Towards System Level Performance Evaluation of Distributed Embedded Systems

Jukka Saastamoinen, Subayal Khan, Jyrki Huusko,
Juha Korpi, Kari Tiensyrjä
VTT Technical Research Center of Finland,
FI-90570, Oulu, Finland
email: {jukka.saastamoinen, subayal.khan,
jyrki.huusko, juha.korpi, kari.tiensyrja}@vtt.fi

Jari Nurmi
Tampere University of Finland,
Department of Computer Systems
P.O. Box 553 (Korkeakoulunkatu 1),
lFIN-33101 Tampere, Finland
jari.nurmi@tut.fi

***Abstract*— In order to manage the increasing complexity and heterogeneity of the distributed embedded systems, the system level performance evaluation must be performed at an early design phase using abstract models of the platforms and applications. Thus, the system level performance simulation techniques for embedded systems play a key role in the architectural exploration phase. The salient performance evaluation methodologies are developed around some key concepts and employ a variety of modelling styles, models of computation and programming languages for performance modelling of embedded systems. The aspects of a performance evaluation methodology might limit it to certain domain(s) of embedded systems and therefore must be investigated. In order to span different domains of distributed systems, a methodology must provide the models for technologies, such as communication protocols and middleware technologies, which are employed in different domains of distributed embedded systems. Once achieved, the methodology will be able to provide an estimate of the contribution of these technologies in the non-functional properties of the distributed system. The goal of this survey is to shortlist the performance modeling methodologies feasible for the performance evaluation of different domains of distributed systems. The abstraction level used to model these protocols is investigated since the accuracy of the related performance numbers depend on the used abstraction level. After comparing the salient methodologies on the basis of modeling style, tools and languages, targeted domain etc., we shortlisted the feasible contributions for performance evaluation of different domains of distributed systems. Afterwards, we describe the models and tools needed by the shortlisted methodologies in order to span the different domain of distributed systems. The article acts as a reference for researchers and industries involved in developing methods and tools for system level performance simulation.**

***Keywords**-Performance Model; Application Model; Platform Mode;, Kahn Process Networks; Y-Char; UML; SystemC*

I. INTRODUCTION

Distributed systems are used in diverse market segments including consumer electronics, medical devices, environment monitoring, industrial control, automotive and office automation. The complexity of these systems has increased enormously in all these industrial domains. Therefore, these systems are accompanied with various design challenges [1].

Firstly, the design space is huge not only due to many alternatives for datalink, transport and middleware technologies (for example, specialized MAC (media access control) protocols in WSN (wireless sensor networks) and multitude of middleware technologies in multimedia applications domain) but also in terms of available platforms and various application implementation alternatives. Secondly, due to computational complexity of many distributed applications and the strict design constraints (non-functional properties), the designer has to make critical design decisions at an early stage in order to compare a particular system design with other possible alternatives before the actual implementation and integration of the system proceeds [1].

Moreover, both the functional and non-functional properties of the overall distributed system not only depends on the computations performed within the network nodes but also on the interaction of the various data streams on the common communication media [1].

Therefore, in order to span different domains of distributed systems, a methodology must provide models of MAC, Transport protocols and middleware technologies. This is important because in case of distributed applications, MAC, Transport and Middleware technologies also contribute to the non-functional properties of the system. These non-functional properties include for example end-to-end packet delays and packet loss rate. Also, the increasing complexity of distributed applications demands that the application design phase shall act as a starting point for the application workload modelling phase to reduce the time and effort in the performance simulation and architectural exploration phase [1].

The main contribution of this article is to provide a literature review of the existing performance simulation methodologies in order to evaluate their feasibility for the performance evaluation of multiple domains of distributed systems. The survey first defines important features which must be investigated in order to evaluate the feasibility of performance evaluation methodologies in various domains of distributed embedded systems. Afterwards, the availability of the models and tools which provide these features are highlighted in salient performance evaluation approaches. Based on this information, the methodologies which fulfil these requirements are shortlisted.

The rest of the paper is organized as follows: in Section 2, a comparison of the salient system level performance evaluation (SLPE) methodologies is provided in order to investigate their feasibility for the performance evaluation of (different domains of) distributed systems. In Section 3, the requirements for the SLPE methodology to span different domains of distributed embedded systems are listed. In Section 4, the methodologies which fulfil these requirements are shortlisted on the basis of the information provided in Section 2. Section 5 evaluates the feasibility of ABSOLUT for the performance evaluation of distributed systems in different domains. The conclusions drawn by the survey are presented in Section 6. This is followed by acknowledgements and list of references.

II. Comparison of methods and tools

The main concepts employed by different SLPE methodologies have been described in detail in [2] and [3] and are therefore not presented in this article. The objective of this case study is to investigate the methods and models employed by the salient performance evaluation methodologies to assess their feasibility for performance evaluation of different domains of distributed systems.

Therefore, in this section, we first classify the SLPE methodologies on the bases of four salient features. These features include the modelling style, used languages and frameworks, non-functional properties validation and targeted system and application domains. In Section 3, we use this information to identify the SLPE methodologies which can span different domains of distributed systems.

