Customising Web Applications Towards Ubiquity
The Notion and the Issues[1]

G. Kappel, W. Retschitzegger
Institute of Software Technology, Vienna University of Technology
{gerti|werner}@bi.tuwien.ac.at

E. Kimmerstorfer
Department of Information Systems (IFS), University of Linz
eugen@ifs.uni-linz.ac.at

B. Pröll
Institute for Applied Knowledge Processing (FAW), University of Linz
bproell@faw.uni-linz.ac.at

W. Schwinger, Th. Hofer
Software Competence Center Hagenberg (SCCH)
{wieland.schwinger|thomas.hofer}@scch.at

Ch. Feichtner
Siemens AG Österreich,
christian.feichtner@siemens.at

Abstract: *Ubiquitous web applications adhering to the anytime/anywhere/anymedia paradigm are required to be customisable meaning the adaptation of their services towards a certain context. When developing ubiquitous web applications one must understand what context is to determine its relevance and how it can be exploited for adapting the provided services accordingly. This paper proposes a design space of customisation providing among others the bases for the development of a comprehensive modelling method for ubiquitous web applications.*

1 Introduction

We are facing a new generation of web applications being characterised by the *anytime/anywhere/anymedia paradigm*, thus providing *ubiquitous access* to services, turning e-commerce into *m-commerce* [CHAKR00]. Ubiquitous computing was first stressed by Marc Weiser [WEISE91], envisioning a scenario where computational power would be available everywhere embedded in walls, chairs, clothing etc. Weiser's goal is to achieve the most effective kind of technology, which is available throughout the physical environment, while making them effectively invisible to the user. In the area of web applications, ubiquity is not seen as visionary in this highly pervasive sense, meaning that computing power is embedded everywhere. Rather, *ubiquitous web applications* build more on existing technology, in that web access is no longer primarily a domain of browsers based on desktop PCs but more and more done by various commercially available mobile devices. In general, ubiquity offers new opportunities and challenges for web applications in terms of *time-aware* [KLEIN99], *location-aware* [GROSS01], *device-aware* [RODRI01] and *personalized services* [KOBSA01]. This implies that ubiquitous web applications have to take into account, individually for each user, time and location of access, together with the different capabilities of devices comprising display resolution, local storage size, method of input and computing speed as well as network capacity. Consequently, the fundamental objective of ubiquitous web applications is to provide services not only to people at any time, any where, with any media but specifically to communicate the *right thing* at the *right time* in the *right way*. The pre-requisite for this is that the application is aware of it's *context* [ABOWD99], [WEISE91]. For developing ubiquitous web applications, one must understand *what context is* to determine its relevancy and *how it can be exploited for adapting* the provided services towards this context, by *customisation*[2].

Considering the notion of customisation from a historical point of view, it represents a major challenge at least since the end user has been put in the middle of concern when developing interactive applications. An area dealing with customisation issues already for a long time is the user interface community, which brought up the notion of *adaptive user interfaces*, cf., e.g., [GOOD84]. Adaptive user interfaces are designed to tailor a system's interactive behaviour considering both individual needs of human users and changing conditions within an application environment. The broader approach of *intelligent or advisory user interfaces* includes adaptive characteristics as a major source of its intelligent behaviour, cf., e.g., [CARRO88]. Another area dealing with customisation but emphasising more on adapting the content of an application are *information filtering and recommender systems* [ARAGA01]. The goal of these systems is to go through large volumes of dynamically generated textual information and present to the user those which are likely to satisfy his/her information requirements. With the emerge of hypertext the need for alternative access paths to information in terms of, e.g., different navigation structures became prevalent leading to another research direction called *adaptive hypertext and hypermedia* [DEBRA99]. Last but not least, the proliferation of mobile computing and mobile web applications, in particular, makes it necessary to consider not only the user preferences but also the environment in terms of, e.g., *location information or device characteristics* in order to adapt the application properly [OPPER99]. Recently, also *standardisation efforts* have been undertaken to collect requirements and provide representation techniques with respect to certain ubiquity issues particularly focusing on device independency and personalisation cf. [W3C99], [W3C000] [W3C01a], [W3C01b].

Learning from these different areas, the work described in this paper complements the efforts mentioned above by proposing a *design space of customisation* which can be characterised, by two orthogonal dimensions, comprising *context* and *adaptation*, and the mapping in between represented by the notion of customisation (cf. Figure 1).

