Designing Privacy-Enhancing Mobile Applications

Koen Decroix, Bart De Decker, Vincent Naessens

Designing privacy-enhancing mobile applications

Koen Decroix[1], Bart De Decker[2] and Vincent Naessens[1]

[1] Katholieke Hogeschool Sint-Lieven, Department of Industrial Engineering
Gebroeders Desmetstraat 1, 9000 Ghent, Belgium
firstname.lastname@kahosl.be
[2] Katholieke Universiteit Leuven, Department of Computer Science,
Celestijnenlaan 200A, 3001 Heverlee, Belgium
firstname.lastname@cs.kuleuven.be

Abstract. This paper evaluates the multi-paradigm modelling methodology for designing controlled anonymous applications, like applications that use trap-door anonymity. The methodology is applied to a class of mobile applications with strong security and privacy requirements. From this study, we detect the constraints in the existing methodology in the context of mobile applications, explore the solution space and define directions for research. Moreover, a first iteration of the realization of the existing design methodology is made.

Key words: Anonymity, privacy, design, methodology

1 Introduction

Multi-paradigm modelling [7] is a promising approach for domain-specific modelling. The design of an application within a particular domain consists of several stages. At each stage, multiple formalisms are introduced to tackle specific challenges at that phase and to evaluate (and estimate) non-functional properties of certain design decisions. Multi-paradigm modelling introduces a specific set of formalisms that are best suited to design applications within a particular domain. In the first phases, formalisms have enough power to capture essential requirements of applications within the domain. In intermediate design steps, formalisms are introduced to estimate major non-functional properties (such as performance, memory usage, etc.). In a late stage, formalisms are selected that allow for an easy mapping to code. This means that software components can easily be instantiated based on information that is extracted from the formalism. The modelling paradigm also foresees transformation rules to map a model within one formalism to a model within another formalism (later in the design).

The methodology has been successfully applied to design specific classes of applications, such as cyber-physical systems [5], component based robotics applications [6] and for visualization and behavior in mobile applications [2]. The modelling strategy has also been applied to the design of controlled anonymous applications [3][4]. Such applications only provide conditional anonymity to the user. When abusive behavior of the user is detected (e.g. sending mail with improper content), control measures must be applied. Control measures can be split

in two categories, namely (1) revoking the permissions to perform an action and (2) accountability (revealing the user's identity). Within that scope, the methodology allows to express anonymity and control requirements at a very high level. It then generates a conceptual model from which anonymity and performance properties can be derived. Finally, the conceptual model can be mapped to instantiations of privacy enhancing technologies (PET). The methodology also provides alternative design decisions that (partially) avoid conflicts between requirements and proposes reasonable conflict resolution strategies.

Contribution. This paper evaluates the methodology proposed in [3] for designing controlled anonymous applications. More specifically, the methodology is applied to a class of mobile applications with strong security and privacy requirements. From this study, we detect constraints in the existing methodology in the context of mobile applications, explore the solution space and define directions for research. Furthermore, a first iteration of the realization of the existing design methodology is made.

The rest of this paper is structured as follows. Section 2 summarizes the major formalisms that are used in the current version of the methodology. Next, some mobile access control scenarios are presented. Those scenarios are used for a critical assessment of the methodology. The constraints of the methodology are presented in Section 4. Moreover, the solution space is explored. Section 5 presents a first iteration of the realization of the existing design methodology. Finally, general conclusions and points for future research are given.

2 Multi-paradigm modelling

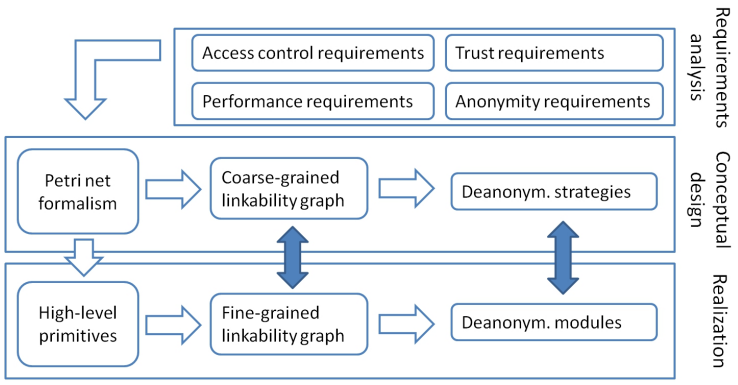

Fig. 1. Overview of the existing multi-paradigm approach for controlled anonymous applications.