A. Modelling style

Different design space exploration methodologies employ either static (analytical) or dynamic (simulation) estimation methods. Usually, the static estimation methods shrink the vast design space briskly but the models employed are very coarse. The models used by dynamic estimation methods are more accurate and detailed but are slow at pruning the vast design space. In other words, the static estimation favour speed instead of accuracy for design space exploration while the dynamic exploration methods favour accuracy instead of speed. Some methods utilize a combination of static and dynamic simulation for exploiting the advantages of both methods. Static methods employ analytical or highly abstract models of applications and usually ignore the dynamic behaviour of the application which depends on the input data. As a result, the static methods do not offer the level of accuracy for exploration and communication scheduling as the dynamic simulation methods. Most of the salient performance simulation approaches utilize the dynamic estimation approach while ARTEMIS, MESH and KOSKI use both static and dynamic estimation methods as shown in Table I. It was observed that different methodologies model the applications and platforms at various levels of abstraction and refinement. ABSOLUT employs layered application and platform models. The platform models operate at the transaction level while the lowest layer of application models comprise of abstract instructions. Detailed description of ABSOLUT modelling methodology is provided in [4].

Some methodologies, for example SPADE, TAPES, ARTEMIS and KOSKI, model applications as KPN (Kahn Process Network) MOC (Model of Computation) [5]. Platform models in SPADE are instantiated via a library of generic building blocks which model different resources in the platform. The processing elements in the platform are modelled as TDEUs (trace driven execution unit). Each process of the modelled application is mapped to a TDEU in the platform [6]. TAPES abstracts the processing of tasks by their execution latencies on the corresponding resources in the platform [7]. Further details of platform modelling in TAPES are mentioned in [7].

The architecture models in ARTEMIS operate at transaction level, which simulate the computation and communication events that are generated by the application model [8]. An architecture model is made from a library of generic building blocks which contains templates of performance models for different platform elements [8]. KOSKI employs UML (unified modeling language) for modelling platform which is later on transformed to an abstract model via UML interfaces [9].

ARTS employs static data flow graphs (SDFG) MOC to model the applications while the architecture models operate at transaction level and simulate the performance consequences of the computation and communication events generated by the platform [10][11].

Baghdadi et al. (2000) [12] describes the system-level specifications in SDL (specification and description language) which results in heterogeneous multi-processor architectures consisting of both hardware and software components. SDL can model a variety of embedded software applications (both real time and non-real time).

Fornaciari et al., (2001,2002) [13], [14] uses software execution profiler for the cycle accurate simulation of the application while the data and address bus streams are generated via a dynamic tracer.

Jabber et al., (2009) [15] model applications via DIPLODOCUS tool. A DIPLODOCUS application comprises of a network of tasks which communicate via communication semantics defined by the methodology. The architecture comprises of a network of physical resources which are abstracted by one of three types of architecture nodes, .i.e., the computation nodes (for example CPUs, DSPs, and hardware accelerators etc.), the communication nodes (for example busses, routers and switches etc.) and the storage nodes (for example memories) [15].

Lahiri et al., [16], [17] uses highly abstract application models and only targets the domain of custom communication architectures for on chip communication architectures.

Mesh models applications as dynamic threads made on top of physical threads [18] which model the platform. MILAN [19] uses HiPerE rapid performance estimation tools for estimating the performance of designs (SoC architectures) at the system level. Applications are modelled as trace files in HiPerE which contain an ordered list of communication and computation tasks. HiPerE uses a generic model (GenM) for modelling SoC architectures [19].

Posadas [24] et al., aims at system level estimation of execution time from a system level performance description written in SystemC. It uses a C++ library for this purpose and therefore does not require any change to the source code of the description. The way different methodologies model the application and platforms are summarised in Table I.

Furthermore, some methodologies are capable of exploiting third party tools for different modelling and refining purposes, for example ABSOLUT's workload generation tool called ABSINTH-2 and SAKE [20] use Valgrind [32] for workload extraction of external libraries. KOSKI employs existing compilers and code generators in order to refine the application to the final processing elements [21]. ARTEMIS [25] uses Laura tool-set for the generation of synthesizable VHDL code from the KPN application description. Furthermore, it was found that all the landmark contributions considered in Table I employ simulation for computing performance numbers.

B. *Languaged, standards and frameworks*

The landmark performance evaluation methodologies described in Table I use a variety of widely used modelling, scripting and Programming languages such as C, C++, Perl, UML and XML for various modelling purposes.

Some methodologies use specification languages such as SDL or LOTOS OSI specification language for system-level specifications. Some methodologies such as MILAN use other tools such as HiPerE and DESERT for modelling and simulation purposes [19]. Lahiri et al. [16] [22] uses POLIS and PTOLEMY frameworks for designing communication architectures of SOCs.

It was observed that some methodologies use modelling languages and simulation frameworks such as SystemC which is widely used for the system-level modelling, architectural exploration, performance modelling etc., of electronic systems. The programing and modelling languages used by the landmark methodologies are shown in Table I.

