Support for Cross-domain Composition of Embedded Systems Using MARTE Models

Maria Vasilevskaya, Simin Nadjm-Tehrani
Dept. of Computer and Information Science
Linköping University, Sweden
Email: [maria.vasilevskaya, simin.nadjm-tehrani]@liu.se

***Abstract*—Embedded systems have evolved from tailormade systems developed by specialist engineers to artefacts built from complex software/hardware components with many extra-functional concerns. Ubiquity of embedded devices demands other facets such as security and safety to be brought to the forefront. At the same time, cost efficiency dictates building systems from reusable building blocks. However, integration of extra-functional building blocks comes with a certain performance resource overhead that must be taken into consideration while designing resource-constraint embedded systems. This paper builds on the premise that functional models can be extended with platform modelling to help the application engineers to select the right extra-functional building blocks accounting for performance implications of their integration. We define a UML profile relating it to relevant parts of the MARTE profile in order to capture the performance analysis results for a reusable building block, and a generic notion of model-based compatibility analysis for platform models. Additionally, our approach rests on creation of ontologies to store MARTE description of hardware components, and is supported by a MagicDraw plugin developed for capturing the analysis results and performing the compatibility analysis.**

I. Introduction

Embedded computing has a ubiquitous presence in almost every facet of our life. As a result, the focus of embedded system engineers is increasingly moving from realising only application functions to providing Extra-Functional Properties (EFPs) for them. Safety and security are among these EFPs that highly impact the design of embedded systems, sometimes originated from certification or legal requirements. Often EFPs have a cross-cutting nature that impact systems in many dimensions. Functions that enforce these properties can be, in turn, encapsulated into reusable units built of software and hardware components and used across many applications. In our work, a reusable unit that provides functions to satisfy EFPs is called *Reusable Building Block* (RBB). For example, such RBBs can encapsulate security functions, e.g. encryption and digital signature, that provide such EFPs as confidentiality and integrity of data respectively. Moreover, RBB encapsulation of functions needed to satisfy EFPs allows domain specialists to study RBBs before their integration into different embedded devices that are complex hardware-software systems. Note that not all EFPs can be implemented in the form of RBBs, but such EFPs are outside of the scope of this work.

Since adding a new feature to a system always comes with resource claims, embedded systems, characterised by their scarce resources, need a special process for integrating RBBs, encompassing expertise from several domains. These range over application domains, e.g. set-top boxes, smart grid infrastructures, health monitoring systems, as well as other expertise domains, e.g. security, safety, and real-time computing. While domain-specific modelling has been proposed for characterising the functional requirements [1], formalisms for extra-functional areas are also emerging. SecureUML [2], UMLSec [3], and AVATAR [4] have been proposed to incorporate safety and security concerns into systems at the design phase. Modeling and Analysis of Real-Time and Embedded systems (MARTE) [5] is a standard UML profile put forward for capturing timing and performance related aspects. This paper suggests that earlier knowledge about results of previously conducted (by extra-functional domain experts) performance analysis of RBBs providing to data about resource footprint and quality of EFPs, increases the efficiency of embedded system design process. Our work contributes to selection of a suitable set of RBBs, by enabling the sensitivity and trade-off analysis at early phases.

Another useful and significant input to the design process is knowledge about platform-specific constraints of RBBs. These constraints originate from the fact that RBBs in embedded systems are often optimised to exploit a particular feature of hardware components on which they are implemented. Thus, they provide an acceptable quality with respect to EFPs while consuming a small amount of limited resources. We argue that these constraints need to be documented and accounted for in order to support integration of RBBs into embedded systems. For example, Preissig [6] reports the results of performance analysis of a Data Encryption Standard (DES) implementation optimised for the memory architecture of the used chip. Similarly, other implementations of DES rely on the presence of a certain instruction set to accelerate permutation operations [7]. However, techniques for a systematic categorisation of such platform-related knowledge are currently missing. Thus, to leverage the results of the RBBs performance analysis along with constraints on platform components, both extra-functional domain experts and embedded system engineers need a consistent framework to capture (and store) these outcomes and to reuse them for later design decisions respectively.

We exploit the MARTE [5] profile and ontology technologies to support (a) extra-functional domain experts in

capturing the results of performance analysis for RBBs and (b) embedded system engineers in selecting the right set of RBBs based on automated analysis of the resource constraints. For this purpose, we use model transformation techniques to bridge the MARTE profile and specific ontologies. These contributions are detailed in this paper as follows:

- Definition of a UML profile to capture performance analysis results for RBBs and creation of an ontology to support storage and querying of this data.
- Definition of the notion of model-based compatibility analysis to match platform-specific constraints of RBBs and the platform resources adopted by an embedded system engineer for development.
- Development of a MagicDraw plugin to support the capturing and compatibility analysis steps above and a scalability study for the ontological tool support.

The approach is validated using a case study that is also used as a running example and presented in Sect. II. Sect. III gives the background. Thereafter, Sect. IV explains the proposed UML profile to record results of the RBB performance analysis. We present our method for the model-based compatibility analysis in Sect. V. Sect. VI shows results of scalability and performance estimations for our approach. The paper ends with a summary of related works followed by conclusions.

II. Smart Metering Application

Fig. 1 depicts a smart grid metering infrastructure developed by the company MixedMode within the European SecFutur project [8]. It is built of a set of metering devices, database servers, client applications, and communication infrastructure. The main goal of this system is to measure energy consumption at households and associate it with client data for billing purposes.

