# Application-specific Architecture Selection for Embedded Systems via Schedulability Analysis

Han Liu*, Hehua Zhang*, Yu Jiang†, Xiaoyu Song‡, Ming Gu* and Jiaguang Sun*
*School of Software Tsinghua University, TNLIST, KLISS, Beijing, China
†Department of Computer Science and Technology, Tsinghua University, TNLIST, KLISS, Beijing, China
‡Department of ECE, Portland State University, Oregon, USA

***Abstract*—Architecting real-time embedded systems is of the top significance during the design phase, especially in complex applications. Due to limited time and resource, to guarantee scheduling eminence without violating application-specific constraints is a challenging problem in architecture level. In this paper, we firstly present an enhanced transformation from AADL models to Cheddar input for schedulability analysis. With subprogram and delayed connection, this transformation is feasible for complex system designs. Based on schedulability analysis, we further propose a novel architecture selection engine, which evaluates scheduling performance through selection standards and application-specific constraints via satisfaction functions. With the proposed selection engine, information from both schedulability and real-time constraints are captured to pick up an optimal architecture. We apply the proposed approach on the architecture selection of an industrial control system in railway applications. Four candidate AADL architectures are transformed and analyzed for schedulability. Then in the selection engine, candidates are ranked within two application constraints. Compared to the selection of general criteria and traditional AHP, our engine excels at better schedulability and satisfaction on real-time application-specific constraints. Moreover, with adjustment on constraints, our engine shows delicate sensitivity by generating a modified selection. We believe the proposed approach can facilitate architecture design of real-time embedded systems.**

***Keywords*—AADL; Schedulability; Architecture Selection Engine**

## I. Introduction

With the widespread application of real-time embedded systems(RTES), architecture modeling is gaining more attention. Its increasing popularity results from two aspects. Firstly, most system failures are ascribed to inappropriate architecture designs. For RTES where timing and resource are strictly constrained, a bad architecture may lead to a weak implementation which is incapable of satisfying constraints. Secondly, preference for architecture modeling comes from the need to uncover design defects at early stage. Based on an architecture model, schedulability analysis can check timing propertie of RTES. As a result, it is easier to fix poor designs at an architecture level to reduce development costs.

However, architecture design is absorbing more complex features as remote calls and synchronous computation. Since these features can hardly be loaded for analysis, detecting design defects is a challenging problem. In addition, a schedulable architecture is not necessarily the optimal for two reasons. Firstly, applications may have various priorites over scheduling parameters. Moreover, application-specific constraints are difficult to capture and analyze in practice, which poses another challenge in architecture design of RTES.

To address these problems, we present a novel approach for architecture selection in RTES with specification in Architecture Analysis and Description Language(AADL) [1]. Main contributions of this paper are: (1) An augmented schedulability analysis method for RTES, where AADL architecture specification is transformed into input of a popular schedulability analyzer Cheddar [2]. Compared to other transformation strategies, our method includes advanced constructs as subprogram and delayed connection. Consequently, complex architecture with various timing of communication can be analyzed for schedulability. (2) A novel architecture selection engine, integrating scheduling information with application-specific constraints. Candidate architectures are evaluated over response time, processor utilization, context switches and flow latency, which are strongly related to timing performance of the system. In addition, we defined a layer of application-specific constraints in the engine to detect and reflect constraint violations in architecture selection, which makes our engine a better alternate for RTES.

## II. Related work

Limited time and computation resource, as the intrinsic nature of embedded systems, mirrors the value of schedulability analysis. In [3], Liu's theory states that processor utilization can be used as a sufficient condition for schedulability. To analyze schedulability of AADL models, transformation-based approaches are proposed. Timed automata and Uppaal are used in [4] to analyze schedulability through verification. Based on Liu's theory, Cheddar [2] is implemented to address this problem in a delicate manner with detailed scheduling information.

