552(97)00026-3

[BS08]   F. Bobillo, U. Straccia. fuzzyDL: An Expressive Fuzzy Description Logic Reasoner. June 2008. http://gaia.isti.cnr.it/straccia/software/fuzzyDL/documents/fuzzyDL.pdf

[BS09]   F. Bobillo, U. Straccia. An OWL Ontology for Fuzzy OWL 2. In *ISMIS '09: Proceedings of the 18th International Symposium on Foundations of Intelligent Systems*. Pp. 151–160. Springer-Verlag, Berlin, Heidelberg, 2009. doi:10.1007/978-3-642-04125-9_18

[BSS05]   P. Bouquet, L. Serafini, H. Stoermer. Introducing Context into RDF Knowledge Bases. In *In Proceedings of SWAP 2005, the 2nd Italian Semantic Web Workshop*. Pp. 14–16. 2005.

[Car10]   M.-A. Caron. IPInfoDB – Free IP address geolocation tools. 2010. accessd: 2010-12-02. http://ipinfodb.com/

[CD99]   J. Clark, S. DeRose. XML Path Language (XPath). November 1999. accessed: 2009-09-28. http://www.w3.org/TR/1999/REC-xpath-19991116

[CDF+98]   M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, S. Slattery. Learning to Extract Symbolic Knowledge from the World Wide Web. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*. AAAI '98/IAAI '98, pp. 509–516. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1998.

[CFB00]   S. Ceri, P. Fraternali, A. Bongio. Web Modeling Language (WebML): a modeling language for designing Web sites. *Computer Networks* 33(1–6):137–157, 2000. doi:10.1016/S1389-1286(00)00040-2

[CK04]   J. J. Carroll, G. Klyne. Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C recommendation, W3C, February 2004. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

[CM00]   D. Connolly, L. Masinter. The 'text/html' Media Type. June 2000. accessed: 2009-09-28. http://www.ietf.org/rfc/rfc2854.txt

[CMZ03]   Y. Chen, W.-Y. Ma, H.-J. Zhang. Detecting Web Page Structure for Adaptive Viewing on Small Form Factor Devices. In *WWW '03: Proceedings of the 12th international conference on World Wide Web.* Pp. 225–233. ACM, New York, NY, USA, 2003. doi:10.1145/775152.775184

[Cow09]   J. Cowan. TagSoup - Just Keep On Truckin'. October 2009. accessed: 2010-11-05. http://tagsoup.info

[CS95]   I. Cooper, R. Shufflebotham. PDA Web Browsers: Implementation Issues. Technical report 11-95*, University of Kent, Computing Laboratory, University of Kent, Canterbury, UK, November 1995. http://www.cs.kent.ac.uk/pubs/1995/57

[DCM08]   DCMI. Dublin Core Metadata Element Set, Version 1.1. January 2008. accessed: 2010-01-13. http://dublincore.org/documents/2008/01/14/dces/

[DEG+03]   S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. A. Tomlin, J. Y. Zien. SemTag and Seeker: Bootstrapping the Semantic Web via Automated Semantic Annotation. In *WWW '03: Proceedings of the 12th international conference on World Wide Web.* Pp. 178–186. ACM, New York, NY, USA, 2003. doi:10.1145/775152.775178

[Dey98]   A. K. Dey. Context-Aware Computing: The CyberDesk Projects. In *AAAI 1998 Spring Symposium on Intelligent Environments.* Pp. 51–54. AAAI Press., Palo Alto, 1998. http://www.cc.gatech.edu/fce/cyberdesk/pubs/AAAI98/AAAI98.html

[Dey01]   A. K. Dey. Understanding and Using Context. In *Personal and Ubiquitous Computing.* Volume Februray, pp. 4–7. Springer-Verlag London Ltd, 2001.

[DME06]   P. J. Danaher, G. W. Mullarkey, S. Essegaie. Factors Affecting Web Site Visit Duration: A Cross-Domain Analysis. In *Journal of Marketing Research.* Volume 43(2), pp. 182–194.

American Marketing Association, May 2006. doi:10.1509/jmkr.43.2.182

[Dou01]   P. Dourish. Seeking a Foundation for Context-Aware Computing. *Hum.-Comput. Interact.* 16(2):229–241, 2001. doi:10.1207/S15327051HCI16234_07

[Dou04]   P. Dourish. What we talk about when we talk about context. *Personal Ubiquitous Comput.* 8(1):19–30, 2004. doi:10.1007/s00779-003-0253-8

[DP04]   Z. Ding, Y. Peng. A Probabilistic Extension to Ontology Language OWL. In *HICSS*. 2004. doi:10.1109/HICSS.2004.1265290

[DRSC09]   M. Dyck, J. Robie, J. Snelson, D. Chamberlin. XML Path Language (XPath) 2.1. W3C working draft, W3C, December 2009. accessed: 2010-01-18. http://www.w3.org/TR/2009/WD-xpath-21-20091215/

[Duc07]   A. T. Duchowski. *Eye Tracking Methodology: Theory and Practice*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[DV00]   L. Denoue, L. Vignollet. An annotation tool for Web browsers and its applications to information retrieval. In *Proceedings of RIAO2000*. Paris, April 2000.

[Ecl10]   Eclipse Foundation. SWT: The Standard Widget Toolkit. 2010. accessed: 2010-11-16. http://www.eclipse.org/swt/

[FC96]   M. Fayad, M. P. Cline. Aspects of Software Adaptability. *Commun. ACM* 39(10):58–59, 1996. doi:10.1145/236156.236170

[FDES98]   D. Fensel, S. Decker, M. Erdmann, R. Studer. Ontobroker: Or How to Enable Intelligent Access to the WWW. In *11th Workshop on Knowledge Acquisition, Modeling, and Management (KAW '98)*. AAAI Press, 1998.

[FGM+99]   R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Updated by RFC 2817. http://www.ietf.org/rfc/rfc2616.txt

[FH05]   Z. Fiala, G.-J. Houben. A Generic Transcoding Tool for Making Web Applications Adaptive. In Belo et al. (eds.), *CAiSE Short Paper Proceedings*. CEUR Workshop Proceedings 161. CEUR-WS.org, 2005. http://www.ceur-ws.org/Vol-161/FORUM_03.pdf

[FHMW03]   Z. Fiala, M. Hinz, K. Meißner, F. Wehner. A Component-based Approach for Adaptive, Dynamic Web Documents. *J. Web Eng* 2(1–2):58–73, 2003.

[Fre91]   Free Software Foundation, Inc. GNU General Public License, version 2. online, June 1991. accessed: 2010-11-16. http://www.gnu.org/licenses/gpl-2.0.html

[FS99]   J. Flinn, M. Satyanarayanan. Energy-aware adaptation for mobile applications. In *Symposium on Operating Systems Principles*. Pp. 48–63. 1999. doi:10.1145/319151.319155

[GF03]   M. Gellner, P. Forbrig. ObSys - a Tool for Visualizing Usability Evaluation Patterns with Mousemaps. In *Proceedings of the Tenth International Conference on Human-Computer Interaction*. Human factors and ergonomics 1, pp. 469–473. 2003.

[GG98]   F. Giunchiglia, C. Ghidini. Local Models Semantics, or Contextual Reasoning = Locality + Compatibility. In Cohn et al. (eds.), *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR-98)*. Pp. 282–291. Morgan Kaufmann Publishers, San Francisco, June 2–5 1998.

