# Review of "Just-in-Time Certification"

John Rushby

*12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS), 2007*

Francesco Mucci

Corso di Laurea Magistrale in Informatica

**Sistemi Critici e Real Time**

Anno accademico 2016-2017

UNIVERSITÀ DEGLI STUDI FIRENZE

## Outline

## Certification

### Certification

provides assurance that deploying a given system does not pose an unacceptable risk of adverse consequences.

is a judgment based on a body of materials that, implicitly or explicitly, consists of three elements:

- claims:
  properties of the system $\Leftarrow$ {adverse consequences, tollerable risk lvl}
- evidence:
  anlysis, review, test;
- argument:
  that justify claims, based on documented evidence.

## Certification

Current methods of assurance explicitly depend on:

- standards and regulations:
  - e.g. DO-178B for airborne software, or IEC-61508 for programmable devices;
- rigorous examination of the whole, finished system.

**Motivation for new approaches:**

modern business practices (e.g., outsourcing):

⇒ system developer and the certifier don't have full visibility into subsystems:

  ⇒ a transition to compositional certification of systems based on certified components.

## Certification

Current methods of assurance explicitly depend on:

- standards and regulations:
  - e.g. DO-178B for airborne software, or IEC-61508 for programmable devices;
- rigorous examination of the whole, finished system.

**Two approaches to assurance:**

- standards-based;
- goal-based.

## Standards-Based Certification

Applicant:

1. follows a prescribed method;
2. delivers prescribed outputs:
   - e.g. documented requirements, designs, analyses, tests and outcomes, traceability among these.

Standard defines only the evidence to be produced:

$\Rightarrow$ the claims and arguments are **implicit**.

**Motivation for Standards-Based Certification:**

Works well in fields that are stable or change slowly:

$\Leftarrow$ institutionalize lessons learned.

## Standards-Based Certification

Applicant:

1. follows a prescribed method;
2. delivers prescribed outputs:

   e.g. documented requirements, designs, analyses, tests and outcomes, traceability among these.

Standard defines only the evidence to be produced:

$\Rightarrow$ the claims and arguments are **implicit**.

**Motivation for new approaches:**

1. less suitable with novel problems, solutions, methods;
2. defines only the evidence to be produced:

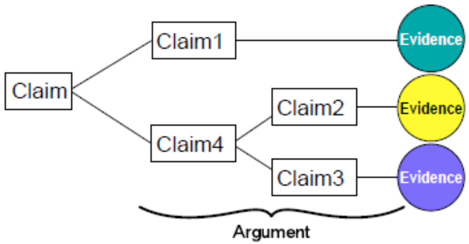
   $\Rightarrow$ is hard to tell whether given evidence meets the intent.

## Goal-Based Certification

Applicant develops an assurance case:

- a structured demonstration that a system is acceptably safe, secure, reliable, etc.;

A comprehensive presentation of **evidence** linked (by **argument**) to a **claim**



**IF** ● **THEN** Claim1; **IF** ● **THEN** Claim2; **IF** ● **THEN** Claim3;
**IF** Claim2 **and** Claim3 **THEN** Claim4; **IF** Claim1 **and** Claim4 **THEN** Claim

## Goal-Based Certification

Applicant develops an assurance case:

- a structured demonstration that a system is acceptably safe, secure, reliable, etc.;
  - whose outline form may be specified by standards;
  - in which the claims, evidence, and argument are presented **explicitly**;

The case is evaluated by independent assessors:

$\Leftarrow$ explicit claims, evidence, argument.

## A "Science of Certification": favor goal-based approaches

Certification is ultimately a judgment:

- judgment should be based on rational argument supported by adequate explicit and credible evidence.

**"Science of Certification":**

would be about ways to develop that argument and evidence.

**Favor goal-based over standards-based approaches:**

- at the very least, expose and examine the claims, arguments and assumptions implicit in standards.

## A "Science of Certification": favor goal-based approaches

Certification is ultimately a judgment:

- judgment should be based on rational argument supported by adequate explicit and credible evidence.

**"Science of Certification":**

would be about ways to develop that argument and evidence.

**Favor goal-based over standards-based approaches:**

- at the very least, expose and examine the claims, arguments and assumptions implicit in standards.

## A "Science of Certification": use formal ("machinable") design descriptions

Methods of certification (similar to formal verification):

1. anticipate all possible state that can arise in the interaction of the system with its environment;
2. show that none of them pose unacceptable risk of adverse consequences.

**Safety analysis methods:**

- hazard analysis;
- fault tree analysis (FTA);
- failure modes and effects analysis (FMEA).

