Containment: from Context Awareness to Contextual Effects Awareness

Boris Dragovic and Jon Crowcroft
the Computer Laboratory, University of Cambridge, UK
{name.surname}@cl.cam.ac.uk

Abstract

Context plays a key role, as recongnized by a wide body of research, in application and entity adaptation in the ubiquitous computing world characterized by extensive platform heterogenity and environment dynamicity and unpredictability. Implicit in the notion of context, as used by context-aware applications, is the actual effects, including constraints, context has on target entities. We believe that making a step further from explicit reasoning about context to explicit reasoning about its implicit effects will facilitate more effective and flexible adaptation. In this work we present an approach to modeling the world based on natural notions of container and containment and show how it enables explicit reasoning about and acting upon context-implied effects on target entities, data objects in particular. We also outline a practical use of the model through its application in a system for autonomic context-aware infromation security and privacy protection.

1 Introduction

The notion of context plays an important role in enabling the vision of ubiquitous and pervasive computing [25]. Characterized by heterogeneity, environment unpredictability and dynamicity ubiquitous computing requires explicit reasoning about context to provide for relevant application and entity adaptation. To meet this aim, the availability of contextual information is exploited in a number of ways in pervasive computing [7].

Despite the huge body of research and knowledge, as outlined and summarized in [19], the notion of context remains elusive with respect to its exact definition or categorization. The former in general lack precise, mathematical style, specification. They are also highly application specific and tend to use natural language geared often resulting in cyclic term definitions, e.g. "context is a entity relevant *state of environment*". Context categorization efforts suffer from a similar set of problems. Although a distinction can be made between categorizations according to application domain and according to point of view [19] a single, unified, approach seems nowhere on the horizon. Contrary to the efforts, universal context definition's and categorization's elusiveness might be a feature rather than a bug, showing us that the right way of thinking about context in general is in its more abstract sense.

Irrelevant of a particular context definition or categorization the very notion of context as used in context-aware computing has two fundamental aspects: a particular set of entities affected by a context and a set of contextual effects of interest. Both are application-specific. The mapping from context description to a set of contextual effects affecting an entity is often implicit in application design or policy specification, e.g.: a set of constraints imposed by a device's rendering capabilities on presentation of a document; a set of options available in "activity-aware" decision support applications; a set of security and privacy threats present for a entity in its environment.

In this paper we neither offer a novel definition of context nor propose a context categorization approach. What we are interested in is modeling, reasoning about and controlling effects that contextual states imply for target entities. The advent of ubiquitous and pervasive computing has necessitated a move away from static security policies to more dynamic models [17], supporting explicit reasoning about context. In analogy,

with a push towards autonomic computing [15] ability to analyze contextual effects explicitly and in a dynamic fashion is required. Establishing dependencies between contextual effects of interest and environment reconfigurations is one of the fundamental steps towards the goal. It will facilitate contextual effects control through a more flexible, effective and autonomic means of adaptation. As the basis of our approach we offer the *container* paradigm as a foundation for structuring the world, Section 2, and show how it may be used to achieve the above goals. We also design the model in an information centric-way i.e. we structure the model so that it facilitates reasoning about the contextual effects affecting the information existing in an ubiquitous setting through its representation in a form of data objects. To further support the proposed model we present, in Section 3 an application in the area of ubiquitous computing security that uses the model to provide autonomic context-adaptive information security and privacy protection.

2 Containers and Containment: Modeling the World

2.1 Container: The Definition

We define a *container* to be a physical or virtual enclosure, a bounded region with a distinctive interior, boundary and exterior, in which another container, or ultimately a piece of information, may exist.

2.1.1 Container Classification

Containers are classified into a container class hierarchy based on their characteristics and primary functionality. Container characteristics are inherited down the hierarchy going from abstract towards more specialized classes. An example classification is depicted in Figure 1. At the top level, we define the classes directly inheriting from the *container* as *physical*, *intermediate* and *virtual* containers while the lower levels are application specific. Physical containers exist in the physical world, i.e. are object of three dimensions, such as e.g. a room or a space within a secure perimeter. Virtual containers, on the other hand, exist solely in the virtual, digital, realm such as e.g. a GUI window, a file system, a file, a TCP packet or a IPSEC/SSL tunnel. Intermediate containers represent a bridge between the physical and the virtual realms by being physical objects or a composition, in the UML sense, of physical objects but containing only virtual entities. Examples of intermediate containers are: a mobile phone, a laptop, a storage device, a display, a communications link etc. Classification granularity impacts on the physical and intermediate class boundaries.