C. *Non-functional properties validation*

The distributed embedded systems support applications which consist of many components running on different networked devices. In such cases, the application components communicate via transport, data link and (possibly) middleware technologies. These distributed applications are generally message based or streaming applications which satisfy the end-user requests by (in turn) requesting one or more services provided by different devices which implement these services. Therefore, the end-user experience is not merely a consequence of the application implementation since the transport protocols, data link protocols as well as physical layer plays a key role in the end-user experience since the end-end delays and packet/frame errors at these layers can deteriorate the end-user experience. Therefore, for a methodology to be able to estimate reliable performance numbers for distributed applications, it must model these OSI model layers with sufficient level of detail. These models must preserve the functionality to a level that the estimated delays show a close correlation with delays estimated by network simulators such as OMNeT++ and ns-2.

It has been noticed that some methodologies are totally focused on one particular domain of applications and systems. The methodologies such as ARTEMIS, KOSKI, SPADE and TAPES which model applications via KPN MOC are limited to the performance estimation of streaming applications since KPN models only model streaming applications very well. Therefore a wide variety of message based distributed applications cannot be modelled via these methodologies. Some methodologies such as Lahiri et al. and ARTS use other models of computations such as CAG (communication analysis graph) and SDFG (static data flow graph) for modelling applications. It was also observed that some methodologies use their own model of computation for describing applications while the others such as ABSOLUT employ a layered application model [4].

In all the methodologies which employ a model of computation to describe the applications, we observe that the functionality of the transport and datalink layer has been abstracted by the communication paradigm employed by the MOC. This means that the non-functional properties of a distributed application (such as end-end delays and packet/frame loss rate) cannot be reliably estimated since the functionality of transport and datalink layers have been swapped by that of the communication means defined by the employed MOC. All the MOCs use simple channels for communication among processes for example KPN MOC use simple FIFO (first-in first-out) channels for passing synchronization tokens among processes. On the other hand in networked devices, datalink layer MAC protocols resolve the contentions for occupancy of the common channel (wired or wireless). The level of abstraction used to model channels should be comparable to the abstraction level employed by network simulators such as OMNeT++ and ns-2 [23]. The MOCs, targeted application domain and the availability of models for transport, datalink and Middleware technologies models are highlighted in Table I for the landmark performance simulation methodologies. In short, the emphasis is on the model of computation employed by the methodologies, since it can potentially restrict an approach to a specific domain of applications. Also, the models of transport and MAC protocols employed by different approaches are investigated since they play an important role in determining the non-functional properties such as end-end message delays in distributed applications.

These non-functional properties can play an important role in end-user experience and must be estimated with reasonable accuracy.

D. *Targeted system and application domains*

The use of multiprocessor based platforms is increasing in high-end mobile handheld devices such as smart phones and internet tablets. On the other hand, in case of wireless sensor networks very low power single processor based systems are commonly used. Hence, for the performance simulation of a wide variety of distributed embedded systems, it is important that the methodology is not restricted to a certain type of platforms (single or multiprocessor based). Also, it should not be strictly targeted at the performance evaluation of a particular subsystem or components of a platform such as performance evaluation of on-chip communication architectures. The system and application domains (of embedded systems) targeted by salient performance evaluation methodologies are listed in Table I. In Table I, we add the domain of distributed networked systems to the three domains of embedded systems elaborated in [21].

Some methodologies are totally focused at the performance evaluation of a particular domain of systems, for example TAPES and Fornaciari et al. target single processor based systems, SPADE and Baghdadi et al. only target Multi-Processor based systems while Lahiri et al. is only focused on the performance evaluation of on-chip communication architectures. The other landmark methodologies are also focused on only two out of the three performance simulation objectives, .i.e., performance evaluation of single processor based SoC, multi-processor based SoC and on-chip communication architectures [21]. Only three methodologies .i.e.; ABSOLUT, ARTEMIS and KOSKI have no such restriction.

TABLE I. ANALYSIS OF SALIENT DESIGN SPACE EXPLORATION METHODOLOGIES

	Modelling Styles, Tool Coupling and Performance Estimation					**Languages, Standards and Other Frameworks Used**		**Non-functional properties validation**				**Targeted System Domain**			
												Non-Distributed Systems			
Name of approach	**Performance estimation type**	**Application Model**	**Architecture Model**	**Tool Coupling**	**Simulation Based Approaches**	**Programming/Modelling Languages and Standards Spanned**	**Other Approaches/Frameworks Used**	**Model of computation**	**Targeted Application Domain**	**Transport and Data-link Model**	**Middleware Layer Workload Models**	**Single Processor based systems**	**Multi-Processor based SOC**	**On-Chip/Intra-Platform Communication Architectures**	**Distributed Networked Systems**
ABSOLUT	D	*X^{11}*	*X^{12}*	*X^{13}*		C/C++/SystemC2.2/TLM2.0/UML		TLM	A			X	X	X	
ARTEMIS	D/S	*X^{21}*	*X^{22}*	*X^{23}*		PEARL, SystemC, RTL	SPADE, SESAME	KPN	ST			X	X	X	
ARTS	D	*X^{31}*	*X^{32}*			SRTS Scripting Language, SystemC		SDFG	ST				X	X	
Baghdadi et al	D	X^{41}	X^{42}			C,RTL,SDL,	MUSIC, CODESIM	N	X^{42}				X		
Fornaciari et al	D	X^{51}	X^{52}			C/C++	MEX, Shade, X^{53}	N	ST			X^{54}			
Jaber et al	D	X^{61}	X^{62}			UML, SystemC, LOTOS OSI Specification Language	DIPLODO-CUS	N	X^{63}			X	X		
Koski	D/S			*X^{71}*		TUT UML Profile, XML	Existing Code Generators and Compilers	KPN	A			X	X	X	