[GHJV95]   E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns – Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[GHM$^+$08] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. F. Patel-Schneider, U. Sattler. OWL 2: The next step for OWL. *Web Semantics: Science, Services and Agents on the World Wide Web* 6(4):309–322, 2008. doi:10.1016/j.websem.2008.05.001

[GJG04]   L. A. Granka, T. Joachims, G. Gay. Eye-Tracking Analysis of User Behavior in WWW-Search. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*. Pp. 478–479. ACM, New York, NY, USA, 2004. doi:10.1145/1008992.1009079

[GL05]   M. Gao, C. Liu. Extending OWL by Fuzzy Description Logic. In *ICTAI*. Pp. 562–567. IEEE Computer Society, 2005. doi:10.1109/ICTAI.2005.65

[Gol10]   S. Golesorkhi. Context Aware Dynamic Adaptation and Optimization of Web User Interfaces. Diplomarbeit, Universität Bonn, November 2010.

[GPZ04]   T. Gu, H. K. Pung, D. Q. Zhang. Toward an OSGi-Based Infrastructure for Context-Aware Applications. *Pervasive Computing, IEEE* 3(4):66–74, October 2004. doi:10.1109/MPRV.2004.19

[Gri10]   K. Grifantini. Eye Tracking for Mobile Control. online, May 2010. accessed: 2010-06-18. http://www.technologyreview.com/computing/25369/?a=f

[Gru93]   T. R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition* 5(2):199–220, 1993.

[GS00]   J. Goecks, J. W. Shavlik. Learning Users' Interests by Unobtrusively Observing Their Normal Behavior. In *IUI*. Pp. 129–132. 2000. http://10.1145/325737.325806

[Hau09]   M. Hausenblas. *Anreicherung von Webinhalten mit Semantik - Microformats und RDFa*. Pp. 147–158. Springer, 2009. doi:10.1007/978-3-540-72216-8_8

[Hay04]   P. Hayes. RDF Semantics. February 2004. accessed: 2009-09-29. http://www.w3.org/TR/2004/REC-rdf-mt-20040210/

[HB99]   P. M. Hallam-Baker, B. Behlendorf. Extended Log File Format. W3C working draft, W3C, february 1999. accessed: 2010-06-18. http://www.w3.org/TR/WD-logfile.html

[HCN08]   R. Hirschfeld, P. Costanza, O. Nierstrasz. Context-oriented programming. *Journal of Object Technology* 7(3):125–151, April 2008.

[HH09]   I. Hickson, D. Hyatt. HTML 5 - A vocabulary and associated APIs for HTML and XHTML, Chapter 5 Microdata. online, August 2009. http://www.w3.org/TR/2009/WD-html5-20090825/microdata.html

[HHL03]   J. Heflin, J. A. Hendler, S. Luke. SHOE: A Blueprint for the Semantic Web. In *Spinning the Semantic Web*. Pp. 29–63. 2003.

[HHRS08]   P. Hitzler, M. Hörtzsch, S. Rudolph, Y. Sure. *Semantic Web*. Springer Akademischer Verlag, 2008.

[HI06]   K. Henricksen, J. Indulska. Developing context-aware pervasive computing applications: Models and approach. *Pervasive and Mobile Computing* 2(1):37–64, 2006. doi:10.1016/j.pmcj.2005.07.003

[HKO+00]   M. Hori, G. Kondoh, K. Ono, S. Hirose, S. Singhal. Annotation-based Web content transcoding. *Computer Networks* 33(1–6):197–211, 2000. doi:10.1016/S1389-1286(00)00068-2

[HM08]   T. Halpin, T. Morgan. *Information modeling and relational databases*. Morgan Kaufmann Publications, 2008.

[HMHN04]   D. W. Hansen, D. J. C. MacKay, J. P. Hansen, M. Nielsen. Eye Tracking off the Shelf. In *ETRA '04: Proceedings of the 2004 symposium on Eye tracking research & applications*. Pp. 58–58. ACM, New York, NY, USA, 2004. doi:10.1145/968363.968375

[HMMS99]   M. Hori, R. Mohan, H. Maruyama, S. Singhal. Annotation of Web Content for Transcoding. July 1999. accessed: 2010-01-13. http://www.w3.org/1999/07/NOTE-annot-19990710

[HOAK04] M. Hori, K. Ono, M. Abe, T. Koyanagi. Generating transformational annotation for web document adaptation: tool support and empirical evaluation. *Web Semantics: Science, Services and Agents on the World Wide Web* 2(1):1–18, 2004. doi:10.1016/j.websem.2004.08.001

[HP05] D. W. Hansen, A. E. Pece. Eye tracking in the wild. *Computer Vision and Image Understanding* 98(1):155–181, 2005. Special Issue on Eye Detection and Tracking. doi:10.1016/j.cviu.2004.07.013

[HPB$^+$04] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean. SWRL: A Semantic Web Rule Language – Combining OWL and RuleML. online, W3C Member Submission, May 2004. accessed: 2010-11-10. http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/

[HPF07] M. Hinz, S. Pietschmann, Z. Fiala. A Framework for Context Modeling in Adaptive Web Applications. In *IADIS International Journal of WWW/Internet*. Volume 5(1), pp. 1–13. IADIS, June 2007.

[HS02] S. Handschuh, S. Staab. Authoring and Annotation of Web Pages in CREAM. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*. Pp. 462–473. ACM, New York, NY, USA, 2002. doi:10.1145/511446.511506

[HSM01] S. Handschuh, S. Staab, A. Maedche. CREAM: creating relational metadata with a component-based, ontology-driven annotation framework. In *K-CAP '01: Proceedings of the 1st international conference on Knowledge capture*. Pp. 76–83. ACM, New York, NY, USA, 2001. doi:10.1145/500737.500752

[Hun09] J. Hunter. JDOM. July 2009. accessed: 2010-11-16. http://www.jdom.org/

[IB04] S. T. Iqbal, B. P. Bailey. Using Eye Gaze Patterns to Identify User Tasks. In *Proceedings of the Grace Hopper Celebration of Women in Computing*. 2004.

[Ins04] Institute of Applied Informatics and Formal Description Methods (AIFB). Annotation Portal. 2004. accessed: 2010-01-19. http://annotation.semanticweb.org/

[Jan07] B. Jankowska. *Architectural Frameworks for Automated Content Adaptation to Mobile Devices Based on Open-Source Technologies*. PhD thesis, European University Viadrina, Frankfurt/Oder, 2007.

[Jen10] Jena Team. Jena – A Semantic Web Framework for Java. September 2010. accessed: 2010-11-13. http://jena.sourceforge.net/

[JK03]   R. J. K. Jacob, K. S. Karn. Eye Tracking in Human-Computer Interaction and Usability Research: Ready to Deliver the Promises. *The Mind's eye: Cognitive The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research*, pp. 573–603, 2003. http://www.sciencedirect.com.lt.ltag.bibl.liu.se/science

[Jud04]   A. Judt. *Konfigurierbare Benutzerschnittstellen zur Vereinfachung formularbasierter Datenerfassung*. PhD thesis, Universität Fridericiana zu Karlsruhe, October 2004.