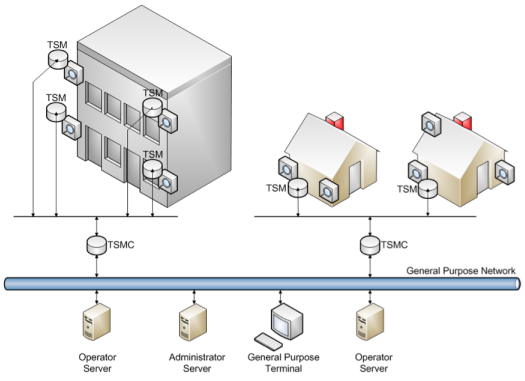

Fig. 1. Smart metering application

The actual measurement is done by Trusted Sensor Modules (TSMs) consisting of a computing platform and physical sensors. The acquired data is transferred via a local bus from each TSM to a Trusted Sensor Module Collector (TSMC) and then eventually sent to an operator server via a general-purpose network. TSMC and TSM are functional modules that are implemented on the same physical platform. In our previous works [9], [10], we presented a technique to systematically extract assets that need protection using functional and execution platform models, illustrated within this networked scenario. This paper takes the process further by selection of suitable RBBs for encryption and decryption functions, as examples of means to satisfy extra-functional properties. We will use the case study in the forthcoming sections to illustrate new concepts when they are introduced in the paper.

III. Background

In Sect. III-A, we make a brief introduction to ontology technologies used in our work. Next, the MARTE profile used for platform modelling is described in Sect. III-B.

A. Ontology technologies

An ontology represents knowledge in a particular domain as a set of concepts and their relations [11]. This knowledge is formalised as a logic-based system and described by a knowledge representation language. In particular, we use the Web Ontology Language (OWL2) [12] which is a commonly used language for creation of large ontologies.

OWL represents an ontology as a set of axioms. These axioms describe *classes*, their *relations*, and *individuals*. An OWL *class* declares the concept of a domain and can be refined by sub-classes. OWL *individuals* are instances of the OWL classes. OWL supports two types of relations. An OWL *object property* defines a relation between two individuals, where one of them plays the role of a domain, and another one the role of a value range. An OWL *datatype property* serves to introduce relations between an individual (domain) and the XML schema datatypes (range) known as XSD [13]. The OWL language supports the importing feature. It allows relating different OWL ontologies using the *owl:import* statement. To design and manage an ontology, one can use tools such as Protégé [14]. In our work, we exploit the Java OWL APIs [15] since the rest of our tool chain is Java-based.

Ontologies allow querying of the declared knowledge by means of ontology reasoners with the help of the SPARQL querying language [16]. SPARQL 1.1 is a standard query language to execute data queries on top of OWL. It supports yes/no-questions (the *ASK* query), a selection which matches a desired pattern (the *SELECT* query), filtering (the *FILTER* modifier), sorting (the *ORDER* modifier), etc. To execute the SPARQL queries, one can load an ontology into the Protégé tool and use its SPARQL plugin [17]. We use Java APIs provided by a widely accepted Jena [18] framework.

B. MARTE

MARTE [5] is a UML profile that contains a rich set of concepts to support design and analysis of embedded systems. Several experience reports on applying this profile to industrial cases exist, among which the work of Iqbal et al. [19].

The MARTE *foundations* define a set of concepts required to model non-functional properties (NFPs). Namely, the Generic Resource Modeling (GRM) package contains general concepts required for modelling of an execution platform consisting of various resources such as computing and storage resources. These foundations serve as a basis for the MARTE

design and analysis packages. The MARTE design package contains sub-packages to describe the hardware and software resource models: Hardware Resource Modeling (HRM) and Software Resource Modeling (SRM). The MARTE analysis package provides facilities to model the context required to perform model-based analysis of real-time and performance characteristics of embedded systems. In particular, the Generic Quantitative Analysis Modeling (GQAM) package defines a set of general components while its extensions refine it to support schedulability and performance analysis. While these extensions facilitate the application of analysis techniques to embedded systems, there are no extensions for storing alternative analysis results for their later reuse, whereas our profile is designed to provide these facilities.

IV. Capturing performance evaluation results

We define a profile for capturing performance evaluation results called Generic Evaluation Model (GEM). This profile is depicted in Fig. 2 and explained in this section.

The feedback loop is a significant step of any analysis as indicated with the step "give feedback to designers" by Petriu [20]. The GEM profile supports this step enriching it with three additional principles: encapsulation of EFP functions into RBBs; separation of a system designer and (extra-functional) domain expert roles; and reuse of the evaluation results. Thus, GEM provides a means for a domain expert to capture the results of RBB performance evaluation in order to support embedded system engineers in selecting a suitable RBB, and to enable automated analysis of the captured data.

gemEvaluation is a central concept of the GEM profile. It relates four sub-concepts that indicate constituents of any evaluation procedure.