We define a *containable* relationship to denote, for each container class, which container classes may "fit", i.e. be contained, within it. For example, a file system may contain a file, a communications link may contain a TCP packet or an IPSEC tunnel etc. In the general case, physical containers may contain further physical or intermediate containers - based on their physical characteristics, namely the size/volume; intermediate containers may contain intermediate or virtual containers while virtual containers may contain only containers of the same class.

We define the *data object* class as an atomic class in a way that it denotes pure information content and may not contain any other container classes. The notion of a data object is different from the traditional notion of a file and represents a collection of information indivisible according to some criteria, e.g. a security classification. A file, being a virtual container, may contain one or more data objects, e.g. a document containing distinctive paragraphs of text, pictures and tables.

2.1.2 Context and Container: The Transparency

Inherent to the notion of container is its boundary, either physical or virtual. With respect to the container boundary we can divide the context into internal and external. To influence the internal context, effects implied by the external context have to cross the container boundary. For example, if it is daylight and if a room has an outside window then there is going to be daylight in the room as well; or if a person has physical access to a mobile device and the device is not tamper resistant then physical access to the stored information is implicit.

Container *transparency* denotes filtering characteristics that a container's boundary poses for different types of contextual effects crossing it. With respect

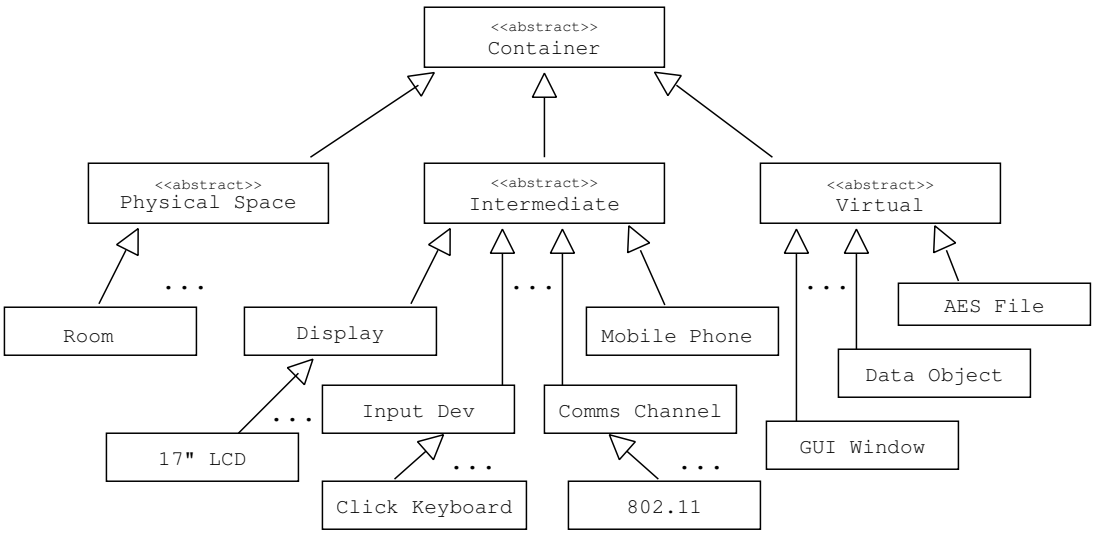


Figure 1. Example container classification.

to a specific contextual effect a container's boundary may be: *opaque*, in which case the effect does not cross the boundary and thus has no influence on container's internal context; *fully transparent*, when the boundary poses no barrier for the effect; and *partially transparent*, when the boundary has an qualitative impact on the effect. Apart from being a function of container's class and contextual effect type, transparency is affected by the internal state of a container, e.g. the level to which glass in a window is dimmed impacts the amount and the specter of daylight that enters the room while the level of tamper resistance of a device determines the skill, determination and knowledge required to access the data stored within.

2.2 Containment: the Model of the World

We model the world, in graph theoretic notion, as a finite-path-length, finite-degree, rooted trees in which nodes represent containers and directed edges represent containment. We call these trees *Containment Trees*. The finiteness of a containment tree is guaranteed by the bounding characteristics of containers in conjunction with the existence of the minimum granularity container - the data object. Containers of class data object are always leaf nodes of a tree. The notion of the Containment Tree is similar to the notion of Information Tree in [6]. When we refer to containment with respect to a particular container we assume the sequence of containers from the containers tree root to the container.

