Model-Based Evaluation Of System Scalability: Bandwidth Analysis For Smartphone-Based Biosensing Applications

Patou, François; Madsen, Jan; Dimaki, Maria; Svendsen, Winnie Edith

Published in:
Proceedings of Euromicro Conference on Digital System Design 2016

Link to article, DOI:
10.1109/DSD.2016.37

Publication date:
2016

Document Version
Peer reviewed version

Model-Based Evaluation Of System Scalability: Bandwidth Analysis For Smartphone-Based Biosensing Applications

François Patou,
Maria Dimaki, Winnie E. Svendsen
dept. of Micro- and Nanotechology
Tech. Univ. of Denmark, DTU, Lyngby, Denmark
email: frpato@nanotech.dtu.dk

Jan Madsen
dept. of Applied Mathematics
and Computer Science
Tech. Univ. of Denmark, DTU, Lyngby, Denmark
email: jama@dtu.dk

***Abstract*—Scalability is a design principle often valued for the engineering of complex systems. Scalability is the ability of a system to change the current value of one of its specification parameters. Although targeted frameworks are available for the evaluation of scalability for specific digital systems, methodologies enabling scalability analysis of multi-domain, complex systems, are still missing. In acknowledgment of the importance for complex systems to present the ability to change or evolve, we present in this work a system-level model-based methodology allowing the multidisciplinary parametric evaluation of scalability. Our approach can be used to determine how a set of limited changes to targeted system modules could affect design specifications of interest. It can also help predict and trace system bottlenecks over several product generations, offering system designers the chance to to better plan re-engineering efforts for scaling a system specification efficaciously.**

We demonstrate the value of our methodology by investigating a smartphone-based biosensing instrumentation platform. Specifically, we carry out scalability analysis for the system's bandwidth specification: the maximum analog voltage waveform excitation frequency the system *could* output while allowing continuous acquisition and wireless streaming of bioimpedance measurements. We rely on several SysML modelling tools, including dependency matrices, as well as a fault-detection Simulink Stateflow executable model to conclude on how the successive re-engineering of 5 independent system modules, from the replacement of a wireless Bluetooth interface, to the revision of the ADC sample-and-hold operation could help increase system bandwidth.

***Keywords*-Scalability; Model-Based Development; SysML; Dependency Matrix; Smartphone;**

I. Introduction

Scalability is often considered an important design principle to abide by for the engineering of complex systems. Together with several other non-functional *-ilities* of systems engineering, scalability is considered by Ross et al. [1] as one of the key factors in providing a complex (e.g cyber-physical) system with the ability to sustain its value in a changing environment. From their extensive semantical and ontological research, Ross et al. and De Weck et al. [1], [2] argued that scalability could be defined as *the ability of a system to change the current value of one of its specification parameters*.

In recent work, we argued that the field of Point-of-Care in-vitro medical diagnostics presented several of the characteristics that justify design for evolvability, a key constituent of which is scalability [3]. We present, in this paper, a model-based methodology for evaluating parametric scalability in complex systems. We illustrate our methodological approach with a case-study: we investigate how the bandwidth of a smartphone-based biosensing instrumentation platform could be scaled up from the successive re-engineering of relevant functionally and structurally independent modules. We define bandwidth for this work as *the maximum voltage waveform excitation frequency for which the system can operate continuously while streaming data to the mobile-software layer*.

II. Methodology

Our methodological approach requires:

1 - The identification of all the *design variables* influencing the specification of interest. Model-Based Design (MBD) can support that identification with an increasing level of refinement, directly related to that of the model itself. Past this identification step, our methodology necessitates:

2- To *identify and trace the dependencies* existing between all identified design variables. This identification process is necessary in order to *aggregate design variables that show dependencies with one another*. Aggregates of inter-dependent variables should be defined so as to be independent from one another. The next step of our methodology then requires:

3- To assess the independence of the functions, structures, or operations implementing these design variable aggregates. This step involves *identifying the various system modules* (e.g. hardware, software, mixed, etc.) responsible for the design specification under consideration. The scalability of the design specification determined by these independent modules can be then be quantitatively evaluated and demands:

4- To *successively replace each independent system module* in the model *by a module that is not limiting for*

scaling the design specification. This step is equivalent to successively nulling the influence of each design variable aggregate defined at the previous step. We illustrate how to apply this methodology with a case-study.