This section outlines the major benefits of multi-paradigm modelling [7] and shows how it is used in the context of designing controlled anonymous applications.

The behavior of a system can be described at different levels of abstraction by means of different models. The model at a given abstraction level depends on the background and goals of the modeller and the system that is modelled. In the current methodology [3], as depicted in figure 1, three abstraction layers are defined.

The first abstraction layer defines formalisms to capture non-functional requirements like anonymity requirements, access control and performance requirements. For instance, anonymity requirements define the set of personal information a user is willing to release during certain transactions. Similarly, performance requirements impose thresholds on acceptable response times.

The second abstraction layer provides formalisms at a conceptual level. A Petri net formalism is used to represent the system at this level. Each transition defines a particular action in the system and each node defines a set of rights that are required to perform that action. Also, a coarse-grained linkability graph is used at this level to detect links between actions (and rights) given certain design decisions. Note that only application level links are captured in the graph. Feasible deanonymization strategies can be derived from the linkability graph.

At the lowest level, formalisms are defined that easily lead to the instantiations of PET software components. Moreover, a fine-grained linkability graph formalism shows which entities in the system can link certain attributes to an individual.

To increase the reliability of the design process, rules are defined to transform a model within a formalism at a high abstraction layer to a model within a lower abstraction layer. The transformation rules are depicted with a white arrow in figure 1. Moreover, design patterns (e.g. patterns for control measures) are applied automatically for often recurring problems (e.g. a set of requirements is recognized and automatically a design pattern is instantiated from a pattern database). Both techniques semi-automate the design process. This increases the reliability of the design and eases the tasks of the designer. Finally, the designer can prove that no additional links appear if the Petri net model is transformed to high-level primitives (see dark arrows in figure 1).

3 Mobile access control scenarios

Three mobile access control scenarios are presented here. The requirements of all scenarios were captured and the design methodology was applied to each of these scenarios. From this analysis, we defined constraints of the current methodology when reusing it for the design of mobile applications. We also outline how to tackle the shortcomings (i.e. we define how to extend the methodology to capture this class of applications):

- The user wants to open an online NFC lock with his mobile. Therefore, he has to prove certain properties. Only if these properties can be proved, he can enter the building.
- The user wants to open an offline NFC lock with his mobile. Therefore, he has to prove certain properties like in the first scenario. However, the lock cannot connect itself to a back end system that takes authorization decisions.
- The user wants to display contents of a remote web server on his workstation. The credentials to retrieve the contents are stored on the user's mobile.

The current methodology focuses on traditional client-server systems. Typically, those systems are less constrained than mobile applications. The selected scenarios deal with different aspects that are related to mobile applications. The first two scenarios cover the limited capabilities of the platform (e.g., memory, battery capacity, processing power). Finally, mobile devices are not always online. This has an impact on the control measures that can be taken. The third scenario focuses on secure data storage (mobile devices gets easily lost or stolen).

4 Assessment of the methodology in the context of mobile access control systems

This section lists constraints in the current instantiation of the methodology for controlled anonymous applications. Throughout the discussion, we also show that alternative non-functional requirements may have a serious impact on design decisions.

4.1 Constraints in the context of mobile applications

The current methodology mainly focuses on traditional client-server systems. This means that a user contacts a powerful service provider using credentials that can be stored on a workstation. Such a workstation typically has a lot of processing power and no real memory constraints. Related to performance, the current methodology allows to evaluate if the response times of authentication protocols that are eventually selected are acceptable from the user's perspective. However, in the context of transactions with mobile devices, new performance constraints arise. For instance, energy consumption and memory usage might be important (especially if protocols – or parts of them – will be run on tamperproof modules within the mobile). The designer therefore wants to evaluate these types of performance constraints very early in the design stage. Similarly, in the NFC door lock scenarios that were introduced above, the service provider does not have unlimited computation capabilities (especially in the case of an offline door lock). The technologies that will be selected when building a particular system will depend on platform constraints of both the mobile device and the locks.

Second, the methodology does not explicitly focus on techniques and patterns for secure storage of data. Although this is less relevant in the context of traditional client-server systems, secure storage of sensitive data and mechanisms to control the activation of credentials are becoming even more important

in the context of mobile devices. As smartphones are more easily lost or stolen, credentials and sensitive data must be stored securely. Multiple alternatives are possible. For instance, credentials can be stored on tamperproof modules protected with various mechanisms. The specific mechanism that is selected will depend on both performance, usability and security requirements. For instance, protecting them with a PIN code might be sufficient in some cases, while biometric technologies are feasible candidates when a higher security level is required.