2.2.1 Containment Expressions

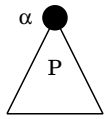
Containment Expressions represent the syntax of Containment Trees and draw from Cardelli's work on Ambient Calculus [6]. To present the syntax of containment expressions we break them down into atomic expressions and provide a graphical representation of the matching containment tree fragment. We also use the following conventions in any further reference to the expressions: lower-case Greek letters, e.g. α, are used to denote a particular container's class without any of its contents; capital letters from standard English alphabet, e.g. P, are used to represent individual containment trees.

To start, absence of contents at any level is represented simply by 0. At the top level, on its own, 0 represents an empty world.

A tree, with only a root node labeled α is written as the expression α:

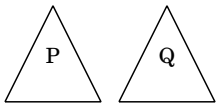
A tree, with a root node labeled α, leading to a subtree represented by *P* is written as the expression $\alpha[P]$:

α[P]



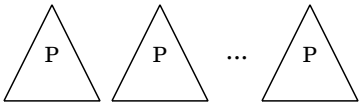
A forest, consisting of two trees *P* and *Q*, is written as the expression $P|Q$:

P | Q



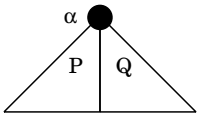
Multiple instances of the same tree *P* is written as the expression $!P$:

!P



A tree obtained by joining two trees *P* and *Q* at the root α is written by the expression $\alpha[P|Q]$:

α[P | Q]



2.2.2 State of the World

Using the containment syntax the state of the world at any point in time can be represented as:

$$
\begin{aligned}
world &\leftarrow world|world \\
world &\leftarrow pspace \\
world &\leftarrow intermediate
\end{aligned}
$$

$$
\begin{aligned}
pspace &\leftarrow pspace|pspace \\
pspace &\leftarrow pspace[pspace] \\
pspace &\leftarrow pspace[intermediate] \\
pspace &\leftarrow 0
\end{aligned}
$$

$$
\begin{aligned}
intermediate &\leftarrow intermediate|intermediate \\
intermediate &\leftarrow intermediate[intermediate] \\
intermediate &\leftarrow intermediate[virtual] \\
intermediate &\leftarrow 0
\end{aligned}
$$

$$
\begin{aligned}
virtual &\leftarrow virtual|virtual \\
virtual &\leftarrow virtual[virtual] \\
virtual &\leftarrow 0
\end{aligned}
$$

where *pspace*, *intermediate* and *virtual* represent instances of container classes physical, intermediate and virtual container or any inheriting classes respectively. The containable relationship needs to be obeyed at each level in a containment tree for it to be *well-structured*.

Figure 2 represents a partial snapshot of a state of the world representing two containment trees. Double circled nodes in the figure denote containers of data object class.

To reflect dynamic changes in the configuration of the world we provide for updating the model through three operations: *enter* operation causes a container, or a containment, to enter another containment; *leave* operation is the converse; while *migrate* operation binds the previous two in an atomic way and denotes change of containment within a realm.

2.2.3 Path Expressions

To be able to reference a containment we use path expressions. A path can be defined as a sequence of containers linked by the *contains* relationship, written as

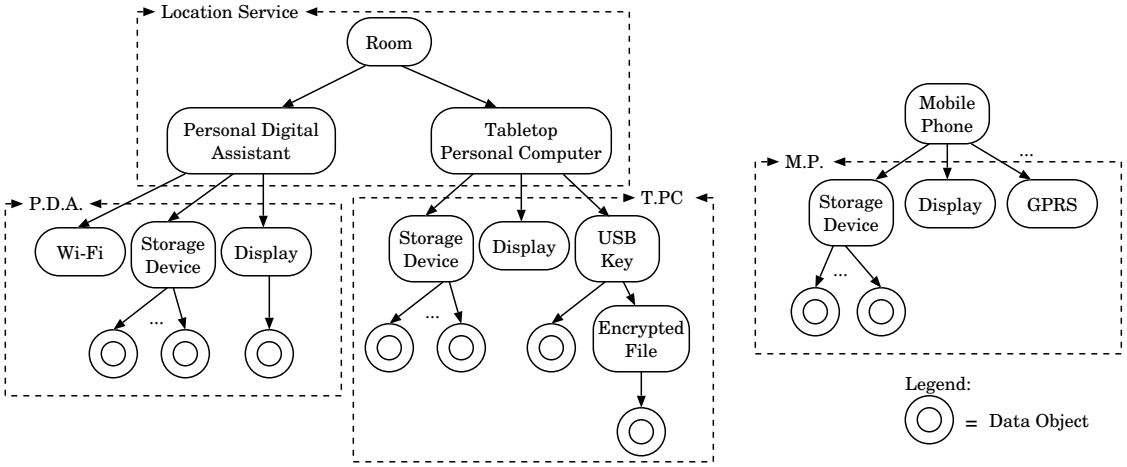


Figure 2. Partial snapshot of the Model of the World state.