From another perspective, architecture selection is already a prevailing topic. It is accepted that the selection is a tradeoff between most vital architectural angles. Due to the simplicity, multi-criteria decision making(MCDM) methods like Analytical Hierarchy Process(AHP) [5] are widely adopted to balance this tradeoff. The related works may be categorized in two groups: (1) selection in embedded systems. [6] presents a combined method with Architecture Tradeoff Analysis(ATAM) and AHP to make decisions on choosing SW/HW integration. Attributes as safety, reliability, modifiability and serviceablity are taken into consideration. In [7], AHP and other MCDM methods are used to analyze performance of the partition design on embedded systems. Although this approach shares some common attributes with ours, including processor utilization, latency and response time, it does not handle the domain

diversity in architecture selection. Aleti et al implements an Eclipse-based tool ArcheOpterix to evaluate architectures in terms of data transmission reliability and communication overhead [8]. (2) selection in software. Based on AHP, many works rely on quality attributes for architecture selection as in [9]. [10] proposes an AHP-GP model to capture nonfunctional requirements in architecture selection. In contrast to selecting an architecture design, [11] leverages AHP to identify critical tradeoffs and sensitive points in design. [12] presents an AHP-based method to accurately capture empirical knowledge of stakeholders.

## III. Architecture Selection Engine

We propose an architecture selection engine with a combination of schedulability analysis and application-specific constraints. The selection process is displayed in Figure 1.

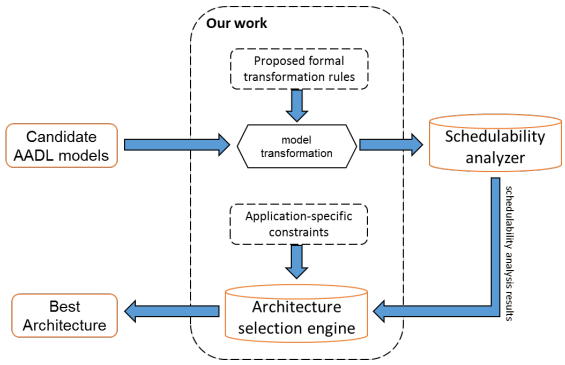



Fig. 1: Architecture selection process

As presented in Figure 1, candidate architectures are transformed for schedulability analysis using the proposed rules. Scheduling results are collected by the analyzer and delivered into the proposed selection engine. Based on the proposed criteria, our engine evaluates candidates and checks performance through a proposed satisfaction function over application constraints. As output of our engine, an optimal architecture is selected.

### A. Architecture transformation

The current Cheddar is incapable of handling AADL models with subprogram and delayed connection. We provide an enhanced transformation from AADL to Cheddar to cover non-support components.

An AADL model $M$ is a tuple ***(Pro,Th,Sp,D,Int)*** where ***Pro*** is a set of processors. ***Th*** denotes thread set. A thread is a tuple $(dp, c, d, p, call)$, where $dp \in \{period, aperiod, sporadic, background\}$ represents the dispatching protocol. $c$ is the execution time. $d$ and $p$ are the deadline and period of the thread. $call$ is a set of subprogram calls in a thread. ***Sp*** is subprograms. ***D*** is the data defined in the model. ***Int*** denotes interactions in port connections($PC$), delayed port connections($DPC$), access connections($AC$) and subprogram calls($SC$).

Input of Cheddar is a tuple ***(Pr,Ad,Task,R,I)***, where ***Pr*** is a set of processors with mapped address space ***Ad***. A ***Task*** is a tuple $(type, cap, dl, p)$. $type \in \{period, aperiod, sporadic, customized\}$ defines the mechanism to activate a task. $cap$ denotes the execution time of the task. $dl$ and $p$ represent deadline and period. ***R*** refers to resource in the model. In this paper, we only consider data resource. ***I*** denotes connections between tasks.

Based on the configuration above, we present transformation rules formally, from an AADL model $M_A$ to a Cheddar input model $M_C$ as in Table I. In the first rule, an AADL processor is transformed to a Cheddar processor with a mapping address space. Scheduler of the Cheddar processor is set according to the $Scheduling\_Protocol$ property of AADL processor. Data components in AADL is mapped to resource of Cheddar. Begin and end time of the resource access is set at the dispatching and execution completion point of the corresponding AADL thread which owns the data. Thread and subprogram components, which are execution units in AADL, are converted to task in Cheddar. The capacity of a task is the sum of execution time of a thread and all its owning subprograms. For another advanced construct delayed port connection in AADL, it is cut into two adjacent connections. The cutting point is transformed as a task which executes between the execution and period end of its predecessor. In respect of other kind of interactions in AADL, the last rule is used to transform them into connections in Cheddar.