III. Case-study: Smartphone-based continuous recovery of bioimpedance signals

A. System overview and SysML model

We consider a smartphone-based platform, introduced in [3] and further elaborated upon in [4]. In an attempt to address the challenges specific to changing contexts and requirements discussed by Fricke et al. [5], the system was designed so as to favour system evolution. Among available functions, it enables the *continuous* recovery of low AC-current signals, such as those exhibited by high-impedance silicon nanowire biological field-effect transistors, using digital lock-in amplification [6]. We analytically identified the design variables involved in bandwidth specification (as defined in section I). We relied on the OMG SysML language to define the system functions, structures and behaviors at a level of abstraction that would allow the definition of parametric relationships between the identified design variables.

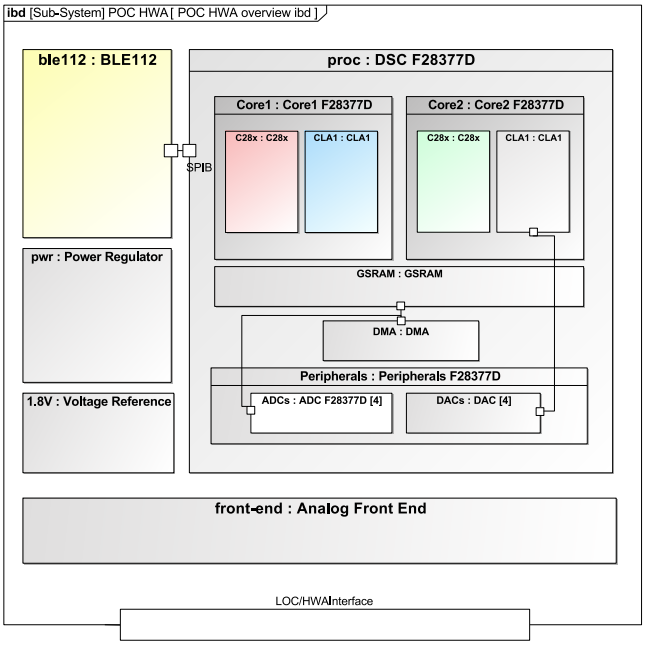

Figure 1. Smartphone hardware accessory architetural overview. The accessory is mainly constituted of a Digital Signal Controller (F28377D DSC) embedding all necessary peripherals to carry out both voltage waveform generation and synchronous sampling. The DSC is interfaced to a Bluetooth Low Energy module (BLE112) via SPI. The analog front end comprehends current amplifiiers. Phase-Sensitive Detection (PSD) is performed digitally in the first core (Core1).

The system overall architecture is given in figure 1. Hardware and embedded software were designed so as to allow lock-in amplification acquisitions to be performed continuously and streamed to the iOS mobile-software layer. The implementation of continuous lock-in acquisitions necessitated the parallelization of the DAC waveform generation, ADC sampling and DSP routines. The latter comprehend a two-stage Phase-Sensitive Detection (PSD Stage 1 and PSD Stage 2) algorithm itself composed of signal mixing and FIR low-pass filtering and decimation [6].

We completed the behavioral model of the system with state-machines for all subsystems influencing the bandwidth specification. A high-level representation of such model is given in figure 2, and illustrates the operation of DSC Core1: while DAC and ADC control is allocated to Core2, Core1's main CPU (C28x) is in charge of PSD Stage1 while its hardware accelerator (CLA) can concurrently execute PSD stage2. Once PSD Stage 2 is completed, Core1's C28x processor transfers the processed data to the BLE module while Core2 is still continuously exciting/sampling. This scheme is made possible via the use of duplicated acquisition buffers, alternating roles: target of the acquisition process for one, source to the DSP and BLE transmission for the other.