$\rightarrow$. A sequence $\alpha \rightarrow \beta$ denotes that β is contained within α and that α is its immediate, first-level, container i.e. direct parent in the Container Tree representation. Path expressions are specified using the following syntax:

$$
\begin{aligned}
element \leftarrow\ & \alpha \\
& \mid \star \\
path \leftarrow\ & element \\
& \mid path\ /\ element \\
& \mid path\ /\ ...\ /\ element
\end{aligned}
$$

A matching set of a path expression is either an empty set or a set with one element where:

- A trivial expression element α matches a container of class α or of a more specialized class.
- Expression element $\star$ matches container of any class.
- Expression e_1/e_2 matches $\alpha_1 \rightarrow \alpha_2$ if e_1 matches α_1 and e_2 matches α_2.
- Expression $e_1/.../e_n$ matches $\alpha_1 \rightarrow ... \rightarrow \alpha_n$ if e_1 matches α_1, e_n matches α_n and all steps in between obey the previous rule.

The use of container classes in matching path elements, rather than unique container identifiers, shall be supported in the next section.

2.3 Containment Realms and Authorities

Although we talk about modeling "the state of the world" we do not envisage a holistic containment-based picture of a ubiquitous system to exist in a centralized fashion, let alone that such a requirement would be infeasible considering the nature and inherent characteristics of the ubiquitous computing world. The model is devised to be established and maintained in an distributed and independent fashion, representing only small portions of what would be a true holistic "state of the world", posing no consistency issues and used locally by ubiquitous devices and infrastructural services.

A device or service that is resource capable to establish and maintain a portion of the model is called a *model authority* or simply *authority*. Portion of the model, maintained by a single authority, is called a *model realm* or just *realm*. In Figure 2 we can distinguish four realms enclosed in dash-lined squares and labeled with their respective authorities: the personal digital assistant (P.D.A.), the table top PC (T.PC), the mobile phone (M.P.) and the location service. The fact that the mobile phone is not a part of the location service's realm denotes that it is either not-locatable by the particular technology employed or is out of its reach. Realm authorities are not necessarily represented as a node in the model, as is the case with the location service in the above example, which depends on they themselves representing a container class relevant for the model application.

The granularity of the model provided by an authority depends on its model establishment and maintenance capabilities. To model the full range of physical and virtual containers and maintain their state an authority needs support and awareness at both the system and application layers. The former is required for device hardware and operating platform software container representation. The latter is needed for modeling the application level containers such as e.g. GUI windows, application level communications channel tunnels or file types. Thus, an authority's individual level of model support defines the minimum local quality and quantity of service for the model application.

2.4 Contextual Effect Propagation

By exploiting container transparency, establishing context at any level in a containment tree allows us to determine its effects at any other level in the model. Consequently, contextual effects a data object is exposed to in a particular context can be determined by identifying a set of effects implicit in the context and reasoning about their propagation across boundaries of containers comprising in the data object's containment.

Figure 3 *a)* shows how a set of effects a container boundary is exposed to (Ψ) is affected by the boundary's transparency ($\Psi_{filtered}$) and combined together with a set of effects originating from inside the container (λ_i) propagated down a containment tree (Ω) - in set theoretic notation:

$$\Omega = \bigcup_{i=1..n}^{i} \lambda_i \bigcup \Psi_{filtered}$$

Figure 3 *b)* represents an illustration of how a contextual effect is propagated down a data object's containment. The changing thickness and solidity of the arrows representing the contextual effect denotes the effect of the container boundaries (horizontal lines).

Reasoning about contextual effect propagation is not to be confused with a much more general notion of inter-container context dependencies. A context dependency can be described as a situation in which a contextual state within a container depends on the contextual state of another container. These dependencies may span containers in ways which can not be expressed in the proposed containment-based model of the world. Resolving and acting upon context dependencies is seen as a job of context awareness mechanisms and techniques and it precedes reasoning about the contextual effects as presented here.

