NASA/TM–2011-217193

A Methodology for Evaluating Artifacts Produced by a Formal Verification Process

Radu I Siminiceanu
National Institute of Aerospace, Hampton, Virginia

Paul S. Miner and Suzette Person
Langley Research Center, Hampton, Virginia

November 2011

NASA/TM–2011-217193

A Methodology for Evaluating Artifacts Produced by a Formal Verification Process

Radu I. Siminiceanu
National Institute of Aerospace, Hampton, Virginia

Paul S. Miner and Suzette Person
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

November 2011

Acknowledgments

This research has been carried out as part of the Autonomous Systems and Avionics (ASA) project led by NASA's Ames Research Center under the Enabling Technology Development and Demonstration (ETDD) program led by the Glenn Research Center. The work has been supported in part by NASA under Cooperative Agreement NNL09AA00A with the National Institute of Aerospace, activity 2705-005.

The use of trademarks or names of manufacturers in this report is for accurate reporting and does not constitute an official endorsement, either expressed or implied, of such products or manufacturers by the National Aeronautics and Space Administration.

Available from:

NASA Center for AeroSpace Information
7115 Standard Drive
Hanover, MD 21076-1320
443-757-5802

Abstract

The goal of this study is to produce a methodology for evaluating the claims and arguments employed in, and the evidence produced by formal verification activities. To illustrate the process, we conduct a full assessment of a representative case study for the Enabling Technology Development and Demonstration (ETDD) program. We assess the model checking and satisfiabilty solving techniques as applied to a suite of abstract models of fault tolerant algorithms which were selected to be deployed in Orion, namely the TTEthernet startup services specified and verified in the Symbolic Analysis Laboratory (SAL) by TTTech. To this end, we introduce the Modeling and Verification Evaluation Score (MVES), a metric that is intended to estimate the amount of trust that can be placed on the evidence that is obtained. The results of the evaluation process and the MVES can then be used by non-experts and evaluators in assessing the credibility of the verification results.

1 Introduction

Formal methods have gradually stepped out of the shadows of research obscurity and are on the verge of claiming a permanent spot in the process of developing safety-critical systems. However, their widespread use in industry and increased exposure has created a dichotomy: formal methods practitioners would like to see their work getting the proper credit for contributions to the overall technology development, especially in the certification process, while certification standards are, naturally, lagging behind the pace of advances in formal methods.

Hence, even though formal verification has earned a de facto recognition in the research world and is strongly advocated even by outsiders of the trade, when formal verification is actually performed in practice, there are no guidelines on how to present a product of formal verification to non-experts and evaluators. In fact, well established methodologies for both delivering and evaluating formal verification products do not currently exist.

The goal of this study is to produce an incipient, prototypical methodology for evaluating the evidence obtained by applying formal verification techniques, such as Model Checking and Satisfiability Modulo Theories (SMT) solving. We illustrate the process by applying it to a hand-picked set of formal models of fault tolerant algorithms. In particular, we have used as primary focus the suite of abstract models of Time-Triggered Ethernet (TTE) [2] startup services developed in the Symbolic Analysis Laboratory (SAL) by TTTech and SRI International [6].

We consider that an evaluation process should comprise a sequence of typical steps, starting with correctly identifying and understanding, and then cataloging and evaluating the three main aspects of such verification products:

(i) the verification **claims**,

(ii) the modeling **assumptions**: in particular, the semantics of the abstractions that are employed and the impact they have on the verification claims.

(iii) the **evidence** to support the verification claims.

The above categories are also the foundation of the *Modeling and Verification Evaluation Score* (MVES), a metric proposed in this study that is intended to estimate the amount of trust that can be placed on the evidence that is presented. The metric is inspired by the Credibility Assessment Score (CAS), part of NASA's standard for models and simulation STD-7009 [1].

The proposed methodology can be extended to similar verification tasks, by abiding to recommended principles of evaluating models, claims, assumptions, evidence, and by following the suggested verification activities and tools for analysis and test.

A high-level representation of the evaluation process is depicted in Figure 1.

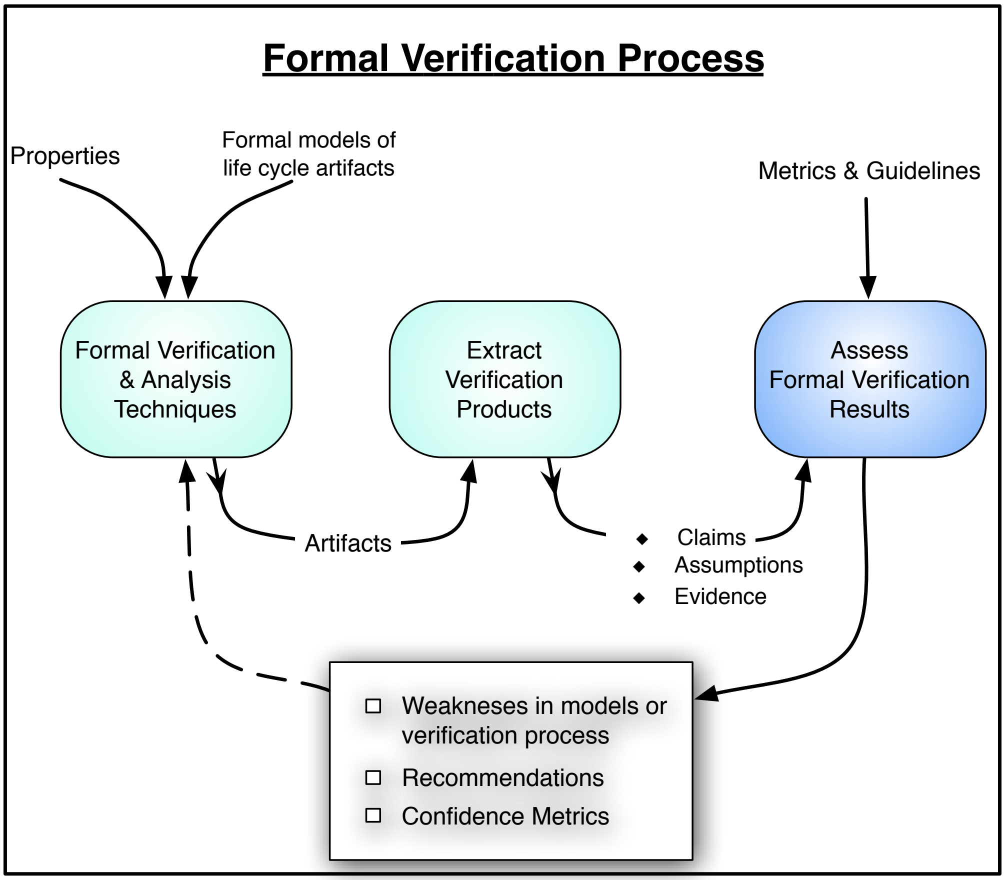
Formal Verification Process
Properties
Formal models of life cycle artifacts
Metrics & Guidelines
Formal Verification & Analysis Techniques
Extract Verification Products
Assess Formal Verification Results
Artifacts
◆ Claims
◆ Assumptions
◆ Evidence
□ Weaknesses in models or verification process
□ Recommendations
□ Confidence Metrics

Figure 1. Evaluation methodology used on a set of abstract models of the Time-Triggered Ethernet communication protocol.