Third, the current methodology assumes that enough resources are available to set up anonymous connections between clients and other entities (i.e. service providers, credential issuers...). The linkability graphs show which information is revealed towards each of these entities. By querying such a graph, the designer can easily derive what information can be collected by a set of collaborating entities in the system. Moreover, the methodology only returns links at the application level. However, within the mobile context, a bunch of information (such as MAC addresses and IP addresses) is revealed to different entities at the communication layer (also depending on the specific communication technology that is used). For instance, using a 3G channel, location information is revealed towards the telecom provider. Similarly, the administrator of a wireless access point knows which devices were connected at a certain time. The same is true for NFC communication. When designing privacy friendly mobile applications, it is especially important to reason about links at communication level.

Finally, the current methodology includes a set of control measures (i.e. strategies) to prevent or discourage misbehavior. For instance, the user's identity may be revealed or a right might be disabled if suspected behavior is detected. Those control measures typically assume a set of online entities. However, new types of control measures are required in mobile applications because of two reasons. First, we cannot assume that mobile devices are always online. Second, devices are easily lost or stolen. Therefore, a criminal should not have the power to impersonate the user.

4.2 Extending the existing design methodology

To tackle the new challenges that are relevant in the context of mobile applications, a set of new formalisms are presented. They are depicted in gray in figure 2. Moreover, extensions are made to existing formalisms.

A set of new requirement formalisms are added, namely activation requirements and communication requirements. The former will lead to mechanisms that will be added during the conceptual design phase for getting access to credentials (e.g., credentials become accessible after a correct PIN code is entered by the user). The latter imposes connectivity constraints for certain actions. Moreover, as depicted in figure 2, the set of performance requirements are extended. For each action, the designer can impose thresholds on for instance energy consumption and memory usage. For instance, thresholds are used to notify the designer when requirements are not satisfied. The designer can redesign some parts of the system until requirements are met.

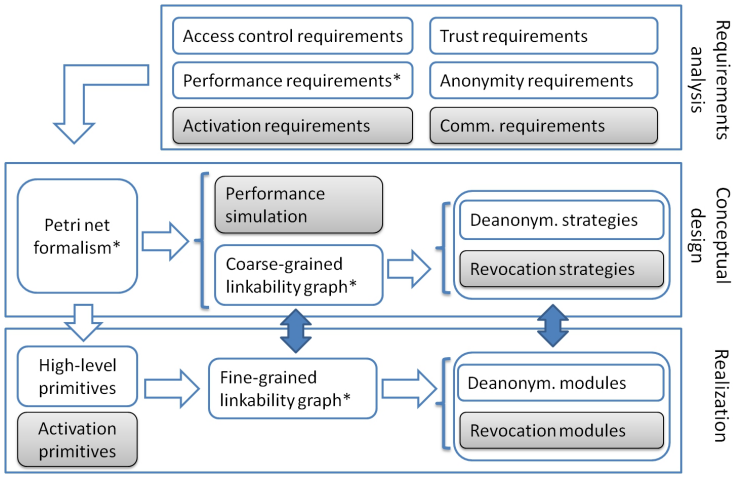

Fig. 2. Overview of extensions to the methodology for designing mobile applications (* extended formalism).

At the conceptual design phase the Petri net formalism is extended to enable the modelling of the new types of requirements. Moreover, a simulation mechanism is introduced to evaluate the performance properties at a very early stage.

Finally, revocation strategies are explicitly modelled. This is essential in the context of mobile applications because credentials possibly become invalid (e.g., after a mobile device that stores credentials is lost or stolen). Moreover, the semantics of the coarse-grained linkability graphs are extended. Application as well as communication level links are represented. Hence, the designer can evaluate which information can be revealed to entities at both levels.

At the implementation level, activation and revocation primitives are instantiated. The fine-grained linkability graph gives a more detailed insight in the links at application and communication level.

5 Realization

To demonstrate the applicability of this methodology, the major formalisms are implemented in AToM^3 [1], which stands for "A Tool for Multi-formalism and Meta-Modelling". The first iteration includes the implementation of the Petri net formalism and the coarse-grained linkability graph of the existing methodology. Moreover, it forms the basis for future work on the methodology. Hence, this section describes the implementation of the two formalisms.