[Kat94]   R. H. Katz. Adaptation and Mobility in Wireless Information Systems. *IEEE Personal Communications* 1:6–17, 1994. doi:10.1109/98.295355

[KC04]   G. Klyne, J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. February 2004. last visited: 2009-09-28. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

[KHPG06] A. Kalyanpur, J. Hendler, B. Parsia, J. Golbeck. SMORE – Semantic Markup, Ontology, and RDF Editor. Technical report, University of Maryland, 2006. http://www.mindswap.org/papers/SMORE.pdf

[Koi05]   M.-R. Koivunen. Annotea and Semantic Web Supported Collaboration. online, 2005. http://www.annotea.org/eswc2005/01_koivunen_final.pdf

[KP88]   G. E. Krasner, S. T. Pope. A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80. *J. Object Oriented Program.* 1(3):26–49, 1988.

[KPG04]   B. Kurz, I. Popescu, S. Gallacher. FACADE – A FrAmework for Context-aware content Adaptation and DElivery. In *Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR'04)*. 2004.

[Kre10]   J. Kreutz. Möglichkeiten zur Erkennung und Nutzung von Mustern in Webseiten mit kontextuellen Struktur-Annotationen. Diplomarbeit, Universität Bonn, Institut für Informatik III, December 2010. (to be published).

[LBP06]   D. Li, J. Babcock, D. J. Parkhurst. openEyes: a low-cost head-mounted eye-tracking solution. In *ETRA '06: Proceedings of the 2006 symposium on Eye tracking research & applications*. Pp. 95–100. ACM, New York, NY, USA, 2006. doi:10.1145/1117309.1117350

[LH05]   T. Laakko, T. Hiltunen. Adapting Web Content to Mobile User Agents. *Internet Computing, IEEE* 9(2):46–53, March–April 2005. doi:10.1109/MIC.2005.29

[LL02]   W. Y. Lum, F. Lau. A Context-Aware Decision Engine for Content Adaptation. *Pervasive Computing, IEEE* 1(3):41–49, 2002. doi:10.1109/MPRV.2002.1037721

[LP00]   Y. Liu, K. M. Passino. Swarm Intelligence: Literature Overview. March 2000.

[LS06]   T. Lukasiewicz, U. Straccia. An Overview of Uncertainty and Vagueness in Description Logics for the Semantic Web. 2006.

[LS08]   T. Lukasiewicz, U. Straccia. Managing uncertainty and vagueness in description logics for the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web* 6(4):291–308, 2008. Semantic Web Challenge 2006/2007. doi:10.1016/j.websem.2008.04.001

[MA08]   M. F. Md Fudzee, J. Abawajy. A Classification for Content Adaptation Systems. In *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*. iiWAS '08, pp. 426–429. ACM, New York, NY, USA, 2008. doi:10.1145/1497308.1497386

[Man90]   W. R. Mande. Evaluation of a proposed handover algorithm for the GSM cellular system. In *Vehicular Technology Conference, 1990 IEEE 40th*. Pp. 264–269. 1990. doi:10.1109/VETEC.1990.110331

[ME99]   P. Martin, P. Eklund. Embedding knowledge in Web documents. *Computer Networks* 31(11–16):1403–1419, 1999. doi:10.1016/S1389-1286(99)00019-5

[Meg98]   D. Megginson. Simple API for XML. 1998. accessed: 2010-11-16. http://www.megginson.com/downloads/SAX/

[Mer89]   D. Merritt. *Building Expert Systems in Prolog*. Chapter 3, pp. 21–34. Springer-Verlag, 1989. accessed: 2010-11-20. http:// www.amzi.com / ExpertSystemsInProlog / 03backwarduncertainty.htm

[Mic10]   Microsoft Corporation. W3C Extended Log File Format (IIS 6.0). online, 2010. last visited: 2010-06-17. http:// go.microsoft.com/fwlink/?LinkId=103790

[Mit97]   T. Mitchell. *Machine Learning*. McGraw Hill, New York, 1997.

[MMR$^+$10] D. Malandrino, F. Mazzoni, D. Riboni, C. Bettini, M. Colajanni, V. Scarano. MIMOSA: context-aware adaptation for ubiquitous web access. *Personal and Ubiquitous Computing* 14:301–320, 2010. doi:10.1007/s00779-009-0232-9

[MRS05]   S. Mukherjee, I. V. Ramakrishnan, A. Singh. Bootstrapping Semantic Annotation for Content-Rich HTML Documents.

In *ICDE '05: Proceedings of the 21st International Conference on Data Engineering*. Pp. 583–593. IEEE Computer Society, Washington, DC, USA, 2005. doi:10.1109/ICDE.2005.28

[MWC10]   E. Miluzzo, T. Wang, A. T. Campbell. EyePhone: Activating Mobile Phones With Your Eyes. In *Proc. of The Second ACM SIGCOMM Workshop on Networking, Systems, and Applications on Mobile Handhelds (MobiHeld'10)*. New Delhi, India, August 2010.

[MYR03]   S. Mukherjee, G. Yang, I. Ramakrishnan. Automatic Annotation of Content-Rich HTML Documents: Structural and Semantic Analysis. In *The SemanticWeb - ISWC 2003*. Lecture Notes in Computer Science 2870, pp. 533–549. Springer Berlin / Heidelberg, 2003. 10.1007/978-3-540-39718-2_34. doi:10.1007/978-3-540-39718-2_34

[NCH$^+$04]   G. Nicol, M. Champion, P. L. Hégaret, J. Robie, L. Wood, A. L. Hors, S. Byrne. Document Object Model (DOM) Level 3 Core Specification. W3C recommendation, W3C, April 2004. accessed: 2010-01-19. http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407

[NSP95]   B. Noble, M. Satyanarayanan, M. Price. A Programming Interface for Application-Aware Adaptation in Mobile Computing. In *MLICS '95: Proceedings of the 2nd Symposium on Mobile and Location-Independent Computing*. Pp. 57–66. USENIX Association, 1995.

[NSS01]   K. Nagao, Y. Shirai, K. Squire. Semantic Annotation and Transcoding: Making Web Content More Accessible. *Multimedia, IEEE* 8(2):69–81, April-June 2001. doi:10.1109/93.917973

[Ora10a]   Oracle Corp. Hudson – Extensible continuous integration server. 2010. accessed: 2010-11-16. http://hudson-ci.org/

[Ora10b]   Oracle Corp. Java Platform, Standard Edition 6 Release. 2010. accessed: 2010-11-16. http://www.java.com/

[Ost08]   S. Ostermiller. Circular Buffers – com.Ostermiller.util Java Utilities. August 2008. accessed: 2010-11-16. http://ostermiller.org/utils/CircularBuffer.html

[Pas98]   J. Pascoe. Adding Generic Contextual Capabilities to Wearable Computers. In *Proceedings of the 2nd International Symposium on Wearable Computers*. Pp. 92–99. IEEE Computer Society, 1998.

[Pas10]   L. Passani. WURFL. October 2010. accessed: 2010-11-16. http://wurfl.sourceforge.net/

[PB05]   A. Poole, L. J. Ball. Eye Tracking in Human-Computer Interaction and Usability Research: Current Status and Future

Prospects. In Ghaoui (ed.), *Encyclopedia of Human Computer Interaction*. IGI Global, December 2005.