### B. Selection Engine

Input of our engine is a tuple $(WCRT, PU, CS, FL)$, which is also the output of Cheddar. $WCRT$ is average worst-case response time for all the tasks. $PU$ denotes the processor utilization. $CS$ represents the number of context switches and $FL$ is the end-to-end flow latency.

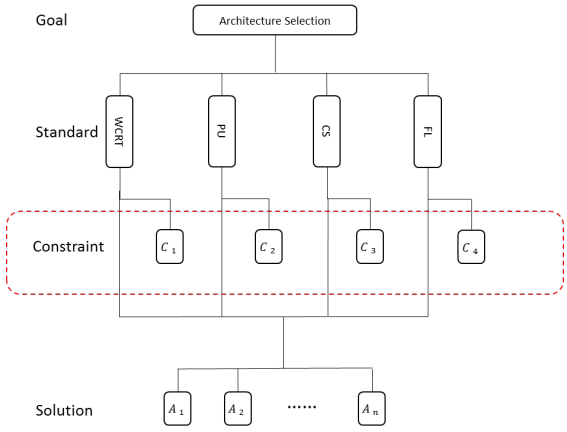



Fig. 2: Selection engine

Presented in Figure 2, the proposed selection engine is an extension of traditional AHP, which has 4 layers including **Goal**, **Standard**, **Constraint** and **Solution**. To accomplish the **Goal**, multiple objectives are defined in the **Standard** layer as $\{ST_{WCRT}, ST_{PU}, ST_{CS}, ST_{FL}\}$, referring to scheduling results. Candidates are listed in **Solution** layer. Traditional AHP uses an integer between 1 and 9 represent the comparative advantage. Although the method is convenient, it is incapable to fully capture the impact of application constraint violation. Thus, we propose the **Constraint** layer, which is a collection of constraints. Each constraint is a triple $(C, S, \Phi_{ij})$. $C$ denotes

TABLE I: The proposed transformation rules

| | |
|---|---|
| (Processor) | $\frac{M_C.Pr.Scheduler{=}M_A.Pro.Scheduling\_Protocol}{M_A.Pro{\rightarrow}(M_C.Pr,M_C.Ad)}$ |
| (Data) | $\frac{r.begin{=}thread.p{\wedge}r.end{=}thread.p{+}thread.c{\wedge}d{\subset}thread}{d{\in}M_A.D{\rightarrow}r{\in}M_C.R}$ |
| (Thread & Subprogram) | $\frac{task.type{=}thread.dp{\wedge}task.cap{=}thread.c{+}\sum M_A.Th.call.c{\wedge}task.dl{=}thread.d{\wedge}task.p{=}thread.p}{thread{\in}M_A.Th{\rightarrow}task{\in}M_C.Task}$ |
| (Delayed port connection) | $\frac{i_C^1.sink{=}i_C^2.source{\wedge}i_C^2.source.p{=}i_C^2.source.cap{=}i_A.source.p{-}i_A.source.c}{i_A{\in}M_A.Int.DPC{\rightarrow}i_C^1,i_C^2{\in}M_C.I}$ |
| (Interaction) | $\frac{i_A.source,i_A.sink{\in}M_A.Th{\wedge}i_C.source,i_C.sink{\in}M_C.Task}{i_A{\in}M_A.Int{\setminus}M_A.Int.DPC{\rightarrow}i_C{\in}M_C.I}$ |

the content. $S$ is the attached standard. $\Phi_{ij}$ is the satisfaction function with a **Solution** $i$ and a **Standard** $j$. $\phi_{ij}$ is defined as in (1).

$$\Phi_{ij} = \begin{cases} 1, & \text{i satisfies the constraint} \\ 1, & \text{j has no constraints} \\ c \in [0,1), & \text{i violates the constraint} \end{cases} \quad (1)$$

As in (1), the satisfaction function has a range from 0 to 1. The closer it is to 1, the lighter the hazard is in constraint violation. The major strength of the **Constraint** layer is the capability to reflect violations directly in the selection.