The following e-shop application is used to illustrate the applicability of the methodology. The shop uses a prepaid payment system. Each user owns an e-shop wallet that contains an amount of money. Only when the user owns a

sufficient amount of money in his wallet, he is able to buy the selected product. To acquire money in his wallet, he can buy and upload it to his wallet. Users need to subscribe to the e-shop before they can join. After subscription, the user acquires an empty wallet along with a buy permission that has a limited validity period (e.g., to avoid accounts that are not used, a limited validity period imposes a renewal of the subscription). A valid buy permission is necessary to buy items. Hence, each buy or upload action requires a verification of the buy permission. The application allows to renew the user's buy permission for a next period. Users that do not use the e-shop anymore can unsubscribe. After unsubscription, the buy permission and the wallet are revoked. The remaining money in the user's wallet is paid back to the user. Note that only a small part of the e-shop system is modelled. Furthermore, abstraction is made of the actions (e.g., the models do not model the payment when upgrading the wallet).

The rest of this section is structured as follows. First AToM^3 is discussed. Next, the implementation of the two formalisms is described followed with a discussion about the transformation from a model in the Petri net formalism to a coarse-grained linkability graph model. This section ends with a discussion about mobile access to the e-shop application.

5.1 Modelling with AToM^3

The AToM^3 [1] modelling tool is a Python based tool that supports multi-paradigm modelling. It allows to build graphical models in different formalisms. Each formalism (e.g., Petri net formalism) is modelled by a meta-model that describes all possible structures in that formalism. The *Entity Relationship* (ER) diagram is the default meta-model formalism that is used in AToM^3 . It expresses the entities and the relations between them that are acceptable in the formalism that is modelled. A graphical layer is also provided in the ER formalism by AToM^3 . It allows to define a graphical representation for every formalism entity (e.g., a circle to represent a state, a directed arc represents a state transition). Constraints in the formalism can be expressed too (in a textual form using *Object Constraint Language*). Finally, a modelling tool for the specified formalism is generated from the meta-model by AToM^3 . The generated tool automatically verifies if the model satisfies the formalism.

AToM^3 has several advantages. First, it is flexible because one tool is capable to process models created in different formalisms. Second, it allows to add functionality (e.g., querying a model, starting a simulation from a model) in the generated modelling tools. Finally, AToM^3 provides an interface to model transformations.

5.2 Petri net model

A Petri net is a graph that consists of a set of places (P), transitions (T) and directed arcs. Arcs connect places to transitions, or vice versa. The set of input arcs (IN) connect a place to a transition ($IN \subseteq P \times T$) while the set of output arcs (OUT) connect a transition to a place ($OUT \subseteq T \times P$). Hence, a tuple

(P, T, IN, OUT) represents a Petri net. Graphically, circles represent places while rectangles represent transitions. Places are containers for tokens of the same type (e.g., a place is marked with a token) and are consumed by a transition when it fires. A transition only fires when all its input tokens hold at least one token. When firing, the transition places one token to each output place. Petri nets are suited to model system state behavior. Hereby, places correspond with states while transitions represent activities in the system.

For the design methodology, the classic Petri net formalism lacks expressive power. This is illustrated by the e-shop example. Five activities or actions are defined in the e-shop example, namely (1) *subscribe*, (2) *unsubscribe*, (3) *renew*, (4) *buy* and (5) *upgrade wallet*. Moreover, two types of rights are defined in the e-shop, namely (1) a *buy right* and (2) a *wallet*. The former represents the right to buy products from the shop. The latter is used for the amount of money that is available for buying products. In the classic Petri net formalism, the actions are modelled as transitions. The two types of rights are represented by two types of tokens, which are kept in two places, *buy perm* and *wallet*, the former contains one *buy* token, the latter contains one *wallet* token. Rights can be associated with attributes. For instance in the e-shop, the validity period of the *buy right* and the amount of money in the *wallet* (other examples of attributes are pseudonym, address, age). Unlike actions and rights, those attributes can not be expressed in the classic Petri net formalism. Furthermore, conditional behavioral decisions like the validity verification ($start \leq now$ and $end \geq now$) of the *buy* action can not be modelled. Finally, it is not possible to model non-functional properties such as the action response time (time it takes to react on a input) or storage space needed for rights.

