A Metrics System for Quantifying Operational Coupling in Embedded Computer Control Systems

DeJiu Chen, Martin Törngren
Division of Mechatronics, Department of Machine Design, Royal Institute of Technology
SE-100 44 Stockholm, Sweden
+46-8-7906000
chen@md.kth.se, martin@md.kth.se

ABSTRACT
One central issue in system structuring and quality prediction is the interdependencies of system modules. This paper proposes a novel technique for determining the operational coupling in embedded computer control systems. It allows us to quantify dependencies between modules, formed by different kinds of relationships in a solution, and therefore promotes a more systematic approach to the reasoning about modularity. Compared to other existing coupling metrics, which are often implementation-technology specific such as confining to the inheritance and method invocation relationships in OO software, this metrics system considers both communication and synchronization and can be applied throughout system design. The metrics system has two parts. The first part supports a measurement of coupling by considering individual relationship types separately. The quantification is performed by considering the topology of connections, as well as the multiplicity, replication, frequency, and accuracy of component properties that appear in a relationship. The second part provides a methodology for combining coupling by individual relationship types into an overall coupling, where domain specific heuristics and technology constraints are used to determine the weighting.

Categories and Subject Descriptors: D.2.m [**Software Engineering**]: Miscellaneous—Embedded computer control; D.2.8 [**Software Engineering**]: Metrics; K.6.4 [**System Management**]: Quality assurance; D.2.11 [**Software Architecture**]: Domain-specific architectures

General Terms: Measurement, Design, Verification

Keywords
System Functions, Coupling Measure, Modularization and Components.

1. INTRODUCTION
Embedded computer control systems (ECS) are computer-based systems for advanced control, diagnostics, and monitoring in machinery [1]. Typical application areas include vehicles, avionics, and robotics. Because of the dynamics under control, such systems differ from other general-purpose computer systems in the aspects of real-time and safety criticality.

In ECS, computer software and hardware together constitute the physical embodiments of system functions for the purpose of system realization. Issues that are of particular concern in the software system design include: identification and classification of requirements and constraints, definition of components and relationships, assessment of feasibility and product qualities, and trade-offs between solution alternatives. Normally, the design is performed at different refinement levels, repeating a means-end pattern in the sense that an allocation of requirements to solutions will in turn derive some new requirements for the underling solutions (e.g., a hierarchy of mapping from end-to-end timing of control loops to code level timing) [2]. From a system engineering point of view, it is the design in the large, targeting pre-code artifacts at levels higher than detailed implementation, that has the key impacts on product qualities, complexity control, costs and time-to-market, see e.g., [3][4][5]. For example, [5] states that the front-end design in general determines 80% of system cost with only 20% of total product development costs spent. For ECS, there is currently a paradigm shift from a "traditional" software design that focuses on coding and testing to a model-based software design that emphasizes quality assessment and optimization using system descriptions at levels higher than code details, see e.g., [28].

One important property of complex ECS, besides the mandatory functionality, real-time timeliness, and dependability, is modularity, indicating the extent to which a system is decomposed and classified into parts. This system property forms a key factor in complexity control, concurrent engineering and product flexibilities (including reusability and modifiability), see e.g., [6][7]. As a rule-of-thumb, software systems with good modularity should exhibit low coupling and high cohesion, where the coupling indicates the strength of interconnections (i.e., the "wiring") between modules and the cohesion shows the strength of holding a module together (i.e., the "glue") [8][9].

For complex ECS, there is a need to support objective and repeatable measurement of coupling throughout the design at different levels of design refinement. Given the importance of design in the large, the designers must be able to verify high-level (e.g., architectural) solutions and perform tradeoffs between solution alternatives by taking modularity into consideration. However, little support exists in this area. Existing approaches to quantitative coupling measurement mainly target detailed software design and consider only program specific dependencies

EMSOFT'04, September 27–29, 2004, Pisa, Italy.

(e.g., calls). Due to their restrictions to implementations details, such approaches have very limited usability in the system level design.

This paper proposes a technique for measuring the operational coupling in ECS, hence providing information for quantitative assessment of modularity in the system design. The metrics system transforms various communication and synchronization relationships, derived from a system meta-model of ECS, into quantitative coupling measures by considering parameters such as frequency and accuracy that affect the strength of dependencies.

This paper includes 7 major sections. In the next section, we discuss the aim of this work and the solution strategy. Section 3 introduces the system model and concepts underlying the metrics system. Section 4 presents the metrics system that supports the measurement by considering individual relationships separately. Section 5 describes a methodology for combining these individual relationships. Finally, Section 6 describes related work and Section 7 concludes and discusses further work.

2. AIM AND APPROACH

The ultimate goal of our work is to provide an effective means of quantifying modularity, and hence a more complete engineering basis for model-based design and optimization of ECS. We consider coupling as a measure of dependency among system parts, established by the relationships connecting these parts in a system solution. See Figure 1.

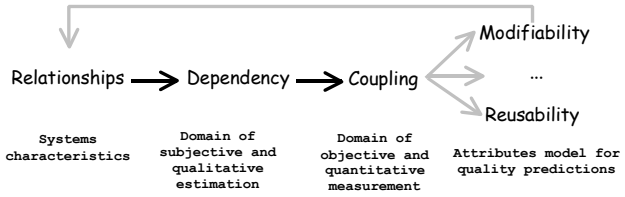


Figure 1. Central terms and their relations.

For this purpose, it is necessary to define the system parts, which can be composed to form modules, and the relationships between these parts, each of which provides information about a kind of dependency. Thereafter, a transformation from dependency to quantitative coupling measures by identifying and combining parameters characterizing the strength is necessary. The metrics system focuses on the operational relationships that are established by communication, synchronization, and implementation. So our approach includes three steps: (1) providing a meta-model that articulates and formalizes various system features of concern into system parameters; (2) elaborating and classifying relevant operational relationships; (3) defining a metrics system that transforms operational dependencies into quantitative coupling figures. This paper focuses on the last step. A detailed description of the meta-model and classification of relationships is given in [10]. The system model refines our previous work in modeling [11][12][13].

Figure 2 depicts the steps of our approach. One of the major points here is that the measurement should support earlier phases of software design, using information from models independent of implementation and technology details. The design in the large can therefore be evaluated for modularity and optimized by tradeoffs with respect to multiple quality attributes. See also Figure 3. A module is a system function that has a single implementation and its own lifecycle, and hence can be modified

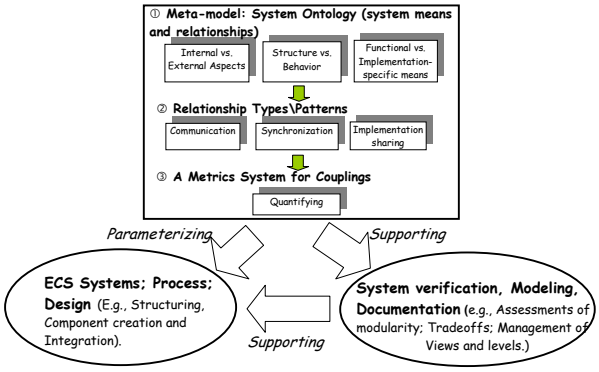


Figure 2. Context and major steps of our work.

and replaced independently.