[1] The work presented herein is supported by the EU-funded project *Ubiquitous Web Applications* IST-2000-25131 and the strategic research project *CustWeb* at the Software Competence Center Hagenberg (SCCH).
[2] Please note that, as will be seen further on, our notion of customisation is broader than the one presented in DIP-5 of [W3C01a]

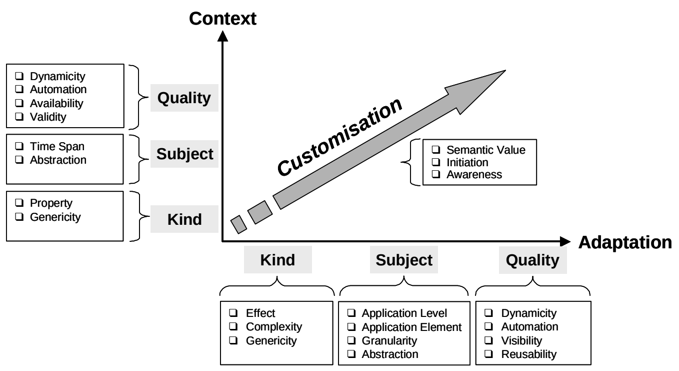

Figure 1. Design Space of Customisation

The benefit of this design space is threefold. First, it allows a structured, uniform view to better understand the various aspects of customisation. Second, it can be used as a conceptual framework for evaluating existing approaches on customisation. Third, from a software engineering point of view, it may be employed for developing a comprehensive method for modelling ubiquitous web applications, cf. Section 5. In the following, the design space of customisation is explained in more detail.

2 Context

The *context* dimension includes the circumstances of consumption relevant for a ubiquitous web application, mainly dealing with the question "why to customise and when". In this respect, we define context as the *reification of certain properties*, describing *the environment of the application* and *some aspects of the application itself,* which are necessary to determine the need for customisation. Context can be further specified by looking at the *kind of context,* which is considered for customisation, its *subject* and some of its *qualitative aspects*.

2.1 Kind of Context

Property. Context properties can be categorized following [SCHMI99] into *natural context*, *technical context*, and *social context*. Natural context comprises properties like the *location* from where the application is accessed and the *time* allowing customisation with respect to certain temporal constraints such as opening hours of shops or timetables of public transportation. Technical context includes properties of the device and browser, *network* properties like bandwidth and network protocol, and also about the state of the *application* itself. Finally, social context holds properties about the *user* accessing the application taking into account the necessity of personalisation.

Genericity. Context can be separated into an *application specific* part which is needed for a particular ubiquitous web application only and an *application independent* part, being *generic* and thus can be reused for other web applications. Application independent context may even be provided by external sources, e.g., third-party providers. An example for such an external source would be a GIS (Geographical Information System) providing geographic data about cities, streets, terrain for a given geographic latitude and longitude. There are already first attempts to develop universal components providing context cf., e.g., the ContextToolkit [ABOWD99] or the NEXUS project [GROSS01].

2.2 Subject of Context

Time Span. Practice has shown that it is useful to broaden the view on context by considering not only the *current context* at a given point in time but also *historical information.* This is necessary to be able to identify changes in the values of the context over time. For example since the bandwidth might be constantly changing it might be more important to be able to trace the average bandwidth instead of just having information about the latest one. In contrast, allowing to customise towards a restricted display size requires information of the current device only. Besides historical context, it might be useful to anticipate possible *future states* of a context too. For example, concerning video streaming, it is not only relevant how the bandwidth changed in the past, but also how the bandwidth will develop or how stable it can be considered in the future, to be able to tune the resolution of the video accordingly, thus guaranteeing a constant video stream.

Abstraction. According to the level ob abstraction where context properties are situated one can distinguish *physical context* which can be directly sensed from the environment (e.g., location in terms of a mobile phone's cellID) and *logical context* which can be built from existing physical or other logical context by applying different kinds of abstraction mechanisms such as *aggregation* and *fusion* [ABOWD99], [SCHMI99]. Aggregation means to derive a certain logical context on bases of the accumulation of values of a *specific* context property (e.g., certain geographical positions relate to the same location in terms of a city), fusion is based on *different* context properties (e.g., "Vienna at night" is a fusion of a certain location and time context).