Lahiri et al	D		X			Languages/Tools used by POLIS & PTOLEMY	POLIS, PTOLEMY	CAG	X^{81}					X	
MESH	D/S		X			C		X^{91}	A			X	X		
MILAN	D	X^{101}	X			Languages/ Tools used by DESERT & HiPerE	DESERT, HiPerE, X^{102}	X^{103}	A						
Posadas et al	D	X^{111}				C++/SystemC		X^{112}	X^{113}			X	X		
SPADE	D	X	X			C/C++	YAPI, TSS	KPN	ST				X		
TAPES	D		X			SytemC, XML		KPN	ST			X			

Abbreviations	
A	No restriction as per our assessment. Also no mention of a particular application domain by the authors.
ST	Streaming Applications.
D	Dynamic
S	Static
N	No MOC (Model of Computation) such as KPN, CAG, TLM and CDFG used or affectively adapted/employed by the methodology. The modelling of applications is elaborated in the corresponding reference in the second column.
TML	Transaction Level Modelling

E. Methodology specific information used in Table I

1) ABSOLUT: **(X^{11})** ABSOLUT uses layered workload models consisting of application, process and Function workload layers. The function workloads consist of abstract instructions and control.

(X^{12}) The platform model is layered and consists of three layers, .i.e. component, subsystem and platform architecture layer.

(X^{32}) The ABSINTH-2 tool uses Valgrind for workload extraction of external libraries.

2) ARTEMIS: (X^{21}) Applications are modelled as KPNs which are either generated by a framework called Compaan or derived manually from sequential C/C++ code.

(X^{22}) Architecture models operate at the transaction level and an architecture model is made from a library of generic building blocks containing template performance models for processing cores, communication media, and various types of memory.

(X^{23}) ARTEMIS uses Laura tool set for automatic generation of VHDL code from KPN based application model.

3) ARTS: **(X^{31})** Applications are modelled using static dataflow task graphs.

(X^{32}) Platform consists of multi-processor models, memories, communications and other platform resources.

4) Baghadi et al.: **(X^{41})** Information related to application and architecture modelling and tool coupling is mentioned in Section 2 A (modelling style).

(X^{42}) The system-level specifications are described in SDL. This results in heterogeneous multi-processor architectures comprising of both hardware and software. SDL does not explicitly specify any particular domain or restriction as far as its ability to model software (both real time and non-real time) is concerned.

5) Fornaciari et al: **(X^{51})** The simulation framework is based on a software execution profiler for cycle-accurate instruction set simulation of the application and a dynamic tracer to generate data and address bus streams.

(X^{52}) Design space exploration is focused on the processor to memory communication through the memory hierarchy and includes configurable bus and memory models, with the latter having behavioural models of on and off-chip level 1 and 2 caches and main memory. The bus and memory models use the bus traces from the software execution profiler as input.

(X^{53}) The architecture exploration is done by using a tool called MEX. It simulates the execution of a program compiled for the Sparc V8 architecture with configurable memory architecture. MEX exploits the Shade [14] library to trace the memory accesses made by a SPARC V8 program and consequently simulates the target memory architecture to obtain accurate memory access statistics. The MEX tool developed by authors uses Shade tool which is based on C/C++ [14] [13].

(X^{54}) Design space exploration is focused on the processor to memory communication through the memory hierarchy. The technique aims at finding the best platform configuration for the application without an exhaustive search of the parameter space. The parameters for the exploration include cache size; block size and instruction cache associativity.

6) Jaber et al.: **(X^{61})** Applications are modelled via DIPLODOCUS tool. A DIPLODOCUS application consists of a network of communicating tasks which can communicate via three communication elements .i.e., the channels, events and requests. The channels exchange the abstract data samples, events exchange signals and the requests ask for and thus trigger the execution of another task.

(X^{62}) The architecture is modelled as a network of physical resources, including computation, communication and storage nodes. All resources have parameters like processing capacity in millions of cycles per second or memory size in bytes.

(X^{63}) A variety of real-time and embedded applications can be modelled with sufficient accuracy by using the data and functional abstraction described by authors in [15].

7) KOSKI: **(X^{71})** KOSKI uses existing compilers and code generators for refining the application to the final processing element.

8) Lahiri et al: **(X^{81})** It is only targeted at design of custom communication architectures of systems on chip.