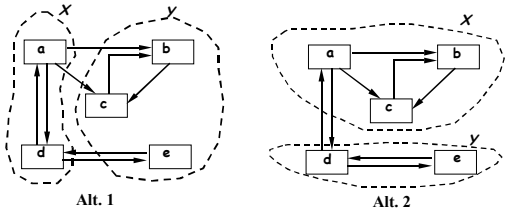


Figure 3. A conceptual illustration of structuring for modularity. Which alternative has a tighter coupling and which parameters affect this assessment?

3. PRELIMINARIES

3.1. Systems model

A system is a synergetic integration of "things" for certain purposes. See e.g., [3][4][5]. We refer to such things as system means. Figure 4 depicts system aspects that are of particular concern in the solution domain of ECS. Each system has a boundary. The internal aspect is concerned with means inside the boundary (e.g., functions, objects or processors) and how they are consolidated into a whole. The external aspect is concerned with means constituting the system context, such as environmental objects, physical conditions (e.g., friction), or restrictions (e.g., operational ranges and forbidden scenarios). From a design-oriented view, a system is discerned along two orthogonal dimensions: content and level. The content consists of structure and behavior, representing "what a system is" and "what the system does" respectively [4][16]. The levels represent the degrees of preciseness and detail of a design with respect to the final realization. There are two major domains of such levels: functional levels and implementation levels. See e.g., [3][5]. At

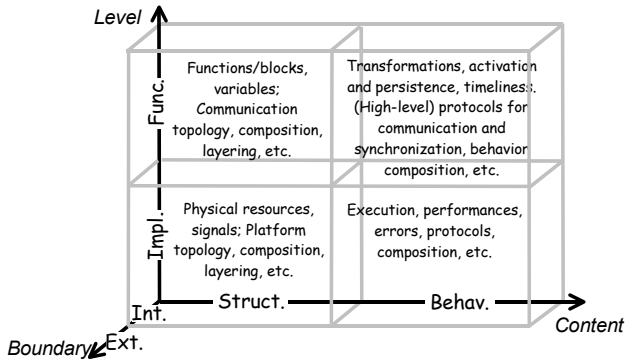


Figure 4. An illustration of system aspects.

functional levels, the structures and behaviors, as well as the internal and external means, are considered in terms of abstractions independent of implementation details. We refer to a system solution at functional levels as a "functional solution". For the system realization, a functional solution is embodied in an "implementation solution", consisting of software programs and electronics hardware.

In our system model [10], a system solution is defined as follows.

DEFINITION 1 (System solutions). *The solutions of a system* ***Sol*** *is a pair* ***Sol***=(***Means***, ***Rel***)*, where*

- ***Means*** *represents the set of things that constitute a system.* ***Means***$=\{Means_1,\ldots, Means_n\}$, $n\in Z^+$.
- ***Rel*** *represents the relationships between system means. It is a subset of binary relations on* ***Means***: ***Rel*** $\subseteq$ ***Means***$\times$ ***RelTyp***$\times$***Means***, *where* ***RelTyp*** *is a finite set of vocabulary representing the types of relationship.*

The relationship types ***RelTyp*** consists of {*COMP, COMM, SYNC, IMPL, SHAR, REFIN, CRIT, REPL*}, denoting whole-parts composition, communication, synchronization (execution ordering and timing), implementation (mapping functions to application software and the system platform), sharing of common features or resources, design refinements, system safety specific relationships, and replication. Accordingly, the triple (*m, RelTyp, n*) denotes a relationship of means *m* with respect to *n*, and ***Rel(M)*** denotes the set of triples representing the relationships of a set of means ***M***. We refer to the functional means as "system functions" (denoted by the set, ***Func*****)**

A relationship between system functions is established by connecting one property of a system function to a compatible property of another system function or an environmental means. Table 1 summarizes the properties of system functions. Some of these properties target other properties. For example, the error property can be applied to any other property such as IO and timing features. We also distinguish properties according to their "direction". For a system function, an "incoming property" is a feature determined by other means, such as a data input. An "outgoing property" is a feature determined by the system function for other means or the entire system, such as a data output. Properties not belonging to these two categories are considered inherent, such as an internal variable.

By means of relationships, the properties of individual system functions are combined, resulting in properties of a system solution as a whole. We refer to the communication, synchronization, as well as implementation-sharing as "operational relationships" since such relationships are concerned with the system behavior. For system functions, we have identified 26 types of operational relationships, summarized in Table 2 and formally defined in [10]. The implementation sharing relationships introduce additional dependencies when a functional solution is partitioned and allocated. In their definitions, the concept of implementation parameters describe artifacts such as logical communication channels and operating system services.

DEFINITION 2 (Operational relationships and interactions). *For a system function* $f\in$***Func***, *let* ***OpRel****(f) be its operational relationships*

$$\textbf{\textit{OpRel}}(f)=\textbf{\textit{Interact}}(f)\cup\textbf{\textit{SImplPRel}}(f)\cup\textbf{\textit{SImplMRel}}(f)\cup\textbf{\textit{DImplMRel}}(f)$$

Where: ***Interact****(f) is the interactions.* ***Interact****(f)*=***CommRel****(f)* $\cup$ ***SyncRel****(f)*, and

- ***CommRel****(f)* = {***ICRel****(f), **IDRel****(f), **IDCRel****(f), **ORel****(f), **OERel****(f), **IDERel****(f), **ICERel****(f), **IDCERel****(f), **IDLRel****(f), **ICLRel****(f), **IDCLRel****(f), **IDELRel****(f), **ICELRel****(f), **IDCELRel****(f), **IDLERel****(f), **ICLERel****(f), **IDCLERel****(f)*}
- ***SyncRel****(f)* = {***RBehRel****(f), **PBehRel****(f), **RBehERel****(f), **PBehERel****(f), **TimRel****(f), **TimERel****(f)*}

Note that these relationships are defined from a single system function point of view. Each relationship constitutes an agreement or a contract that a system function has within a system solution by specifying the features that the system function depends upon or is obligated to produce. E.g., the data input communication relationships of a system function *f* are: ***IDRel****(f)*={(f, r, e)$\in$***InRel****(f)* $|$ $f\in$***Func*** $\wedge$ r=*COMM* $\wedge$ $e\in$***Func***$\backslash\{f\}$ $\wedge$ ($\exists p\in$***O****(e)*: $\exists q\in$***ID****(f)*: $p\Rightarrow q$)}, where: ***InRel*** **–** incoming relationships, ***O*** **–** communication output, ***ID*** **–** communication data input.

Table 1. An overview of system functions' properties.

Property	Definition
Form	Features concerning structure (e.g., layering, topology).
IO	Communication inputs and outputs (e.g., data).
Trans	Functional transformations, each of which consists of a domain, a range, and a transformation rule.
Mode	Application specific mode logic consisting of states and transitions.
Var	Variables that appear in transformations and modes.
Exe	Executional behavior features of other properties (e.g., ***IO***, ***Trans***, ***Mode,*** and ***Var***) such as in terms of triggering, execution modes, persistence, and computation-models.
Tim	Timing of other properties (e.g., ***Exe***, ***IO***, ***Trans***, and ***Mode***) such as in terms of triggering frequency, time stamp, delay, and deadline.
Err	Features concerning erroneous conditions of other properties (e.g., ***Tim***, ***Exe***, and ***IO***,)

Table 2. Operational relationships of system functions.