2.3 Quality of Context

Automation. Context information can be provided *manually*, *automatically* or *semi-automatically*. For ubiquitous web applications it is desirable to automatically gather as much context information as possible to reduce user interaction. Additionally, context information may be automatically computed on the basis of other context information available or the application itself. Projection of bandwidth available in the future is an example of context information constructed automatically on bases of the history of bandwidth. Building user categories on the bases of interaction patterns would be another example. Semi-automatic means that automatic construction is accompanied by information entered manually by, e.g., a user, a designer or the vendor of a device.

Availability. It need to be considered that the values of certain context properties might not be available at some point in time. For example, the availability of information about the location obviously depends on the capability of the device used. The goal should be, of course, to design the application to be robust enough to deal with missing context, in the most extreme case, to run without getting any given context [BADRI00]. In this case, there could be a *default context* provided, or the user is requested to *provide the context manually*, or the service simply cannot be requested.

Dynamicity. Considering the frequency of context changes, context can be either *static*, i.e., determined once at application start up (e.g., the device used to select the appropriate interaction style), without considering any further changes or *dynamic*, i.e., determined during runtime, on every change (e.g., the bandwidth to adapt the resolution of an image on the fly).

Validity. Context properties may not be valid during the complete period until they are sensed from the environment again. In a mobile scenario an example would be that if location information doesn't change within a certain period, the device might not be online any longer, thus the location information might no longer be valid. Another example would be the validity of opening hours, which for example, may change during the seasons. Thus it may be required to specify the validity period during which a context is valid.

3 Adaptation

The second dimension is covered by the notion of adaptation, characterised by the *kind of adaptation*, i.e., what changes have to be done, *the subject of adaptation* in terms of what to change and the *quality of adaptation*.

3.1 Kind of Adaptation

Effect. Basically, adaptation may *enhance* a web application by adding certain parts (e.g., presenting a personalised advertisement), *reduce* a web application by removing certain parts (e.g., disabling all images) or perform some *transformations* which are often a combination of enhancement and reduction (e.g., showing textual descriptions instead of video clips).

Complexity. Adaptation can be *atomic* or composed of *series of other adaptations*. A transformation of the presentation of a movie from a video sequence into an alternative series of scene pictures is an example of a complex adaptation since it consists of disabling the video and enabling the presentation of the scene pictures.

Genericity. Likewise the context, also adaptation may be *application specific*, i.e., just meaningful for a certain web application (e.g., an adaptation performing the calculation of a certain discount) or may be *generic*, i.e., independent of any web application (e.g., a link can be enabled or disabled, the resolution of an image can be reduced or increased).

3.2 Subject of Adaptation

Application Level. All levels of a web application like described in [KAPPE00] may be subject to adaptation, comprising *content level* (i.e. domain-dependent data), *hyperbase level* (i.e., the logical composition of web pages together with the navigation structure), *presentation level* (i.e., the layout of each page and user interaction), along with the customisation itself. Regarding a certain adaptation, one or more of these levels may be affected. In some cases the adaptations are local to one level. Changing the display colour of headlines in a page is an adaptation solely performed at the presentation level, none of the other levels need to reflect that adaptation. In other cases it will be required that adaptations done at a certain level are propagated to the other ones. For example, should the content of the web application be reduced from an image rich presentation to a simpler text based version, not only the content representation needs to be adapted but likewise, the hyperbase as well as the presentation needs to reflect the changes, thus being adapted accordingly. Adaptation may not only affect the web application but may also change the customisation itself by enabling and disabling certain customisations.

Application Elements. Each of the levels mentioned above comprises a number of different application elements like *pages*, *links*, *access structures*, *input fields*, *lists*, and *media types*. For each of these application elements, different adaptations may be applied. For example, video data may be adapted by means of compression methods reducing the resolution or dropping frames whereas access structures may be adapted from a guided tour navigation style to an index navigation style.

Granularity. Another important issue concerns the *granularity of adaptation*, indicating the number of application elements affected by a certain adaptation. The granularity ranges from *micro adaptation* to *macro adaptation*. Whereas micro adaptation is concerned with fine-grained adaptations by affecting a single application element only (e.g., disabling a certain link on a certain page), macro adaptation means that rather large parts of an application are adapted, thus affecting multiple application elements (e.g., changing the language effects every application element visible to the user). Note that, there is no exact border between micro and macro adaptation. In its most extreme form, macro adaptation simply means that depending on the context, the whole application realising a certain service is substituted by another one, thus better fitting the context.