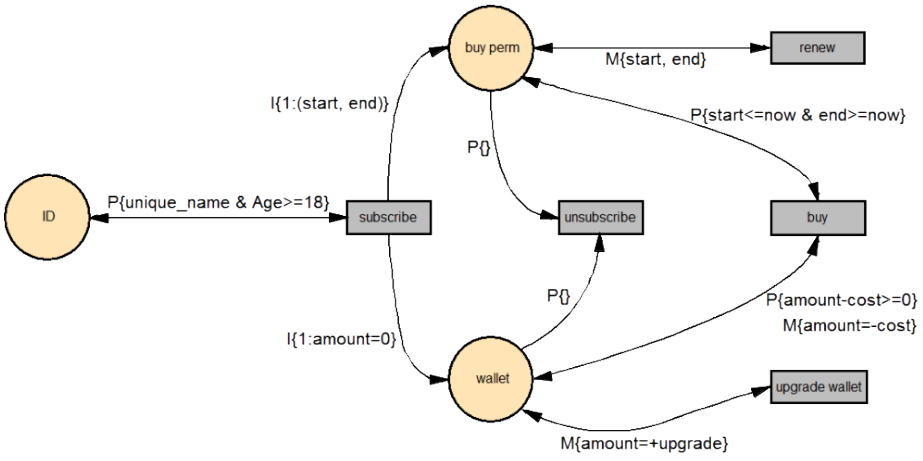

Fig. 3. Example of a Petri net created with AToM^3 .

Hence, semantical extensions on the classic Petri net formalism are needed for the design methodology. Figure 3 depicts the model of the e-shop example in the extended Petri net formalism (this model is created using the Petri net tool generated in AToM^3). First, tokens can hold multiple attributes. For instance in case of the e-shop, a *buy* token contains the attributes *start* and *end*. They define the *buy* right's validity period. The *wallet* token contains an attribute *amount* that represents the amount of money in the user's wallet.

Second, different types of arcs are defined, namely (1) *initiators*, (2) *mutators* and (3) *proofs*. In the e-shop two initiators are present. They are represented by output arcs. Both are connected to the *subscribe* action and are indicated by the I-specifier. The first initiates one *buy* token in the *buy perm* place with a specified start and end date ($I\{1 : (start, end)\}$). The second initiator creates one *wallet* token in the *wallet* place with an initial amount of zero ($I\{1 : amount = 0\}$).

A mutator can update the attribute values of existing tokens. They are depicted with a bidirectional arc with a label that contains the M-specifier. Examples in the e-shop are $M\{amount+ = upgrade\}$ and $M\{amount- = cost\}$. The former adds the upgrade amount of money to the amount of money in his wallet. The latter decreases the amount of money in the user's wallet with the price of the item he bought.

Proofs are used to express conditional behavioral decisions before an action executes. An action only gets the token when the proof's expression (label) validates to `TRUE`. Performing an action does not only imply that a token is spent (one time proof). Tokens can also be used to prove a property (lifetime proof). The former is depicted as an input arc, the latter is displayed as a bidirectional arc. The bidirectional arc is a short-hand notation for the combination of an input arc and an output arc that initializes a token with the same attributes as the input token. Proofs are indicated by the P-specifier. An example of a proof in the e-shop is $P\{start \leq now \ \& \ end \geq now\}$. This proof verifies the validity of the buy permission when the user buys a product. Another example of a proof is $P\{amount - cost \geq 0\}$. This proof verifies if a user owns a sufficient amount of money in his wallet.

Third, the extended Petri net formalism is also able to express some non-functional properties such as (1) *action response time*, (2) *storage needed by an action* (temporal storage of right during execution) and (3) *storage needed by a right*. Non-functional properties are not displayed. They are used to evaluate if a system satisfies its non-functional requirements (e.g., maximum response time for an action is 10ms). Finally, also other properties can be derived from the Petri net formalism. Examples are *reachability* (can a certain state be reached) and *liveness* (how many times can an action be executed) properties.

Petri net models are modelled with a tool that is generated automatically by AToM^3 . This tool is used for the Petri net model of the e-shop that is depicted in figure 3. To generate such a tool, AToM^3 requires a meta-model that describes the Petri net formalism.