Design variables influencing system bandwidth were identified by the electrical, embedded-software, and mobile-software designer. They were then informed in the various diagrams of the SysML model and are summarized in the dependency matrix given in figure 3 (step 1). As mentioned in section II, these dependencies can be used to define *aggregates* of design variables showing interdependencies and their associated system functions, structures or operations (step 2). If an arrow displays a dependency between two design variables in a matrix cell, then the corresponding row and column design variables are aggregated together. These aggregates correspond in figure 3 to the groups of design variables with overlays of the same color. The next step of our methodology (step 3) demanded us to identify independent system modules (functional, structural or operational) *specifying or encompassing* the independent design variables aggregates from the previous step: The final step of our methodology consists in successively replacing these independent modules by ones that are *not* bandwidth limiting. This process was achieved using a state-machine-based fault-detection simulation model and is explicited further in the next section.

B. Model-based recovery of bandwidth specification

We carried out the last step of our methodology by relying on a refined state-machine model enabling fault-detection. Part of this model is presented in figure 2. Faulty states were specified so as to be able to identify the independent module responsible for bandwidth limitation for a given trial excitation frequency. For instance, triggering the $\mathrm{StartDSPStage2}$ event while the CLA hardware accelerator is still processing samples (i.e. is in the DSP Stage 2 state) is a faulty operation that would result in the input data for PSD Stage 2 being

wrongly overwritten. Such event trigger is thus redirected in the model to the bwLimitedByPSDStage2Duration faulty state (one the red states in figure 2), pinpointing PSD Stage 2 as the responsible module for bandwidth limitation, since we asserted in the previous steps of the methodology that the design variables associated to bandwidth specification in DSP Stage 2 were independent from the other modules. Executing a simulation for a specific voltage waveform excitation frequency *above* our system bandwidth would thus bring the state-machine to a given faulty-state, whereas a simulation at an excitation frequency *below* the bandwidth would run continuously, recovering the impedance measurement in the mobile software layer model independently of the acquisition duration. We considered that the simulation were fault-free if the model could successfully fill at least two acquisition buffers while transmitting and recovering data in the mobile-software layer model.

In order to recover the bandwidth of our initial system via simulation, we derived an iterative dichotomous algorithm (figure 4) made responsible for attempting to simulate the state-machine model at excitation frequencies redefined iteratively in the following maner: The initial condition for running this algorithm was to set the initial excitation frequency to an arbitrary value, in this case 100 kHz. Should this first simulation be successful, this frequency would the set as the lower bound for the minimum system bandwidth at the next iteration where the trial frequency would be doubled. If instead the simulation ended in a faulty state, then the excitation frequency would be set as an upper bound for maximum bandwidth and the next trial frequency would be halved. By repeating this scheme iteratively, the algorithm converges towards the system's bandwidth value within a given tolerance. Given the possibility of a large number of iterations and related computational load, we opted for replicating our state-machine model in MathWorks Simulink StateFlow and implemented our algorithm in Matlab.

C. System scalability over generations

Our initial system i.e. *Gen1* on figure 5 successfully passed all simulations at excitation frequencies below 3.1624Hz. The first faulty-state in which the model would end up beyond this value was that associated with a trigger event received for initiating the PSD routines while DSC Core1 was still transferring data to the BLE module. As the simulation enters in the faulty-state it is aborted, thereby pinpointing the BLE module to be the bottleneck for *Gen1*'s bandwidth. We determined in step 3 and 4 that three design variables influencing system bandwidth have been confined to the BLE system communication module, namely spi8BitWordTransferIntervalDuration, attributeNotificationUpdateInterval, and bleSPIWordProcessingTime. The first of these variables is dependent on the last (BLE-DSC communication software dependencies). attributeNotificationUpdateInterval does not show dependencies with the other two but also influences the system bandwidth. It is a property of the BLE protocol specification.