9) MESH: **(X^{91})** The framework is based on a layered composition of threads, with the dynamic logical threads made on the top of physical threads. The physical threads model the hardware components of a platform and represent their computational power. The application software is modelled as logical threads. The execution of a dynamic number of logical threads is scheduled (by the scheduling layer of MESH) onto a processing element (for example a processor modelled as a physical thread).

10) MILAN: **(X^{101})** Applications are modelled as trace files by HiPerE and consist of a list of communication and computation tasks.

(X^{102}) DESERT and HiPerE are used for rapid design space exploration. DESERT shrinks the design space by shortlisting designs and HiPerE estimates the performance.

(X^{103}) The methodology proposed by [19] is only aimed at system level estimation of execution times from a system level performance description written in SystemC. No MOC is employed by this methodology.

11) Posadas et al.: **(X^{111})** This methodology aims at system level estimation of execution time from a system level performance description written in SystemC and therefore does not employ application models. It estimates the execution time of the application via a C++ library.

(X^{112}) Application is modelled as a set of Processes which can only interact with each other via predefined channels.

(X^{113}) Only C++ applications can be simulated.

12) Summary: In this section, the important aspects of salient system level performance simulation methodologies were elaborated. We observed that these methodologies employ a variety of tools and modelling languages and mostly focus on a few modelling (targeted system domain) objectives shown in Table I. Different methodologies describe the application and platform models at different levels of abstraction and employ different models of computation for describing the application models.

Also, some of the methodologies use third party tools for modelling or simulation purposes and some provide tools coupling for extending the usability of the methodology for other simulation objectives. In the next section we further investigate the feasibility of landmark performance evaluation approaches described in this section for the SLPE of distributed embedded systems.

III. TOWARDS PERFORMANCE EVALUATION OF DISTRIBUTED EMBEDDED SYSTEMS

After investigating the salient features of landmark performance evaluation approaches in Section 2, we conclude that in order to validate the non-functional properties of distributed embedded systems in different domains the methodology must fulfil the following salient requirements.

A. MOC agnostic

The methodology must not employ a specific model of computation for modelling applications since this will restrict the methodology to a particular domain of applications or systems, for example the methodologies which uses KPN MOC for application or platform are mainly targeted at performance evaluation of streaming applications.

B. Multithreaded applications modelling

In order for the methodology to evaluate the performance of multi-threaded applications, the methodology must model the multi-threading support for system level performance simulation of these applications.

C. Physical and transport layer models

Physical layer models such as channel models, coding and modulation techniques as well as the functional models of datalink and transport layer protocols must be provided for evaluating their contribution in non-functional properties [23]. Also, the methodology must be capable of evaluating the performance of protocols operating on a particular layer of the OSI model in isolation just like OMNeT++ and ns-2 [23].

D. No domain restrictions

In order to span the domain of distributed systems such as WSNs, the methodology must be capable of evaluating the percentage utilization of the platform by data link and transport Protocols. WSNs in particular employ highly efficient and specialized datalink protocols to reduce power consumption.

E. Workload modelling of user-spacecode, libraries and system calls

From an implementation perspective, all the applications processes use user-space code, external libraries, background processes and system calls. Therefore, the methodology must provide tools and methods for generating the workload models of not only the user space code but also the external libraries, background processes and system calls.

F. *Workload generation of middleware technologies*

It must be capable of workload extraction of API functions of the various Middleware technologies such as NoTA (network on a terminal architecture) SOA (service oriented architecture). This will enable the methodology to span the domain of distributed streaming and context aware applications.

G. *Detailed as well as highly abstracted workload modelling*

The methodology must provide/define application workload modelling tools/techniques for generating the application workload models with varying degrees of refinement and detail. The more refined and detailed workload models result in slower simulation speed due to increased structure and control while the less detailed workload models usually result in faster simulation speed [29] [4] [30]at the expense of accuracy. Once this is achieved, the system designer can freely choose the workload models that will provide the right balance between accuracy and speed for the modelling objective.

H. *Integration of application design and performance evaluation*

For early phase evaluation of the distributed applications, the methodology must automate the workload extraction process by seamless integration of application design and performance simulation phase. This can be achieved if the application and workload modelling phases are linked such that application models act as a starting point for the application workload modelling. The proposed technique must be experimented with modern SOAs such as GENESYS and NoTA.

I. *Non-functional properties validation*

The non-functional properties must be carried through the application design phase and validated by the performance simulation approach. The non-functional properties are usually modelled and elaborated in the application model views [1][27][28].

IV. FEASIBILITY OF EXISTING SLPE APPROACHES

As shown in Table I, none of the methodologies is capable of providing reliable estimates of the non-functional properties of distributed applications. The reason is that in case of distributed embedded systems, the transport, datalink and (possibly) middleware technologies contribute to the non-functional properties such as end-end frame and packet delays. In order for a methodology to accurately estimate the effects of these protocols on non-functional properties; it must employ functional MAC and Transport Protocols.

As shown in Table I, majority of the performance modelling techniques are limited to a particular domain of embedded systems and applications due to which they cannot be employed for the performance evaluation of different domains of distributed embedded systems.