Figure 4 depicts the ER diagram. This is the meta-model that describes the extended Petri net formalism. Two classes `PN_right_container` and `PN_action`

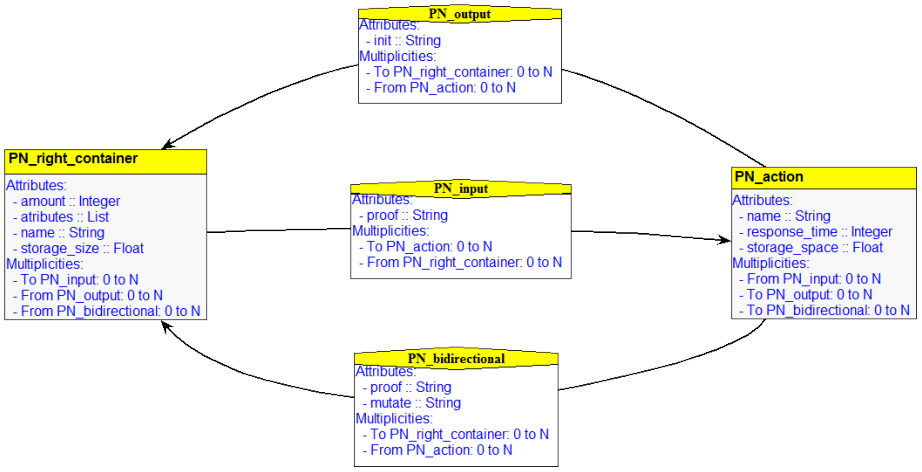

Fig. 4. ER diagram of the Petri net formalism created with AToM^3 .

represent a *place* and an *action* respectively. Places and actions are connected with initiators, mutators, proofs or a combination of them. Each of them can have its specific behavior and label syntax. Hence, multiple associations are specified between the two classes, namely (1) `PN_input`, (2) `PN_output` and (3) `PN_bidirectional`. These represent input, output and bidirectional arcs respectively. Furthermore, each component in the Petri net formalism has one or more properties. To model properties, attributes are added to the classes and associations in the meta-model. Action properties are (1) *name*, (2) *response time* and (3) *storage space* needed during execution. Note that at this stage in the design only an estimation of the storage space, based on the used right's attributes, can be made. For instance, this storage is used for rights that the action needs to process. The properties of a place (container of rights of the same type) are (1) *name* (2) *amount* of rights (tokens) that the place holds, (3) *right attributes* and (4) *storage size* that is needed to store one right (at this stage in the design the storage can only be estimated). Different types of rights can have different attributes (e.g., pseudonym, age, validity period). Hence, to model *right attributes* in the meta-model, a list of attributes is used (`attributes` in the `PN_right_container` class). Properties for the arcs are (1) *name* and (2) *behavior* (expression represented by the label of the arc). Depending on the type of the arc, different attributes are used in the meta-model, namely `proof`, `init` and `mutate`. Often, an input and an output arc are combined into one bidirectional arc. Hence, a bidirectional arc contains a *proof* and a *mutate* attribute.

5.3 Coarse-grained linkability graph

At the conceptual level in the existing design methodology, linkabilities between actions, rights and unique environmental attributes (e.g., identity, pseudonym, e-mail address) are presented in a coarse-grained linkability graph (V, L). A coarse-grained linkability graph contains different units of linkability that are represented by vertices (V), namely (1) *actions*, (2) *rights* and (3) *unique environmental attributes*. A subject or set of subjects is assigned to each unit. Subjects are entities to which the unit is revealed. For instance in the e-shop example, the *subscribe* action is revealed to the server of the service provider that executes this.

Furthermore, undirected edges ($L \subseteq V \times V$) between vertices represent links between those vertices. Two types of links exist, namely *unconditional links* and *conditional links*. The latter is a link that is revealed between an action and a right after for instance, detecting misbehavior of a user. Furthermore, different properties are assigned to links. Firstly, the *multiplicity*, denoted as a ratio, specifies how many instances of the unit can be linked directly. For instance, the multiplicity ratio of the link between the *subscribe* action and the *buy permission* is `1:1`. This means that only one instance of each unit can be linked to each other. Secondly, a *link condition* specifies the condition that must be fulfilled to reveal the link (e.g., the link is revealed when misbehavior of a user is detected). In case of an unconditional link, the link condition is always `TRUE`. Finally, the *set of subjects* that are required to reveal the link between two units of linkability (e.g., a trusted third party is needed to reveal the link).