We consider that this type of evaluation and analysis is of indisputable value, but it comes with its own caveats. We cannot, and do not intend to make any claims regarding the thoroughness of our approach. While efforts should be made to be as thorough as possible in collecting and documenting the elements that are subject to evaluation, achieving comprehensiveness in this regard is rarely, if ever, possible. We are very aware of the fact that claims, assumptions, and evidence may have been missed in our analysis.

Regarding the choice of an example to illustrate our proposed process, we have selected a representative application for the Enabling Technology Development and Demonstration (ETDD) program. TTEthernet is one of the advanced technologies that was selected for deployment in NASA's Orion crew exploration vehicle. This is in line with our goal of defining a specification of the verification process for a portion of the fault-tolerant avionics on a flagship mission, suitable for the Preliminary Design Review (PDR).

By picking the existing TTE models we have also made a conscious decision to select a product that is viewed as above par compared to the current state of affairs in industry, and thereby present a positive example of adherence to good practices. At the same time, we recognize the fact that our evaluation was performed in a manner that the developers of the models did not anticipate, and, therefore did not plan for. From that perspective, our evaluation should not be interpreted as imposing criticism on the TTE models but, rather, as strengthening the case for establishing an evaluation methodology via a good example, that inevitably has its share of shortcomings.

The Models

The Time-Triggered Ethernet (TTEthernet) protocol is a communication infrastructure that facilitates the use of a single physical communication infrastructure for distributed applications with mixed-criticality requirements. This is achieved via a fault-tolerant, self-stabilizing synchronization strategy, which establishes a temporal partitioning and ensures isolation of the critical dataflows from the non-critical dataflows. TTEthernet is supported by a vast array of documentation [5], including a proposed SAE aerospace standard [2].

The executable specification of the TTEthernet startup protocols that we set forth to evaluate includes three separate SAL [3] models:

- A parameterized SAL model of the TTE *startup/restart* protocol; The bounded model checker `sal-bmc` is used to establish several properties related to the eventual stabilization (synchronization) of the system in the presence of one or two omission-inconsistent faults.

- A SAL model of the *permanence function*, which is the TTE mechanism for exchanging clock values between components by taking into account transmission delays. The model checker `sal-inf-bmc`, based on the calendar automata formalism [4] is used to study the behavior of the transparent clock mechanism when static and dynamic delays are imposed on the protocol (i.e., a desired maximum transmission delay is established).

- A SAL model of the *compression function*, which computes the maximum deviation between two correct local clocks in the system (also refered to as precision) during an observation window, by using a fault tolerant average as correction value. The infinite bounded model checker is used to establish the correctness of a membership vector and the bound on the observation window size.

We note that there is currently no formal linkage between the models.

2 Claims

In this section, we catalog statements, provided by the developer in [6], that can be viewed as claims about the verification process and its outcomes. They are classified from the perspective of the incremental nature of the work. Presently, it is rarely the case that models and techniques are built from scratch. They are routinely developed based on previous experiences and artifacts. Therefore, we established the main classification of the claims as: *general*, *legacy*, or *incremental*.

A. General

[GC1] *The work is incremental: it builds on previous results.*

Page 2: “this work advances previous results by tolerating multiple failures. In particular, here our failure model allows an inconsistent faulty end-to-end communication flow which was not addressed in our previous work.”

[GC2] *The system's ability to tolerate two faults is verified.*

Page 3: "Our focus in formal methods was to facilitate a formal exploration of the algorithmic properties and to validate that the integrated system behavior was able to tolerate two faults under all system modes."

[GC3] *There are potential caveats when using non-assured tools.*

[GC4] *The soundness of the abstraction is not assumed.*

Page 3: "During the development we have also been very cautious in relying on the results of formal method tools, not only because of the potential incorrectness of non-assured tools, but also because of the need of limited reliance on results due to the abstraction required for application of formal methods."

[GC5] *The SMT solver allows the completion of the analysis.*

Page 4: "We initially used `sal-smc`, but, unfortunately, the formal analysis of stabilization from an arbitrary system state exceeded its capabilities. On the other hand, the bounded model checker `sal-bmc` now incorporates the powerful YICES SMT solver. Switching to `sal-bmc` allowed us to to finalize the TTEthernet startup/restart strategy (see Section 2.2)."

B. Legacy: from previous work

[LC1] *TTP startup has been "verified".*

Page 3: "Simplified versions of the TTP startup protocol have also been formally verified using the SAL infinite bounded model checker [DS04]."

[LC2] *Exhaustive "fault simulation" for one-fault scenarios has been performed.*

Page 3: "In [SRSP04] we present exhaustive fault simulation as SAL-based verification method using `sal-smc`. [...] we are able to show successful synchronization after network power-on, within a given power-on interval, in presence of either a faulty TTP controller or faulty central guardian."

C. Incremental: specific to the current work

[IC1] *Synchronization: the worst startup time is bounded.*

Page 27: "the derived worst case startup/restart times have their safe upper bounds at about 60 verification steps."

[IC2] *The permanence function computes a correct interval for the permanence point in time.*

Page 43: "We are interested in the relation of the dispatch point in time to the permanence point in time. [...] We expect that this property holds in case when the system is free of cumulative error [...]. Otherwise [...], we need to weaken the property as follows. [...] The test property is verified when `worst_case_cumulative` error is set to zero. Otherwise, it is falsified with a counterexample at depth seven."

[IC3] *The compression function computes a correct fault tolerant average.*

Pages 60–65: "Although, the approach is not scalable for high k it is sufficient for the verification of dual fault-tolerance."[1]

[1] Note: It is difficult to infer what the main verification claim for the compression function is. There are three properties, one abstraction, and five invariants listed as part of the verification task, but no indication

Recommendations for eliciting and evaluating claims

We can establish a core set of guidelines both for eliciting and evaluating claims. Beside the obvious recommendation of demonstrating the validity and relevance of claims, additional tasks are warranted whenever the work is deemed incremental. The correspondence between all the elements of the previous framework and the current framework has to be established, by answering the following questions:

[ICR1] Are the models representing the same, modified, or an augmented version of the protocol?

[ICR2] If abstraction is used, is the abstraction the same, similar, modified, extended?

[ICR3] What are the properties of interest and what properties are addressed?

[ICR4] Does the abstraction preserve the properties of interest?

[ICR5] In what ways are the tools and techniques the same, different, or more advanced?

3 Assumptions

TTEthernet specifies a fault-tolerant Multi-Master synchronization strategy, in which each component is configured either as a Synchronization Master (SM), Synchronization Client, or Compression Master (CM). The Synchronization Masters are the nodes (end systems) that initiate the clock synchronization service by sending an integration frame (message) to the Compression Masters. The Compression Masters, which are typically configured as network switches, collect the frames, calculate a fault-tolerant median from their timing and send a new frame back to the Synchronization Masters. All other components in the network are configured as Synchronization Clients and only react passively to the synchronization strategy. The synchronization information is exchanged in Protocol Control Frames.

The SAL model of TTEthernet startup is a synchronous composition of k Synchronization Master modules (SM), m Compression Master modules (CM), two sets of communication media modules ($k \times m$ bidirectional channels), and a diagnosis module.