Abstraction. Another important issue concerning adaptation is whether it is done at *type level* or at *instance level*. An example for adaptation at type level, also known as schema evolution, would be to change the structure of a page by adding or removing attributes. Adaptation at the instance level is conducted if, e.g., the instances within a list of restaurants are filtered to show only particular restaurants according to the user's preferred price level.

3.3 Quality of Adaptation

Dynamicity. Likewise the context, adaptations may be either *static* or *dynamic*. Static adaptation means that the result of adaptation is already defined at design time whereas dynamic adaptation means that the result of adaptation is generated at runtime of the application. An example for static adaptation is to predefine two versions of an image, one with a high resolution intended to be presented on desktop computers and a low-resolution black-and-white one intended for handheld devices. The actual version of the image which is presented is then just chosen at runtime. In contrary an image might be dynamically adapted relative to the bandwidth currently available since the different network bandwidth situations can hardly be pre-assumed. Most of the times a ubiquitous web application is neither completely statically adapted nor completely dynamically adapted but rather incorporate both forms of adaptation in order to appropriately deal with different real world situations.

Automation. Adaptation may be applied fully *automatically,* so that the user can not take influence on the adaptation, *manual*, i.e., the user can decide which adaptation should be applied or *semi-automatic* meaning that the user has some influence whether an adaptation is applied or not. In general it should be dependent on the preferences of the user whether the adaptation is automatic or not. For this, it is necessary that the user is able to understand the consequences of adaptations which could be facilitated by means of proper tool support [MCILH98]. The replacement of images because of device restrictions for example, should be done automatically, whereas the decision whether response time or resolution quality is of higher importance should be made by the user.

Visibility. Visibility of adaptation addresses the issue for whom a certain adaptation is done and perceivable. Adaptations may be visible *globally*, i.e., by all users of the ubiquitous web application meaning that the adaptation affects the state of all sessions. An example would be to readjust the navigation structure of a web site. Adaptations may also be visible by certain *classes of users* or certain *individual users* only. The former is often found when applications are adapted according to different user expertise, separating into novice and expert users. Adaptation towards individual users would be, e.g., if certain recommendations were made, depending on the items bought by the user.

Reusability. Adaptation may be done *from scratch*, i.e., each time adaptation is required, it is done based on the original (non-adapted) version of the service [W3C99]. For example, transforming a video to audio and to a series of picture scenes need to be conducted from the original video. In contrary, adaptation may also be performed *incrementally*, meaning that subsequent adaptations are conducted on the results of previous adaptations, hereby reusing the adaptation output of the previous adaptation operation.

4 Customisation

Regarding customisation as a whole, there are some crucial issues which have to be discussed, comprising the *semantic value of customisation,* the *initiation of customisation,* and finally, *customisation awareness*.

Semantic Value. In the course of customisation it is important to reason about the semantic value, which should and can be provided by a customised service. The semantic value describes the quality of the output of the customisation relative to a non-customised version of the service. The adaptation may *enhance the semantic value* of a service, *reduce the semantic value*, or it may *preserve the semantic value*, thus achieving *semantic equivalence*. Especially personalisation and location-aware adaptations endow the application with semantic enhancement, in that each particular user is provided with specific added value. On the other hand, the same application customised for the same user may (and certainly does) look different when it is run on different devices and/or in different situations. This is inevitable (for example it is impossible to show that beautiful applet on a PDA with no virtual machine installed), but the service (or the added value) provided to the user should nevertheless be the same. In this case customisation enables to maintain semantic equivalence, which means that, despite the different context, the value provided to the user should still be the same. The semantic value may be judged on either *objective* or *subjective* bases. Some adaptations may result in an objectively semantically equivalent version (e.g. translating a text from English into Sanskrit) but may be perceived by the user as semantically not equivalent (supposing, e.g., the user is not able to understand Sanskrit). An example where the information is reduced (thus objectively not equivalent) but perceived as semantically equivalent by the user, is the transformation of a digital audio into mp3 format. Though the latter includes only a subset of the information of the digital audio version, the loss is (nearly) not perceivable by the user.