Graphically, actions, rights and unique environmental attributes are represented by rectangles, ellipses and circles respectively. A Link is depicted as an undirected edge together with its link condition (depicted in an ellipse) and its multiplicity. Figure 5 depicts the coarse-grained linkability graph of the e-shop that contains three unconditional links, each with a multiplicity ratio of `1:1`.

Figure 6 depicts the meta-model of the coarse-grained linkability graph. Three classes `CG_LNK_action`, `CG_LNK_right` and `CG_LNK_unique_attribute` represent *actions*, *rights* and *unique environmental attributes* respectively. Only direct links are possible between an action and a right or an action a unique environmental attribute. Hence, two associations `CG_LNK_link_attr` and `CG_LNK_link_right` model the links between the units of the coarse-grained linkability formalism (in case only one association was used, then direct links between actions were also possible). The first one represents a link between an action and a unique environmental attribute. The second one represents a link between an action and a right. This meta-model is used by the AToM^3 tool to generate automatically the coarse-grained linkability graph tool (which was used to create the linkability model of the e-shop).

Coarse grained-linkability graphs are useful to analyze linkabilities in the system. Moreover, strategies for control measures can be applied (e.g., creating conditional links to reveal a link between an action and a right in case misbehavior of a user is detected). Also, the impact of revoking a permission can be estimated using this graph.

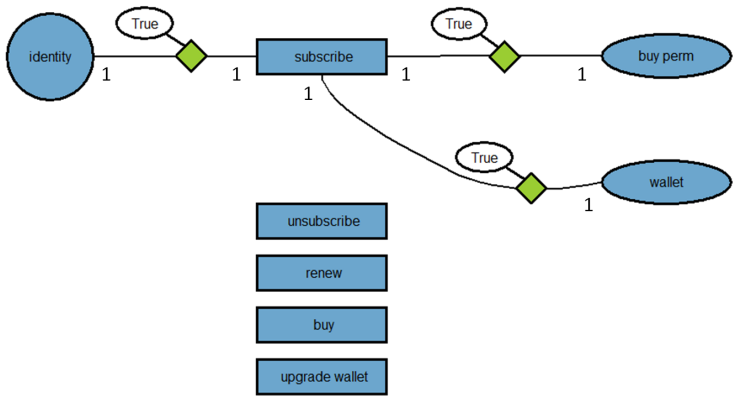

Fig. 5. Example of a coarse-grained linkability graph created with AToM^3 .

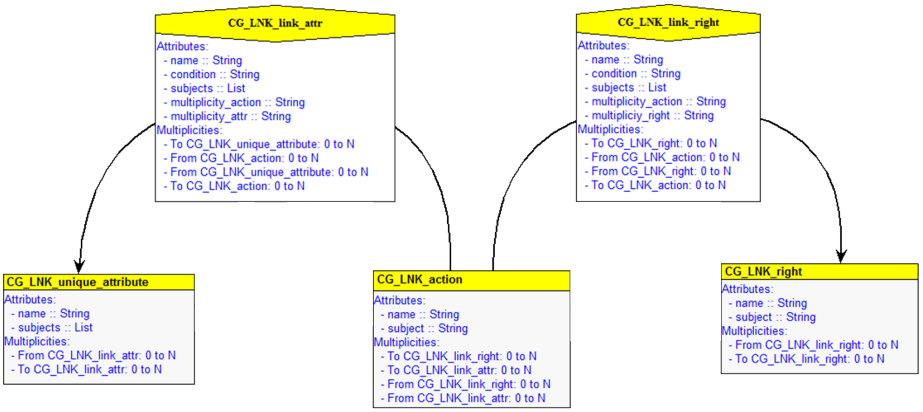

Fig. 6. ER diagram of the coarse-grained linkability graph created with AToM^3 .

5.4 Towards a transformation from a Petri net model to a coarse-grained linkability graph

Future work on the realization of the design methodology will elaborate on the automatic transformation between a Petri net model and a coarse-grained linkability graph. The transformation is based on graph rewriting. Therefore, multiple rewriting rules must be defined. Each rule contains a left hand side pattern (LHS) and a right hand side pattern (RHS). The LHS rule represents a pattern (sub model) of the source model in the source formalism. The RHS rule specifies how the source sub model must be rewritten in the destination formalism. Those rewriting rules are part of a graph grammar and each rule has a priority. The highest priority rule is executed first.