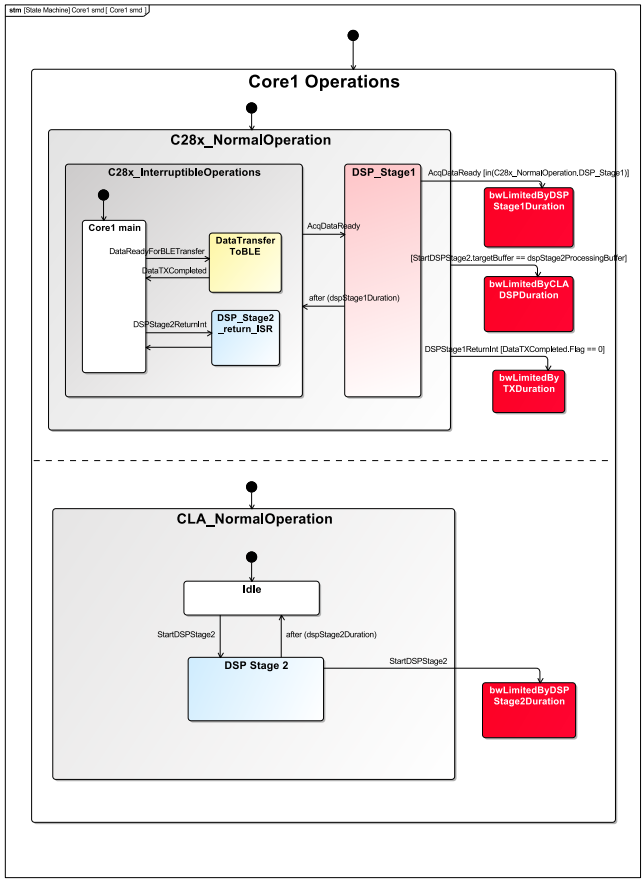

Figure 2. The value of the successive system's generations bandwidth specification is retrieved by simulation of the system's state machines and the recovery of faulty executions. Entry to any of the predefined faulty states (red states in the figure) translates a fault in execution and effectively stops the simulation. More specifically, simulations at various AC excitation frequencies are carried out by dichotomy: If a simulation is faulty, then the excitation frequency is set as an upper bound for the minimum system bandwidth. If the simulation is successful, it is set as a lower bound for the maximum system bandwidth. Simulations are repeated until an excitation frequency within the specified tolerance interval outputs a successful simulation.

We were able to shunt the influence of the BLE module on bandwidth by setting these three design variables to infinitesimals. This, in turn, helped us identify which system module would be the bottleneck for bandwidth in *Gen2*: the evolved system corresponding to *Gen1* but *replacing the BLE module and its relevant design variables* by non-limiting implementations. The execution of our iterative dichotomous algorithm now converged towards 27.498Hz, this time limited in bandwidth by the PSD Stage2 hardware-software module. By using the same reasoning we could investigate which modules would be successively limiting system bandwidth, each time circumventing the bottleneck