Relationships	Definitions
ORel, ***OERel***	Communication relationships by outputs (O) to other system functions or environmental means (E).
IDRel, ***ICRel***, ***IDCRel***	Communication relationships by inputs (I) of: 1. Data (D), i.e., variables in domains of functional transformations; 2. Control (C), i.e., variables used as conditions for transitions between application specific modes; 3. Hybrid data& control (DC), i.e., variables that are involved in both functional transformations and mode transitions, from other system functions
IDERel, ***ICERel***, ***IDCERel***	Communication relationships by inputs (I) of: 1. Data (D); 2. Control (C); 3. Hybrid data&control (DC), from environmental means (E).
IDLRel, ***ICLRel***, ***IDCLRel***	Communication relationships by inputs (I) of: 1. Data (D); 2. Control (C); 3. Hybrid data&control (DC), from other system functions and the inputs are looped (L) (i.e., which can be traced back to the output(s) of the same system function by communications of some system functions)
IDELRel, ***ICELRel***, ***IDCELRel***	Communication relationships inputs (I) of: 1. Data variables (D); 2. Control variables (C); 3. Hybrid data&control variables (DC), from environmental means (E) and the inputs are looped (L).
IDLERel, ***ICLERel***, ***IDCLERel***	Communication relationships by inputs (I) of: 1. Data variables (D); 2. Control variables (C); 3. Hybrid data&control variables (DC) with other system functions and the inputs are looped (L) via some environmental means (E).
RBehRel, ***PBehRel***	Synchronization relationships in terms of precedence/ordering by: 1. requiring executional behavior feature (RBeh), 2. providing executional behavior feature (RBeh), from/to other system functions.
RBehERel, ***PBehERel***	Synchronization relationships in terms of precedence/ordering by: 1. requiring executional behavior feature (RBeh), 2. providing executional behavior feature (RBeh), from/to environmental means (E).
TimRel	Synchronization relationships by timing features (Tim) with other system functions.
TimERel	Synchronization relationships by timing features (Tim) with environmental means (E).
SImplPRel	Implementation specific relationships due to sharing (S) an implementation parameter (ImplP), such as a task, an inter/intra-node communication channel, a clock service) with other system functions.
SImplMRel	Implementation specific relationships due to sharing (S) an implementation means (ImplM), such as a device and a CPU, with other system functions.
DImplMRel	Implementation specific relationships due to dependencies (D) between the implementation means (ImplM) on which the system functions are allocated, such as by means common failure modes.

3.2. Measurement

3.2.1. Context

The target of coupling measurement is a single system function. From a requirement assignment point of view, we distinguish between the core and the adaptation of a system solution as well as a system function. The core refers to the fundamental portion that accounts only for mandatory system requirements (i.e., functionality, (RT)performance, and dependability). Often, however, when other more "soft" qualities such as maintainability and modifiability are of concern, there is normally a gap between what the core inherently grants and what is expected. Under the circumstances, adjustments on the core in terms of restructuring (targeting relationships) and tuning of system parameters (focusing on system means and their properties) have to be performed given that such actions will not violate the mandatory requirements. The results constitute to as the adaptation of a system solution. Examples of parameter-tuning in system design can be introducing additional system means in a system solution, changing execution frequencies, or increasing implementation resources. Accordingly, the coupling of a system function can be either "intrinsic", i.e., the dependency is within the core, or "extrinsic", i.e., if the dependency is within the adaptation.

The system function under measurement should exist in a "stable" system solution in terms of a "static" configuration. In the case of dynamic configuration, we view the system as composed of several static configurations. In our system model, the mapping between two stable configurations in a design hierarchy is given by the refinement design relationship. In the development, the measurement can be performed several times at different design levels, targeting abstract functional solutions as well as technology specific implementation solutions. By evaluating the results, problems due to implementation technologies (e.g., due to inconsistent emergent properties in implementation) can be revealed.

3.2.2. Parameters affecting coupling

Coupling is a measure that quantifies the strength of the dependency a system function has with respect to its operational context. In our approach, two parameters are chosen to characterize the strength: *intensity* and *target-cardinality*. The intensity is a factor that aggregates the magnitude per time unit and the accuracy of features being utilized in a relationship. The target-cardinality is concerned with the scope of dependencies, e.g., in the cases of fan-in and fan-out topologies.

An agreement (or a contract) is considered strong under the following conditions: a large magnitude of features per time unit being used (e.g., received/provided variables); a high accuracy is required by the context of a system with respect to value, ordering, and timing; and a large amount of partners are involved.

Note that some of these parameters can also be involved in other system quality attributes (e.g., performance and reliability). Since such parameters affect multiple quality attributes simultaneously, they are often considered as the sensitive points of a system solution, see e.g., [18].

In system design, it would be convenient to have an overall coupling figure for a system function taking all individual relationships into account. To this end, a methodology has been developed. To make the cross type combination of coupling possible, the problem due to the lack of a common basis has to be resolved (e.g., a communication coupling in numbers of variables per time unit vs. a synchronization coupling in number of behavior features per time unit are not directly comparable). For this reason, we normalize the measured couplings and combine the results using a linear combination technique (i.e., weighted-sum).

4. INDIVIDUAL COUPLING

The operational coupling of a system function is a union of coupling by each type of interaction defined as follows.

DEFINITION 3 (Operational coupling of system functions by individual relationships). *For each* $f \in \mathbf{Func}$ and $\mathbf{r}(f) \in \mathbf{Interact}(f)$,

$$Coupling(f) = \bigcup_{\forall \mathbf{r}(f) \in \mathbf{Interact}(f)} \{(\mathbf{r}(f), \kappa(\mathbf{r}(f)))\}$$

Where: κ - *coupling factor of a type of relationships.*

We write $\kappa(\mathbf{r}(f))$ for the coupling factor of a system function $f \in \mathbf{Func}$ for one type of its relationships $\mathbf{r}(f) \in \mathbf{Interact}(f)$. The coupling factor is the measure of dependency strength defined as follows.

DEFINITION 4 (Coupling factor). *For each* $\mathbf{r}(f) \in \mathbf{Interact}(f)$

$$\kappa(\mathbf{r}(f)) = \rho(\mathbf{r}(f)) \cdot \delta(\mathbf{r}(f))^{C_\delta}$$

where: ρ – *intensity factor*
δ – *target-cardinality*
C_δ – *coefficient relating* δ *to* ρ

4.1. Intensity factor -ρ

The intensity factor quantifies the magnitude of features per time unit that appear in a type of relationship by considering the number, the frequency, and the accuracy of involved properties.