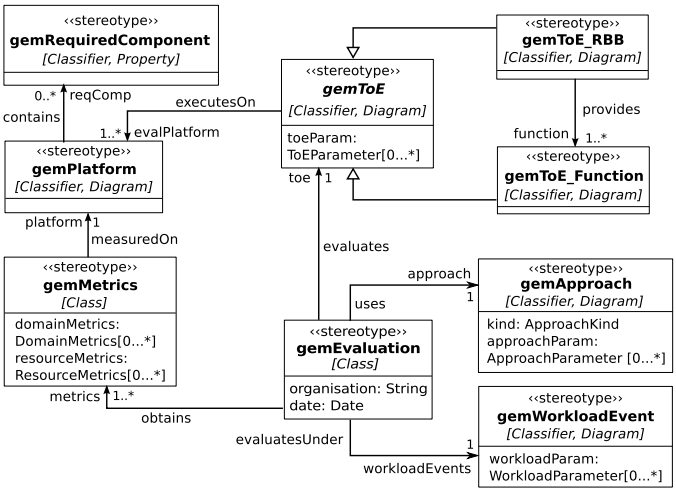

Fig. 2. A general UML profile to capture performance evaluation results

First, *gemToE* (Target of Evaluation) refers to an RBB (*gemToE_RBB*) or its functions (*gemToE_Function*) that are under performance evaluation. A gemToE can be characterised by a set of parameters introduced by the *toeParam* tag (i.e. a property of the stereotype), e.g. a key size and cipher mode for an encryption RBB. Any ToE executes on some evaluation platforms reflected by the *gemPlatform* stereotype. We use a UML class model annotated with stereotypes from the HRM MARTE package to describe this platform. An execution platform can describe resources that take a variety of forms, e.g. hardware, software, or logical resources. In this presentation, we consider only hardware components. However, the general concept is scalable to include other forms of resources for analysis. The *gemRequiredComponent* stereotype is used to introduce those components of an evaluation platform that are significant for a considered ToE to obtain the captured performance results. For example, it can be a particular instruction set exploited by a ToE implementation.

Second, *gemApproach* denotes the description of the used evaluation approach. The *kind* tag defines the type of the evaluation method, e.g. simulation, emulation, or analytical analysis. These types are represented as a UML enumeration, i.e. *ApproachKind*. An approach can be parameterised using the *approachParam* tag.

Third, *gemWorkloadEvent* introduces the workload used during the performance evaluation. Similarly, a workload can be parameterised using the *workloadParam* tag.

Finally, *gemMetrics* defines a set of metrics adopted for the performance evaluation. These metrics can be of two categories. The first category (*resourceMetrics*) describes the resource footprint created by ToE, e.g. execution time. The second category (*domainMetrics*) refers to the obtained indicators that characterise the quality of EFPs, e.g. the security level provided by a ToE. The presence of these two categories reflects the fact that for selection of suitable RBBs both resource footprint and quality of service indices play a significant role.

Note that multiplicity of the *toe*, *approach*, and *workloadEvents* association ends are equal to one. Each time a new approach (likewise workload or ToE) is used a new evaluation should be defined. During one evaluation of a ToE several evaluation platforms can be used. Thus, several metrics sets can be obtained as indicated by the multiplicity of the *metrics* association end. In turn each set of metrics is uniquely associated with one evaluation platform (see *measuredOn*).

gemToE, *gemApproach*, *gemWorkloadEvent* can be used to annotate either a complex or a very simple model of a corresponding constituent. The level of detail does not play a significant role for the GEM profile. Nevertheless, the richer these models are the more informed decisions can be made by an embedded system engineer when selecting RBBs. All parameters and metrics types (ToEParameter, AppproachParameter, WorkloadParameter, DomainMetrics, and ResourceMetrics) are defined as UML data types the attributes of which can be of the MARTE NFP types or other basic types (not shown in Fig. 2).

The MARTE GQAM sub-profile (mentioned in Sect. III) significantly helps the domain experts to design their evaluations. In order to demonstrate how our profile can be used to capture performance evaluation results modelled in GQAM, we identify the correspondence between the stereotypes and tags of GEM and GQAM shown in Table I.

Note that our profile is not restricted to capture results

TABLE I
CORRESPONDENCE BETWEEN OUR GEM AND (MARTE) GQAM

GEM	GQAM
gemToE	The GaScenario stereotype
toeParam	No direct mapping. It can be mapped to some variables declared in the *contextParam* tag (the *GaAnalysisContext* stereotype) where *sourceKind* is defined as required (req).
gemPlatform	The GaResourcePlatform stereotype
gemRequired-Component	Any element from a GaResourcePlatform model can be annotated with this stereotype.
gemApproach	Not represented.
kind	Not represented. It is defined by an analysis formalism that underlies the GQAM model.
approachParam	Not represented.
gemWorkloadEvent	The GaWorkloadEvent stereotype
workloadParam	No direct mapping. It can be mapped to some variables declared in the *contextParam* tag where *sourceKind* is defined as required (req).
gemMetrics	No direct mapping. It can be mapped to any declared variables where *sourceKind* is set to calculated (calc), estimated (est), or measured (msr).
resourceMetrics	No direct mapping. It can be mapped to some variables declared as tags of the GaStep stereotype.
domainMetrics	No direct mapping. These variables are usually defined by domain experts and introduced as a part of the *contextParam* tag.

obtained only when GQAM or its refinements are used. For example, results presented by Preissig [6], that are obtained through a traditional approach, can also be described by GEM (not shown in this paper). However, GQAM can facilitate this task, since the mapping identified in Table I can be automated as a transformation directly feeding relevant data into GEM.

Since GEM is defined as a general UML profile, a domain expert can refine some of its concepts to tailor it to a certain domain. In particular, an expert needs to defined data types for *ToEParameter*, *DomainMetrics*, and *ResourceMetrics*. Fig. 3 shows such a refinement for the security domain as an example where a cipher RBB is considered. *ToEParam_Cipher* says that a cipher mechanism can be characterised by its key size and cipher mode. Quality of service metrics for a cipher (see *DM_Cipher*) are resistance to attacker's capabilities in terms of skill, motivation, and duration of the attack. Finally, a pair of resource metrics defined for this RBB are used memory and data rate (see *RM_Cipher*).