In selection, pairwise comparisons are used to compare two objects. For $n$ objects, a pairwise comparison $P = (a_{ij})_{n\times n}$ satisfies that $a_{ii} = 1$, $i < n$ and $a_{ij} = \frac{1}{a_{ji}}$, $i < n$, $j < n$. $a_{ij}$ is set in "1-9" scale as in [5]. In addition, comparisons of **Solution**s are built over each **Standard**. Overall five comparisons are constructed. According to [5], normalized maximal eigenvector of a pairwise comparison is defined as the weighted vector. Object with a larger weight has greater impact on architecture selection. Assuming the weighted vector of **Standard**s is $W_{standard} = [w_{ST_1}, w_{ST_2}, w_{ST_3}, w_{ST_4}]$. For $Solution$ with $m$ candidates, there are four $m \times m$ pairwise comparisons. The weighted vector of $i_{th}(i \leq m)$ matrix is $W^i_{solution} = [w^i_{SO_1}, w^i_{SO_2}...w^i_{SO_m}]$. Four weighted vectors of $Solution$ are assembled into a weighted matrix as following. $R = \{(W^i_{solution})^T \mid i = 1, 2, 3, 4\}$

The total weighted vector $W$ of our selection engine is defined as follows, representing scores of all the candidate architectures considering scheduling performance and application-specific constraints.

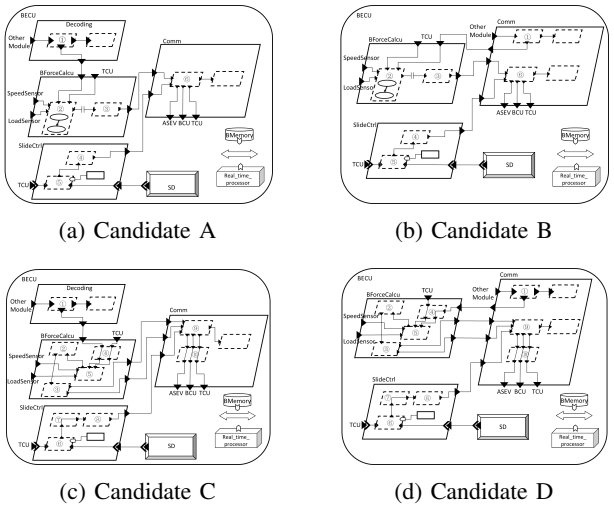
$$W = [\omega_1, \omega_2...\omega_m],\ \omega_i = \sum_{j=1}^{4} w_{st_j} R_{ij} \Phi_{ij},\ i = 1, 2...m$$

## IV. Case Study

The proposed approach is applied on a real industrial system, the Braking Electronically Control System(BECU). Based on railway applications, two constraints are:

1) Latency of braking flow must not exceed 1450$\mu s$.
2) $WCRT$ of the sending thread in BECU must not exceed 150$\mu s$.

As in Figure 3, four candidate architectures are modeled in graphical AADL [1]. A and B have centralized calculation threads with calls to subprograms and delayed connections, while C and D spread functions to distributed threads. Scheduling properties of candidates are listed in Table II. Through transformation, schedulability analysis is shown in Table II.