DEFINITION 5 (Intensity factor). *For* $f \in \mathbf{Func}$, $\mathbf{r}(f) \in \mathbf{Interact}(f)$, *and* $r(f) \in \mathbf{r}(f)$, *let the properties forming* $\mathbf{r}(f)$ *be* $\mathbf{Prop}(\mathbf{r}(f))$. *The intensity of* $\mathbf{r}(f)$ *is*

$$\rho(\mathbf{r}(f)) = \sum_{\forall p \in \mathbf{Prop}(\mathbf{r}(f))} \sum_{\forall r(f) \in \mathbf{r}(f)} R^{r(f)}(p) \cdot F^{r(f)}(p) \cdot s_A{}^{r(f)}(p)$$

Where: $R^{r(f)}(p)$ – *number of replications of p in r(f)*
$F^{r(f)}(p)$ – *frequency of p in r(f)*
$s_A{}^{r(f)}(p)$ – *accuracy factor of p in r(f).*

In Definition 5, the outer summation distinguishes different properties that appear in a type of relationship. E.g., in the configuration shown in Figure 5, the system function f_1 has data variables *p1*, *p2*, *p3*, *p4* in its data input relationships, $\mathbf{Prop}(\mathbf{IDRel}(f_1)) = \{p1, p2, p3, p4\}$, and control variables *p5* and *p6* in its control input relationships, $\mathbf{Prop}(\mathbf{ICRel}(f_1)) = \{p5, p6\}$, and properties *p7* and *p8* in its output relationships, $\mathbf{Prop}(\mathbf{ORel}(f_1)) = \{p7, p8\}$. The system function f_1 thus has properties belonging to three different relationship types. The inner summation identifies the relationship instances that use each of these properties. It covers the cases where one property appears in multiple instances of a single relationship type. E.g., in Figure 5, the data output of f_1 has two instances: $\mathbf{ORel}(f_1) = \{r', r''\}$, where: $r' = (f_1, COMM, f_2)$ and $r'' = (f_1, COMM, f_3)$. (See also Definition 1). The output variable *p7* is involved in both r' and r''.

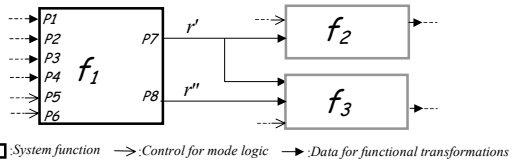


Figure 5. An example configuration of three system functions.

4.1.1. Number of replications – R

Given a property in an interaction, the parameter *R* describes the redundancy that the property has each time when it is involved in the interaction, $R \in \mathbf{Z}^+$ (i.e., a positive integer). The default value is *1*, meaning there is no redundancy. A duplication has *R=2*, a triple-redundancy has *R=3*, and so on. The redundancy can be either spatial or temporal, where a property is replicated for the reasons of dependability. For example, consider f_1 in Figure 5, the data variable *p8* may need to be sent to function f_3 twice at each occurrence of communication compared to its nominal frequency.

4.1.2. Frequency – F

Given a property in an interaction, this parameter describes the rate at which this property has to be provided or produced. We assume that the system under consideration is deterministic in the sense that interactions are either periodic or aperiodic, where the frequency in the latter case is the rate necessary to handle burst conditions (i.e. the worst case).

4.1.3. Accuracy factor – s_A

The accuracy factor, s_A, accounts for the strength of coupling related to the required and provided quality of properties within their compatibility range. It differentiates relationships that are based on properties of the same type and with the same magnitude per time unit, but of different accuracy. This factor is defined as follows.

DEFINITION 6 (Accuracy factor s_A). *The accuracy factor for property p in relationship r(f) is*

$$s_A{}^{r(f)}(p) = \frac{1}{1 - C_{RM}{}^{r(f)}(p) \cdot \ln\left(1 - RM_A{}^{r(f)}(p)\right)}$$

Where: $RM_A{}^{r(f)}(p)$ *– relative accuracy margin of p in r(f).* $0 \le RM_A{}^{r(f)}(p) < 1$.
$C_{RM}{}^{r(f)}(p)$ *– constant for tuning the effect of the* RM_A. $C_{RM}{}^{r(f)}(p) > 0$.

The relative accuracy margin, RM_A, indicates the freedom to further change the quality of connected features (e.g., deceasing the quality of data sent to a system function).

DEFINITION 7 (Relative accuracy margin RM_A). *The relative accuracy margin for property p in relationship r(f) is*

$$RM_A{}^{r(f)}(p) = \begin{cases} \dfrac{\left| RA_0{}^{r(f)}(p) - RA^{r(f)}(p) \right|}{Reg_A{}^{r(f)}(p)}, & \text{if } RA^{r(f)}(p) \neq RA_0{}^{r(f)}(p) \\ 0, & \text{if } RA^{r(f)}(p) = RA_0{}^{r(f)}(p). \end{cases}$$

Where: $RA^{r(f)}(p)$ *– inherent relative accuracy of p in r(f),* $0 < RA^{r(f)}(p) \le 100\%$.
$RA_0{}^{r(f)}(p)$ *– context provided or assumed relative accuracy of p in r(f),* $0 < RA_0{}^{r(f)}(p) \le 100\%$.
$Reg_A{}^{r(f)}(p)$ *– length of accuracy compatible region for p in r(f),* $0 < Reg_A{}^{r(f)}(p) \le 100\%$.

If p is an incoming property: $RA_0{}^{r(f)}(p) \ge RA^{r(f)}(p)$, *and* $Reg_A{}^{r(f)}(p) = RA_0{}^{r(f)}(p)$. *Otherwise*: $RA_0{}^{r(f)}(p) \le RA^{r(f)}(p)$ *and* $Reg_A{}^{r(f)}(p) = 1 - RA_0{}^{r(f)}(p)$.

Both *RA* and RA_0 are defined in terms of the agreement of a property to its nominal value in percentage. The relative accuracy, *RA*, is the accuracy determined by the system function itself. For example, a data input can have a *RA* of *99.5%*, meaning *0.5%* of deviation to the nominal value can be tolerated (i.e., either ignored or handled by the system function). The reference relative accuracy, RA_0, is the required accuracy given by the context. For example, the data input mentioned above can have $RA_0 = 99.9\%$, meaning that data sent to the system function has *0.1%* of deviation. The case *RA=100%* and $RA_0=100\%$ represents an ideal condition where the accuracies are not of concern or not taken into account, for example when the design is at an early stage.

The length of the accuracy compatible region, Reg_A, indicates the amount of permissible difference between the accuracy of a property and its required value in a relationship. For an incoming property (e.g., input data), the region is $(0, RA_0]$, meaning that any value of *RA* from RA_0 down to *0* is acceptable. When $RA > RA_0$, the connection is considered *INCOMPATIBLE* since errors of the incoming feature can no longer be controlled and will eventually result in system failure. For an outgoing property (e.g., output data), the region is $[RA_0, 1]$, meaning that any value of *RA* from RA_0 up to is *100%* acceptable. If $RA < RA_0$, an outgoing connection becomes *INCOMPATIBLE* since the provided feature does not meet the required quality. See also Figure 6.

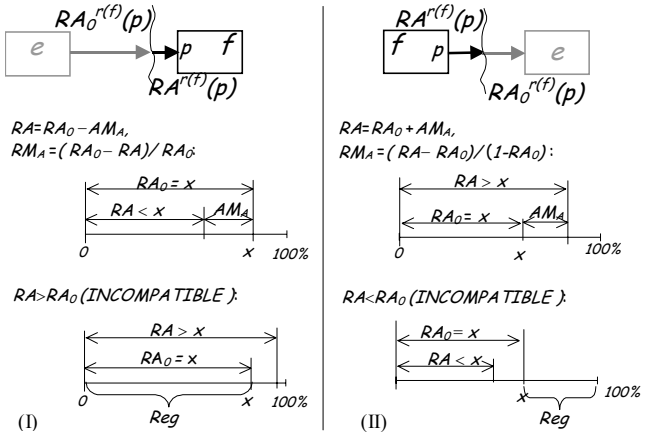


Figure 6. An illustration of accuracies of (I) input (II) output.