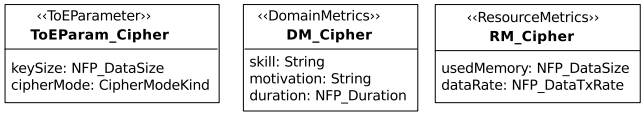

Fig. 3. Refinement of the general evaluation profile for the security domain

A. Storing and querying captured results

To assist an embedded system engineer in reusing performance evaluation data captured by a domain expert, we exploit ontology technologies (see Sect. III-A). In particular, the profile in Fig. 2 is transformed into an ontology called the *core evaluation ontology* (the owl-file is available at [21]). Each refinement of this profile for an extra-functional domain (e.g. the one depicted in Fig. 3) is transformed into a separate ontology called *[domain name] evaluation ontology* that imports the core evaluation ontology enriching it with additional concepts from this domain, i.e. concrete ToE parameters, domain, and resource metrics. Since description of actual performance evaluation results is essentially an instance of the profile, we transform it into axioms on individuals and call it *[domain name] evaluation record*. For example, a class annotated with the ToE stereotype becomes an individual of the ToE class in the core ontology.

A variety of queries can be defined to retrieve different information from these ontologies, for example:

- Retrieve values of relevant performance metrics for a certain RBB.
- Retrieve a set of RBBs of a particular domain that satisfy required values w.r.t. certain performance metrics.
- Retrieve a set of RBBs, for which the platform constraints are compatible with a platform adopted for an embedded system under development.

Queries like the first two are implemented directly as SPARQL queries [16]. However, the task of compatibility analysis requires more sophisticated support. Hence, in the rest of this paper, we present the developed method for cross-domain model-based compatibility analysis, where one domain is an extra-functional domain and the other one is an application domain. To demonstrate introduced concepts and methods, we use the security domain as an example of EFPs.

V. MODEL-BASED COMPATIBILITY ANALYSIS

Model-based compatibility analysis allows automatic accounting for platform constraints while selecting a set of RBBs to be used for a system design. Recall, that an extra-functional domain engineer provides these constraints by annotating elements of a MARTE model, which describes an evaluation platform, with the *gemRequiredComponent* stereotype (see Fig. 2). The core of our method is a set of ontologies and SPARQL queries. They are designed to infer whether RBBs and an adopted platform for the system are compatible with respect to formulated platform constraints for an RBB and platform declarations of the system being configured. We develop the LiU (LInköping University) MagicDraw[1] plugin to support transformation of MARTE models into ontologies and to execute compatibility analysis. Below, Sect. V-A explains the developed ontology hierarchy and Sect. V-B defines the notion of compatibility.

A. Ontologies for compatibility analysis

Fig. 4 depicts the developed set of ontologies. These ontologies are organised in three layers, i.e. expert, vendor, and engineer, and related to each other through the *import*, *use*, and *refer to* relations.

1) ***Expert layer****:* Ontologies of the *Expert layer* are maintained by experts of the embedded and extra-functional domains. It contains three ontologies. The first two, i.e. *NFPType ontology* and *Resource ontology*, are obtained as transformation of MARTE packages dedicated for resource modelling.

[1]MagicDraw is adopted as an integrating environment within the European SecFutur project [8].

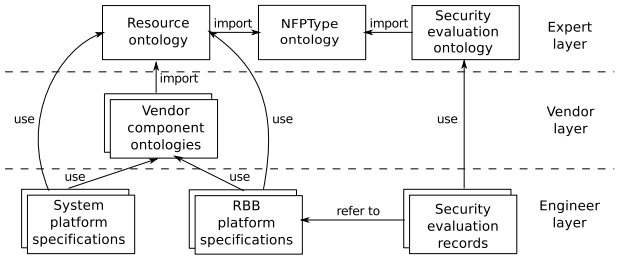

Fig. 4. Ontologies for compatibility analysis

The third ontology is the refinement of the core evaluation ontology for the security domain already explained in Sect. IV.

Techniques to transform UML class models into ontologies are studied and presented, e.g. by Xu et al. [22]. Additionally, Xu et al. prove that the presented UML (class models) to OWL transformation is semantics-preserving. Transformation of MARTE packages into OWL can be approached in a similar manner where a UML class is replaced by a UML stereotype and an attribute is replaced by a tag. Some basic rules that we apply in our work may be summarised as follows:

- Each MARTE stereotype is represented as an OWL class.
- Each tag is represented as an OWL object or data property, where the domain is the stereotype that owns the tag and the range is the defined type of the tag.
- Generalisation relations of the MARTE profile are represented as sub-class relations in the ontology.
- Composition relations are represented as the part-whole object properties, i.e. the "hasPart" relation.

We avoid using the Ontology UML profile [23] that allows designing ontologies as UML models since this requires in-depth understanding of the underlying ontologies from an engineer. Our goal is to exploit advantages of ontology technologies (e.g. querying services), but to allow an engineer to operate only with terms of a considered extra-functional domain.

The *NFPType ontology* of the expert layer contains a set of types and their relations needed to characterise a hardware/software component, e.g. data rate (Mbps, Kbps, etc.) and frequency (Hz, KHz, etc.). This ontology is derived from the MeasurementUnits, MARTE_DataTypes, and Basic_NFP_Types sub-packages of MARTE that enable specification of non-functional properties (see Chapter D.2 in MARTE [5]). The *Resource ontology* contains concepts needed to describe platform components. It is derived from the MARTE HRM package described in Chapter 14.2 [5]. Both ontologies can be accessed on our web page [21].