[Pem02]   S. Pemberton. XHTML^{TM} 1.0 The Extensible HyperText Markup Language (Second Edition). W3C recommendation, W3C, August 2002. accessed: 2010-01-18. http://www.w3.org/TR/2002/REC-xhtml1-20020801

[PN09]   K. Pernice, J. Nielsen. Eyetracking Methodology - How to Conduct and Evaluate Usability Studies Using Eyetracking. August 2009. last accessed 2010-06-17. http://www.useit.com / eyetracking / methodology / eyetracking-methodology.pdf

[Pop09]   A. Popescu. Geolocation API Specification. Last call WD, W3C, July 2009. accessed: 2010-10-25. http://www.w3.org/TR/2009/WD-geolocation-API-20090707/

[PSK09]   T. Plumbaum, T. Stelter, A. Korth. Semantic Web Usage Mining: Using Semantics to Understand User Intentions. In *UMAP '09: Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization*. Pp. 391–396. Springer-Verlag, Berlin, Heidelberg, 2009. doi:10.1007/978-3-642-02247-0_42

[RAB+04] A. Ranganathan, J. Al-Muhtadi, J. Biehl, B. Ziebart, R. Campbell, B. Bailey. Evaluating Gaia using a Pervasive Computing Benchmark. 2004. http://choices.cs.uiuc.edu/ranganat/Pubs/Gaia_Evaluation.PDF

[RAB+05] A. Ranganathan, J. Al-Muhtadi, J. Biehl, B. Ziebart, R. Campbell, B. Bailey. Towards a Pervasive Computing Benchmark. In *PerWare '05 Workshop on Support for Pervasive Computing*. Pp. 194–198. 2005.

[Ras86]   L. Rastrigin. Random Search as a Method for Optimization and Adaptation. In Arkin et al. (eds.), *Stochastic Optimization*. Lecture Notes in Control and Information Sciences 81, pp. 534–544. Springer Berlin / Heidelberg, 1986. doi:10.1007/BFb0007129

[Rei06]   G. Reif. *Semantic Web - Wege zur vernetzten Wissensgesellschaft*. Chapter Semantische Annotation, pp. 405–418. Springer Berlin, 2006. doi:10.1007/3-540-29325-6

[RHJ99]   D. Raggett, A. L. Hors, I. Jacobs. HTML 4.01 Specification. W3C recommendation, W3C, December 1999. accessed: 2010-01-18. http://www.w3.org/TR/1999/REC-html401-19991224/

[RSC06]   T. Rho, M. Schmatz, A. B. Cremers. Towards Context-Sensitive Service Aspects. In *ECOOP Workshop 06*. July 2006. http://roots.iai.uni-bonn.de/research/logicaj/downloads/papers/RhoSchmatz-OT4AmI06.pdf

[SAW94]   B. N. Schilit, N. Adams, R. Want. Context-Aware Computing Applications. In *First International Workshop on Mobile Computing Systems and Applications*. Pp. 85–90. 1994.

[SBG99]   A. Schmidt, M. Beigl, H.-W. Gellersen. There is more to context than location. *Computers & Graphics* 23(6):893–901, 1999. doi:10.1016/S0097-8493(99)00120-X

[Sch95]   W. N. Schilit. *A System Architecture for Context-Aware Mobile Computing*. PhD thesis, Columbia University, 1995.

[Sch98]   B. Schneiderman. *Designing the User Interface*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[Sch00]   A. Schmidt. Implicit Human Computer Interaction Through Context. *Personal and Ubiquitous Computing* 4(2–3):191–199, June 2000.

[Sch09]   F. Schmedding. Content-Sensitive User-Interfaces for Annotated Web Pages. In *Proceedings of the 39. Jahrestagung für Informatik*. 2009.

[SFYW98] R. Stiefelhagen, M. Finke, J. Yang, A. Waibel. From Gaze to Focus of Attention. In *Proceedings of Workshop on Perceptual User Interfaces: PUI 98*. Volume 1614, pp. 25–30. 1998.

[SG00]   D. D. Salvucci, J. H. Goldberg. Identifying Fixations and Saccades in Eye-Tracking Protocols. In *ETRA '00: Proceedings of the 2000 symposium on Eye tracking research & applications*. Pp. 71–78. ACM, New York, NY, USA, 2000. doi:10.1145/355017.355028

[SL01]   A. Schmidt, K. V. Laerhoven. How to Build Smart Appliances? *IEEE Personal Communications*, pp. 66–71, 2001.

[SLF03]   T. Strang, C. Linnhoff-Popien, K. Frank. CoOL: A Context Ontology Language to enable Contextual Interoperability. In Stefania et al. (eds.), *DAIS 2003, LNCS 2893*. Pp. 236–247. 2003.

[SLM00]   G. South, A. Lenaghan, R. Malyan. Using Reflection for Service Adaptation in Mobile Clients (T4). Technical report, Kingston University-UK, 2000. http://technology.kingston.ac.uk/ncg/research/publications/2000/Reflection_Mobile/APL_TC_XVII_T4.pdf

[SLP99]   C. R. Sastry, D. P. Lewis, A. Pizano. Webtour: a System to Record and Playback Dynamic Multimedia Annotations on Web Document Content. In *ACM Multimedia (2)*. Pp. 175–178. 1999. http://10.1145/319878.319925

[Spi00]   M. Spiliopoulou. Web Usage Mining for Web Site Evaluation. *Commun. ACM* 43(8):127–134, 2000. doi:10.1145/345124.345167

[SPTH05]   G. B. Stamou, J. Z. Pan, V. Tzouvaras, I. Horrocks. A Fuzzy Extension of SWRL. In *Rule Languages for Interoperability*. W3C, 2005. http://www.w3.org/2004/12/rules-ws/paper/52

[Sri10]   V. Srini. Projects:TestGen4Web. June 2010. accessed: 2010-11-15. http:// developer.spikesource.com / wiki / index.php?title=Projects:TestGen4Web&oldid=3487

[SSM$^+$10]   J. San Agustin, H. Skovsgaard, E. Mollenbach, M. Barret, M. Tall, D. W. Hansen, J. P. Hansen. Evaluation of a Low-Cost Open-Source Gaze Tracker. In *ETRA '10: Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*. Pp. 77–80. ACM, New York, NY, USA, 2010. doi:10.1145/1743666.1743685

[SSSK06]   G. Stoilos, N. Simou, G. Stamou, S. Kollias. Uncertainty and the Semantic Web. *IEEE Intelligent Systems* 21:84–87, 2006. doi:10.1109/MIS.2006.105

[SST$^+$05]   G. Stoilos, G. B. Stamou, V. Tzouvaras, J. Z. Pan, I. Horrocks. Fuzzy OWL: Uncertainty and the Semantic Web. In Grau et al. (eds.), *OWLED*. CEUR Workshop Proceedings 188. CEUR-WS.org, 2005. http://ceur-ws.org/Vol-188/sub16.pdf

[STHK01]   B. N. Schilit, J. Trevor, D. M. Hilbert, T. K. Koh. m-Links: An Infrastructure for Very Small Internet Devices. In *Proceedings of the 7th annual international conference on Mobile computing and networking*. MobiCom '01, pp. 122–131. ACM, New York, NY, USA, 2001. doi:10.1145/381677.381689

[SWM04]   M. K. Smith, C. Welty, D. L. McGuinness. OWL Web Ontology Language Guide. Technical report 20040210, World Wide Web Consortium, 2004. http://www.w3.org/TR/owl-guide/

[The09]   The Apache Software Foundation. Log Files - Apache HTTP Server - Access Log. October 2009. last visited: 2010-06-17. http://httpd.apache.org / docs / 2.2 / logs.html#accesslog