## A "Science of Certification": use formal ("machinable") design descriptions

Methods of certification (similar to formal verification):

1. anticipate all possible state that can arise in the interaction of the system with its environment;
2. show that none of them pose unacceptable risk of adverse consequences.

**Safety analysis methods:**

applied by hand to informal descriptions (documents describing requirements, specifications, and assumptions) of the design and environment of the system.

## A "Science of Certification": use formal ("machinable") design descriptions

Methods of certification (similar to formal verification):

1. anticipate all possible state that can arise in the interaction of the system with its environment;
2. show that none of them pose unacceptable risk of adverse consequences.

**Use formal "machinable" design descriptions (aka model-based descriptions):**

$\Rightarrow$ can then use automated analysis methods for certification;

$\Rightarrow$ automated requirements development and analysis;

$\Rightarrow$ be sure that the lower levels of design and implementation do not introduce new risks.

## A "Science of Certification": favor multi-legged cases supported by Bayesian analysis

Different levels of criticality:

- Levels A to E in DO-178B
- Evaluation Assurance Levels (EALs) in the Common Criteria
- Safety Integrity Levels (SILs) in IEC-61508

**More evidence required at higher Levels/EALs/SILs:**

what's the argument that these deliver increased assurance?

1. An implicit appeal to diversity;
2. belief that diverse methods fail independently;
    - not true in n-version software, should be viewed with suspicion here too.

## A "Science of Certification": favor multi-legged cases supported by Bayesian analysis

**More evidence required at higher Levels/EALs/SILs:**

Need to know the arguments supported by each item of evidence, and how they compose. Want to distinguish rational multiple forms of evidence from "nervous demands for more and more".

**Bayes theorem:**

- evidence $E$ supporting a claim $C$;
- measure of the extent to which $E$ supports $C$ = relation between $P(C|E)$ and $P(C)$ (subjective probabilities that indicate the strength of our posterior and prior beliefs in the claim $C$).

The task is more complicated when we have multiple items of evidence.

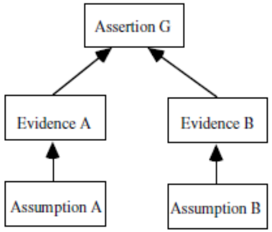
## A "Science of Certification": favor multi-legged cases supported by Bayesian analysis

The task is more complicated when we have multiple items of evidence.

### Multi-legged arguments:

use of diversity in arguments;

- "two-legged" argument: evidence from testing is combined with that from formal verification and the conclusion or claim is based on both.



**The structure of a two-legged argument in support of a claim *G***

## A "Science of Certification": favor multi-legged cases supported by Bayesian analysis

**Bayesian Belief Networks (BBNs):**

the nodes of the graph represent judgments about components of the argument and the arcs indicate dependencies between these.

BBNs allow us to explore our beliefs about the extent to which multiple items of evidence support a claim: under some combinations of prior belief, increasing the number of failure-free tests may decrease our confidence in the test oracle rather than increase our confidence in system reliability.

**Be wary of demands for more and more evidence, with implicit appeal to diversity and independence:**

$\Rightarrow$ favor explicit multi-legged cases supported by Bayesian analysis.

## A "Science of Certification": conclusions

Scientifically principled approach to certification:

- goal-based approach,
- that makes extensive use of unconditional claims delivered by automated formal methods,
- and that combines these with other evidence to form multi-legged assurance cases
- supported by Bayesian analysis.

## Compositional Certification: motivation and problems

Certification authorities consider only whole systems.

- Subcontracting and component-based development:
  - $\Rightarrow$ little visibility into subsystem designs.

**Compositional Certification:**

compositional approach to certification, in which components can be separately certified and systems using these do not need to reexamine their content.

It's the interactions that matter :

1. safety is a system (considered in its entirety) property;
2. compositional reasoning do not extend to certification issues;
3. compositional extensions to certification are too onerous or lack credibility.

## Compositional Certification: motivation and problems

Certification authorities consider only whole systems.

- Subcontracting and component-based development:
  - $\Rightarrow$ little visibility into subsystem designs.

**Compositional Certification:**

compositional approach to certification, in which components can be separately certified and systems using these do not need to reexamine their content.

It's the interactions that matter :

1. safety is a system (considered in its entirety) property;
2. compositional reasoning do not extend to certification issues;
3. compositional extensions to certification are too onerous or lack credibility.

## Compositional Certification: extending compositional verification to certification

Compositional verification of programs:

- Program $A$ guarantees $P$ if environment ensures $Q$;
- Program $B$ guarantees $Q$ if environment ensures $P$;
- Conclude that $A||B$ guarantees $P$ and $Q$.

Programs interact only through explicit computational mechanisms: shared variables.

But, software and systems can interact through other mechanisms:

- computational context: shared resources;
- noncomputational mechanisms: the controlled plant.