Only three out of all the approaches mentioned in Section 2, .i.e., ABSOLUT, ARTEMIS and KOSKI are not restricted to any particular domain of embedded systems. Furthermore, out of these approaches, ARTEMIS and KOSKI use KPN MOC for modelling applications which can only model streaming applications well [6]. Therefore, out of all the system level performance evaluation methodologies presented in Table I, ABSOLUT is most feasible for the performance evaluation of distributed embedded systems since it is not limited to any particular system or application domain. As explained before, this is due to the fact that it does not employ any MOC for modelling applications or platforms. In the next section, we describe the tools and models employed by ABSOLUT for fulfilling the requirements mentioned in Section 3.

V. EVALUATIN FEASIBILITY OF ABSOLUT

A. *MOC agnostic*

ABSOLUT uses SystemC for modelling platform components. ABSOLUT methodology does not employ any specific MOC for modelling platforms and applications. It employs a component library for instantiating platform models and ABSINTH-2 and SAKE tools for automatic application workload generation.

B. *Multithreaded applications modelling*

Multi-Threaded support has been modelled and integrated to ABSOLUT due to which it can be used for the performance evaluation of multithreaded applications. The approach has been described via a case study in [26].

C. *Performance evaluation of protocols*

ABSOLUT provides an operating system (OS) model which is hosted on the processor model in the platform. The ABSOLUT OS model consists of a scheduler and provides the possibility to model different OS Services. The scheduler schedules the application model processes. Platform services can be implemented by the system designer as described in [23]. The implementation of services closely mimics the way services are scheduled by the widely used platforms (mostly via scheduling queues). Highly accurate transport, datalink and physical layer models have been designed and integrated to ABSOLUT [23] [27].

D. *No domain restriction*

ABSOLUT does not model the application workload models and platform capacity models by employing a certain MOC. This property allows the system designers to employ ABSOLUT for the performance evaluation of different domains of embedded systems.

E. *Workload modelling of user-space code, libraries and system calls*

ABSOLUT can model the workload models of user-space code and external libraries. Furthermore, the automatic workload modelling of system calls for a variety of platforms is performed via CORRINA **Error! Reference source not found.**.

F. *Workload modelling of middleware technologies*

The workload modelling of middleware technologies is performed via ABSINTH-2. The middleware technologies can also be modelled as system calls. The workload modelling of NoTA device interconnect protocol (DIP) has been demonstrated in a case study aimed at the SLPE of distributed NoTA systems [27].

G. *Detailed as well as highly abstract workload modelling*

ABSOLUT provides different tools for modelling application workloads at various abstraction levels and refinement. This allows the system designer to choose the tools which provide the right compromise between speed and accuracy.

H. *Integration of application design and performance evaluation*

ABSOLUT application workload models can be easily modelled by extending the application model which acts as a blue print for application workload models. The seamless integration of application design and performance evaluation minimizes the time and effort involved in performance evaluation phase.

I. *Non-functional properties validation*

For the validation of non-functional properties, the non-functional properties must be carried through the application design phase and validated by the performance evaluation phase. The seamless integration of application design and ABSOLUT workload modelling has been demonstrated for different service oriented application architecture design methodologies such as GENESYS and NoTA [1] [27].

ABSOLUT has been successfully employed for the performance simulation of NoC based SOCs and distributed embedded systems [1] [27]. We now list the features mentioned in Section 3 which are provided by ABSOLUT and also provide the references to the research articles which demonstrate these features via case studies. This information is presented in Table II.

TABLE II. FEATURES PROVIDED BY ABSOLUT FOR THE PERFORMANCE EVALUATION OF DISTRIBUTED EMBEDDED SYSTEMS

Number	**Feature**	**References**
I	*MOC Agnostic*	[4]
II	*Multithreaded Applications Modelling*	[26]
III	*Performance Evaluation of Protocols*	[23]
IV	*No Domain Restriction*	[1][27][28]
V	*Workload Model Generation of User-Space code, External Libraries and System Calls*	[4][20][30]
VI	*Workload Generation of Middleware technologies*	[27][20]
VII	*Detailed and Highly abstract workload modelling*	[29][4][30]
VIII	*Integration of Application Design and Performance Evaluation*	[1][28][27]
IX	*Non-functional Properties Validation*	[1][28][27]

From Table II it is clear that all the features which are required by a SLPE methodology to evaluate the performance of distributed embedded systems in different domains are provided by ABSOLUT.

VI. CONCLUSION AND FUTURE WORK

We therefore conclude from the survey that many methodologies have been developed for the SLPE of distributed systems. Those methodologies which employ a MOC for modelling applications or platforms are restricted to a particular domain of embedded systems. Also, these methodologies cannot provide reliable performance numbers of certain non-functional properties of distributed systems since the MAC and Transport protocols models are absent or abstracted out. These protocols play a key role in the end-user experience by contributing to non-functional properties. These non-functional properties such as end-end packet delays, packet loss rate and frame loss rate must be taken into account for performance modelling of distributed embedded systems so as to ensure a good end-user experience after the development and deployment of the distributed system.