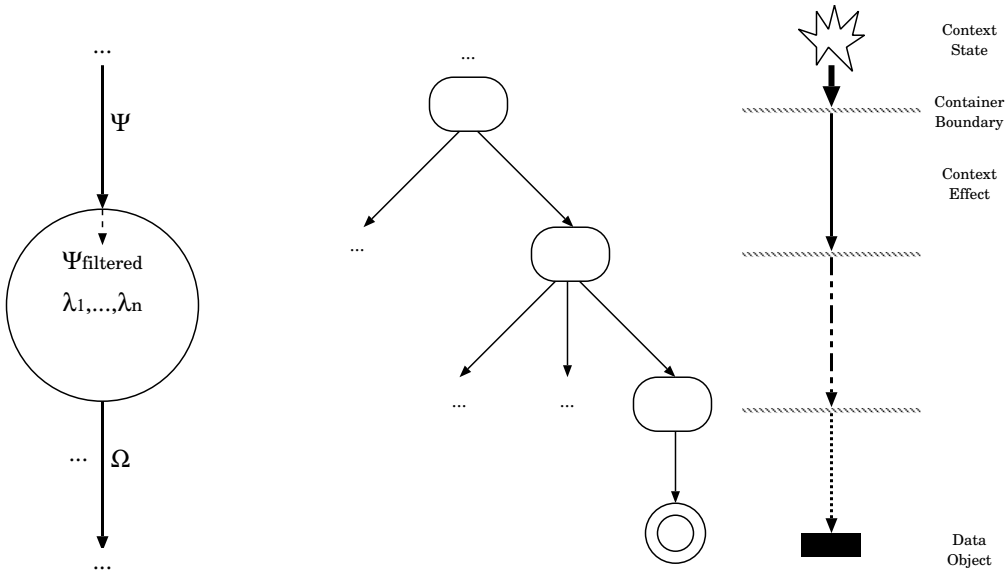
A container's boundary transparency, as stated previously, can change based on container's state, e.g. open vs. closed door of a room with respect to sound permeability. This enables, by controlling state of containers on the containment propagation path between the context occurrence and a data object, to affect the set and the degree of contextual effects experienced by the data object. Furthermore, the same can be accomplished by purposeful insertion of a new container on the path.

The main application target of the model is acting upon analysis of contextual effects experienced by a data object in an environment. For this we leverage container transparency as specified by the relevant container class. Thus using container classes, and not unique container identifiers, to express containment paths (Section 2.2.3) facilitates specifying policies to match *all* data objects affected by a set of contextual effects; rather than having a policy rule on per container instance basis causing unnecessary duplication. For example, a path */pda/storage_device* matches both */pda/hard_drive* and */pda/sd_card*, where *sd_card* and *hard_drive* are specializations of *storage_device* and where it is the *storage_device* class that defines transparency for the particular contextual effect.

2.5 Other Modeling Approaches - Related Work

Spatial models have been a focus of research in several different areas of computer science such as mobility theory, ubiquitous computing [14, 1] and spatial databases [21]. The very idea of a container as a basis for the proposed model stems from Egenhofer's geoinformation systems work on spatial reasoning algebras [10].

Theoretical foundations of mobility models have been laid by the work on the π-calculus [16], aimed at modeling distributed communications systems, and its variants such as asynchronous, distributed or nomadic π-calculi that added new concepts such as migration, site failure, located channels, permissions etc. The ap-



a) Propagation across container boundary.

b) Propagation down a containment.

Figure 3. Contextual effect propagation.

proaches to structuring the world in mobility models range from flat, in Mob_{adtl} [11], to hierarchical, as in Join-Calculus [12] or Ambient Calculus [5].

The presented model is significantly influenced by Cardelli's work on Ambient Calculus for mobile ambients [6] and draws from the later specialization of the model by Scott et al. [20]. Both the notion of a mobile ambient and of an entity, their nesting and migration, as used by Cardelli and Scott respectively resemble the role of a container and containment. However, both the Ambient Calculus, in a more formal way and with more expressive power, and Scott's work deal with enforcing policies on migrations of mobile computations and mobile agents respectively.

The key difference is in the underlying philosophy. We model the world using passive entities, containers, that have no computational power whatsoever. Rather than being concerned about the legality of migrations, i.e. model updates, and control over them we are interested in the way in which the represented entities affect external forces imposed on them, i.e. contextual effects, passively. We go further to analyse how compositions, i.e. nesting, of such entities and model reconfigurations affect contextual effects propagation. We depart from a binary, allow-deny, decision model to provide for a plethora of container and model operations to control the degree of contextual effect propagation.

Scott's approach, being more practical, is founded on premises of pervasive location service and embedding of a notion of an owner to the entities which both bare scalability issues. The proposed model, being inherently distributed in an independent fashion poses no such issues. One of the consequences of this is a support for variable levels of granularity at which different realms are maintained - which greatly aids the model deployment in ubiquitous computing environments.