[Tob08]   Tobii Technology AB. Tobii X60 & X120 Eye Trackers. online, October 2008. accessed: 2010-06-18. http:// www.tobii.com / archive / files / 17992 / Tobii_X120_Eye_Tracker_leaflet.pdf.aspx

[Tob09]   Tobii Technology AB. Tobii Eye Tracker IS. online, February 2009. accessed: 2010-06-18. http://www.tobii.com/archive/files/19966/Tobii+Eye+Tracker+IS+Leaflet.pdf.aspx

[Tre08]   Treasury Board of Canada Secretariat. Client-side CSS for enhancing accessibility. online, February 2008. accessed: 2010-11-15. http://www.tbs-sct.gc.ca/clf2-nsi2/tb-bo/acch-aacc-eng.asp

[W3C07]   W3C - WorldWideWeb Commitee. XHTML Role At-
          tribute Module, A module to support role classification
          of elements. October 2007. accessed: 2010-01-16. http://
          www.w3.org/TR/2007/WD-xhtml-role-20071004/

[W3C08]   W3C Incubator Group on Uncertainty Reasoning for the
          World Wide Web. Final Report. Technical report, W3C, 2008
          2008. accessed: 2010-03-08. http://www.w3.org/2005/Incu-
          bator/urw3/XGR-urw3-20080331/

[W3C10]   W3C Semantic Web - Contributors. Semantic Web
          Development Tools. online, January 2010. accessed:
          2010-11-15.   http:// www.w3.org / 2001 / sw / wiki / in-
          dex.php?title=Tools&oldid=1441

[WA06]   M. Wnuk, R. Atterer. UsaProxy - Usability Proxy for
          Websites. December 2006. accessed: 2010-01-19. http://
          fnuked.de/usaproxy/

[WAL$^+$00] P. L. Wizinowich, D. S. Acton, O. Lai, J. Gathright, W. Lup-
          ton, P. J. Stomski. Performance of the W.M. Keck Obser-
          vatory Natural Guide Star Adaptive Optic Facility: the
          first year at the telescope. In *Proc. SPIE Vol. 4007, p. 2-13,
          Adaptive Optical Systems Technology, Peter L. Wizinowich; Ed.*
          Pp. 2–13. 2000.

[WDC$^+$04] X. Wang, J. S. Dong, C. Chin, S. Hettiarachchi, D. Zhang.
          Semantic Space: An Infrastructure for Smart Spaces.
          IEEE Pervasive Computing 3(3):32–39, 2004. doi:10.1109/
          MPRV.2004.1321026

[Wei93]   M. Weiser. Some Computer Science Issues in Ubiquitous
          Computing. *Commun. ACM* 36(7):75–84, 1993. doi:10.1145/
          159544.159617

[Wic10]   M. Wick. GeoNames. 2010. accessed: 2010-11-16. http://
          geonames.org

[Wir01]   Wireless   Application   Forum.   WAG   UAProf.   on-
          line, October 2001. accessed: 2010-11-09. http://
          www.openmobilealliance.org / tech / affiliates / wap /
          wap-248-uaprof-20011020-a.pdf

[WM87]   C. Ware, H. H. Mikaelian. An Evaluation of an Eye Tracker
          as a Device for Computer Input. *SIGCHI Bull.* 17(SI):183–
          188, 1987. doi:10.1145/30851.275627

[Wnu05]   M. Wnuk. Usability Proxy for Websites. Technical report,
          Media Informatics Group, University of Munich, October
          2005.

[XTL08]   Y. Xiao, Y. Tao, Q. Li. Web Page Adaptation for Mobile De-
          vice. In *4th International Conference on Wireless Communica-
          tions, Networking and Mobile Computing, 2008. WiCOM'08.*
          Pp. 1–5. oct. 2008. doi:10.1109/WiCom.2008.1182

[Yee98]   K.-P. Yee. CritLink: Better Hyperlinks for the WWW. on-
            line, April 1998. last visited: 2010-11-02. http://zesty.ca/crit
            /ht98.html

[YRP99]   M. Yarvis, P. L. Reiher, G. J. Popek. Conductor: A Frame-
            work for Distributed Adaptation. In *Workshop on Hot Top-
            ics in Operating Systems*. Pp. 44–49. 1999. doi:10.1109/HO-
            TOS.1999.798376

[Zad65]   L. Zadeh. Fuzzy Sets. *Information and Control* 8(3):338–353,
            1965. doi:10.1016/S0019-9958(65)90241-X

[Zha07]   D. Zhang. Web Content Adaptation for Mobile Handheld
            Devices. *Commun. ACM* 50(2):75–79, 2007. doi:10.1145/
            1216016.1216024

Glossary

AJAX *Asynchronous JavaScript and XML*, a technique to combine →*JavaScript* functionality with dedicated server support to allow incremental updates without issuing a complete page request.

API *Application Programming Interface*, definition of classes and methods to interact with a given framework.

Browser Application to navigate the →*WWW* and visualize its →*HTML* pages.

CSS *Cascading Style Sheets*, a declarative language to control the visualisation of →*HTML* elements.

DOM *Document Object Model* [NCH$^+$04], a cross-platform and language-independent convention for representing and interacting with objects in →*HTML*, →*XHTML* and →*XML* documents. The data in a DOM is structured as a tree.

GUI *Graphical User Interface*, a →*HMI* based on visual interaction.

HCI *Human Computer Interaction*, the field of research examining the way, humans can interact with computing machines.

HMI *Human Machine Interface* or *User Interface*, the aggregate of means by which people (the users) interact with a particular machine, device, computer program or other complex tool.

HTML *HyperText Markup Language* [RHJ99], language to enrich text with markup information and interconnection links, primarily used for web pages.

HTTP *HyperText Transfer Protocol* [FGM$^+$99], an application-level protocol for data exchange in distributed, collaborative, hypermedia information systems. Main protocol used to request resources in the →*WWW*.

Internet Using standardized protocols, local computer networks are interconnect to route data from one network to any other transitively connected network. Allows to run e. g. protocols from the →*WWW* to access world wide distributed documents and files.

IP Address *Internet Protocol Address*, numerical identifier assigned with rough geographic relation to every participant of an →*Internet* communication.

JavaScript Scripting language, which can be embedded in →*HTML* pages to modify the page →*DOM*. The corresponding interpreter is embedded in the →*Broswer*.

MIME *Multipurpose Internet Mail Extensions* a coding standard for emails and other →*Internet* data, containing an agreed way to identify the type of transferred data.

OWL Web Ontology Language [SWM04], ontology format to represent knowledge bases and content interrelationships in the Semantic Web.

PDA *Personal Digital Assistant*, handheld devices that were originally designed as personal organizers.

RDF *Resource Description Framework* [KC04], an W3C standard to describe semantics of any identifiable resource.

SemS *Semantic Shadow*, the concept elaborated in this thesis.

Service An application without →*HMI* performing a specific task and being accessible via an application programming interface.

$\mathcal{SHOIN}$**(D)** The description logic of OWL DL, providing the ontology semantics [SSSK06].

Smartphone Mobile phone with enhanced user input and output capabilities, e. g. taller screen, direct manipulation options via pen or touch.

SWRL *Semantic Web Rule Language* [HPB$^+$04], combination of →*OWL* with a rule markup language.

UAprof *User Agent Profile*, protocol to exchange device capability information, part of the →*WAP* protocol collection [Wir01].

UI *User Interface*, →*HMI*.