(a) Candidate A (b) Candidate B

(c) Candidate C (d) Candidate D

Fig. 3: Candidate architectures of BECU

TABLE II: Scheduling properties of candidates. P: period; ET: execution time; Unit:$\mu s$

| Thread | A | | B | | C | | D | |
|---|---|---|---|---|---|---|---|---|
| | P | ET | P | ET | P | ET | P | ET |
| 1 | 150 | 10 | 150 | 10 | 150 | 10 | 150 | 10 |
| 2 | 400 | 70 | 400 | 75 | 200 | 30 | 200 | 30 |
| 3 | 420 | 20 | 420 | 25 | 210 | 10 | 210 | 15 |
| 4 | 460 | 20 | 460 | 25 | 180 | 10 | 180 | 15 |
| 5 | 440 | 50 | 440 | 50 | 190 | 10 | 190 | 10 |
| 6 | 480 | 10 | 480 | 15 | 180 | 10 | 180 | 10 |
| 7 | - | - | - | - | 200 | 10 | 200 | 10 |
| 8 | - | - | - | - | 210 | 20 | 210 | 25 |
| 9 | - | - | - | - | 220 | 20 | 220 | 25 |
| 10 | - | - | - | - | 230 | 10 | 230 | 15 |
| Results of schedulability analysis | | | | | | | | |
| Candidate | $WCRT$ | | $PU$ | | $CS$ | | $FL$ | |
| A | 118.3 | | 46.7% | | 30 | | 1460 | |
| B | 130.0 | | 51.3% | | 30 | | 1465 | |
| C | 71.0 | | 70.8% | | 83 | | 1390 | |
| D | 81.0 | | 82.8% | | 91 | | 1395 | |

TABLE III: $Standard$ pairwise comparison for high-speed railway and subway applications

| Railway | $WCRT$ | $PU$ | $CS$ | $FL$ |
|---|---|---|---|---|
| $WCRT$ | 1 | 4 | 3 | $\frac{1}{3}$ |
| $PU$ | $\frac{1}{4}$ | 1 | $\frac{1}{2}$ | $\frac{1}{9}$ |
| $CS$ | $\frac{1}{3}$ | 2 | 1 | $\frac{1}{7}$ |
| $FL$ | 3 | 9 | 7 | 1 |
| Subway | $WCRT$ | $PU$ | $CS$ | $FL$ |
| $WCRT$ | 1 | $\frac{1}{2}$ | $\frac{1}{6}$ | $\frac{1}{2}$ |
| $PU$ | 2 | 1 | $\frac{1}{3}$ | 4 |
| $CS$ | 6 | 3 | 1 | 5 |
| $FL$ | 2 | $\frac{1}{4}$ | $\frac{1}{5}$ | 1 |

Targeting at high-speed railway and subway applications, $Standard$ pairwise comparisons are shown in Table III. $WCRT$ and $FL$ are more valued in railway while $PU$ and $CS$ are prioritized in subway. The selection is in Figure 4.

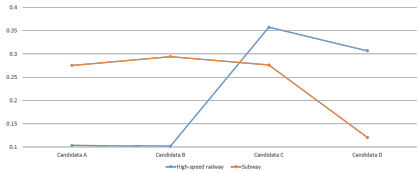



Fig. 4: Selection in high-speed railway & subway applications

In Figure 4, the y-axis denotes the weight calculated by our engine. The results display the impact of application-specific requirements on architecture selection. Specifically, C is selected as the best architecture for high-speed railway applications due to its short flow latency, while B with less context switches is selected out for subway applications.

We also compare the proposed engine with two selection approaches: a) general criteria selection [11], including modifiability, scalability, development effort and portability and b) traditional AHP with scheduling information in subway application domain. The comparison is exhibited in Figure 5.

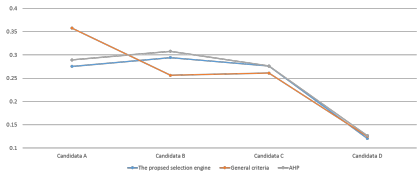



Fig. 5: Selection comparison to general criteria and AHP

Under general criteria, candidate A is selected out. However, the selection suffers from poor schedulability with longer $WCRT$ and flow latency than selection in high-speed railway applications. As an alternate, traditional AHP presents a similar result as the proposed engine. Nevertheless, without insights into application constraints, A is considered better than C in traditional AHP with the fact that A fails to meet constraints of subway domain.