We begin by identifying the assumptions used throughout the verification process. We classify these observations depending on the three distinct stages of this process: **design**, **modeling**, and **verification**. While not all observations fall in a perfectly delimited category (some may be relevant to more than one phase, or might not fit any) and not all assumptions have the same degree of impact, we see it a reasonable initial approach.

For each (explicit or implicit) assumption that is identified, we attempt to provide two evaluation criteria:

- Relevance: which captures how likely the assumption is to affect the verification results;
- Impact: which estimates how many important behaviors in the real system may be affected;

A complete set of assumptions identified in the TTE models is listed in Tables 1, 2, and 3, where we include the location (as page number reference in the report [6]), the assumption, and the two metrics: relevance and impact. For the two elements of the metric we use a coarse domain (high, medium, low, none), given that estimating precise values is usually a subjective assessment.

on what the goal (main result/theorem) is. Tables 4.1 and 4.2 in [6] “show the verification results” but only in terms of runtime, no mention on whether some/all properties/lemmas are true or false. The statements, collectively, seem to be part of a larger argument, but no larger argument is explicitly presented.

3.1 Design assumptions

We include in this category the "operational" assumptions about the system and environment.

Table 1: Design assumptions

Id	Loc	Assumption	Relev.	Impact
Faults				
DA1	11	Number of faults: two parallel	High	High
DA2	11	Number of fault scenarios: three (two faulty SMs, two faulty CMs, one faulty SM and one faulty CM)	High	High
DA3	12	Nature of faults: inconsistent-omissive for both input and output	High	High
DA4	12	Fault transience: no "short time stability" is assumed	Med	Low
DA5	12	"we decided to restrict the failure mode for SMs for two-fault tolerant configurations"	Med	High
Open Systems Interconnection (OSI) model				
DA6	6	Interference of TT services with OSI level services is ignored (they run in parallel in the same system).	Low	Low
High integrity				
DA7	11	High integrity design of switches	High	High
Permanence function				
DA8	7	"The transparent clock mechanism and the permanence function are used to mitigate network imposed jitter *almost* entirely". The transparent clock field's value "is *almost* exact". The meaning of "almost" needs to be clarified (qualitatively, if not quantitatively)	Low	Low
DA9	7	"the permanence function artificially increases the network delay from a dynamic actual to the constant maximum"	Low	Low
DA10	27	"for this approach the ideal state of the system has to restrict the faulty SM from staying in a critical state in the state machine which would potentially cause the synchronized Synchronization Master to abort synchronization."	Med	Low
Compression function				
DA11	31	"The Protocol Control Frames are transmitted on the same physical wire as the dataflow messages and, consequently, the temporal characteristics of frames belonging to dataflow and protocol control flow are not independent anymore and temporal interferences are unavoidable."	Med	Low
DA12	31	"Sources of interferences can be the end systems as well as the switches in the communication infrastructure."	Med	Med
DA13	32	"The end systems and switches in this network may impose dynamic transmission delays on frames, and in particular Protocol Control Frames"	Low	Low

Continued on next page

Table 1 – continued from previous page

Id	Loc	Assumption	Relev.	Impact
DA14	35	"However, in real world measurement errors occur, such that the permanence point in time will occur within an interval around the nominal permanence point in time."	High	Med
DA15	45	"Due to drifts of the oscillators the actual dispatch points in the SMs and consequently the permanence points in time in the CMs will deviate."	Med	Low
DA16	45	"the CMs realize a so called compression function that runs *unsynchronized* to the synchronized global time."	Low	Low
DA17	47	The definition of the *correction_value* is a fault tolerant mid-point.	Med	Low
DA18	50	"k defines the number of faulty SMs that have to be tolerated and N, the number of overall SMs required to tolerate the defined number of failures is then given by $N = 3{*}k{+}1$."	High	High

3.2 Modeling assumptions

These are restrictions imposed by the abstraction process: the modeling decisions about *what* parts of the real system/algorithm to represent in the model (as model variables), *how* to represent them (continuous, discrete, finite – including bounds, variable ranges, etc.), and their interactions (the transition relation).

A special case is the representation of **time**. The abstraction process will reduce/simplify the system and its behavior. It has to be argued in a defensible way that the abstraction preserves enough detail in order to make the verification meaningful.

The second key element of the model is the **fault injection** mechanism. The faulty (inconsistent-omissive) behavior is modeled by interposing two boolean matrices between the output of CMs and the channels, and the channels and input to SMs, respectively. The boolean values are independently set to true or false to represent whether a transmitted value is sent/received or dropped. Setting all values to true corresponds to correct behavior (absence of faults).

Table 2: Modeling assumptions

Id	Loc	Assumption	Relev.	Impact
		Time: real vs. discrete		
MA1		Time unit representation: real time or discrete time (integer counter). Example: the RT Clock Module for the permanence function.	High	High
MA2		The modeled device is clearly separated from modeling tricks.	High	Med
MA3		The impact of floating point representation on manipulating real time variables.	Med	Med
MA4		The *synchronous* module composition means certain aspects of a real system cannot be captured, such as jitter (drifts, delays, small ϵ's etc.)	High	High
		Infrastructure		

Continued on next page

Table 2 – continued from previous page

Id	Loc	Assumption	Relev.	Impact
MA5	11	CRC functions not represented	Low	Low
MA6		Connections: no intuition is given on what is the communication delay represented in the model, i.e. how long (ticks, steps, seconds) does it take for a message to travel to its destination(s).	Med	Med
Synchronization				
MA7	21	“In our studies we modeled systems with four and five SM which are sufficient to show two-fault tolerance”.	Low	Low
MA8	22	“In the model, due to the granularity of the simulation step, we simulate the sequentialized transmission as a parallel transmission.”	Med	Med
MA9	23	“Concurrent frames lead to an indeterminism in the reception order [...] Those SMs that receive the CS and the CA at the same point in time will select the CS frame and also transit to SM_FLOOD state. In the model it is, therefore, sufficient to react to the CS frames”.	Med	Low
MA10	24	“The TRANSITION part is used to delay the output until the next simulation step.”	Med	Low
MA11	30	“In terms of failure injection: if the faulty CM decides not to relay a frame to a given SM all parallel messages will be lost for this simulation step to this SM.”	Med	Low
Permanence function				
MA12	37	The meaning of the randomized parameters: `measured_transmission_delay` and `cumulative_error`.	Low	Low
MA13	42	“The first two transitions of the permanence function are functional transitions of the permanence function, the third transition is a modeling necessity to avoid the overall system from deadlocking.”	Low	Low
MA14	42	“In the formal model of the permanence function there are no implicit synchronized events and all synchronization is done via the event calendar.”	Med	Low
Compression function				
MA15	45	“This executable formal specification adds to this method a parameterizable fault-tolerance capability that leaves the number of faulty SMs to be tolerated configurable.”	Low	Low
MA16	48	“The dispatch process maintains a local timer variable”, which is discrete.	Low	Low
MA17	43	The analysis of one sender, one channel, and one receiver for the permanence function is enough to capture the concurrent nature of the protocol.	High	Med
MA18	50	“the observation window is the only other parameter that has to be assigned by hand”	Low	Med