URI *Unique Resource Identifier*, a string that uniquely identifies a digital resource, such as a web page. Consists of protocol definition, domain and location string and might be extended by a fragment part.

URL *Uniform Resource Locator*, a →*URI* that identifies an object in the →*WWW*.

WAP *Wireless Application Protocol*, a collection of techniques and protocols to make the content of the →*WWW* accessible for mobile phones with low bandwidth connections.

Web Service A computing service, which supports an API that is build upon the →*WWW* protocols.

WWW *World Wide Web*, a collection of protocols to access →*HTML* documents and other files from distributed servers connected through the →*Internet*.

XHTML *Extensible HyperText Markup Language* [Pem02], an →*XML* compliant variant of →*HTML*.

XML *Extensible Markup Language* [BPM$^+$08], a text-based general-purpose markup language to structure data semantically.

XPath *XML Path Language*, a query language to identify elements in an →*XML* document.

XPointer *XML Pointer Language*, extension of →*XPath* which allows to include element queries in →*URI*s.

Index

Source-code and documentation of
the framework, annotation derivation,
and adaptation prototypes available online at
http://sam.iai.uni-bonn.de/people/PascalBihler/sems

Liste der erschienenen Berichte / Publication list

IAI-TR-91-1 Axioms for Concurrent Processes
Armin B. Cremers and Thomas N. Hibbard
Juni 1991.

IAI-TR-92-1 The Categorical Abstract Machine:
Basics and Enhancements
Ralf Hinze
Dezember 1992.

IAI-TR-92-2 On Perfect Introspection with Quantifying-in
Gerhard Lakemeyer
Dezember 1992.

IAI-TR-92-3 All You Ever Wanted to Know about Tweety
Gerhard Lakemeyer
Dezember 1992.

IAI-TR-93-1 Exploration and Model Building in Mobile
Robot Domains
Sebastian B. Thrun
Januar 1993.

IAI-TR-93-2 Best Approximate General Ellipses on Integer
Grids
Dieter W. Fellner and Christoph Helmberg
Januar 1993.

IAI-TR-93-3 Programming in Propositional Logic
or Reductions: Back to the Roots (Satisfiability)
Hermann Stamm-Wilbrandt
März 1993

IAI-TR-93-4 Extensible Image Synthesis
Dieter W. Fellner
April 1993.

IAI-TR-93-5 Extracting Provably Correct Rules
from Artificial Neural Networks
Sebastian B. Thrun
Mai 1993.

IAI-TR-93-6 Adaptive Steuerung von Strategieparametern
bei genetischen Algorithmen
Andreas Bergmann
Juni 1993.

IAI-TR-93-7 Lifelong Robot Learning
Sebastian B. Thrun and Tom M. Mitchell
Juli 1993.

IAI-TR-93-8 Situation Schemata
as Representation Format and Query Language
for a Non-Standard Database System
Armin B. Cremers, Thomas Lemke,
Volker Paul and Marion Schulz
August 1993.

IAI-TR-93-9 CGI++
A 2D Graphics Interface
Based on CGI
Dieter W. Fellner and Martin Fischer
August 1993.

IAI-TR-93-10 A simple linear time algorithm
for embedding maximal planar graphs
Hermann Stamm-Wilbrandt
Oktober 1993.

IAI-TR-94-1 Logic Program Modules for
Interoperable Information
Systems
Armin B. Cremers, Thomas Lemke and
Mechthild Rohen
Mai 1994.

IAI-TR-94-2 Planar embedding of hamiltonian graphs
via efficient bipartation of circle graphs
Christoph Hundack and
Hermann Stamm-Wilbrandt
Juni 1994.

IAI-TR-94-3 Implementation of Dynamic Delegation
in Strongly Typed Inheritance-Based Systems
Günter Kniesel
Oktober 1994.

IAI-TR-94-4 Update Propagation in Chimera,
an Active DOOD Language
Ulrike Griefahn and Rainer Manthey
Oktober 1994.

IAI-TR-94-5 Reflections on Some Fundamental Issues of
Rule-Based Incremental Update Propagation
Rainer Manthey
Oktober 1994.

IAI-TR-95-10 Distributed CGI^{++} - eine Client/Server-Lösung zur Entwicklung leistungsfähiger Computergraphik-Anwendungen
Dieter W. Fellner and Martin Fischer
Juli 1995.

IAI-TR-95-11 Runtime Type for Class Hierarchies
Martin Fischer
Juli 1995.

IAI-TR-95-12 Reference Pointer for Class Hierarchies
Martin Fischer
August 1995.

IAI-TR-95-13 The Dynamic Window Approach to Collision Avoidance
Dieter Fox, Wolfram Burgard
and Sebastian Thrun
August 1995.

IAI-TR-96-1 Estimating the Absolute Position of a Mobil Robot Using Position Probability Grids
Wolfram Burgard, Dieter Fox
Daniel Hennig and Timo Schmidt
Februar 1996.

IAI-TR-96-2 A Deterministic Annealing Framework for Unsupervised Texture Segmentation
Thomas Hofmann, Jan Puzicha and
Joachim M. Buhmann
Februar 1996.

IAI-TR-96-3 Map Learning and High-Speed Navigation in RHINO
S. Thrun, A. Bücken, W. Burgard,
D. Fox, T. Fröhlinghaus, D. Hennig,
T. Hofmann, M. Krell and T. Schmidt
Juli 1996.

IAI-TR-96-4 Belief Revision in a Nonclassical Logic
Gerhard Lakemeyer and Wilfried Lang
Juli 1996.

IAI-TR-96-5 Bildverarbeitung und Mustererkennung
W. Förstner und A.B. Cremers
Juli 1996.

IAI-TR-96-6 Implementing Chimera on Top of an Active Relational Database System
Ulrike Griefahn and Thomas Lemke
Juli 1996.

IAI-TR-96-7 PROPAGATION RULE COMPILER:
TECHNICAL DOCUMENTATION
Ulrike Griefahn and Thomas Rath
Juli 1996.

IAI-TR-96-8 DATA VISUALIZATION BY MULTIMENSIONAL SCALING:
A DETERMINISTIC ANNEALING APPROACH
Joachim Buhmann and Hansjörg Klock
Oktober 1996.

IAI-TR-96-9 MONADIC-STYLE BACKTRACKING
Ralf Hinze
Oktober 1996.

IAI-TR-96-10 THE SYSTEM COBREP: AUTOMATIC CONVERSION OF
CSG BUILDING DATA TO BOUNDARY REPRESENTATION
Roman Englert
Oktober 1996.

IAI-TR-96-11 OBJECTS DON'T MIGRATE!
Günter Kniesel
April 1996.

IAI-TR-96-12 ENCAPSULATION - VISABILITY AND ACCESS RIGHTS
Günter Kniesel
November 1996.

IAI-TR-96-13 MAINTAINING LIBRARY CATALOGUES WITH AN RDBMS
- A PERFORMANCE STUDY -
O. Balownew, T. Bode, A.B. Cremers,
J. Kalinski, J.E. Wolff, H. Rottmann
November 1996.

IAI-TR-96-14 HIERARCHICAL RADIOSITY ON TOPOLOGICAL DATA STRUCTURES
Heinzgerd Bendels, Dieter W. Fellner,
Stephan Schäfer
November 1996.

IAI-TR-97-1 CLOSING THE GAP BETWEEN MODELING AND RADIOSITY
Dieter W. Fellner, Stephan Schäfer
Januar 1997.