Initiation. An important issue refers to the question at which point in time customisation is actually initiated. First of all, customisation can be initiated each time a change in the context comprising the environmental properties occurs, independent of any user's request, thus being referred to as *immediate customisation*. In this respect, the adapted version of the affected part of the application is only pre-generated for further use or additionally directly pushed to the respective user(s). Immediate customisation can be found for example in [FOX96] where as soon as the bandwidth changes the appropriate customisations are performed. A second possibility would be to initiate customisation on the fly, i.e., not before a user requests a certain page being affected by customisation, which is called *deferred customisation*. A third alternative would be to trigger customisation periodically, called *periodic customisation*.

Awareness. Another crucial question is how far the ubiquitous web application is aware of customisation in terms of knowing about *context* and *adaptation*. When looking at existing approaches, one can encounter three different alternatives. Many of the existing approaches employ customisation *external* to the application itself, meaning that the application responds to a request and delivers the result regardless of any customisation (cf. [KAPPE00]). Customisation is rather done in a successive step on basis of the result, delivered by the application. In this case the application is *completely unaware of customisation*, thereby, however, limiting the potential of customisation. Such approaches very often use a proxy-based architecture where the proxy is responsible for customisation (cf. e.g., [SMITH98], [ORACL02], [FOX96]). In case that, not only the complete response of an application is considered by customisation but also the intermediate results, produced by a certain application level, before passing further on to the next level, *inter-level customisation* takes place. Consequently, at least the subsequent level is aware of the customisation done upon the previous level (cf. e.g., [CERI00], [FRATE99]). The most powerful form of customisation is achieved, if the adaptation can be performed already on the results of each single computational step within each level called *intra-level customisation* (cf. e.g., [ROSSI01], [KAPPE01b]).

5 Outlook

Based on the design space described in this paper we currently develop a modelling method for ubiquitous web applications focusing on customisation using UML [RUMBA98] as the basic formalism. Our approach [KAPPE01b] , [KAPPE01c] adopts a broad view on customisation in that it takes into account all issues resulting from the *anywhere/anytime/anymedia paradigm*, herewith providing both, semantic enhancement and semantic equivalence. Special focus is given on *dynamic context* and *dynamic adaptation* as well as on both *immediate* and *deferred customisation* to reflect the dynamic nature of ubiquitous web applications. Finally, we adhere to *intra-level customisation* to be able to customise even single model elements within a certain application level. The basis for accomplishing such a broad view of customisation is provided by a *conceptual architecture for customisation design*, consisting of three major components. First, the *context component* provides detailed information about the environment of an application and the application itself, thereby triggering the actual customisation as soon as the context changes. Second, a rule-based mechanism [KAPPE98], [KAPPE01a] in terms of *customisation rules* is employed in order to specify the actual customisation. Third, for separation of concerns, the application is divided into a *stable part*, comprising the default, i.e., context-independent structure and behaviour of the application and a *variable, context-dependent part*, thus being subject to adaptations. To support this architecture, a *generic customisation model* is proposed, offering to the customisation designer appropriate model elements.

References

[ABOWD99] G. D. Abowd: "Software Engineering Issues for Ubiquitous Computing", Int. Conf. on Software Engineering, Los Angeles, 1999.

[ARAGA01] V. Aragao, A. Fernandes, C. Goble: "Towards an Architecture for Personalization and Adaptivity in the Semantic Web", Int. Conf. on Information Integration and Web-based Applications & Services, Linz, Austria, Sept., 2001.

[BADRI00] B. Badrinath et al. "A conceptual framework for network and client adaptation", Mobile Networks and Applications, 5(4), 2000.

[CARRO88] J. M. Carroll, A. P. Aaronson, "Learning by Doing With Simulated Intelligent Help", CACM, 31(9), Sept. 1988.

[CERI00] S. Ceri, P. Fraternali, A. Bongio: "Web Modeling Language (WebML): a modeling language for designing Web sites", 9th World Wide Web Conference, Amsterdam, May 2000.

[CHAKR00] D. Chakraborty, H. Chen: "Service Discovery in the future for Mobile Commerce", ACM Crossroads, Winter 2000.

[DEBRA99] P. De Bra: "Design Issues in Adaptive Web-Site Development", 2nd Workshop on Adaptive Systems and User Modeling on the WWW of the 8th International Word Wide Web Conference, Toronto, Canada, 1999.

[FOX96] A. Fox, E. Brewer, S. Gribble, E. Amir: "Adapting to Network and Client Variability via On-Demand Dynamic Transcoding", ACM Seventh International Conference on Architectural Support for Programming Languages and Operating Systems, 1996.