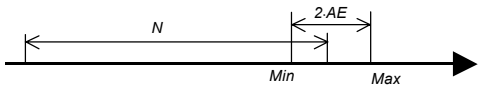
The relative accuracies, *RA* and RA_0, are both concerned with permissible errors of a property from single system functions' point of view. By definition, an error is a condition where a feature deviates from its nominal value [14]. *RA* and RA_0 are derived from two tolerances respectively: *RE* and RE_0. While the relative error (*RE*) describes the degree of permissible errors inherent in a system function, the reference relative error (RE_0) describes what is supported/required by its context. For property *p* in *r(f)*, $RA^{r(f)}(p) = 1 - RE^{r(f)}(p)$ and $RA_0{}^{r(f)}(p) = 1 - RE_0{}^{r(f)}(p)$.

The accuracy factor, s_A, has a range of $(0, 1]$. Plot-I of Figure 7 depicts its relationship to the relative accuracy margin, RM_A, given as the inverse of a S-curve. When $RM_A=0$, meaning that the accuracy provided by a system function precisely matches what is required by the context, s_A has a value of *1*. As the margin increases, s_A approaches *0* asymptotically. Plot-II of Figure 7 depicts the relationships between s_A and the relative accuracies of a property, *RA* and RA_0, when $C_{RM}=1$. As indicated by the plots, to reduce the quality dependencies of a system function, one needs to increase the relative accuracy margin of its properties. This can be achieved by decreasing the *RA* or increasing the RA_0 of its incoming properties, and by increasing the *RA* or decreasing the RA_0 of its outgoing properties (e.g., by applying a more robust design).

The constant C_{RM} is introduced to shape the curves. For example, as shown in Plot-I and III of Figure 7, if a decrease of RA has a large impact on the coupling, a large C_{RM} should be chosen. The choice of value depends on several factors in system development (see Section 7 for a discussion).

The accuracies have their semantics given by the targeting properties.

Communication relationships. For communication relationships (i.e., ***CommRel***), the accuracies *RA* and RA_0 are concerned with permissible value deviations of communication variables. The relative errors, *RE* and RE_0, are then the ratio of permissible value deviation of a variable (i.e., the absolute error (*AE*) or value tolerance) to its nominal value range. See Figure 8. For example, consider the environmental temperature input of a system function with a required tolerance of $\pm 1°C$ over $0\sim200°C$. Assume the environmental data has a tolerance of $\pm 0.5°C$ in a range over $-50°C\sim350°C$, reflecting the constraints of sampling device (e.g., temperature sensor and A/D converter) that will eventually implement the communication. Then, we have *RA* in *99.5%* (i.e., $(1-1/200)\cdot 100\%$) and RA_0 in *99.875%* (i.e., $(1-0.5/400)\cdot 100\%$). If $C_{RM}=1$, the relative margin (RM_A) and accuracy factor (s_A) for the data input will be in *0.375%* and *0.9963* respectively. Replacing the system function by a new one with a tolerance of $\pm 2°C$, *RA* becomes *99%*. This will contribute to a lower coupling by reducing the quality dependency in terms of s_A to *0.9913*.



N – Nominal value; AE – absolute error; Min – minimum value; Max – maximum value.

Figure 8. Relative and absolute error of values.

Synchronization relationships by executional behavior features. For ordering relationships (i.e., ***PBehRel***, ***RBehRel***, ***PBehERel***, or ***RBehERel***), the accuracies *RA* and RA_0 are concerned with permissible errors of executional behavior features in the connections. The relative errors, *RE* and RE_0, are the ratio between the number of permissible occurrences of errors (e.g., a combination of omission and commission) to the number of its nominal occurrence over a time duration. Note that an executional behavior feature is "binary" in nature, e.g., a triggering is either successful or not. For example, consider a system function that is required to read its data input periodically sent by another system function at a rate of *100Hz*. Assume the sender (as well as the communication link) has one omission of its sending action over *10s*. If one omission of this receiving action over *1s* is tolerable, we have *RA=99%* and $RA_0=99.9\%$. If $C_{RM}=1$, the relative margin (RM_A) and accuracy factor (s_A) for the input behavior will be *0.88%* and *0.9913* respectively.

Synchronization relationships by timing features. For timing relationships (i.e., ***CoTimRel*** and ***CoTimERel***), the relative accuracies are concerned with permissible errors of timing features in the connections. The relative errors, *RE* and RE_0, are given by the ratio of permissible deviation (i.e., jitters) to the nominal interval of a timing feature (e.g., periodicity or delay). For example, consider a system function (*f*) that performs data output at a rate of *100Hz*. Assume the output can be traced back to an input of another system function (f'). The timing of the output can be written as $t_2=t_1+\tau_{ex}+\tau_{in}$, where: t_2 – time instant of the output, t_1– time instant of the input, τ_{ex}– delay until the input data propagates to an input of *f*, and τ_{in}– computational delay of *f* (which also may include estimated operational interference). Assume an end-to-end timing requirement as follows; $t_2 = t_1 + 10\pm0.5(ms)$. Given $\tau_{ex}=6\pm0.3(ms)$, it will be required that $\tau_{in}=4\pm0.2(ms)$. If the system function *f* provides $\tau_{in}= 4\pm0.1(ms)$, we have *RA=97.5%* for the output (i.e., $(1-0.1/4)\cdot100\%$) and $RA_0=95\%$ (i.e., $(1-0.2/4)\cdot100\%$). With $C_{RM}=1$, the relative margin (RM_A) and accuracy factor (s_A) for the output timing will be *50%* and *0.5906* respectively.

4.2. Target-cardinality - δ

The target-cardinality describes the number of other system means connected to a system function by relationships of a particular type, defined as follows.

DEFINITION 8 (Target-cardinality). For ***r(f)***$\in$***Interact(f)***, *f*$\in$***Func***, *the target-cardinality of the relationship is*[1]

$$\delta(r(f)) = \left| \{ g \in \textbf{Func} \mid \exists (f,\ R,\ g') \in \textbf{r}(f),\ g' \in \textbf{Func} \setminus \{f\},\ R \in \textbf{RelType}:\ g = g' \} \right|$$

The coefficient C_δ is introduced due to the fact that the intensity (ρ) and target-cardinality (δ) of a relationship can have different effects on dependency. The choice of value depends on several factors in system development (see Section 7 for a discussion). We assume $C_\delta = 2$ for all communicational relationships, meaning that target-cardinality has a much larger effect on communication coupling than intensity. For all synchronization relationships, we assume $C_\delta = 1$, meaning that target-cardinality and intensity have the same effect on the coupling. For example, consider two system functions: f_1 and f_2. Both of them receive data. While f_1 is