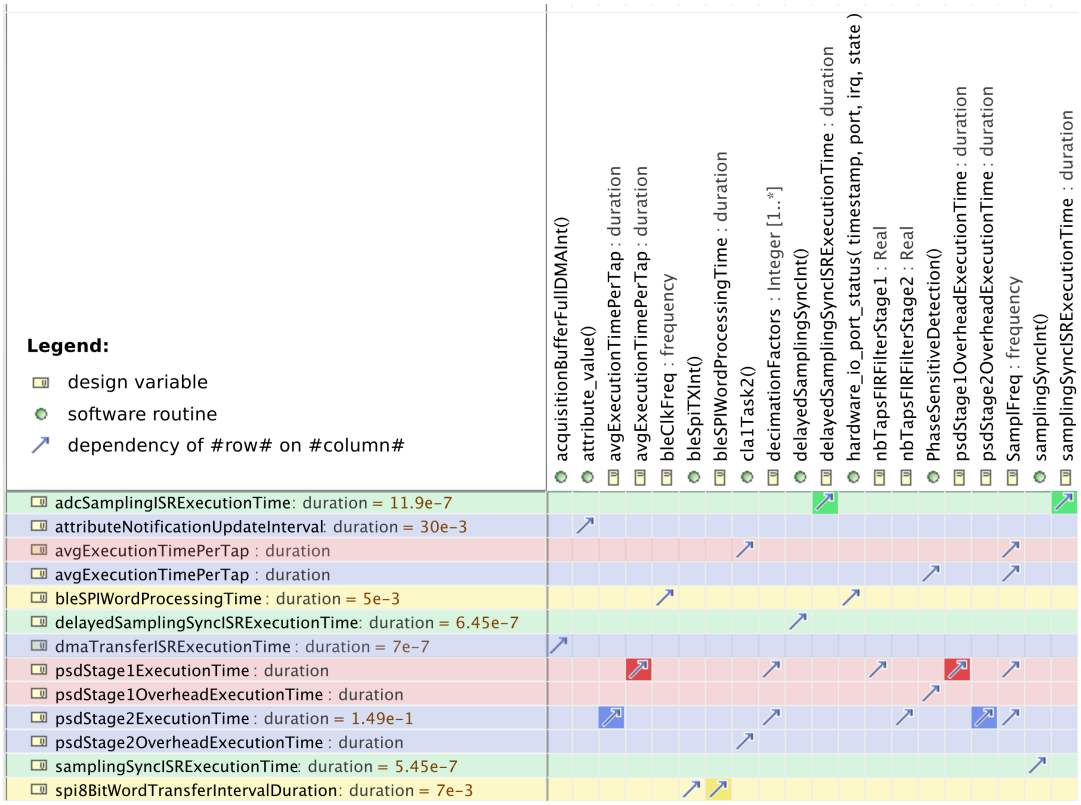

Figure 3. Dependency matrix presenting dependencies between all the design variables influencing the specification of interest: the system bandwidth. The same set of design variables fill both the rows and columns of the matrix. The dependencies are read from row to column: arrows for a given matrix row represents a dependency *of* the design variable corresponding to the arrow's cell row *on* the design variable in the arrow's cell column. Design variables presenting a dependency relation are aggregated together. These aggregates are represented with overlays of the same color

of the previous system generation. The corresponding successive systems generations could thus be determined and consist of:

- *Generation 1*: Original system such as depicted in figure 1.
- *Generation 2*: System evolved so that the BLE communication module (i.e. the yellow-overlay design variable aggregate + the attributeNotificationUpdateInterval variables) is not limiting to system bandwidth: the duration design variables were set to infinitesimals.
- *Generation 3*: Generation 2 evolved so that stage 2 PSD is not limiting to bandwidth (i.e. the blue-overlay design variables aggregate): stage2PSDExecutionTime set to an infinitesimal.
- *Generation 4*: Generation 3 evolved so that stage 1 PSD is not limiting to bandwidth (i.e. the red-overlay design variables aggregate): stage1PSDExecutionTime set to an infinitesimal.
- *Generation 5*: Generation 4 evolved so that the ADC sample and hold duration is not limiting (i.e. the green-overlay design variables aggregate): adcSamplingISRExecutionTime set to an infinitesimal
- *Generation 6*: Generation 5 evolved so that the independent dmaTransferIsrExecutionTime variable is not limiting: set to an infinitesimal.

IV. Discussion

The presented 4-steps methodology provides a way to evaluate the scalability of complex systems by leveraging model-based design and analysis. The successive identification of design variables, the assessment of their interdependencies, and the determination of the independent system modules engulfing these variables are prerequisites for being able to consider how *successive re-engineering of each of these modules* will affect the design specification of interest. Our systems engineering approach enables scalability assessment to be carried out for complex (e.g. cyber-physical) systems where specifications are likely to arise from design variables spanning over various engineering disciplines and where system modules may comprehend anything from a mechanical add-on to a software toolbox.

Our methodology is valuable in that it can not only help identifying system bottlenecks, but it also provides a means to evaluate the focus and scope of the re-engineering effort.

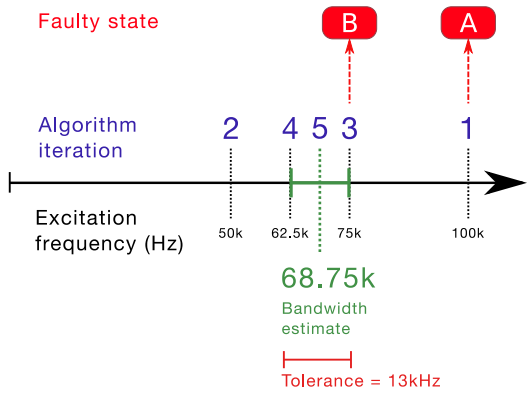

Figure 4. Principle of the iterative dichotomous algorithm for the estimation of system bandwidth. An initial condition sets the original excitation frequency to an arbitrary value, in this case 100 kHz. The state-machine simulation ends in faulty state A. 100kHz is thus set as an upper bound for the maximum system bandwidth and the next trial frequency is halved to 50kHz. At this frequency the simulation runs successfuly setting 50kHz as the lower bound for the minimum system bandwidth. By repeating this scheme iteratively, the algorithm converges towards the system's bandwidth value: it ends with a successful run at frequency 68.75kHz with a lower bound at 62.5kHz and upper bound of 75kHz. The difference between which is 12.5kHz, inferior to the maximum uncertainty tolerance set for the bandwidth estimate.