2) ***Vendor layer****:* This layer in Fig. 4 consists of *vendor component ontologies*, where a vendor is a provider of platform components available for construction of execution platforms for embedded systems (e.g. Texas Instruments). Each ontology encapsulates description of platform components that belong to some vendor. The *Resource ontology* of the expert layer serves as a language to describe these components.

In order to provide the description of available components, a vendor needs to perform the following steps. First, a vendor uses the MagicDraw tool [24] to create models of platform components. These models are UML class models annotated with stereotypes from the HRM package. Second, a vendor launches the LiU MagicDraw plugin to transform the created MARTE models into an ontology. We use the Java OWL APIs [15] and Acceleo [25] tools to realise this transformation.

Fig. 5 depicts an example of the OMPA3530 board provided by the vendor Texas Instruments [26]. This board includes computing (TMS320C64x+ and ARM Cortex-A8) and storage (NAND Flash and LPDDR) elements, communication interfaces (I2C, SDIO, and 10/100Mbps NIC), a daughter card, and the LCD display. The daughter card is connected to the ADE7758 sensor via a Serial Peripheral Interface (SPI) bus. Finally, a 10/100Mbps NIC is used to connect the OMAP3530 board to a communication channel (LAN). Note that TMS320C64x+ and ARM Cortex-A8 have also their own models that we omit for the sake of brevity.

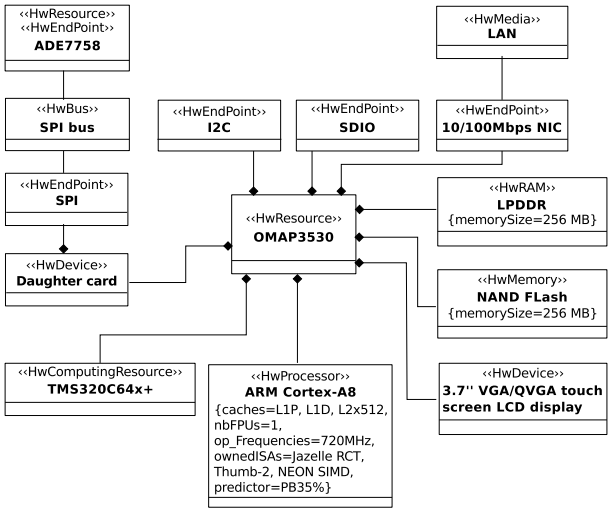

Fig. 5. A model of the OMAP3530 component created at the vendor layer

3) ***Engineer level****:* The bottom layer in Fig. 4 is the *Engineer layer*. At this level, system and security engineers use the ontologies created at the expert and vendor layers to model the adopted platform and components used for evaluation of RBBs. In particular, an engineer uses existing components provided by vendors and stored in vendor ontologies, whereas the resource ontology is used if such a component is not known or not present in vendor ontologies. Thereafter, a security engineer creates a *security evaluation record* (i.e. an instance of the security evaluation ontology) where the *gemPlatform* concept refers to the *RBB platform specification* (see Fig. 4).

In the case study outlined in Sect. II, TSM devices are built on the OMAP3530 board from Fig. 5. Since this component will have been described and stored in the vendor ontology a system engineer only needs to load the corresponding ontology and use this component as a part of the TSM model. The LiU MagicDraw plugin supports this functionality.

As mentioned in Sect. II, the data transmitted between a TSM and TSMC must be protected against confidentiality threats. We consider two candidate RBBs, namely the AES (Advanced Encryption Standard) [27] and DES (Data Encryption Standard) [6] implementations by Texas Instruments [26], which can provide this EFP. AES RBB requires the use of the

C64x+ processor while the DES RBB requires the use of the TMS320C6211 chip. In the next section, we explain how the suggested architecture of ontologies enables selection of RBBs based on platform compatibility analysis.

B. Compatibility analysis

We identify two types of platform compatibility: *logical* and *environmental*. The notion of *logical compatibility* is based on the pairwise logical compatibility of an RBB and system platform components as it is defined below.

Definition: Two components *A* and *B* are *logically compatible* if one of the following holds: (a) *A* is identical with *B*; (b) *A* has *B* as a part; (c) *A* is a part of *B*; (d) *A* can be connected to *B*; (e) *B* can be connected to *A*; (f) a disjunction of (b)-(e).

We employ the ontology querying services to automate a check of the above definition. We use the ASK operator of SPARQL [16] that returns a boolean value indicating whether a path that matches a query pattern exists. For example, the query for case (b) where the relation "hasPart" is examined has the following form $PREFIX\ hrm: [the\ ontology\ IRI]\ ASK\ WHERE\{?A\ hrm:hasPart\ ?B\}$. Queries for cases (c) - (e) have a similar structure replacing "hasPart" with the "isPartOf", "hasConnected", and "isConnectedTo" object properties respectively. To support the check of case (f), we use a special construct defined by the SPARQL 1.1 syntax, i.e. the so called path properties [28]. It allows examining a path of an arbitrary length. Hence, the query for case (f) replaces the "hasPart" property with a path expression, namely, $(hrm:hasPart \mid hrm:isPartOf \mid hrm:connectedTo \mid hrm:hasConnected)*$. In this expression, the symbol "|" denotes the "OR" operator, while the symbol "$*$" means that any number of occurrences is allowed.

In the query above, the $?A$ and $?B$ symbols denote variables. They are replaced by components of a system platform and components of an RBB platform (annotated with the "gemRequiredComponent" stereotype) respectively. In our case study (see Section V-A), these are OMAP3530 and C64x+ for the AES RBB, and OMAP3530 and TMS320C6211 for the DES RBB. Since TMS320C64x+ has a C64x+ processor as its part, the query returns *true*. In contrast, no path is found between TMS320C6211 and TMS320C64x+ for the DES RBB. Thus, we conclude that the particular implementation of the DES algorithm is not logically compatible with the current design of a system platform that is based on the OMAP3530 board, while AES can be selected as a RBB to provide secure communication for the TSM device.