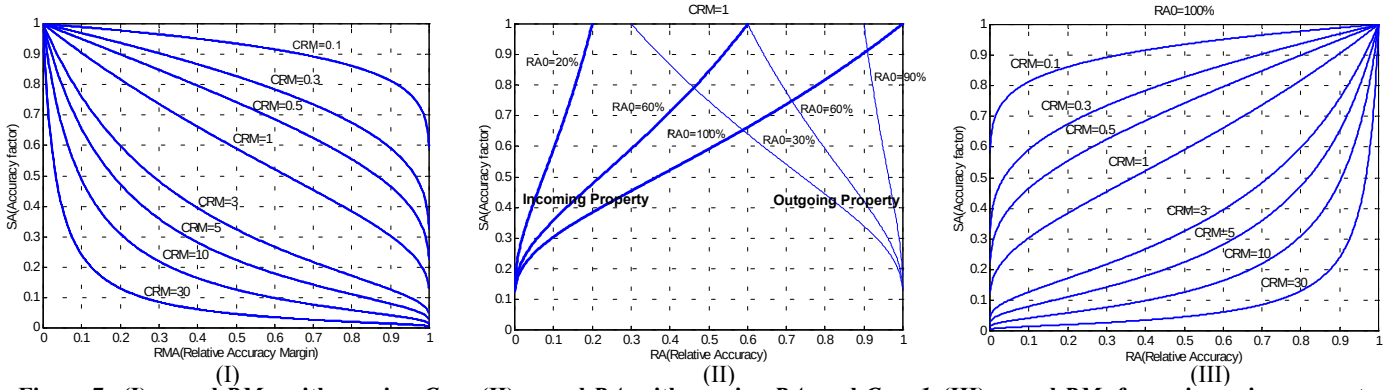


Figure 7. (I) s_A and RM_A, with varying C_{RM}. (II) s_A and *RA*, with varying RA_0 and $C_{RM}=1$. (III) s_A and RM_A for an incoming property, with varying C_{RM} and $RA_0=100\%$

connected to *5* other system functions (δ=*5*) in *1 Hz*, f_2 is connected to one other system function (δ=*1*) in *25 Hz*. Assume R=*1*, s_A=*1*, and C_δ=*1*, then these two system functions will have the same coupling value. However, in most cases, f_1 is normally more difficult to modify or replace than f_2, indicating a stronger dependency on the external functions. In f_2, the compatibility (both operational and analytical) concerns only two system functions and is easier to manage.

4.3. Example

Assume there are six system functions shown in Figure 9. Assume the accuracies equal to 100%.

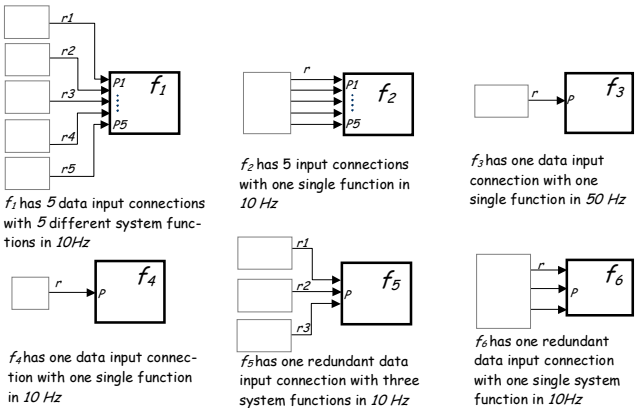


Figure 9. Example: six configurations of data input.

The coupling of these system functions are: $\kappa(\boldsymbol{r}(f_1))=1250$, $\kappa(\boldsymbol{r}(f_2))=50$, $\kappa(\boldsymbol{r}(f_3))=50$, $\kappa(\boldsymbol{r}(f_4))=10$, $\kappa(\boldsymbol{r}(f_5))=270$, and $\kappa(\boldsymbol{r}(f_6))=30$. The following shows how the computation is performed:

For f_2: $\boldsymbol{r}(f_2)=\boldsymbol{IDRel}(f_2)=\{r\}$, $\delta(\boldsymbol{r}(f_2))=1$, and $\boldsymbol{Prop}(\boldsymbol{r}(f_2))=\{P_1, P_2, P_2, P_4, P_5\}$. For each property, we have

P_1: $R^r(P_1)=1$, $F^r(P_1)=10$; P_2: $R^r(P_2)=1$, $F^r(P_2)=10$;
P_3: $R^r(P_3)=1$, $F^r(P_3)=10$; P_4: $R^r(P_4)=1$, $F^r(P_4)=10$;
P_5: $R^r(P_5)=1$, $F^r(P_5)=10$;

Hence, $\rho(\boldsymbol{r}(f_2))=50$ and $\kappa(\boldsymbol{r}(f_2))=50$.

Although f_1 and f_2 have the same input intensity, f_1 has a much tighter coupling due to its high target-cardinality. The configuration in f_2 can be considered as a result of modularizing the configuration of f_1 where the sender functions to f_1 are composed into a single module. For the same reason, f_6 has a much lower coupling than that of f_5, although inputs to these system functions have the same intensity. The relationships of f_3, f_4, f_6 have the same amount of variables and target-cardinality. f_3 has a higher coupling than f_4 and f_6 because of its frequency. f_6 has a higher coupling than f_4 due to its replicated inputs.

5. INTEGRATED COUPLING

In the previous section, the coupling of a system function is a collection of couplings by individual relationships. In this section, we describe the approach to an integrated measure where the coupling of individual relationships are normalized, weighted, and summed together. This coupling is defined as follows.

DEFINITION 9 (Overall coupling of system functions). *For* $f \in \boldsymbol{Func}$ and $\boldsymbol{r}(f) \in \boldsymbol{Interact}(f)$, *the overall coupling is*

$$Overall\_Coupling(f) = \sum_{\forall \boldsymbol{r}(f) \in \boldsymbol{Interact}(f)} \omega(\boldsymbol{r}) \cdot \kappa'(\boldsymbol{r}(f))$$

Where: $\omega(\boldsymbol{r})$ – *weighting factor of relationship* $\boldsymbol{r}$.
$\kappa'(\boldsymbol{r}(f))$ – *relative coupling factor of* $\boldsymbol{r}(f)$.

5.1. Relative coupling factor - κ'

The relative coupling factor κ' for a type of relationship is obtained by normalizing the coupling of a system function as follows.

DEFINITION 10 (Relative coupling factor). *For* $f \in \boldsymbol{Func}$ and $\boldsymbol{r}(f) \in \boldsymbol{Interact}(f)$, *the relative coupling factor is*

$$\kappa'(\boldsymbol{r}(f)) = \begin{cases} \dfrac{1}{1+\left(\dfrac{\kappa_B(\boldsymbol{r}(f))}{\kappa(\boldsymbol{r}(f))}\right)^{2C_s(\kappa_B(\boldsymbol{r}(f))-\kappa(\boldsymbol{r}(f)))}} & \text{When } \kappa(\boldsymbol{r}(f)) \le \kappa_B(\boldsymbol{r}(f)) \\ 1-\dfrac{1}{1+\left(\dfrac{\kappa_U(\boldsymbol{r}(f))-\kappa_B(\boldsymbol{r}(f))}{\kappa_U(\boldsymbol{r}(f))-\kappa(\boldsymbol{r}(f))}\right)^{2C_s(2\kappa(r_U(f))-\kappa_B(\boldsymbol{r}(f))-\kappa(\boldsymbol{r}(f)))}} & \text{When } \kappa_B(\boldsymbol{r}(f)) \le \kappa(\boldsymbol{r}(f)) \le \kappa_U(\boldsymbol{r}(f)) \end{cases}$$

Where: $\kappa(\boldsymbol{r}(f))$ – *coupling of* $\boldsymbol{r}(f)$.
$\kappa_B(\boldsymbol{r}(f))$ – *baseline coupling of* $\boldsymbol{r}(f)$.
$\kappa_U(\boldsymbol{r}(f))$ – *upper bound coupling of* $\boldsymbol{r}(f)$.
C_s – *constant for tuning slope of the curve* $\kappa'(\boldsymbol{r}(f))$ *at* $\kappa_B(\boldsymbol{r}(f))$, $C_s>0$.