TABLE IV: Sensitivity analysis on constraints

| Candidate | Tight constraints | | Loose constraints | |
|---|---|---|---|---|
| | Weight | Rank | Weight | Rank |
| A | 0.2754 | 3 | 0.2837 | 2 |
| B | 0.2942 | 1 | 0.2995 | 1 |
| C | 0.2762 | 2 | 0.2762 | 3 |
| D | 0.1206 | 4 | 0.1267 | 4 |

For subway applications, we loose the $WCRT$ constraint from $150\mu s$ to $180\mu s$. The comparison results are shown in Table IV. Comparatively, the relative merit between A and C reverses. To sum up, adjustment on application-specific constraints can be reflected through the proposed selection engine in a delicate manner.

## V. Conclusion

In this paper, we propose an enhanced transformation strategy on AADL models for schedulability analysis and a scheduling-based architecture selection engine for embedded systems. With the transformation strategy, complex AADL models including subprograms and delayed connections can be involved in schedulability analysis. With the proposed engine, both scheduling information and application-specific constraints are captured for selection. Our approach is applied on a real complex industrial system, BECU. Candidate architectures in AADL are analyzed for schedulability. The proposed selection engine then calculates a selection. In comparison with general criteria and traditional AHP, our engine excels at guaranteeing schedulability and meeting application-specific constraints. Furthermore, our engine shows delicate sensitivity to modification on constraints. In the future, we plan to apply this approach on more large-scale systems.

## Acknowledgment

This research is sponsored in part by NSFC Program (No. 61202010, 91218302), National Key Technologies R&D Program (No.SQ2012BAJY4052), 973 Program (No.2010CB 328003) of China and Tsinghua University Initiative Scientific Research Program(20131089331).

## References

[1] "Sae as5506a: Architecture analysis and design language(aadl)," 2009.

[2] F. Singhoff, J. Legrand, L. Nana, and L. Marce, "Cheddar: a flexible real time scheduling framework," in *2004 annual ACM SIGAda international conference on Ada*. ACM, 2004, pp. 1–8.

[3] L. Liu and W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM(JACM)*, vol. 20, no. 1, pp. 46–61, 1973.

[4] A. Johnsen, K. Lundqvist, P. Pettersson, and O. Jaradat, "Automated verification of aadl-specifications using uppaal," in *2012 IEEE 14th International Symposium on High-Assurance Systems Engineering (HASE)*. IEEE, 2012, pp. 130–138.

[5] T. Saaty, "The analytical hierarchy process," 1980.

[6] P. Wallin, J. Froberg, and J. Axelsson, "Making decisions in integration of automotive software and electronics: A method based on atam and ahp," in *Fourth International Workshop on Software Engineering for Automotive Systems*. IEEE, 2007.

[7] P. Garg, A. Gupta, and W. Rozenblit, "Performance analysis of embedded systems in the virtual component co-design environment," in *11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*. IEEE, 2004, pp. 61–68.

[8] A. Aleti, S. Bjornander, L. Grunske, and I. Meedeniya, "Archeopterix: An extendable tool for architecture optimization of aadl models," in *ICSE Workshop on Model-Based Methodologies for Pervasive and Embedded Software*. IEEE, 2009.

[9] M. Razavi, F. Aliee, and K. Badie, "An ahp-based approach toward enterprise architecture analysis based on enterprise architecture quality attributes," *Knowledge and Information Systems*, vol. 28, no. 2, pp. 449–472, 2011.

[10] D. Babu, P. Govindarajulu, R. Reddy, and A. Kumari, "An integrated approach of ahp-gp and visualization for selection of software architecture: A framework," in *2010 International Conference on Advances in Computer Engineering*. IEEE, 2010, pp. 334–338.

[11] L. Zhu, A. Aurum, I. Gorton, and R. Jeffery, "Tradeoff and sensitivity analysis in software architecture evaluation using analytic hierarchy process," *Software Quality Control*, vol. 13, no. 4, pp. 357–375, 2005.

[12] J. Lee, S. Kang, and C.-K. Kim, "Software architecture evaluation methods based on cost benefit analysis and quantitative decision making," *Empirical Software Engineering*, vol. 14, no. 4, pp. 453–475, 2009.