Continued on next page

Table 2 – continued from previous page

Id	Loc	Assumption	Relev.	Impact
MA19	50	“In this example setup we set *observation_window* = 5. [...] it does not matter to which value it is set... [...] The definition of this interval contributes to the hypothetically worst-case [...]”	Low	Med
MA20	53	“The correct dispatch functions are initialized by setting their dispatch timeout to an arbitrary point within the interval defined by `earliest_correct_dispatch` and `latest_correct_dispatch`. Also, the correct dispatch functions will start execution in the wait state. The faulty dispatch processes are free to dispatch at any time.”	Low	Med
MA21	54	Meaning of “pointer” and “clock synchronization stack” are not explained.	Low	Low
MA22	55	“Note that we abstract from the transmission delays that would naturally occur in the TTEthernet network. [...] As all PCF transmissions are affected in the same way, we conclude that the particular value of the transmission delay will not have an impact on the properties we are interested verification.”	Med	Low
MA23	57	“Hereafter, the local variables will be re-initialized and the compression function is restarted. This transition will not be done in the real hardware.”	Med	Low
MA24	59	“Here, the check `reading_index` $> k + 1$ and the following transition to cm_wait state are done in the model only, again, to avoid the modeling of multiple concurrent compression functions.”	Low	Low
Code				
MA25	72	A large number of parameters are not explained, such as the 12 threshold parameters (e.g. `sm_integrate_to_sync_threshold`) which are key to understanding the transitions in Figure 2.1 on page 16.	Med	Med
MA26	73	Comments in the code suggest that some of the thresholds were adjusted, either increased or decreased, without a statement of what motivated the change and what the change implies.	Med	Low
MA27	75	The code should match the diagram in Figure 2.1, page 17.	High	Med

3.2.1 Model restrictions.

Due to scalability issues, the variable ranges used are often drastically reduced. This heavy abstraction can have direct implications in terms of interpreting the verification output.

- Very small cycle duration: 5.
- One SM state (`SM_WAIT_4_CYCLE_START_1020`) and two CM states (`CM_RELAY`, `CM_STABLE_2080`) are commented out.

3.3 Verification assumptions

The verification assumptions are concerned with three issues:

- what properties are verified
- the confidence in the tool/method itself, and
- the interpretation of the output produced by the tool(s).

Table 3: Verification assumptions

Id	Loc	Assumption	Relev.	Impact
Tool integrity				
VA1		There are no known errors in the SAL model checkers.	High	None
Method				
VA2		The iterative method is presumed sound. When a counterexample of length k if found and a counterexample of length $k+1$ is not, it is implied that $k+1$ is the worst-case scenario for reaching startup stabilization.	High	None
VA3	27	The proof depends on the meaning of an *ideal* state. Perfect synchronization of ideal states may be a stronger or weaker property than what is actually needed. More precisely, small jitter may lead to clock values differing by (say) one, while the ideal state requires identical clock values, which would mean that the property is too strong. On the other hand, counterexamples at increased depth are not excluded, which means the property (given the iterative approach using BMC) is too weak.	Med	None
VA4		Composition of the three main verification results (from the three separate models) is not pursued to determine how do they connect and influence each other.	High	None
VA5	30	"The reasoning on the convergence in case of a higher number of components has to be done informal."	High	High
Interpretation of results				
VA6	29	Translate the abstract counterexample to the real system (e.g. $RTD + CAO + RES + RTD + 3 * RES + \ldots$) into μseconds	Low	Low
VA7	67	"[...] we conclude that this was an issue of improper use rather than a tooling issue [...]"	Med	None
VA8	67	"[...] the completeness and quality metrics for this validation process are subject to further research [...]"	Med	Med
VA9	68	"SAL provides guidance in the development of the proof by producing counterexamples. This is a practical and powerful feature that allows systematically strengthening of the invariant."	Med	Med
Synchronization				
VA10	27	The correlation between the parameter `depth` and the parameter `worst_case_counter` is not very well explained.	Low	Low

Continued on next page

Table 3 – continued from previous page

Id	Loc	Assumption	Relev.	Impact
		Permanence function		
VA11	43	The two lemmas for the permanence function are rather trivial	Med	None
VA12	43	Lemma `test` is a special case of lemma `test_cumulative` for `worst_case_cumulative_error` = 0	Low	None
VA13	43	"The `test` property is verified when worst case cumulative error is set to zero. Otherwise, it is falsified with a counterexample at depth seven."	High	High
		Compression function		
VA14	45	"It has to be guaranteed that faulty SMs that may send early or late will not cause the compression function to use only a subset of PCFs from correct SMs."	Med	Low
VA15	48	Notation k is used inconsistently in the three definitions on p.47-48 It first denotes the total number of SMs, then it denotes the number of faulty SMs.	Low	Low
VA16	54	The expression `time` + `end_of_time` is an upper bound for all clock values.	Low	Low
VA17	57	"Here the restart of the one compression function is equal to the execution of a second compression function in parallel."	Low	Low
VA18	59	"Hence, we assume the calculation phase to take zero time."	Low	Low
VA19	60	The case when the clock correction value is not a multiple of the atomic unit of measurable time (one oscillation) is not treated.	Low	Low
VA20	61	"Note that the `window` and `correction` properties do not account for the message transmission overhead from the SMs to the CM. Hence in the real world the nominal `cm_compressed_pit` will occur `max_transmission` delay later than reflected in the properties above."	Low	Low
VA21	61	"We then proof the correctness of the abstraction."	High	High
		Code		
VA22	72	The use of certain parameters is not fully explained: `par_proof_implicit`, `par_proof_explicit`, `par_testcase`, `par_is_proof`, `par_is_testcase`, `par_SMSM_failures`, `par_SMCM_failures`, `par_high_integrity`, `par_standard_integrity`, `par_cm_full_cbg`.	Med	Med

3.4 Modeling recommendations.

In conjunction with the above observations, we can set forth a number of basic principles that should be followed in the modeling stage. Recommendations can be organized in a hierarchy that starts at the most general level, applicable to any model based approach, and branches down into particular activities (e.g. interactive theorem proving, model checking, SAT solving), and finally into specific tools and techniques (e.g. PVS, SAL, SMV, SPIN, etc.)

The list below refers solely to the specifics of our case study: state-machine models of clock synchronization algorithms.

[MR1] Real-time (continuous) vs digital time (discrete).

Digital components keep digital time (integers), but the specification often needs to reason about continuous time (reals). There has to be a clear separation of the two frameworks: while real time aspects may be safely used for *reasoning*, *computations* involving reals performed by digital components have to be avoided. Mixing artifacts from the two realms, especially in computation, may introduce in the model entities that are not strictly part of the model, such as elements of the test bench becoming part of the system under test. Additionally, there are concerns about the floating point arithmetic issues (approximations, error accumulation, etc.). Finally, the guiding principle should be that digital clocks cannot measure any unit of time less than one tick. “Small epsilons” do not belong in the digital framework.

[MR2] The correspondence between the specification (if it exists) and the implementation needs to be rigorously established. For example, if any high-level description of a system is given, such as charts and diagrams (in this case the diagram in Figure 2.1), the correspondence between state transition diagrams and the code needs to be rigorously established.