This definition is derived from the general-purpose standard scoring function (SSF) of Wymore [5], by assuming that the lower bound of a coupling factor is *0*. The baseline, $\kappa_B(\boldsymbol{r}(f))$, represents the design goal or the condition in an initial solution (from which other solution alternatives are generated). The relative coupling, $\kappa'(\boldsymbol{r}(f))$, of the baseline coupling is always *0.5*. The upper bound, $\kappa_U(\boldsymbol{r}(f))$, represents the maximum value and has $\kappa'=1$. Figure 10 depicts the shape of the curve and the effect of C_s

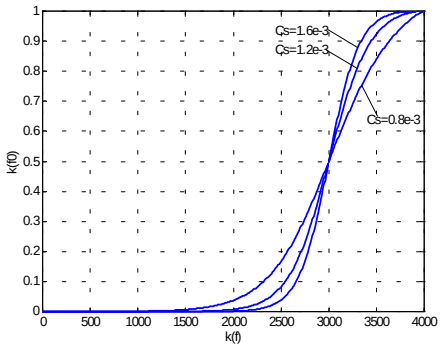


Figure 10. κ' ***and*** κ ***with varying*** C_s**, when** κ_B**=3000,** κ_U**=4000.**

To determine the baseline and upper bound for each system function, one needs to take the implementation-sharing relationships of a system function (i.e., ***SImplPRel***, ***SImplMRel***, and ***DImplMRel***) as well as the scheduling decisions into consideration. Each of these implementation-sharing relationships indicates a special kind of technology constraints.

For example, consider an axis control system within which there

[1] |A| - cardinality of set A; A\B - exclusion of set B from A.

are a torque control function (*TC*) and a power stream control function (*IC*). The *TC* sends a data variable (i.e., current set point) to the *IC* at *2kHz*. Given a target platform consisting of two nodes and a CAN-bus, *TC* and *IC* can be allocated either to a single node or to different nodes separately. Accordingly, the baseline and upper bound for the communication relationships differ, resulting in different values of the relative coupling.

5.2. Weighting factor - ω(r)

The weighting factor is used to indicate the relative tightness of dependencies due to different types of relationships. It is based on the assumption that different relationships can have different degrees of contribution to flexibility. The weight is obtained as follows.

DEFINITION 11 (Weighting factor). *For each* $\mathbf{r} \in \mathbf{Interact}$,

$$\omega(\mathbf{r}) = \omega(\mathbf{Gr}) \cdot \omega^{\mathbf{Gr}}(\mathbf{r})$$

Where: ***Gr*** *– category of interactions, i.e.* ***CommRel*** *and* ***SyncRel****.*

$\omega(\mathbf{Gr})$ *– cross category weights.*

$\omega^{\mathbf{Gr}}(\mathbf{r})$ *– cross type weights within the same group.*

The values are determined using a binary comparison technique, i.e., *analytical hierarchy process* (AHP) [19]. Some useful heuristics for the weighting is listed in Table 3. One result for the relationships in a particular system is shown in Table 4.

Table 3. General heuristics taken for the weighting.

Communication vs. synchronization	Synchronization is tighter. Such relationships connect dynamic features that are in general more challenging to manage (e.g., requiring careful scheduling and clock synchronization).
Internal vs. environmental relationships	Environmental relationships are tighter. Such a relationship normally involves a mapping from the continuous physical domain to the discrete digital domain. Moreover, the environment is often predetermined, with little freedom to be changed.
Outgoing vs. incoming relationships	Incoming relationships are tighter. An incoming relationship indicates a client role, while an outgoing relationship indicates a server role. A sever can work independently without its clients, but a client needs the services provided by its servers.
Data vs. control communications	Communication relationships of control variables are tighter. Instead of transformations, the targets of control variables are mode transitions where a small divergence or change to the expected values may result totally different outcomes.
Simple vs. looped relationships	Looped relationships are much tighter since a loop indicates a symmetric or mutual dependency.
Ordering vs. timing relationships	Timing relationships are tighter than pure ordering. Such relationships require a fixed timing between executional behaviors, and are hence more difficult to guarantee.

Table 4. Example of weighting results.

Relationships - ***r***	Group/category - ***Gr***	Cross-group weights - ω (***Gr***)	Cross-weight within a group - $\omega^{Gr}(r)$	Final weight - ω (***r***)
ORel	CommRel	0.25	0.008	**0.002**
OERel	CommRel	0.25	0.010	**0.003**
IDRel	CommRel	0.25	0.014	**0.004**
IDERel	CommRel	0.25	0.020	**0.005**
ICRel	CommRel	0.25	0.028	**0.007**
IDCRel	CommRel	0.25	0.033	**0.008**
ICERel	CommRel	0.25	0.040	**0.010**
IDCERel	CommRel	0.25	0.048	**0.012**
IDLRel	CommRel	0.25	0.058	**0.015**
PBehRel	SyncRel	0.75	0.025	**0.019**
IDELRel	CommRel	0.25	0.082	**0.021**
ICLRel	CommRel	0.25	0.116	**0.029**
PBehERel	SyncRel	0.75	0.043	**0.032**
IDCLRel	CommRel	0.25	0.141	**0.035**
ICELRel	CommRel	0.25	0.176	**0.044**
IDCELRel	CommRel	0.25	0.224	**0.056**
RBehRel	SyncRel	0.75	0.080	**0.060**
RBehERel	SyncRel	0.75	0.144	**0.108**
CoTimRel	SyncRel	0.75	0.256	**0.192**
CoTimERel	SyncRel	0.75	0.452	**0.339**

6. RELATED WORK

Some of the earliest definitions on coupling-and-cohesion for software have been proposed by Stevens, Myers, Yourdon, and Constantin in 1970's, see [20][21]. Their aim is to provide a basis for modularizing software in a structured programming language and for evaluating the quality. They distinguish 7 levels of cohesion and 5 levels of coupling, found in structured software programs. These basic coupling types have been refined and extended over the years. For instance, Offutt et. al. [22] differentiate three ways of information usage: C-uses (computation uses), P-uses (predicate uses), and I-uses (indirect uses). In [23], Lounis and Melo have proposed a suite for identifying and counting various coupling types in C-language. In this work, the couplings between C-modules are arranged in two major groups distinguished by the kind of interconnection mechanism: unit-call interconnection and common interconnection. These basic coupling types are then sub-classified according to the type of transmitted information (by data or by address/reference), the usage of shared information (in computation or in procedure control as in [22],), as well as some language specific issues (interconnection by call or by return). To measure the coupling of each module or in the entire system, the numbers of import and export connections of each interconnection type are counted.