Possibly, rewriting rules are divided over several graph grammars, depending on their purpose (e.g., some rules are part of a graph optimization such as reducing non-used vertices). After sequentially executing the graph grammars on the Petri net, a corresponding coarse-grained linkability graph is automatically generated. However, a transformation is only correct when the set of rewriting rules of the transformation are correct and complete. Hence, proofs for *correctness* and *completeness* are crucial.

5.5 Towards mobile access to the e-shop

The buy permission and wallet can be stored on a mobile device. This improves the usability of the e-shop application because the user only needs his mobile phone to access the shop. Mobile devices are vulnerable for theft. Hence, a revocation strategy is needed to prevent a malicious user to enter the e-shop. The coarse-grained linkability graph allows to determine a suitable revocation strategy. For instance, the buy permission can be revoked. This prevents a malicious user to buy products in the shop. The wallet is still valid but it is not possible to perform harmful actions (a malicious user can only upgrade the user's wallet).

Because mobiles get easily lost or stolen, activation requirements are required. Credentials must be stored on tamperproof modules (e.g., a secure element) and can be protected with various mechanisms. In the e-shop, the buy permission and the wallet are stored on a secure element. The activation mechanism for the buy permission must have a high security level. The activation mechanism of the wallet only requires a lower security level (e.g., PIN code) because a user already needs a buy permission to buy products.

6 Conclusions and future work

This work is a first iteration to adapt the multi-paradigm design methodology to increase its usability when designing privacy-friendly mobile applications. New types of requirements are collected during the analysis phase. More precisely, communication and activation requirements and new types of performance requirements are formalized. During the conceptual design, those requirements are

captured. The performance analysis estimates energy consumption and memory required to execute the protocols. They can be tuned for different types of mobiles (depending on the capabilities of the device). At the application level, the conceptual models are mapped to high level primitives. Future work will elaborate on the realization of the design methodology. Furthermore, future work will elaborate also on the extensions of the methodology and offer new types of requirements and patterns that can be instantiated automatically if certain requirements are present. Moreover, the performance evaluation can already constrain the types of mobile devices that are feasible to run the application.

Acknowledgement This research is partially funded by the Interuniversity Attraction Poles Programme Belgian State, Belgian Science Policy, and by the IWT-SBO project (DiCoMas) "Distributed Collaboration using Multi-Agent System Architectures".

References

1. Juan Lara and Hans Vangheluwe. AToM^3 : A tool for multi-formalism and metamodelling. In Ralf-Detlef Kutsche and Herbert Weber, editors, *Fundamental Approaches to Software Engineering*, volume 2306 of *Lecture Notes in Computer Science*, pages 174–188. Springer Berlin / Heidelberg, 2002.
2. Raphael Mannadiar and Hans Vangheluwe. Modular synthesis of mobile device applications from domain-specific models. In *Proceedings of the 7th International Workshop on Model-Based Methodologies for Pervasive and Embedded Software*, MOMPES '10, pages 21–28, Antwerpen, Belgium, 2010. ACM.
3. Vincent Naessens and Bart De Decker. A methodology for designing controlled anonymous applications. In Simone Fischer-Hübner, Kai Rannenberg, Louise Yngstrm, and Stefan Lindskog, editors, *Security and Privacy in Dynamic Environments*, volume 201 of *IFIP International Federation for Information Processing*, pages 111–122. Springer Boston, 2006.
4. Vincent Naessens, Liesje Demuynck, and Bart De Decker. A fair anonymous submission and review system. In *Communications and Multimedia Security*, pages 43–53, 2006.
5. Akshay Rajhans, Shang-Wen Cheng, Bradley R. Schmerl, David Garlan, Bruce H. Krogh, Clarence Agbi, and Ajinkya Bhave. An architectural approach to the design and analysis of cyber-physical systems. In *Proceedings of the 3rd International Workshop on Multi-Paradigm Modeling*, Denver, Colorado, USA, 2009.
6. Andreas Schuster and Jonathan Sprinkle. Synthesizing executable simulations from structural models of component-based systems. In *Proceedings of the 3rd International Workshop on Multi-Paradigm Modeling*, Denver, Colorado, USA, 2009.
7. Hans Vangheluwe, Juan de Lara, and Pieter J. Mosterman. An introduction to multi-paradigm modelling and simulation. In *In Proceedings of the AIS'2002 Conference (AI, Simulation and Planning in High Autonomy Systems), ed. F. Barros and N. Giambiasi*, pages 9–20, Lisboa, Portugal, 2002.