3 Model Application

The presented containment-based model of the world is suitable for a class of context-aware applications that satisfy the following two fundamental requirements: the world can be structured as a forest of containment trees based on container classes that have an explicit role in reasoning about an external set of forces, e.g. contextual effects; those forces can be explicitly identified, e.g. application-specific effects implicit in context. Model flexibility allows for its application in a wide variety of settings.

We briefly present our application of the model in an *Autonomic System for Context-Adaptive Informa-*

tion Security and Privacy Protection.

3.1 Motivation

Ubiquitous computing vision [25] has brought about a number of challenges for security and privacy of information stemming from a number of technological and socio-technological reasons [24]. Some of the problems can be solved by adapting existing solutions from traditional distributed systems while the others need novel solutions. Examples of the latter would be secure device associations [23], location limited channels for authentication [3] or methods addressing specific usability issues [22].

The availability of contextual information plays an important part in reasoning about information security and privacy in the face of frequent and unpredictable context changes, as inherent in the ubiquitous computing world. In more traditional environments, characterized by the existence of a secure perimeter and its implications together with the limited means of information access and usage, contextual factors, being predictable, are reasoned about implicitly and built into static security policies. For adequate information security and privacy protection in ubiquitous computing we need explicit reasoning about the context, this is especially true for authorization and access control mechanisms [8] and for development of more dynamic, context-adaptive, security policies [18].

In our work on context-adaptive security, presented in detail in a companion publication [9], we address a subset of information leakage threats particularly exacerbated in the ubiquitous computing scenarios that we call *information exposure threats*. Their distinguishing characteristic is that they do not involve a malicious custodian[1]. Information exposure threats represent information leakage into the environment as a side-effect of the information management and handling procedures deployed in a particular context. They stem from a mismatch between: information sensitivity; context surrounding the information - determining the threat model; and a particular information management procedure employed - granting a level of protection in the context. Simple instances of the threat class involve sensitive information being: displayed in a form and on a screen visually accessible by a third party [**?**] [**?**]; taken out of a secure perimeter on a mobile computing or storage device unaccounting for the shift in threat model; transmitted in plain-text over a corporate wireless link whose signal penetrates into a publicly accessible area, etc.

Expecting users to reason about and act upon security issues of such complexity is highly unrealistic and contrary to the vision of the "disappearing computer". Thus, we develop an autonomic system for context-adaptive information security and privacy protection founded on the previously presented containment-based model of the world. The main goal of the system is to provide maximum information availability for information custodian while protecting its security and privacy according to the perceived threat level implied in current context at any point in time.

3.2 Levels of Exposure

An information exposure threat has two main characteristics: type and degree. The former determines the nature of a threat while the latter denotes the actual risk or the likelihood of the particular threat materializing in the given context. The notion of information exposure, as we define it, assumes information access. For illustrative purposes only we can typify information exposure threats according to the nature of information access implied as: physical, visual, audio and network access. Thus, for example, we could say that an information exposure threat described in natural language as "mobile device outside secure perimeter" implies a risk of physical access to information stored on the device due to increased likelihood of device abduction.

Unlike the usual binary, nothing or all, decision model of authorization policies we strive to provide maximum information availability while adequately protecting its security and privacy. The perceived or estimated degree of an information exposure threat plays an important role in this process. While the mere presence or absence of a threat would force a binary protective action employment such as e.g. information destruction i.e. deletion in the presence of a threat, the degree allows for a choice of matching

[1] A person in a legitimate possession or access to information as determined by external authentication and authorization mechanisms.

actions which balance information availability with its exposure. For example, considering the following three contexts: "inside a secure perimeter", "outside a secure perimeter" and "outside secure perimeter and owner away"; we could establish the respective degrees of physical access to information stored on a mobile device as: low, medium and high. This enables us to perform, for example, the following protective actions: none i.e. retain the information in its current form, encrypt the information and erase the information, respectively. Similar considerations would apply for a piece of sensitive information displayed on a public screen exposed to threats described as: "inside a secure perimeter" and "a third-person present" or "outside a secure perimeter". To mitigate these threats the GUI window hosting the information representation could be shrunk or migrated to an available mobile phone's display lowering the observability of the information.

Levels of Exposure (LoEs) are introduced to quantify the degree or likelihood of information exposure due to a specific threat or collection of threats present in an context and are used to discriminate between appropriate protective actions to be applied as hinted at above. As information of different sensitivity classes is expected to have different handling policies in the face of information exposure threats we specify LoE models on per information sensitivity class basis. The granularity of a LoE model for a sensitivity class depends directly on context capturing capabilities of the policy enforcement device and on the range of available protective actions. Individual LoE models can take the form anywhere from independent points in the threat-action space to structures like hierarchies or lattices.