There are also efforts in defining metrics of coupling and cohesion for object-oriented software systems, considering especially the object-oriented features such as classes, encapsulation, inheritance and polymorphism. Eder et al. [24] have identified three dimensions of cross-class coupling by investigating the configuration and dependencies of OO programs: interaction coupling, component coupling, and inheritance coupling. Chidamber and Kemerer [25] have proposed a metrics suite for object-oriented design based on a study of the ontology of OO concepts concerning objects and their relationships. An OO object is a representation of the application domain, defined by a name and a set of properties in terms of instance variables and operation methods. Objects are related to each other by encapsulation, independence and inheritance. A coupling is established between two classes when methods declared in one class use methods or instance variables of the other class. It is referred to as CBO (Coupling between object classes), and measured by the number of classes a class is coupled with. In [26], Briand et.al. have proposed an integrated measurement framework for object-oriented coupling metrics. This is based on a formal definition of characteristics of such software programs such as polymorphisms, static and dynamic method invocations. In [27], Allen and Khoshgoftaar propose an information-theory based approach to coupling and cohesion measures, motivated by the need for a more effective assessment method than counting. For the understanding of coupling and cohesion a formal framework is used to describe the components in a system and their relationships. This approach measures the overall coupling of inter-module and intra-module types by quantifying the symbolic information content (i.e., entropy) based on connection patterns.

From an ECS point of view, all the above-mentioned approaches are delimited by their software engineering based perspective. The lack of consideration with respect to ECS specific concerns such as timing and concurrency means that the produced measures are only partial figures of coupling.

7. CONCLUSIONS AND FUTURE WORK

A metrics system has been developed to support an objective and repeatable measurement of operational dependencies between system functions, hence a more systematic approach to structuring and quality prediction of ECS. Instead of implementation-technology, the measurement targets the overall system structure and behavior, existing throughout the refinement hierarchy of a system. Based on a fine grained classification of such relationships, the quantification takes ECS specific issues into concern such as replication and timing.

There are several obvious avenues for further work. In the short term there is a need to gather more empirical evidence mainly for studying the influence and tuning of the following parameters part of the coupling metric: C_{δ}, C_{RM}, and the cross-type weighting. We believe that these depend on both human decision factors and system factors, such as system characteristics, the insight of the developers, the available modeling and tool support. I.e. we do not expect that coupling as a soft quality attribute will obey any natural laws. However, for the purpose of early estimations there is less of a need for highly precise metrics; the value lies in being able to reveal the sensitive points and to make multi-attribute trade-offs possible.

8. ACKNOWLEDGMENTS

The work described in this paper is funded in part by ARTES (a network for real-time research and graduate education), and in part by KTH (Royal Institute of Technology) in Sweden.

9. REFERENCES

[1] J. Wikander, M. Törngren, Mechatronics as an Engineering Science*, Proc of the 6th UK Mechatronics Forum Int Conf*, 1998.

[2] NG Leveson, Intent Specifications: An Approach to Building Human-Centered Specifications, *IEEE Trans on SW Eng*, VOL.26 (1), 2000

[3] NP Suh, *Axiomatic design: Advances and Applications*. Oxford University Press. 2001.

[4] E Rechtin, MW Maier. *The Art of System Architecting*, CRC Press. 1997.

[5] WL Chapman, AT Bahill, AW Wymore, *Engineering Modeling and Design*, CRC Press. 1992.

[6] JE Cooling, *Software Design for Real-time Systems*, Chapman and Hall. 1991.

[7] H Gommaa, *Designing Concurrent, Distributed, and Real-Time Applications with UML*, Addisson-Wesley, 2000.

[8] W Stevens, GJ Myers, L Constantin, Structured Design, *IBM Syst Jour*, VOL.13 (2) 1974.

[9] EL Yourdon, L. Constantin, (1979), *Structured Design – Fundamentals of a Discipline of Computer Program and Syst Design*, Prentice-Hall, 1979.

[10] DJ Chen, M Törngren, A Systematic Approach for Identifying Operational Relationships in Embedded Computer Control Systems, *30th Euromicro Conf. on Component-Based Software Engineering Track, Aug, 2004*.

[11] DJ Chen, M Törngren, Towards A Framework for Architecting Mechatronics Software Systems, *7th IEEE Int Conf on Eng of Complex Comp Sys* (ICECCS), Jun, 2001.

[12] DJ Chen, J El-Khoury, M Törngren, A Modeling Framework for Automotive Embedded Control Systems, *SAE Tech Paper Series* (2004-01-0721), 2004 SAE World Cong, Detroit, March 8-11, 2004.

[13] J El-khoury, DJ Chen, M Törngren, A Survey of Modeling Approaches for Embedded Computer Control Systems, *Technical Report*, TRITA-MMK 2003:36 ISSN 1400 –1179, ISRN KTH/MMK/R-03/11-SE, 2003.

[14] NG Leveson, *Safeware – System Safety and Computers*, Addison-Wesley Publishing Company, 1995.

[15] N Storey, *Safety-Critical Computer Systems*, Addison-Wesley, 1996.

[16] R Wieringa, A survey of structured and object-oriented software specification methods and techniques, *ACM Computing Surveys (CSUR)*, Vol 30(4), Dec, 1998.

[17] SW Kercel, Why should engineers be interested in bizarre systems?, *IEEE International Conference on Systems, Man, and Cybernetics* Volume: 3 , 8-11 Oct. 2000.

[18] MR Barbacci, SJ Carriere, PH Feiler, R Kazman, MH Klein, HF Lipson, TA Longstaff, CB Weinstock, Steps in an Architecture Trade-off Analysis Method: Quality Attribute Models and Analysis, *Tech Report, SEI, CMU*, 1997.

[19] TL Saaty, *Multicriteria decision making: The analytic hierarchy process*. 1980, McGraw-Hill.

[20] W. Stevens, G. J. Myers, L. Constantin, Structured Design, *IBM System Journal*, Vol. 13, no. 2. 1974.

[21] EL Yourdon, L. Constantin, (1979), *Structured Design – Fundamentals of a Discipline of Computer Program and System Design*, Prentice-Hall, 1979

[22] AJ Offutt, MJ Harrold, and P Kolte, A Software Metric System for Module Coupling, *Journal on Software and System*, 1993

[23] H. Lounis and W. Melo, Identifying and Measuring Coupling on Modular Systems, *Proceedings of the 8th International Conference on Software Technology* (ICST'97), Curitiba, Brazil, June 1997.

[24] J Eder, G Kappel, and M Schrefl, Coupling and Cohesion in Object-Oriented Systems, *Tech. Rep.*, *Univ. of Klagenfurt*, 1994.

[25] SR Chidamber and CF Kemerer, A Metrics Suite for Object Oriented Design *IEEE Trans.s on Software Eng.*, vol. 20, Issue: 6, June 1994.

[26] LC Briand, JW Daly, and JK Wüst, A Unified Framework for Coupling Measurement in Object-Oriented Systems, *IEEE Trans. on Software Eng.*, VOL. 25 (1), Jan./Feb. 1999

[27] EB Allen, TM Khoshgoftaar, Measuring Coupling and Cohesion: An Information-Theory Approach, *Proceedings of the Sixth International Software Metrics Symposium,* pages 119-127, Boca Raton, Florida, USA, Nov. 1999, IEEE Computer Society.

[28] *Roadmaps for Embedded Software and Systems*, ARTIST Project IST-2001-34820. < http://www.artist-embedded.org/Roadmaps/ >