In this case, there are transitions in the code that do not appear in the diagram in Figure 2.1 (page 17):

From state		To state
`SM_UNSYNC`	→	`SM_TENTATIVE_SYNC`
`SM_FLOOD`	→	`SM_WAIT_4_CYCLE_START`
`SM_TENTATIVE_SYNC`	→	`SM_FLOOD`
`SM_SYNC`	→	`SM_FLOOD`
`SM_SYNC`	→	`SM_UNSYNC`
`SM_SYNC`	→	`SM_INTEGRATE`
selfloops		

[MR3] For SAL models and similar formalisms derived from state-transition systems (e.g. Petri nets, SMV, SPIN), the transition relation has to be complete, i.e. for each state, the disjunction of guards on outgoing arcs has to be equivalent to *true* (no gaps, nothing “falling through the cracks”).

[MR4] Similarly, for SAL models and other formalisms employing the guarded commands paradigm (e.g. Petri nets), any non-determinism introduced by overlapping guards has to be justified.

An example of branch completeness: the four transitions out of state `SM_INTEGRATE` have the following guards:

$$\begin{aligned}
g_1 &\equiv \neg P_1 \wedge \neg P_2 \wedge P_3 \\
g_2 &\equiv \neg P_1 \wedge \neg P_2 \wedge \neg P_3 \\
g_3 &\equiv \neg P_1 \wedge P_2 \\
g_4 &\equiv P_1
\end{aligned}$$

Where,

$$\begin{aligned}
P_1 &\equiv \exists\ \texttt{ch} : \texttt{message}[\texttt{ch}] = \texttt{coldstart\_ack} \\
P_2 &\equiv \texttt{mem2nat}(\texttt{best\_message}) \geq \texttt{sm\_integrate\_to\_sync\_threshold} \\
P_3 &\equiv \texttt{SM\_local\_timer} > 0
\end{aligned}$$

The disjunction of all guards is $g_1 \vee g_2 \vee g_3 \vee g_4$
$\equiv (\neg P_1 \wedge \neg P_2 \wedge P_3) \vee (\neg P_1 \wedge \neg P_2 \wedge \neg P_3) \vee (\neg P_1 \wedge P_2) \vee P_1$
$\equiv (\neg P_1 \wedge \neg P_2 \wedge (P_3 \vee \neg P_3)) \vee (\neg P_1 \wedge P_2) \vee P_1$
$\equiv (\neg P_1 \wedge \neg P_2) \vee (\neg P_1 \wedge P_2) \vee P_1$
$\equiv (\neg P_1 \wedge (\neg P_2 \vee P_2)) \vee P_1$
$\equiv \neg P_1 \vee P_1$
$\equiv \mathit{true}$.

[MR5] Sink states should not be artificially masked out.

4 The Evidence

Evidence can be provided by the developer or, when feasible, generated by the evaluator. In our case study, we had the luxury of having access to an automated script to instantiate, execute, and collect output from the parameterized models. Instantiating the models through the original scripts is less likely to assign inconsistent values to parameters. On top of the provided script, we have used our own shell script, that sets the arguments for the model checker (further details in section 5.2) and then runs all the instances in batch mode, instead of one by one.

Model checking output, execution traces

In model checking, the evidence generally consists of answers to temporal logic queries. The output can be as concise as a minimal "yes/no" answer to whether a property holds or not, but it is commonly accompanied by an execution trace (in the model) that explicitly proves or disproves the statement under consideration, called *witness* or, respectively, *counter-example*.

Central to clock synchronization algorithms is the representation of time and the management of clocks. An inspection of how the variables such as local clocks and timers are manipulated can give an indication on how to interpret the output traces. For the TTE models, we make the following observations.

- The variable `SM_local_clock` is updated by the transitions in the SM module, to:
  - value 0, a number of 32 times;
  - `inctime(SM_local_clock)`, a number of 9 times;
  - `inctime(smc_scheduled_receive_pit)`, which is a constant (equal to 2), a number of 3 times.

  It is expected that for the vast majority of cases, the "nominal" behavior should be to increment the value of the clock by 1; However, most transitions set the local clock to either 0 or 2. Moreover, the clock "tick" (incrementing the value) is done for only 3 out of 8 SM states: `SM_TENTATIVE_SYNC_1060`, `SM_SYNC_1070`, and `SM_STABLE_1080`.
- Similarly, `CM_local_clock` is mostly reset to 0.
- The variable `SM_local_timer` is updated in 45 instances, only twice through the "nominal" operation, `decrement_timer(SM_local_timer)`, and four times through an "unsafe" decrement `SM_local_timer' = SM_local_timer - 1`.

Table 4: Execution trace of length 30 for the TTE clock synchronization protocol with 5 Synchronization Masters.

step	SM_1		SM_2		SM_3		SM_4		SM_5	
	state	clk	state	clk	state	clk	state	clk	state	clk
0	power-up	0	power-up	0	power-up	0	power-up	0	power-up	0
1	tentative	4	tentative	1	tentative	2	tentative	4	tentative	2
2	tentative	0	unsync	2	tentative	3	tentative	0	tentative	3
3	tentative	1	unsync	0	tentative	4	tentative	1	tentative	4
4	unsync	2	sync	2	tentative	0	unsync	2	tentative	0
5	unsync	0	sync	3	tentative	1	unsync	0	tentative	1
6	unsync	0	sync	4	unsync	2	unsync	0	unsync	2
7	unsync	0	sync	0	unsync	0	unsync	0	unsync	0
8	unsync	0	sync	1	unsync	0	unsync	0	unsync	0
9	unsync	0	integrate	0	unsync	0	unsync	0	unsync	0
10	unsync	0	integrate	0	unsync	0	unsync	0	unsync	0
11	unsync	0	integrate	0	unsync	0	unsync	0	unsync	0
12	unsync	0	integrate	0	unsync	0	unsync	0	unsync	0
13	unsync	0	integrate	0	unsync	0	unsync	0	unsync	0
14	unsync	0	integrate	0	unsync	0	unsync	0	unsync	0
15	unsync	0	integrate	0	unsync	0	unsync	0	unsync	0
16	unsync	0	integrate	0	unsync	0	unsync	0	unsync	0
17	flood	0	integrate	0	flood	0	unsync	0	flood	0
18	flood	0	integrate	0	flood	0	unsync	0	flood	0
19	flood	0	integrate	0	flood	0	unsync	0	flood	0
20	unsync	0	wait2	0	wait2	0	wait2	0	wait2	0
21	unsync	0	wait2	0	tentative	0	wait2	0	tentative	0
22	unsync	0	wait2	0	tentative	1	wait2	0	tentative	1
23	unsync	0	wait2	0	unsync	2	wait2	0	unsync	2
24	unsync	0	tentative	0	unsync	0	tentative	0	unsync	0
25	unsync	0	tentative	1	unsync	0	tentative	1	unsync	0
26	unsync	0	unsync	2	unsync	0	unsync	2	unsync	0
27	unsync	0	unsync	0	unsync	0	unsync	0	unsync	0
28	unsync	0	unsync	0	unsync	0	unsync	0	unsync	0
29	unsync	0	unsync	0	unsync	0	unsync	0	unsync	0
30	unsync	0	unsync	0	unsync	0	unsync	0	unsync	0

An example of an execution trace produced by the model checker is synthesized in Table 4.[2] In the trace of length 30, the SM local clock variable `SM_local_clock[0]` is "stuck" at 0 for 27 out of 30 steps and it only "ticks" 3 times.