The definition of *environmental compatibility* is built upon the *Env_Condition* type from the HRM package (see Figure 14.72 in MARTE [5]) which defines five condition types: temperature, humidity, vibration, shock, and altitude. Environmental (env.) condition of each type has a value range. An engineer needs to annotate the components with the "HwComponent" stereotype and define the "r_Conditions" tag to assign env. conditions to a component. We use the following terms and functions to define *environmental compatibility*:

- K and U are sets of the env. condition types and measurement units respectively, where $K = \{temperature, humidity, vibration, shock, altitude\}$ and $U = \{^\circ C, \%, m/s^2, g, m\}$.
- A set ENV_COND defined as $I \times U \times K$ that describes each env. condition as a tuple of a value interval (a set I), a unit (from the set U), and a type (from the set K).
- A function defining env. conditions of a component, i.e. $env\_cond : COMP \rightarrow 2^{ENV\_COND}$, where $COMP$ is the set of components.
- Projection functions extracting from an env. condition the corresponding type: $kind : ENV\_COND \rightarrow K$, unit: $unit : ENV\_COND \rightarrow U$, and value interval $range : ENV\_COND \rightarrow I$.

Given these terms and functions we introduce two other functions to define the notion of environmental compatibility.

Definition: Environmental compatibility is a function $env\_comp : COMP \times COMP \rightarrow \{true, false\}$. A component A is *environmentally compatible* with a component B if $env\_comp(A, B)$ is evaluated to $true$ as defined below:

$$env\_comp(A,B) \triangleq true\ if\ \{\langle\langle i_1,u,k\rangle,\langle i_2,u,k\rangle\rangle \mid a \in env\_cond(A), b \in env\_cond(B), k \in kind(a) \cap kind(b), i_1 = range(a), i_2 = range(b), i_1 \cap i_2 \neq \emptyset\} \neq \emptyset\ or\ \{k \mid a \in env\_cond(A), b \in env\_cond(B), k \in kind(a) \cap kind(b)\} = \emptyset$$

The intuition is that env. conditions of a platform component A adopted for an embedded system and a component B required by an RBB are compatible if corresponding interval values of env. conditions of the same type are overlapping. These components are also compatible if there are no common types of conditions that are defined for both components.

In addition, we define another function that specifies the environmental conditions under which a pair of components can not operate (although each could operate individually under respective conditions). We refer to such env. conditions as env. constraints. This function is defined as $env\_constr : COMP \times COMP \rightarrow 2^{ENV\_COND} \times 2^{ENV\_COND}$ as follows:

$$env\_constr(A,B) \triangleq \{\langle\langle i_1,u,k\rangle,\langle i_2,u,k\rangle\rangle \mid a \in env\_cond(A), b \in env\_cond(B), k \in kind(a) \cap kind(b), i_1 = range(a) \setminus range(b), i_2 = range(b) \setminus range(a), u = unit(a)\}$$

The intuition is that while each component might operate in an interval that is a subset of its operational condition, the composition with another component that disallows that subset dictates that the first component is prohibited from operating in that condition. The presence of these constraints for components guides a system engineer to implement a mechanism (e.g. cooling system) that ensures that the corresponding components sustain the allowed env. conditions (e.g. to keep within a given temperature range). The LiU MagicDraw plugin generates these env. constraints automatically from given environment conditions attached to individual components.

In our case study, the temperature conditions for OMAP3530 and TMS320C64x+ (the AES RBB) have the same ranges of $[0; 90]^{\circ}C$. Therefore, the system and RBB are environmentally compatible without any additional constraints.

The above definitions for computing the compatibility relation and their pairwise imposed constraints allow us to reason about env. conditions of assemblies based on constraints for its constituent components.

VI. Scalability and performance

So far, we have used the smart metering network to illustrate the compatibility analysis and knowledge management ideas supported by our methods and tool. This section proceeds to show that our approach is scalable to domains with large data sets. We design experiments to estimate the size of resulting vendor ontologies as well as the execution time for the transformation of MARTE models into an ontology.

We focus on microcontrollers (MCUs) provided by some of the most common vendors (Renesas, Texas Instruments, Fujitsu, Atmel, and Microchip Technology). We estimate potential complexity of corresponding MARTE models and the size of corresponding ontologies in terms of the number of generated axioms. Three classes of embedded systems and MCUs commonly used for their design [29] are considered: small scale (8-bit MCUs), medium scale (16-bit MCUs), and sophisticated systems (32-bit and ARM-based MCUs). Then, we study how many models are currently available on the market for each vendor (see Table II). The data has been extracted from the official Internet resources of the vendors.

TABLE II
Scalability and performance estimations

		8-bit	16-bit	32-bit
1	Renesas	933	2290	1817
2	Texas Instruments	0	406	292
3	Fujitsu	103	207	630
4	Microchip Technology	348	334	79
5	Atmel	238	0	179
6	Total amount of units	1622	3237	2997
7	Av. number of axioms per unit	68	105	133
8	Total number of axioms	110 296	339 885	398 601
9	Av. transformation time (ms)	1455.1	1627.92	2497.92

To estimate the potential number of generated axioms, we select five commonly used MCUs of each class, create their MARTE models, and execute their transformations. This study shows that the simplest 8-bit MCUs creates an average of 68 axioms, while the most sophisticated 32-bit MCUs generate 133 axioms (see Table II, row 7). Row 8 shows the number of produced axioms when all models are added into ontologies. Finally, we compare these numbers with scalability studies of the OWL APIs and Jena technologies [30], [31]. In particular, Horridge and Bechhofer [30] show that OWL APIs can easily handle ontologies that contain 1 651 533 axioms consuming 831 MB. As a result, we conclude that the used technologies (OWL APIs and MARTE) allow handling ontologies for a significant number of vendors in a potential real world deployment. This capacity allows loading multiple vendors' ontologies to execute compatibility analysis. Additionally, some techniques for swapping ontologies in memory can be implemented to handle even bigger datasets.