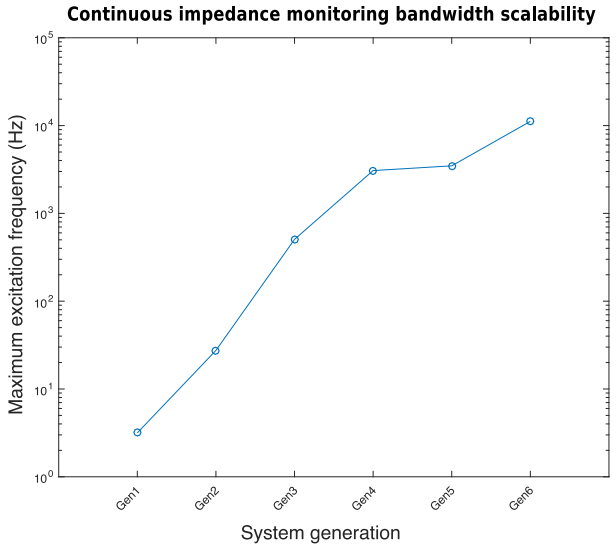

Figure 5. Successive systems are evolved so that the bottleneck module identified of a given generation is replaced by a non-limiting module in the next generation.

The analysis of dependencies helps certify that the benefit of scaling a specification by changing a system module will not necessitate the re-engineering of other modules dependent on the former in any way.

Our strategy of iterative system evolution, circumventing bottleneck system modules with non-limiting implementations enables to predict which unchanged module will become the new specification bottleneck in the evolved system.

V. Conclusion

We proposed a model-based methological approach for the evaluation of system scalability and demonstrated its utility with a cyber-physical biomedical system application. Our endeavor addresses some of the shortcomings associated with the assessment of change-related *-ilities* in the engineering of complex-systems. It finally acknowledges that engineering *change* has become predominant over design from blank-canvas when dealing with complex systems, and that system-wide tools and methodologies for easing the evaluation of change-related *-ilities* are still needed.

Acknowledgment

This project is a part of the EU Marie Curie Initial Training Networks (ITN) Biomedical engineering for cancer and brain disease diagnosis and therapy development: EngCaBra. Project no. PITN-GA-2010-264417.

References

[1] A. M. Ross, D. H. Rhodes, and D. E. Hastings, "Defining changeability: Reconciling flexibility, adaptability, scalability, modifiability, and robustness for maintaining system lifecycle value," *Systems Engineering*, vol. 11, no. 3, pp. 246–262, 2008.

[2] O. L. De Weck, A. M. Ross, and D. H. Rhodes, "Investigating Relationships and Semantic Sets amongst System Lifecycle Properties (Ilities)," *Third International Engineering Systems Symposium CESUN 2012, Delft University of Technology, 18-20 June 2012*, no. June, pp. 18–20, 2012.

[3] F. Patou, M. Dimaki, W. E. Svendsen, K. Kjaegaard, and J. Madsen, "A Smart Mobile Lab-on-Chip-Based Medical Diagnostics System Architecture Designed for Evolvability," *2015 Euromicro Conference on Digital System Design*, pp. 390–398, 2015. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7302301

[4] F. Patou, M. Dimaki, W. E. Svendsen, C. Kjægaard, and J. Madsen, "Smartphone-based biosensing platform evolution : implementation of electrochemical analysis capabilities ." in *International Symposium on Medical Communication and Information Technologies*, 2016, p. [In Press].

[5] E. Fricke and A. P. Schulz, "Design for changeability (DfC): Principles to enable changes in systems throughout their entire lifecycle," *Systems Engineering*, vol. 8, no. 4, pp. 342–359, 2005.

[6] J. Leis, C. Kelly, and D. Buttsworth, "Sampling , Quantization and Computational Aspects of the Quadrature Lock-In Amplifier," in *Signal Processing and Communication Systems (ICSPCS), 2012 6th International Conference on*, 2012, pp. 1–7.