IAI-TR-97-2 P-SUBGRAPH ISOMORPHISM COMPUTATION
AND UPPER BOUND COMPLEXITY ESTIMATION
Roman Englert, Jörg Seelmann-Eggebert
Januar 1997.

IAI-TR-97-3 ACTIVE MOBILE ROBOT LOCALIZATION
Wolfram Burgard, Dieter Fox and Sebastain Thrun
Februar 1997.

IAI-TR98-8 Aufnahme und Intelligente Auswertung
Grossflächiger Szenen
A. Brunn, R. Englert, Th. Hau,
A.B. Cremers und W. Förstner
September 1998.

IAI-TR-98-9 Komponentenbasierte Anpassbarkeit von Groupware
Oliver Stiemerling
September 1998.

IAI-TR-98-10 Polytypic Functions Over Nested Datatypes
Ralf Hinze
Oktober 1998.

IAI-TR-98-11 Generalizing Generalized Tries
Ralf Hinze
November 1998.

IAI-TR-98-12 Numerical Representations as Higher-Order
Nested Datatypes
Ralf Hinze
Dezember 1998.

IAI-TR-98-13 The MyVIEW Project: from Heterogeneous Bibliographic
Information Repositories to Personalized Digital Libraries
Jens E. Wolff and Armin B. Cremers
Dezember 1998.

IAI-TR-98-14 The Use of Cooperation Scenarios in the Design and
Evaluation of a CSCW System
Oliver Stiemerling and Armin B. Cremers
Dezember 1998.

IAI-TR-98-15 Portfoliooptimierung mit Verfahren aus der Informatik
Armin B. Cremers, Christoph Hundack, Jens Lüssem
Dezember 1998.

IAI-TR-99-1 Deriving Monad Transformers
Ralf Hinze
Januar 1999.

IAI-TR-99-2 Polytypic Programming With Ease
Ralf Hinze
Februar 1999.

IAI-TR-99-3 Experience- and Model-based Transformational Learning
of Symbolic Behavior Specifications
Michael Beetz and Thorsten Belker
März 1999.

IAI-TR-99-4 Perfect Trees and Bit-reversal Permutations
Ralf Hinze
März 1999.

IAI-TR-99-5 MANUFACTURING DATATYPES
Ralf Hinze
April 1999.

IAI-TR-99-6 CONSTRUCTING RED-BLACK TREES
Ralf Hinze
Mai 1999.

IAI-TR-99-7 NONLINEAR DISCRIMINANT ANALYSIS USING KERNEL FUNCTIONS
Volker Roth and Volker Steinhage
Juni 1999.

IAI-TR-99-8 EFFICIENT GENERALIZED FOLDS
Ralf Hinze
Juni 1999.

IAI-TR-99-9 A NEW APPROACH TO GENERIC FUNCTIONAL PROGRAMMING
Ralf Hinze
Juli 1999.

IAI-TR-99-10 DISTRIBUTED COMPONENT-BASED TAILORABILITY FOR
CSCW APPLICATIONS
Oliver Stiemerling, Ralph Hinken, and Armin B. Cremers
März 1999.

IAI-TR-99-11 FLEXIBLE ALIASING WITH PROTECTION
Günter Kniesel and Dirk Theisen
Juni 1999.

IAI-TR-99-12 XPRES: A RANKING APPROACH TO RETRIEVAL ON
STRUCTURED DOCUMENTS
J.E. Wolff, H. Flörke, A.B. Cremers
Juli 1999.

IAI-TR-99-13 THE EVOLVE TAILORING PLATFORM: SUPPORTING THE
EVOLUTION OF COMPONENT -BASED GROUPWARE
Oliver Stiemerling, Ralph Hinken, Armin B. Cremers
September 1999.

IAI-TR-99-14 MUTIPLE SIP STRATEGIES -
A DYNAMIC APPROACH TO
DEDUCTIVE QUERY PROCESSING
Andreas Behrend
Dezember 1999.

IAI-TR-99-15 POLYTYPIC VALUES POSSESS POLYKIND TYPES
Ralf Hinze
Dezember 1999.

IAI-TR-2000-1 PROBABILISTIC HYBRID ACTION MODELS FOR
PREDICTING CONCURRENT PERCEPT-DRIVEN ROBOT BEHAVIOR
Michael Beetz and Henrik Grosskreutz
Juni 2000.

IAI-TR-2001-1 NewsSIEVE
Ein selbstadaptiver Filter für textuelle Informationen
Elmar Haneke
Januar 2001.

IAI-TR-2001-2 Perspektiven: Persistente Objekte mit anwendungsspezifischer
Struktur und Funktionalität
Wolfgang Reddig
November 2001.

IAI-TR-2001-3 A New Adaptive Algorithm for the Polygonization of
Noisy Imagery
Lothar Hermes and Joachim M. Buhmann
Dezember 2001.

IAI-TR-2002-1 Second Workshop on aspect-Oriented Software development
P. Costanza, G. Kniesel, K. Mehner, E. Pulvermüller and A. Speck
Februar 2002.

IAI-TR-2002-2 Constructing Tournament Representations: An Exercise in
Pointwise Relational Programming
Ralf Hinze
Februar 2002.

IAI-TR-2002-3 Church Numerals, Twice!
Ralf Hinze
Februar 2002.

IAI-TR-2002-4 Schnelle Numerische Verfahren zur Bewertung von
Europäischen Optionen in Erweiterten Black-Scholes
Marktmodellen
Stefan Alex Popovici
Mai 2002.

IAI-TR-2002-5 Optimal Cluster Preserving Embedding of Non-Metric
Proximity Data
Volker Roth, Julian Laub, Motoaki Kawanabe, Joachim M. Buhmann
Juni 2002.

IAI-TR-2002-6 Internet-Based Robotic Tele-Presence
Dirk Schulz
Mai 2002.

IAI-TR-2002-7 Path Based Clustering
Bernd Fischer and Joachim M. Buhmann
August 2002.

IAI-TR-2002-8 The Generalized Lasso: a Wrapper approach to
gene selection for microarray data
Volker Roth
August 2002.

IAI-TR-2002-9 PLANNING TO OPTIMIZE THE UMTS CALL SET-UP FOR THE EXECUTION OF MOBILE APPLICATIONS
Roman Englert
Oktober 2002.

IAI-TR-2003-1 PROGRAM-INDEPENDENT COMPOSITION OF CONDITIONAL TRANSFORMATIONS
Günter Kniesel and Helge Koch
Juni 2003.

IAI-TR-2003-2 THE JMANGLER TUTORIAL
Günter Kniesel
Juli 2003.

IAI-TR-2003-3 STATIC DEPENDENCY ANALYSIS FOR CONDITIONAL PROGRAM TRANSFORMATIONS
Günter Kniesel
Juli 2003.

IAI-TR-2003-4 OBJEKTORIENTIERTE DATENMODELL-INTEGRATION FÜR EIN OFFENES, PALÄOÖKOLOGISCHES INFORMATIONSSYSTEM
A.B. Cremers, R. Dikau (Hrsg.)
Dezember 2003.

IAI-TR-2003-5 CONCEPTS FOR THE REPRESENTATION, STORAGE, AND RETRIEVAL OF SPATIO-TEMPORAL OBJECTS IN 3D/4D GEO-INFORMATION-SYSTEMS
Jörg Siebeck
Dezember 2003.