3.3 The Role of the Containment Model

Information exposure threats are, as outlined previously, implicit in the context. We classify containers according to their primary functionality, e.g. a display, a keyboard or a storage device, but choose classes to be represented based on their distinctive transparency characteristics. Transparency of each of the container class' boundary is defined in terms of threat types, e.g. physical, visual, audio or network access, and its impact on the threat degree.

For example, consider a piece of information classified as *SECRET* within containment specified using a path expression as */pda/hard_drive/encrypted_file* and a threat described as "outside secure perimeter" characterized with types physical, visual and audio access and a degree of 8 out of 10. The LoE establishment for the relevant data object would proceed as follows: container class *hard_drive* is opaque for threats of visual and audio access but it is fully transparent for threats of the type physical access, therefore only the physical access threat of degree 8 is propagated further down the containment tree; container class *encrypted_file* is partially transparent for the threat and impacts its degree by 40%, the reasoning being that it is much easier to access information stored as plain-text than in an encrypted fashion, provided that the encryption key is secure; the resulting threat that the data object is exposed to is of type *physical access* with the degree of 4.8; considering that the information is classified as *SECRET* the degree may imply the LoE defined as *HIGH* which requires information destruction, in other words, the information is not allowed to leave the secure perimeter. For another data object classified at *CONFIDENTIAL* the same threat type and degree might have implied a LoE of *LOW* thereby denoting that information of that classification level may leave the secure perimeter stored on the hard drive if encrypted. Although simplistic, the example illustrates the containment based reasoning about context implied information security and privacy threats.

3.4 Protective Actions

From the point of view of information security and privacy protection the goal of the system is to maintain the lowest LoE for all data objects in all contexts - what we call the state of *homeostasis*. This is accomplished by two sets of protective actions: containment manipulation and information reduction. Containment modification actions are aimed at blocking threats before they reach data objects in question by exploiting container transparency characteristics. Containment modification actions consist of: a new container insertion somewhere on the threat's propagation path, e.g. file encryption, SSL tunneling; state alteration of an container already on the path, e.g. GUI window shrinking or a data object migration to a different

containment. Information reduction [13, 2] actions are aimed at reducing information content so as to lower its sensitivity classification and thus the LoE.

Ubiquitous device's context sensing capabilities, including user profiling i.e. context information solicitation, and containment-model granularity supported by the platform determines the granularity of the protective actions that may be applied ranging from: the binary, all or nothing, decision model to fine-grained container manipulation and information reduction techniques. Our system for autonomic context-adaptive security [9], thus, forms a specific set of policies based on *finite state automata with tautness functions* [4] for each of the enforcement devices based on their platform profiles to maximize information availability while enforcing appropriate information protection with respect to perceived information exposure threats.

4 Discussion and Conclusion

4.1 Discussion

Scalability. The main factor considering model scalability is the possible size of the container classification hierarchy. Individual container classes to be represented are chosen based on their distinguishing transparency characteristics for model application-relevant set of contextual effects. Thus, comparing to the envisaged software and hardware heterogeneity in the ubiquitous world, we expect the size of the classification to remain manageable. For example, considering the presented model application, we can divide all available storage devices into: fixed and removable, denoting the level of available control over stored data at all times; and tamper-resistant and otherwise, denoting the ease of physical access to the stored information.

The size of the model at every individual authority will, in-line with the container classification granularity, depend on container classes supported by the authority as determined by its hardware and software configuration. Representation of the model at an authority may range, application-specifically, from implicit, in cases where model is used only in an abstract way, e.g. to form policies, and there is no actual data structure representing the model, to explicit, in cases where containment configuration is needed run-time for reasoning about contextual effects or otherwise. Formal methods may be used to approximate individual containers into larger ones, maintaining model correctness while reducing its overall size.

Complexity. Complexities involved in model maintenance and use are highly application specific. Although the model maintenance overheads depend on the chosen representation, judging by its structure and the nature of update operations, we expect them to be close to trivial. Model use involved complexities may vary from, again, trivial where the model is used just for querying environment configuration at a point in time to substantially more significant in cases where explicit contextual effect reasoning process is performed on the model.

4.2 Conclusion

In this work we have presented a model that provides a unified representation of space, joining physical and virtual realms, based on the notions of a container and containment. We leverage inherent characteristics of a container, and its class, to model contextual effects propagation across its boundary. Together, these two pieces of work facilitate reasoning about and provide a means of localized reaction to the quality and quantity of contextual effects as experienced by a target entity in a dynamically reconfigurable environment. The model allows for independent and distributed maintenance at granularities matching available resources and capabilities of devices it is deployed on. This provides for minimum level of service guarantees to the model applications - making it particularly attractive for ubiquitous computing environments. To demonstrate its effectiveness we have briefly presented the use of model in a system for autonomic, context-adaptive, and fine-grained information security protection.