Next, we execute 50 runs of the transformation and measure the execution time for each run (see Table II, row 9). The hardware used is a system with 2.8 GHz Intel Core i7 and 8 GB of RAM running Mac OS. In our case, the transformation time does not vary substantially for small and medium MCUs, while a one second increase is observed for the sophisticated 32-bit MCUs. This increase can be explained by naturally larger, in comparison with 8-bit and 16-bit MCUs, complexity of 32-bit MCUs in both number of elements and their attributes.

VII. Related Work

We briefly review the earlier work on analysis support for RBB integration and use of ontologies for a system design.

The use of performance analysis at the early design phase is a subject of active research. Woodside et al. [32] apply an approach for performance analysis of RBBs represented as aspects. Bondarev et al. [33] present a toolkit for performance evaluation where RBBs conform to a specific component model. These methods use models as a means to input required data into performance tools. Our work complements these approaches since it enables *reuse* of outcomes of the RBBs performance evaluation conducted by EFPs domain experts.

Ciccozzi et al. [34] propose a meta-model which is used to propagate results of monitoring extra-functional properties at the code level back to a system model in order to improve it. This approach is developed for the CHESS modelling language [35]. In our work, we exploit MARTE models to capture platform-specific constraints of embedded systems and RBBs. The proposed MARTE compatible profile allows capturing more information about outcomes of RBBs performance analysis, e.g. the used workload and the observed resource footprint. This information, structured, captured, and searchable, aids an embedded system engineer to select an appropriate RBB to satisfy required extra-functional properties.

The use of ontologies to support tasks of model-driven engineering is a promising research topic [36]. The potential of ontology technologies applied to system engineering to formalise general modelling is outlined by Tetlow et al. [37]. Walter et al. [38] employ ontologies to improve the practice of domain-specific modelling. The authors develop a framework to validate models conformed to a certain domain-specific language employing ontology reasoning services (e.g. an inconsistency checker). Vasilevskaya et al. [10] use the ontology and domain-specific modelling paradigms to create a process for integration of security reusable building blocks into embedded systems. However, the resource concerns are not treated in that process. In comparison, we provide ontological support for the standard MARTE profile as a means of managing performance analysis and compatibility analysis for RBB selection.

Another interesting topic is the use of ontologies to facilitate design of complex systems. Dibowski et al. [31] present an ontology framework to describe devices for the building automation domain. Tang et al. [39] use ontology to configure embedded control systems based on a functional model of a

system that is intended to express user demand. Wagelaar [40] combines the ontology technology with the model-driven architecture principles to enable reuse of platform-independent models to Platform-Specific Models (PSMs) transformations. Tekinerdoğan et al. [41] employ ontologies to support selection of PSMs, where a system platform is described as a set of high level properties. We use ontologies to support composition of embedded systems from RBBs that provide extra-functional properties. Furthermore, we use MARTE models and extend them with additional concepts to formulate platform-specific constraints as a basis for such composition. Specifically, we elaborate on the notion of model-based compatibility as one of possible criteria for selection of RBBs.

We implement the compatibility analysis as queries to the vendor ontologies searching for potentially hidden relations. This allows accounting and reusing the expert knowledge captured by vendors as opposed to querying unrelated models using a tool such as EMF query [42].

VIII. Conclusions

In our work, we recognise the need to reuse the performance evaluation results of RBBs for their integration into embedded system designs when considering EFPs already at the early development phases. Thus, we present an infrastructure that unifies UML profiling, ontologies, and transformation techniques to support the domain engineer in describing these results and an embedded system engineer in using this knowledge. Additionally, we elaborate the model-based compatibility analysis as a basis to reuse the performance analysis results across domains. We developed the LiU MagicDraw plugin to support the above tasks. The case study of smart metering devices is used to illustrate our proposal.

In further work, we will enhance our compatibility technique considering other MARTE packages (e.g. those to model the software layer). Additionally, we will explore other criteria and strategies to reuse the RBBs performance analysis results, validating them on smart metering devices and other case studies given by industrial partners of the SecFutur project [8].

References

[1] S. Kelly and J.-P. Tolvannen, *Domain-Specific Modeling: Enabling Full Code Generation*. John Wiley & Sons, 2008.

[2] T. Lodderstedt, D. Basin, and J. Doser, "SecureUML: A UML-Based Modeling Language for Model-Driven Security," in *Int. Conf. on The Unified Modeling Language*. Springer-Verlag, 2002.

[3] J. Jürjens, *Secure System Development with UML*. Springer-Verlag, 2005.

[4] G. Pedroza, L. Apvrille, and D. Knorreck, "AVATAR: A SysML Environment for the Formal Verification of Safety and Security Properties," in *IEEE Int. Conf. on New Technologies of Distributed Systems*, 2011.

[5] *UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems, version 1.1*, Object Management Group, 2011.

[6] R. S. Preissig, "Data Encryption Standard (DES) Implementation on the TMS320C6000 ," *Application Report*, 2000.