IAI-TR-2004-1 MODELL- UND DIAGNOSEBASIERTES TRANSFORMATIONSLERNEN VON SYMBOLISCHEN PLÄNEN FÜR MOBILE ROBOTER
Michael Beetz, Thorsten Belker, Armin B. Cremers
Februar 2004.

IAI-TR-2004-2 UNANTIZIPIERTE SOFTWARE-EVOLUTION - ABSCHLUSSBERICHT DES TAILOR PROJEKTS (DFG-PROJEKT CR 65/13)
Pascal Costanza, Günter Kniesel, Armin B. Cremers
Oktober 2004.

IAI-TR-2004-3 A META MODEL FOR ASPECTJ
YAN Han, Günter Kniesel, Armin B. Cremers
Oktober 2004.

IAI-TR-2004-4 UNIFORM GENERICITY FOR ASPECT LANGUAGES
Tobias Rho, Günter Kniesel
Dezember 2004.

IAI-TR-2006-1 A LOGIC FOUNDATION FOR PROGRAM TRANSFORMATIONS
Günter Kniesel
Januar 2006.

IAI-TR-2006-2 "Scrap Your Boilerplate" Reloaded
Ralf Hinze, Andres Löh, and Bruno C. d. S. Oliveira
Januar 2006.

IAI-TR-2006-3 'AOSD.06 Industry Track Proceedings'
Fifth International Conference on Aspect-Oriented Software Development,
Bonn, Germany, March 20-24
Matt Chapman, Alexandre Vasseur, Günter Kniesel (Eds.)
März 2006.

IAI-TR-2006-4 Independent Evolution of Design Patterns and Application
Logic with Generic Aspects - A Case Study
Tobias Rho, Günter Kniesel
April 2006.

IAI-TR-2006-5 Nonparametric Density Estimation for Human Pose Tracking
Thomas Brox, Bodo Rosenhahn, Uwe Kersting, and Daniel Cremers
April 2006.

IAI-TR-2006-6 EWAS 2006
3rd European Workshop on Aspects in Software
Günter Kniesel (Ed.)
August 2006.

IAI-TR-2006-7 A Fixpoint Approach to State Generation for
Stratifiable Disjunctive Deductive Databases
Andreas Behrend
Dezember 2006.

IAI-TR-2006-8 Interaktive, plan-basierte Softwareagenten
im verteilten Supply Chain Management
Abschlussbericht im Rahmen des DFG-Schwerpunktprogramms
Intelligente Softwareagenten und betriebswirtschaftliche
Anwendungsszenarien
Michael Beetz, Armin B. Cremers, Axel Kuhn, Bernd Hellingrath,
Jürgen Schumacher, Andre Alberti
Dezember 2006.

IAI-TR-2006-9 "Zeitsynchronisation in TmoteSky Sensornetzen"
Serge Shumilov, Boris Iven, Oliver Mali, Roman Saul
Dezember 2006.

IAI-TR-2007-1 Introducing Curvature into Globally Optimal
Image Segmentation: Minimum Ratio Cycles
on Product Graphs
Thomas Schoenemann, Daniel Cremers
April 2007.

IAI-TR-2007-2 GLOBALLY OPTIMAL IMAGE SEGMENTATION WITH
AN ELASTIC SHAPE PRIOR
Thomas Schoenemann, Daniel Cremers
April 2007.

IAI-TR-2007-3 DUAL-SPACE GRAPH CUTS FOR MOTION LAYER DECOMPOSITION
Thomas Schoenemann and Daniel Cremers
April 2007.

IAI-TR-2007-4 ITERATED AND EFFICIENT NONLOCAL MEANS
FOR DENOISING OF TEXTURAL PATTERNS
Thomas Brox, Oliver Kleinschmidt, and Daniel Cremers
August 2007.

IAI-TR-2007-5 DESIGN AND IMPLEMENTATION OF A WORKFLOW ENGINE
Sebastian Bergmann
September 2007.

IAI-TR-2007-6 ON LOCAL REGION MODELS AND THE STATISTICAL INTERPRETATION
OF THE PIECEWISE SMOOTH MUMFORD-SHAH FUNCTIONAL
Thomas Brox and Daniel Cremers
September 2007.

IAI-TR-2007-7 TIME-BASED TRIGGERS FOR SQL:
A LINGERING ISSUE REVISITED
Andreas Behrend, Christian Dorau, and Rainer Manthey
November 2007.

IAI-TR-2007-8 HIGH RESOLUTION MOTION LAYER DECOMPOSITION
USING DUAL-SPACE GRAPH CUTS
Thomas Schoenemann and Daniel Cremers
Dezember 2007.

IAI-TR-2007-9 MATCHING NON-RIGIDLY DEFORMABLE SHAPES ACROSS IMAGES:
A GLOBALLY OPTIMAL SOLUTION
(DFG-PROJEKT CR 250/1-1)
Thomas Schoenemann and Daniel Cremers
Dezember 2007.

IAI-TR-2007-10 GLOBALLY OPTIMAL SHAPE-BASED TRACKING IN REAL-TIME
(DFG-PROJEKT CR 250/1-1)
Thomas Schoenemann and Daniel Cremers
Dezember 2007.

IAI-TR-2008-1 SOFTWARE SECURITY METRICS FOR MALWARE RESILIENCE
Hanno Langweg
Februar 2008.

IAI-TR-2008-2 RULE-BASED PROGRAMMING
Günter Kniesel, Jorge Sousa Pinto
Juni 2008.

IAI-TR-2008-3 Efficient Planar Graph Cuts with
Applications in Computer Vision
Frank R. Schmidt, Eno Töppe and Daniel Cremers
Juni 2008.

IAI-TR-2008-4 The Elastic Ratio: Introducing Curvature into
Ratio-based Globally Optimal Image Segmentation
Thomas Schoenemann, Simon Masnou and Daniel Cremers
Dezember 2008.

IAI-TR-2008-5 A convex approach for computing minimal partitions
Antonin Chambolle, Daniel Cremers, Thomas Pock
November 2008.

IAI-TR-2009-1 Anisotropic Minimal Surfaces for Integration
of Multiview Stereo and Normal Information
Kalin Kolev, Thomas Pock, Daniel Cremers
Juni 2009.

IAI-TR-2009-2 Witnessing Patterns: A Data Fusion Approach
to Design Pattern Detection
Alexander Binun, Günter Kniesel
Januar 2009.

IAI-TR-2009-3 DPDX - A Common Exchange Format for Design
Pattern Detection Tools
Günter Kniesel, Alexander Binun, Péter Hegedűs, Lajos Jenő Fülöp,
Alexander Chatzigeorgiou, Yann-Gaël Guéhéneuc and Nikolaos Tsantalis
Oktober 2009.

IAI-TR-2009-4 Moment Constraints in Convex Relaxation Methods
for Shape Optimization
Daniel Cremers
September 2009.

IAI-TR-2010-1 3D-Objekterkennung in dynamischen Szenen
J. Behley, V. Steinhage, A.B. Cremers
November 2010.

IAI-TR-2011-1 An Adequate, Denotational, Functional-Style
Semantics for Typed FlatCurry without Letrec
Jan Christiansen, Daniel Seidel, and Janis Voigtländer
März 2011.

IAI-TR-2011-2 The Semantic Shadow
Combining User Interaction with Context Information
for Semantic Web-Site Annotation
Pascal Bihler
Juli 2011.