The risks of automation. We have generated traces of increasing length, from 5 to 70, using our own script to expedite the process. However, in one instance, the ouput did not correspond to the expected shape, which raised the issue of relying too much on automation, especially for evaluation purposes. Whenever there is a chain (or hierarchy) of automated tools or scripts, it is difficult to gauge and assign the proper degree of confidence in each component.

[2] N.B.: this is not the actual output of `sal-bmc`: we have written a C program to extract the information of interest from the large (up to and in excess of one hundred thousand lines) unstructured output file and print it in a LaTeX table.

5 Evaluation of the TTE SAL Model

5.1 Modeling

We have collected a brief list of possible violations of the modeling recommendations (MR) set forth in Section 3.4:

1. None of the SM states, except `SM_INTEGRATE`, satisfy the branch completeness property [MR3].

   Example: The four transitions out of `SM_WAIT_4_CYCLE_START_CS` have the following guards:

   $$\begin{array}{lcl} g_1 & \equiv & P_1 \\ g_2 & \equiv & P_2 \\ g_3 & \equiv & \neg P_1 \wedge P_3 \wedge P_4 \\ g_4 & \equiv & \neg P_1 \wedge P_3 \wedge \neg P_4 \end{array}$$

   Where,

   $$\begin{array}{lcl} P_1 & \equiv & \texttt{best\_message} = \texttt{coldstart} \\ P_2 & \equiv & \texttt{best\_message} = \texttt{coldstart\_ack} \\ P_3 & \equiv & \neg\exists\ \texttt{coldstart\_ack message} \\ P_4 & \equiv & \texttt{SM\_local\_timer} > 0 \end{array}$$

   The disjunction of all guards does not cover $\neg P_1 \wedge \neg P_3$, unless it is proven that $\neg P_3 \Rightarrow \neg(P_1 \vee P_2)$.

2. There are cases of branch overlap [MR4].

   Example: the guards for transitioning from `SM_TENTATIVE_SYNC` to `SM_FLOOD` and

   `SM_WAIT_4_CYCLE_START_CS`.

3. There is a stuttering step transition for both the SM and CM module, which looks like it is there to cover all the cases that are otherwise not covered. This will guarantee the absence of sink states, but it is not clear whether this safety net thrown at the end is the intended semantics or not. It is actually better to remove the safety net in order to expose sink states [MR5].

5.2 Scalability

The SAL model is a synchronous composition of k Synchronization Master modules, m Compression Master modules, two sets of communication media modules ($k \times m$ unidirectional channels), and a diagnosis module.

5.2.1 Parameters

The minimal instance of the model that can be used to study dual fault scenarios ($f = 2$) has 7 nodes. The number 7 can be viewed as $(2f + 1) + f$, where $2f + 1$ end systems (SMs) are needed to tolerate two faulty SMs, and f switches are needed to tolerate $f - 1$ CMs.

The case of $f = 2$ faulty CMs can be dismissed, but the meta-argument to support it is missing.

The components of the model are organized as follows:

Components	
Number of Synchronization Masters (SM)	5
Number of Compression Masters (CM)	2
Number of input channels to SMs	$5 * 2 = 10$
Number of input channels to CMs	$2 * 5 = 10$

Other relevant parameters	
Number of Integration Cycles	4
Cycle duration	5 ticks
Channel “buffer” size	$4 + 2 = 6$ entries
Message size	9 bits
SM/CM states	8

5.2.2 Model Size

The Yices input file (the satisfiability problem in conjunctive normal form) generated for the BMC problem of depth 67 is $\sim$ 280 MB long. The number of boolean variables involved corresponds to the number of bits needed to encode a system state, multiplied by two (“from” and “to” states in the transition relation) and then by the number of unfoldings of the transition relation which gives the BMC depth.

This total number of bits is computed from the following definitions and declarations.

Type	definition	range	#bits
TYPE_SM_ids	[1..max_SM]	[1..5]	3
TYPE_SM_number	[0..max_SM]	[0..5]	3
TYPE_SM_states	enum(8)	[0..7]	3
TYPE_channels	[1..max_channels]	[1..2]	1
TYPE_CM_ids	[1..max_channels]	[1..2]	1
TYPE_CM_states	enum(8)	[0..7]	3
TYPE_worst_case_counter	[0..200]	[0..200]	8
TYPE_integration_cycles	[0..max_integration_cycle-1]	[0..3]	2
TYPE_time	[0..(integration_cycle_duration-1)]	[0..4]	3
TYPE_timer	[0..25*integration_cycle_duration]	[0..125]	7
TYPE_membership	ARRAY TYPE_SM_ids OF BOOLEAN	[1..5]×[0..1]	5
TYPE_transparent_messages	[0..max_transparent_messages-1]	[0..5]	3
TYPE_message_type	enum(4)	[0..3]	2
TYPE_message	record(#TYPE_message_type	[0..3] +	
	TYPE_integration_cycles	[0..3] +	
	TYPE_membership#)	[1..5]×[0..1]	9
TYPE_transition_index	[1..max_number_transitions]	[1..50]	6
TYPE_transition_marker	ARRAY TYPE_transition_index OF BOOLEAN	[1..50]×[0..1]	50

Variables:

Synchronization Master		
Variable	range	# bits
SM_state	TYPE_SM_states	3
SM_local_clock	TYPE_time	3
SM_local_integration_cycle	TYPE_integration_cycles	2
SM_local_async_membership	TYPE_membership	5
message_out	ARRAY TYPE_channels OF TYPE_message	18
SM_local_timer	TYPE_timer	7
SM_local_sync_membership	TYPE_membership	5
SM_num_stable_cycles	[0..6]	3
flood_receive	BOOLEAN	1
best_message	TYPE_message	9
best_in_message	TYPE_message	9
current_async	TYPE_membership	5
Total size		70

Compression Master		
Variable	range	# bits
message_out	ARRAY TYPE_SM_ids OF ARRAY TYPE_transparent_messages OF TYPE_message	270
CM_state	TYPE_CM_states	3
CM_local_clock	TYPE_time	3
CM_local_integration_cycle	TYPE_integration_cycles	2
CM_local_timer	TYPE_timer	7
CM_local_sync_membership	TYPE_membership	5
CM_local_async_membership	TYPE_membership	5
CM_num_stable_cycles	[0..6]	3
cm_current_async	TYPE_membership	5
compressed_membership	ARRAY TYPE_transparent_messages OF TYPE_membership	30
next_message_out	ARRAY TYPE_SM_ids OF ARRAY TYPE_transparent_messages OF TYPE_message	270
next_message_out_block_cs	ARRAY TYPE_SM_ids OF ARRAY TYPE_transparent_messages OF TYPE_message	270
cm_best_in_message	TYPE_message	20
Total size		893

Connections		
Variable	range	# bits
SM_messages_in	ARRAY TYPE_SM_ids OF ARRAY TYPE_channels OF ARRAY TYPE_transparent_messages OF TYPE_message	540
CM_messages_in	ARRAY TYPE_channels OF ARRAY TYPE_SM_ids OF TYPE_message	90
connectivity_CM_in	ARRAY TYPE_channels OF ARRAY TYPE_SM_ids OF BOOLEAN	10
connectivity_CM_out	ARRAY TYPE_channels OF ARRAY TYPE_SM_ids OF BOOLEAN	10
connectivity_SM_in	ARRAY TYPE_SM_ids OF ARRAY TYPE_channels OF BOOLEAN	10
connectivity_SM_out	ARRAY TYPE_SM_ids OF ARRAY TYPE_channels OF BOOLEAN	10
Total size		670

Diagnosis		
Variable	range	# bits
worst_case_counter	TYPE_worst_case_counter	8
Total size		8

The total number of bits for representing a single system state is: $5{*}70{+}2{*}893{+}670{+}8 = 2,814$ boolean variables (bits). Hence, the unfolding of the transition relation for depth 67 requires: $2 * 67 * 2814 = 377,076$ boolean variables.