References

[1] N. Adly, P. Steggles, and A.Harter. Spirit: a resource database for mobile users. In *Proceedings of The ACM CHI'97 Workshop on Ubiquitous Computing*, 1997.

[2] D. E. Bakken, R. Parameswaran, D. M. Blough, A. A. Franz, and T. J. Palmer. Data obfuscation: Anonymity

and desensitization of usable data sets. *IEEE Security and Privacy*, 2(6):34–41, November/December 2004.

[3] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *Proceedings of Network and Distributed System Security Symposium*, 2002.

[4] J. Baliosian and J. Serrat. Finite state transducers for policy evaluation and conflict resolution. In *IEEE 5th International Workshop on Policies for Distributed Systems and Networks*, 2004.

[5] L. Cardelli. Semistructured computation. In *7th International Workshop on Database Programming Languages: Research Issues in Structured and Semistructured Database Programming*, September 1999.

[6] L. Cardelli and A. D. Gordon. Mobile ambients. In *Proceedings of The FOSSACS '98*, 1998.

[7] D. Chalmers. *Contextual Mediation to Support Ubiquitous Computing*. PhD thesis, Department of Computing, Imperial College London, 2002.

[8] M. J. Covington, W. Long, S. Srinivasan, A. K. Dev, M. Ahmad, and G. Abowd. Securing context-aware applications using environmental roles. In *Proceedings of the 6th ACM symposium on Access Controls models and technologies*, pages 10–20, 2001.

[9] B. Dragovic, J., P. Vidales, and J. Crowcroft. Autonomic system for context adaptive security in ubiquitous computing environments. Submitted for publication at ESORICS 2005.

[10] M. Egenhofer and A. Rodr?guez. Relation algebras over containers and surfaces: An ontological study of a room space. *Spatial Cognition and Computation*, 1(2):155–180, 1999.

[11] G. Ferrari, C. Montagnero, L. Semini, and S. Semprini. The mobadtl model and method to design network aware applications. Technical report, Computer Science Dept., University of Pisa, 2003.

[12] C. Fournet and G. Gonthier. The reflexive chemical abstract machine and the join-calculus. In *23rd ACM Symposium on Principles of Programming Languages*, 1996.

[13] A. Heuer and A. Lubinski. Data reduction - an adaptation technique for mobile environments. In *Interactive Apllications of Mobile Computing (IMC'98)*, 1998.

[14] J. Hightower, B. Brumitt, and G. Borriello. The location stack: A layered model for location in ubiquitous computing. In *4th Intl. Workshop on Mobile Computing Systems and Applications (WMCSA 2002)*, 2002.

[15] IBM. Autonomic computing manifesto. WWW, October 2001.

[16] R. Milner. *Communicating and Mobile Systems: The Pi Calculus.* Cambridge University Press, 1999.

[17] G. K. Mostefaoui and P. Brezillon. Context-based security policies: A new modeling approach. In *2nd IEEE Conference on Pervasive Computing and Communications Workshops*, 2004.

[18] G. K. Most?faoui and P. Br?zillon. Modeling context-based security policies with contextual graphs. In *Workshop on Context Modeling and Reasoning*, 2004.

[19] G. K. Most?faoui and J. Pasquier-Rocha. Context-aware computing: A guide for the pervasive compting community. In *The IEEE/ACS International Conference on Pervasive Services (ICPS'04)*, July 2004.

[20] D. Scott, A. Beresford, and A. Mycroft. Spatial security policies for mobile agents in a sentient computing environment. In *Proceedings of The FASE 2003*, 2003.

[21] S. Shekhar, S. Chawla, S. Ravada, S. Fetterer, and C. Liu. Spatial databases: Accomplishments and research needs. *IEEE Trans on Knowledge and Data Engineering*, 11(1):45–55, 1999.

[22] F. Stajano. One user, many hats; and, sometimes, no hat?towards a secure yet usable pda. In *Proceedings of Security Protocols Workshop*, 2004.

[23] F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ubiquitous computing. *IEEE Computer Supplement on Security and Privacy*, pages 22–26, April 2002.

[24] F. Stajano and J. Crowcroft. *The Butt of the Iceberg: Hidden Security Problem of Ubiquitous System*. Kluwer, 2003.

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