We found that ABSOLUT can be used for the performance evaluation of distributed systems in different domains. The reason is that it does not employ any MOC for modelling applications and platforms. Also, ABSOLUT provides the models of different protocols (for example MAC and transport) which are important for the SLPE of distributed systems. Of course, all the performance evaluation methodologies have not been covered in this survey. The main contribution of this article is to describe a disciplined approach for classifying the performance evaluation methodologies which helps to determine their feasibility for the performance evaluation of distributed embedded systems. By evaluating a methodology on the basis of requirements looking at the requirements we can evaluate its potential for SLPE of distributed embedded systems. If all the requirements mentioned in Section 3 are fulfilled by a methodology, it can be used to evaluate the performance of a wide range of distributed embedded systems.

ABSOLUT fulfils these requirements and provides the tools for the workload modelling at different levels of abstraction and refinement. Also, the ABSOLUT application workload models can be obtained by extending application model. The extended layered application architecture is analysed to identify the corresponding ABSOLUT workload model layers. This reduces the time and effort in performance modelling which is important to consider due to increasing complexity of distributed applications.

In the future, a number of widely used MAC and Transport protocol models will be designed and integrated to ABSOLUT. Currently IEEE 802.11 DCF, UDP and TCP models are provided. Also, a GUI front-end will be beneficial for easy instantiation of ABSOLUT performance models. It will also help to reduce the learning curve for SLPE via ABSOLUT toolset.

ACKNOWLEDGMENT

This work was performed in the Artemis SOFIA and SMECY projecst partially funded by Tekes – The Finnish Funding Agency for Technology and Innovation and the European Union. The work was performed in cooperation with Finnish ICT SHOK research project Future Internet. Authors would like to thank all their colleagues for valuable discussions about the topic.

REFERENCES

[1] I. Lee, J.Y.T Leung, S. H. Son. Handbook of Real-Time and Embedded Systems. Publisher: Chapman and Hall/CRC (July 23, 2007) .800 pages. Language: English.ISBN-10: 1584886781. ISBN-13: 978–1584886785.

[2] S. Khan, S. Pantsar-Syväniemi, J. Kreku, K. Tiensyrjä, J.-P. Soininen, ”Linking GENESYS application architecture modelling with platform performance simulation,” Forum on Specification and Design Languages 2009 (FDL2009). Sophia Antipolis, France, September 22-24, 2009. ECSI. France (2009)

[3] S. Khan, E. Ovaska, K. Tiensyrjä, J. Nurmi, “From Y-chart to seamless integration of application design and performance simulation,” Proceedings 2010 International Symposium on System-on-Chip - SOC. Tampere, Finland, 29-30 Sept. 2010. IEEE. Piscataway, NJ, USA (2010), pp. 18-25.

[4] J. Kreku, M. Hoppari, T. Kestilä, Y. Qu, J.-P. Soininen, P. Andersson, K. Tiensyrjä, “Combining UML2 application and systemc platform modelling for performance evaluation of real-time embedded systems,” EURASIP Journal on Embedded Systems. DOI: 10.1155/2008/712329.

[5] M. Gries, “Methods for evaluating and covering the design space during early design development,” Integration, the VLSI journal, vol. 38, 2004, pp. 131–183.

[6] P. Lieverse, P. van der Wolf. E. Deprettere, “A trace transformation technique for communication refinement,” Proc. 9th International Symposium on Hardware/Software Codesign (CODES 2001), pp. 134–139.

[7] T. Wild, A. Herkersdorf, G.Y. Lee, “Tapes—trace-based architecture performance evaluation with SystemC,” Design Automation for Embedded Systems 10(2–3): Special Issue on SystemC-based System Modelling, Verification and Synthesis , pp. 157–179..

[8] A.D. Pimentel, L. Hertzberger, P. Lieverse, P. van der Wolf, E. Deprettere, “Exploring embedded systems architectures with Artemis,” IEEE Computer 34(11), Nov 2001, pp. 57–63.

[9] T. Kangas, P. Kukkala P, H. Orsila, “UML-based multiprocessor SoC design framework. ACM Transactions on Embedded Computing Systems (TECS), Vol. 5 Issue 2, May 2006, pp. 281–320.

[10] S. Mahadevan, F. Angiolini, M. Storgaard, R. Olsen, J. Sparso J, J. Madsen, “A network traffic generator model for fast network-on-chip simulation,” Proceedings of the Design, Automation and Test in Europe (DATE05), 2005, pp. 780–785.

[11] S.Mahadevan, M. Storgaard, J. Madsen, K. Virk, “Arts: a system-level framework for modeling MPSoC components and analysis of their causality,” Proceedings of the International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005, pp. 480–483.

[12] A. Baghdadi, N. Zergainoh, W. Cesario, T. Roudier T, A.A. Jerraya, “ Design space exploration for hardware/software codesign of multiprocessor systems,” Proc. 11th International Workshop on Rapid System Prototyping (RSP), 2002, pp. 8–13.

[13] W. Fornaciari, D. Sciuto, C. Silvano, V. Zaccaria, “A sensitivity-based design space exploration methodology for embedded system,” . Design Automation for Embedded Systems, 7, 2002, pp. 7-33.