5.2.3 Runtimes

The runtime for the largest reported instance of the model is 272, 306 seconds (which is > 75 hours > 3 days). However, when running the script in an attempt to replicate the results, even though it does take more than 3 days indeed, SAL/Yices reports a total execution time of just 37 seconds. The actual runtime had to be collected by other means, e.g. with the `time` command.

We have set up an additional script to collect data for a number of instances with varying depth of the BMC problem, in order to assess the scalability of the technique. To this end, we have slightly modified the SRI script to correlate the `depth` and `worst_case_counter` parameters. The runtime (rt), in hours:minutes format, for various instances of the worst case counter (wcc) is listed in the table below.

Table 11. Scalability results for the synchronization model: model checking runtime (hh:mm) for various worst case counter values.

wcc	rt	wcc	rt
5	00:03	40	08:59
10	00:03	45	17:11
15	00:04	50	39:58
20	00:04	55	62:25
25	00:12	60	72:52
30	01:45	65	124:22
35	05:01	70	185:26

The numbers indicate that the bounded model checking technique scales reasonably well, given the size and complexity of the model.

6 Assessing the Value of the Verification Results

The evaluation methodology proposed in this report is general in nature and can be systematically applied to evaluate evidence from formal verification tasks commonly used in practice. In this section, we assess the results of the formal verification of the SAL models and TTEthernet Startup algorithms and suggest a metric, the *Modeling and Verification Evaluation Score*, to quantify the results of our assessment.

The results of an assessment are usually a combination of objective (measurable) qualities and subjective value-judgment evaluations (typically done by a group of experts or critics). For complex processes, it is rarely the case that there is a consensus on what exactly is objectively measurable and verifiable, what is relevant for an evaluation and what is not. In our case, for example, we would have to measure the degree of "correctness" of the verification process and the of amount trust that can be placed in its outcome.

The subjective part of an evaluation is most commonly referred to as a "rating." Rating systems may come with an algorithm to compute a (numeric or non-numeric) score/value, either monolithic or composite, i.e. from a number of sub-scores, on a predefined scale. Some ratings are the result of a review process, which may be purely a matter of judgment. A review scale may establish a set of criteria to be met for each level, which can be more easily observed by an indepedent evaluator.

For our purposes, we consider a composite numeric score with minimum level criteria on each component as a way to describe a verification product. We acknowledge that this approach of assigning numeric weights to subjective assessments may be unsuitable in certain situations, however, it does enable us to illustrate our methodology with a straightforward approach for summarizing and assessing the credibility of the formal verification results.

6.1 Other Evaluation Metrics

An assessment scale that is closely related to our work was proposed in the NASA standard STD-7009. The standard was developed in response to Action 4 from the 2004 report *"A Renewed Commitment to Excellence"* [7] that solicited a standard for the development, documentation, and operation of models and simulations (M&S). The Credibility Assessment Scale establishes a more rigorous framework for evaluating models for simulation.

CAS consists of eight factors grouped into three categories:

- M&S Development
  - Verification: Were the models implemented correctly? What was the numerical error uncertainty?
  - Validation: Did the M&S results compare favorably to the referent data, and how close is the referent to the real-world system?
- M&S Operations
  - Input Pedigree: What is the confidence in the input data?
  - Results Uncertainty: What is the uncertainty in the M&S results?
  - Results Robustness: How thoroughly are the sensitivities of the current M&S results known?
- Supporting Evidence
  - Use History: Have the current M&S been used successfully before?
  - M&S Management: How well managed were the M&S processes?
  - People Qualifications: How qualified were the personnel?

As the modeling phase is common to CAS and our framework, we can adapt the CAS metric by shifting the scope from simulation to verification. Our adaptation promotes elements from factors to categories and de-emphasizes or eliminates others.

6.2 The Modeling and Verification Evaluation Score (MVES)

We propose a composite score scheme, where a numeric score R is the result of composing ($\bigoplus$) n subscores: $R = \bigoplus_{i=1}^{n} s_i$. In turn, each score s_i may be a composition of n_i subscores within the same range, by a composition rule $\otimes_i$, $s_i = \otimes_{j=1}^{n_i} x_{i,j}$.

The range of scores is usually arbitrarily pre-determined (e.g., 1 to 4 for CAS). For uniformity, we prefer a universal range for all scores, such as the compact interval $[0,1]$, to which any discrete set or dense interval can be mapped via uniform scaling. For example, the CAS discrete values $0,1,2,3,4$ may be mapped to 0, 0.25, 0.5, 0.75, 1. The composition operators ($\bigoplus, \otimes$) may take various forms (additive, multiplicative, weighted sum, lower/upper bounds, etc.) depending on the circumstances.

- *Additive score*: $s_i = \sum_{j=1}^{n_i} x_{i,j}$.
- *Multiplicative score*: $s_i = \prod_{j=1}^{n_i} x_{i,j}$.
- *Weighted Sum*, a generalization of the additive score: $s_i = \sum_{j=1}^{n_i} (w_{i,j} x_{i,j})$, where the weights are in $[0,1]$ and add up to 1: $\sum_{j=1}^{n_i} w_{i,j} = 1$.
- *Average*, a special case of weighted sum, where all the weights are equal to $\frac{1}{n_i}$: $s_i = \frac{\sum_{j=1}^{n_i} x_{i,j}}{n_i}$.

1. **Design** (0.20)
    - *Specification*: are all the requirements correctly captured in the design? (0.10)
    - *Claims*: are the claims clearly stated, relevant, and valid? (0.10)
2. **Modeling** (0.40)
    - *Assumptions*: are the assumptions valid, consistent, and fully documented? (0.15)
    - *Verification*: is the system modeled correctly? are the abstractions (if any) sound? (0.15)
    - *Validation*: is the correct (intended) system modeled? (0.10)
3. **Evidence** (0.30)
    - *Scalability*: can the verification be performed on increasingly larger models? (0.10)
    - *Provability*: is the output sufficient to support the claims? (0.10)
    - *Credibility*: what is the level of confidence in the output? can it be reproduced independently? (0.10)
4. **Qualifications** (0.10)
    - *Use History*: have the tools/approaches been used successfully before? (0.05)
    - *Personnel*: how qualified were the personnel? (0.05)

Table 12. The categories of the Modeling and Verification Evaluation Score (MVES)

- *Bounds*: minimum or maximum of the values $\{x_{i,j}\}$.