[7] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, "Security in Embedded Systems: Design Challenges," *ACM Transactions on Embedded Computing Systems*, 2004.

[8] SecFutur: Design of Secure and Energy-efficient Embedded Systems for Future Internet Application. http://www.secfutur.eu, visited May 2012.

[9] M. Vasilevskaya, L. A. Gunawan, S. Nadjm-Tehrani, and P. Herrmann, "Security Asset Elicitation for Collaborative Models," in *Model-Driven Security Workshop at MoDELS, ACM*, 2012.

[10] ——, "Integrating security mechanisms into embedded systems by domain-specific modelling," *Sec. and Comm. Net., Wiley*, 2013.

[11] B. Chandrasekaran, J. R. Josephson, and V. R. Benjamins, "What are ontologies and why do we need them?" *IEEE Intelligent Systems*, 1999.

[12] B. Motik, P. F. Patel-Schneider, and B. Parsia. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax, http://www.w3.org/TR/owl2-syntax/, visited Jan. 2013.

[13] XSD type system. http://www.w3.org/, visited Mar. 2013.

[14] Protégé editor. http://protege.stanford.edu, visited Mar. 2013.

[15] "The OWL API. http://owlapi.sourceforge.net, visited Mar. 2013."

[16] E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. http://www.w3.org/TR/rdf-sparql-query/, visited Jan. 2013.

[17] SPARQL in Protégé. http://protege.stanford.edu/doc/sparql/, visited Mar. 2013.

[18] Apache jena project. http://jena.apache.org/, visited Feb. 2013.

[19] M. Z. Iqbal, S. Ali, T. Yue, and L. Briand, "Experiences of Applying UML/MARTE on Three Industrial Projects," in *Model Driven Engineering Languages and Systems, Springer*, 2012.

[20] D. C. Petriu, "Software model-based performance analysis," *book chapter in Model Driven Engineering for distributed Real-Time Systems: MARTE modelling, model transformations and their usages*, 2010.

[21] Platform Ontologies. http://www.ida.liu.se/~marva/ontologies/, visited May 2013.

[22] Z. Xu, Y. Ni, W. He, L. Lin, and Q. Yan, "Automatic extraction of OWL ontologies from UML class diagrams," *World Wide Web*, 2012.

[23] *Ontology Definition Metamodel, version 1.0*, Object Management Group, May 2009, document number: formal/2009-05-01.

[24] MagicDraw. http://www.magicdraw.com, visited May 2012.

[25] Acceleo. http://www.eclipse.org/acceleo/, visited Feb. 2013.

[26] Texas Instruments. www.ti.com, visited Mar. 2013.

[27] Cryptography for C64x+-based Devices. http://www.ti.com/tool/c64xpluscrypto, visited Jan. 2013.

[28] A. Seaborne. "SPARQL 1.1 Property Paths", http://www.w3.org/TR/2010/WD-sparql11-property-paths-20100126/, visited Jan. 2013.

[29] R. Kamal, *Embedded Systems: Architecture, Programming and Design*. Tata McGraw-Hill, 2009.

[30] M. Horridge and S. Bechhofer, "The owl api: A java api for owl ontologies," 2011.

[31] H. Dibowski and K. Kabitzsch, "Ontology-Based Device Descriptions and Device Repository for Building Automation Devices," *EURASIP J. Embedded Syst.*, 2011.

[32] M. Woodside, D. C. Petriu, D. B. Petriu, J. Xu, T. Israr, G. Georg, R. France, J. M. Bieman, S. H. Houmb, and J. Jürjens, "Performance Analysis of Security Aspects by Weaving Scenarios Extracted from UML models," *Journal of Syst. and Soft., Elsevier Science Inc.*, 2009.

[33] E. Bondarev, M. Chaudron, and P. H. N. de With, "CARAT: a Toolkit for Design and Performance Analysis of Component-Based Embedded Systems," in *The Conf. on Design, Automation and Test in Europe, EDA Consortium*, 2007.

[34] F. Ciccozzi, A. Cicchetti, and M. Sjödin, "Round-trip support for extra-functional property management in model-driven engineering of embedded systems," *Information and Soft. Tech., Elsevier*, 2012.

[35] ARTEMIS-JU-216682, CHESS. http://chess-project.ning.com/, visited Mar. 2013.

[36] D. Gasevic, D. Djuric, and V. Devedzic, *Model Driven Engineering and Ontology Development*. Springer Publishing Company, 2009.

[37] P. Tetlow, J. Z. Pan, D. Oberle, E. Wallace, M. Uschold, and E. Kendall. "Ontology Driven Architectures and Potential Uses of the Semantic Web in Systems and Software Engineering". http://www.w3.org/2001/sw/BestPractices/SE/ODA/, 2006.

[38] T. Walter, F. S. Parreiras, and S. Staab, "An ontology-based framework for domain-specific modeling," in *Softw. Syst. Model*. Springer-Verlag, 2012.

[39] B. Tang, J. Yi, Y. Cheng, and C. Gu, "A Reconfigurable Design Method of Embedded Control System based on Ontology," in *IEEE Int. Conf. on E-Product E-Service and E-Entertainment*, 2010.

[40] D. Wagelaar, "Platform ontologies for the model-driven architecture," Ph.D. dissertation, 2010.

[41] B. Tekinerdoğan, S. Bilir, and C. Abatlevi, "Integrating platform selection rules in the model driven architecture approach," in *Conf. on Model Driven Architecture: Foundations and Applications*. Springer, 2003.

[42] EMF, Model Query. http://www.eclipse.org/, visited Jul. 2013.