[14] W. Fornaciari, D. Sciuto, C. Silvano, V. Zaccaria, “A design framework to efficiently explore energy-delay tradeoffs,”. Proc. Ninth International Symposium on Hardware/Software Codesign (CODES), 2001, pp. 260–265.

[15] C. Jaber, A. Kanstein, L. Apvrille, A. Baghdadi, P.L. Moenner, R. Pacalet, “High-level system modeling for rapid hw/sw architecture exploration,” Proc. IEEE/IFIP International Symposium on Rapid System Prototyping (RSP ’09), Paris, France, pp. 88–94.

[16] K. Lahiri, S. Dey, A. Ragunathan,”Evaluation of the traffic-performance characteristics of system-on-chip communication architectures,” Proc. Proceedings of 14th International Conference on VLSI Design, 2001, pp. 29–35.

[17] K. Lahiri K, A. Ragunathan, S. Dey, “Efficient exploration of the SoC communication architecture design space,” Proc. IEEE/ACM International Conference on Computer Aided Design (ICCAD), 2000, pp. 424–430.

[18] J. Paul, A. Bobrek, J. Nelson, J. Pieper, D. Thomas, “Schedulers as model-based design elements in programmable heterogeneous multiprocessors,” Proc. Design Autamation Conference, 2003, pp. 408-411

[19] S. Mohanty, V. Prasanna, “Rapid system-level performance evaluation and optimization for application mapping onto soc architectures,”. Proc. Proceedings of the IEEE International ASIC/SOC Conference, 2002, pp. 160–167.

[20] J. Saastamoinen, J. Kreku, ”Application workload model generation methodologies for system-level design exploration,” Proceedings of the 2011 Conference on Design and Architectures for Signal and Image Processing, DASIP 2011. Tampere, Finland, 2-4 Nov. 2011. IEEE Computer Society (2011), pp. 254-260

[21] T. Kangas, "Methods and Implementations of Automated System for a Chip Architecture Exploration", PhD Thesis, Tampere University of Technology, Publication 616, 2006, 181 pages.

[22] K. Lahiri, A. Ragunathan, S. Dey, ”System-level performance analysis for designing on-chip communication architectures,” IEEE

Transactions on Computer-Aided Design of Integrated Circuits and Systems 20(6), 2001, pp. 768–783.

[23] S. Khan, J. Saastamoinen, M. Majanen, J. Huusko, J. Nurmi, " Analyzing transport and MAC layer in system-level performance simulation," 2011 International Symposium on System on Chip, SoC 2011. Tampere, Finland, 31 Oct. - 2 Nov. 2011. IEEE Computer Society (2011), 8 p.

[24] H. Posadas, F. Herrera, P. Sanchez, E. Villar E, F. Blasco, "System-level performance analysis in SystemC," Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE 2004), Paris, France, 2004, pp. 378–383.

[25] A.D. Pimentel, P. van der Wolf, E. Deprettere, L. Herzberger, "The Artemis Architecture Workbench," Proceedings of the Progress Workshop on Embedded Systems, Utrecht, Netherlands, 2000, pp. 53-62.

[26] J. Saastamoinen, S. Khan, K. Tiensyrjä, T. Taipale, "Multi-threading support for system-level performance simulation of multi-core architectures," ARCS 2011. 24th International Conference on Architecture of Computing Systems 2011, Workshop Proceedings. VDE Verlag Gmbh, 2011, pp. 169-177

[27] S. Khan, J. Saastamoinen J. Nurmi, "System-level performance evaluation of distributed multi-core NoTA systems," 2nd IEEE International Conference on Networked Embedded Systems for Enterprise Applications. NESEA 2011, Fremantle, Dec. 8-9, 2011. IEEE (2011)

[28] S. Khan, J. Saastamoinen, K. Tiensyrjä, J. Nurmi, "SLPE of distributed GENESYS applications on multi-core platforms," The 9th IEEE international symposium on Embedded Computing (EmbeddedCom 2011). Sydney, Dec 12-14, 2011

[29] J. Kreku, M. Hoppari, T. Kestilä, Y. Qu, J.-P. Soininen, K. Tiensyrjä, "Languages for Embedded Systems and their Applications" volume 36 of Lecture Notes in Electrical Engineering, chapter Application Workload and SystemC Platform Modeling for Performance Evaluation, pp. 131–148. Springer.

[30] J. Kreku, J. Penttilä, J. Kangas, J.-P. Soininen, "Workload simulation method for evaluation of application feasibility in a mobile multiprocessor platform," Proc. Proceedings of the Euromicro Symposium on Digital System Design, 2004, pp. 532–539.

[31] S. Khan, J. Saastamoinen, K. Tiensyrjä, J. Nurmi, "Application workload modelling via run-time performance statistics,". IJERTCS-2012 *(Submitted-Under Review)*

[32] N. Nethercote, J. Seward, "Valgrind: A Framework for Heavyewight Dynamic Binary Instrumentation, in Proceedings of ACM SIGPLAN 2007 Conference of Programming Language Design and Implementation (PLDI2007), San Diego, California, USA, June 2007