For our purposes, the weighted sum is a good fit, due to its generality and flexibility. Moreover, there is a natural correspondence between the notions of relevance and impact in Section 3, and weights and subscores.

The MVES score consists of ten factors grouped into four categories ($n = 4$), as shown in Table 12, where the weight of each category and element is given in paranthesis. The weights reflect only our initial assessment, but they may be subject to further adjustments. They are nearly uniformly distributed across the ten factors, with slightly more emphasis on modeling than on personnel qualifications.

Full details on the elements and levels of MVES are listed in Table 13.

The score obtained by applying the MVES metric to the TTE models is listed in Table 14.

7 Future Work

The preliminary work presented here can be extended in many ways. The framework can be used in several other contexts where formal verification is performed. The spectrum of verification techniques has diversified tremendously in recent years, therefore a boilerplate approach can no longer be applied. We intend to investigate the specific differences when applying other archetypal approaches, including theorem proving, SMT solving, SAT solving, for timed, embedded and hybrid system verification.

Level	Specification	Claims	Assumptions	Verification	Validation
1.00	Complete formal specification	Complete and unambiguously formulated claims	Fully documented, no limiting assumptions	Full models exhaustively verified	No discrepancies vs specification and requirements
0.75	Semi-formal specification	Indirectly formulated claims	Few limiting assumptions	Reduced or abstract instances fully verified	Indirect validation, via transformations
0.50	Incomplete or partial specification	Semi-formal description, traceable to models	Partial implementation	Partial verification, simulation, or testing	Partial validation
0.25	Informal description	Can be inferred from informal statements	Incomplete implementation	Qualitative estimates	Discrepancies with specification
0.00	Insufficient	Insufficient	Insufficient	Insufficient	Insufficient

Level	Scalability	Provability	Credibility	Tools	Personnel
1.00	Sufficiently large instance verified	Fully general approach + automation	Code, proofs, scripts available, fully reproducible	Advanced, robust, validated techniques	Extensive experience
0.75	Large instances + meta-arguments	General sub-class of models can be verified in practice	Pseudo-code or equivalent	Experimental techniques or standard recommended practices	Advanced degree or good experience
0.50	Small instances, base cases can be verified	Partial or incomplete proof	Partial evidence	Non-standard or obsolete technique	Formal training experience and recommended practice training
0.25	Qualitative estimates	Informal proof	Conceptual proof of concept	Some qualitative evidence or expert opinion	Engineering or science degree
0.00	Insufficient	Insufficient	Insufficient	Insufficient	Insufficient

Table 13. Elements and levels of MVES

Table 14. MVES score for the TTEthernet models

Category	Weight	Subsc.	Explanation	Score
Specification	0.10	1.00	Executable formal specification	0.10
Claims	0.10	0.80	Indirectly stated	0.08
Assumptions	0.15	0.70	Limiting abstractions employed	0.10
Verification	0.15	0.80	Small number of violations	0.12
Validation	0.10	1.00	Correct system modeled	0.10
Scalability	0.10	0.50	Runtimes: hours to days	0.05
Provability	0.10	0.80	Additional meta-arguments needed	0.08
Credibility	0.10	1.00	Results fully reproducible	0.10
Use History	0.05	1.00	Good track record, state-of-the-art	0.05
Personnel	0.05	1.00	Extensive expertise	0.05
Total				**0.83**

References

1. National Aeronautics and Space Administration. Standard for models and simulation, NASA-STD-7009, Nov 2008.

2. SAE International AS-2 Embedded Computing Systems Committee. SAE AS6802 - Time-Triggered Ethernet, version 1.0.26, Jan 2010.

3. Leonardo de Moura, Sam Owre, and Natarajan Shankar. The SAL Language Manual. Technical Report SRI-CSL-01-02, CSL Technical Report, 2003.

4. Bruno Dutertre and Maria Sorea. Modeling and verification of a fault-tolerant real-time startup protocol using calendar automata. In *Formal Techniques, Modeling and Analysis of Timed and Fault-Tolerant Systems, Joint International Conferences on Formal Modeling and Analysis of Timed Systems (FORMATS/FTRTFT)*, pages 199–214, 2004.

5. Hermann Kopetz, Astrit Ademaj, Petr Grillinger, and Klaus Steinhammer. The time-triggered ethernet (TTE) design. In *Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*, pages 22–33, Seattle, WA, May 2005.

6. Wilfried Steiner. TTEthernet Executable Formal Specification. Technical Report Contract no. 236701, Marie Curie International Outgoing Fellowship – Complexity Management for Mixed-Criticality Systems (CoMMiCS), 2009.

7. The “Diaz” Report. A Renewed Commitment to Excellence: An Assessment of the NASA Agency-wide Applicability Investigation Board Report, Jan 2004.

Table 15. List of Acronyms

BMC	Bounded Model Checking
CAS	Credibility Assessment Score
CM	Compression Master
DA	Design Assumption
GC	General Claim
IC	Incremental work Claim
ICR	Incremental Claim Recommendation
LC	Legacy work Claim
MA	Modeling Assumption
MR	Modeling Recommendation
MVES	Modeling and Verification Evaluation Score
SAL	Symbolic Analysis Laboratory
SAT	Satisfiability Checking
SM	Synchronization Master
SMT	Satisfiability Modulo Theories
TTE	Time-Triggered Ethernet
TTP	Time-Triggered Protocol
VA	Verification Assumption

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 01- 11 - 2011	**2. REPORT TYPE** Technical Memorandum	**3. DATES COVERED** *(From - To)* 01/2011–07/2011

4. TITLE AND SUBTITLE A Methodology for Evaluating Artifacts Produced by a Formal Verification Process	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Siminiceanu, Radu I.; Miner, Paul S.; Person, Suzette	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER 402600.04.04.04
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) NASA Langley Research Center Hampton, VA 23681-2199	**8. PERFORMING ORGANIZATION REPORT NUMBER** L-20068
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001	**10. SPONSOR/MONITOR'S ACRONYM(S)** NASA
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) NASA/TM-2011-217193

12. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified Unlimited
Subject Category 59
Availability: NASA CASI (443) 757-5802

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The goal of this study is to produce a methodology for evaluating the claims and arguments employed in, and the evidence produced by formal verification activities. To illustrate the process, we conduct a full assessment of a representative case study for the Enabling Technology Development and Demonstration (ETDD) program. We assess the model checking and satisfiabilty solving techniques as applied to a suite of abstract models of fault tolerant algorithms which were selected to be deployed in Orion, namely the TTEthernet startup services specified and verified in the Symbolic Analysis Laboratory (SAL) by TTTech. To this end, we introduce the Modeling and Verification Evaluation Score (MVES), a metric that is intended to estimate the amount of trust that can be placed on the evidence that is obtained. The results of the evaluation process and the MVES can then be used by non-experts and evaluators in assessing the credibility of the verification results.

15. SUBJECT TERMS

certification; formal methods; formal specification; software lifecycle; software quality

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			STI Help Desk (email: help@sti.nasa.gov)
					19b. TELEPHONE NUMBER ***(Include area code)***
U	U	U	UU	28	(443) 757-5802