[FRATE99] P. Fraternali: "Tools and approaches for data-intensive Web applications: A survey", ACM Computing Surveys, 31(3), Sept. 1999.

[GOOD84] M. D. Good, J. A. Whiteside, D. R. Wixon, S. J. Jones: "Building a User-Derived Interface", CACM, 27(10), Oct. 1984.

[GROSS01] M. Großmann, A. Leonhardi, B. Mitschang, K. Rothermel: "A World Model for Location-Aware Systems". Informatik, 8(5), 2001.

[KAPPE98] G. Kappel, W. Retschitzegger: "The TriGS Active Object-Oriented Database System - An Overview", ACM SIGMOD Record, 27(3), Sept. 1998.

[KAPPE00] G. Kappel, W. Retschitzegger, W. Schwinger: "Modeling Customizable Web Applications - A Requirement's Perspective", Int. Conf. on Digital Libraries: Research and Practice, Koyoto, Japan, Nov. 2000.

[KAPPE01a] G. Kappel et al.: "Bottom-up design of active object-oriented databases", CACM, 44(4), 2001.

[KAPPE01b] G. Kappel, W. Retschitzegger, W. Schwinger: "A holistic view on web application development - the WUML approach", Tutorial notes at First International Workshop on Web-oriented Software Technology, Valencia, Spain, June 2001.

[KAPPE01c] G. Kappel, W. Retschitzegger, W. Schwinger: "Modeling Ubiquitous Web Applications - The WUML Approach", Int. Workshop on Data Semantics in Web Information Systems, Yokohama, Japan, Nov. 2001.

[KLEIN99] L. Kleinrock: "Nomadicity: Anytime, Anywhere In A Disconnected World", Mobile Networks and Applications, 1(4), Jan. 1996

[KOBSA01] A. Kobsa: "Generic User Modeling Systems", User Modeling and User-Adapted Interaction, Vol. 11, 2001.

[MCILH98] M. McIlhagga: A. Light, and I. Wakeman: "Towards a Design Methodology for Adaptive Applications", Fourth annual ACM/IEEE International Conference on Mobile Computing and Networking, Dallas, TX, USA, Oct. 1998.

[OPPER99] R. Oppermann, M. Specht: "A Nomadic Information System for Adaptive Exhibition Guidance", Int. Conf. on Hypermedia and Interactivity in Museums, Washington, Sept. 1999.

[ORACL02] P. Waddington, P. J. Gill, "Oracle9i Application, Server Wireless Edition in Action", Oracle Magazine, Jan. - Feb. 2002.

[RODRI01] J. R. Rodriguez et al. :"Extending e-business to Pervasive Computing Devices - Using WebSphere Everplace Suite Version 1.1.2", IBM Redbooks, International Technical Support Organisation, SG24-5996-00, 2001.

[ROSSI01] G. Rossi, D. Schwabe, R. M. Guimarães: "Designing Personalized Web Applications", Int. World Wide Web Conf., Amsterdam, 2001.

[RUMBA98] J. Rumbaugh, I. Jacobson, G. Booch: "The Unified Modeling Language Reference Manual", Addison-Wesley, 1998.

[SCHMI99] A. Schmidt, K. A. Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven, W. Van de Velde: "Advanced Interaction in Context", First Int. Symposium on Handheld and Ubiquitous Computing, Karlsruhe, Germany, 1999; LNCS; Vol 1707, Springer, 1999.

[SMITH98] J. R. Smith, R. Mohan, C.-S. Li: "Content-based Transcoding of Images in the Internet". Int. Conf. on Image Processing, Oct. 1998.

[W3C99] W3C: "PIDL - Personalized Information Description Language". W3C Note, http://avocado.w3.mag.keio.ac.jp/TR/NOTE-PIDL, 1999.

[W3C00] W3C: Composite Capabilities/Preference. Profiles, http://www.ccpp.org/, 2000.

[W3C01a] W3C: "Device Independence Principles", W3C Working Draft, http://www.w3.org/TR/di-princ/, Sept. 2001.

[W3C01b] W3C: Platform for Privacy Preferences (P3P) Project, http://www.w3.org/P3P, 2001.

[WEISE91] M. Weiser: "The Computer for the 21st Century", Scientific American, 265, 3, September 1991.