Compositional certification is harder than verification.

## Compositional Certification: extending compositional verification to certification

Compositional verification of programs:

- Program $A$ guarantees $P$ if environment ensures $Q$;
- Program $B$ guarantees $Q$ if environment ensures $P$;
- Conclude that $A||B$ guarantees $P$ and $Q$.

Programs interact only through explicit computational mechanisms: shared variables.

**But, software and systems can interact through other mechanisms:**

- computational context: shared resources;
- noncomputational mechanisms: the controlled plant.

Compositional certification is harder than verification.

## Why Compositional Verification does not extend to Certification?
## Unintended interaction through shared resources

Compositional verification of correctness assumes the integrity of the analyzed components:

**Verifying a protocol with sender ($S$) and receiver ($R$):**

verification of $S$ performed on the basis of assumptions about $R$:

- including failure modes;
    - $R$ may drop or duplicate messages, respond unpredictably;
- not include that a failure in $R$ disrupts integrity and functionality of $S$ through shared resources;
    - $S$ and $R$ share a processor with inadequate isolation between processes $\Rightarrow$ malfunctioning $R$ can write into the program or data memory of $S$, or to monopolize the CPU.

## Why Compositional Verification does not extend to Certification? Unintended interaction through shared resources

Compositional verification of correctness assumes the integrity of the analyzed components:

**Verifying a protocol with sender ($S$) and receiver ($R$):**

verification of $S$ performed on the basis of assumptions about $R$:

- including failure modes;
    - $R$ may drop or duplicate messages, respond unpredictably;
- not include that a failure in $R$ disrupts integrity and functionality of $S$ through shared resources;
    - $S$ and $R$ share a processor with inadequate isolation between processes $\Rightarrow$ malfunctioning $R$ can write into the program or data memory of $S$, or to monopolize the CPU.

## Compositional Certification: avoiding unintended interaction through shared resources

**Integration framework (i.e., an architecture) that guarantees:**

- Composability: properties of a component are preserved when it is used within a larger system;
- Compositionality: properties of a system can be derived from those of its components.

Partitioning mechanism creates an environment in which:

- application subsystems cannot interfere with one another, but they can cooperate;
- guarantees the integrity of these subsystems.

We can now reason about their composition using methods for compositional verification?

## Compositional Certification: avoiding unintended interaction through shared resources

**Integration framework (i.e., an architecture) that guarantees:**

- Composability: properties of a component are preserved when it is used within a larger system;
- Compositionality: properties of a system can be derived from those of its components.

Partitioning mechanism creates an environment in which:

- application subsystems cannot interfere with one another, but they can cooperate;
- guarantees the integrity of these subsystems.

We can now reason about their composition using methods for compositional verification?

## Why Compositional Verification does not extend to Certification? Unintended interaction through the controlled plant

Certification of a subsystem must be performed relative to a model of its plant and physical environment.

**Plant models = hybrid systems:**

- state machines + differential equations over real-valued variables;

$\Rightarrow$ extend compositional verification from state machines to hybrid systems.

- Methods for formal analysis of hybrid systems;

  e.g. hybrid abstraction, hybrid assume-guarantee reasoning;

- methods for computing invariants of hybrid systems;

$\Rightarrow$ develop and adapt these methods for certification: feasible, but additional work is needed.

## Why Compositional Verification does not extend to Certification?
## Unintended interaction through the controlled plant

Certification of a subsystem must be performed relative to a model of its plant and physical environment.

**Plant models = hybrid systems:**

- state machines + differential equations over real-valued variables;

⇒ extend compositional verification from state machines to hybrid systems.

- Methods for formal analysis of hybrid systems;
  e.g. hybrid abstraction, hybrid assume-guarantee reasoning;
- methods for computing invariants of hybrid systems;

⇒ develop and adapt these methods for certification: feasible, but additional work is needed.

## Why Compositional Verification does not extend to Certification? Domino behavior of cascading faults

Formal methods for compositional verification need to deal with systematic avoidance of a cascading faults in the case of component failures.

**Combination of faulty air data subsystem and the autopilot:**

autopilot can detect or mask failure of air data subsystem, it can still function, but possibly in a degraded manner;

$\Rightarrow$ deal with degraded behaviors.

- Control:
  1. number of fault cases;
  2. domino behavior of cascading faults.

## Why Compositional Verification does not extend to Certification? Domino behavior of cascading faults

Formal methods for compositional verification need to deal with systematic avoidance of a cascading faults in the case of component failures.

**Combination of faulty air data subsystem and the autopilot:**

autopilot can detect or mask failure of air data subsystem, it can still function, but possibly in a degraded manner;

$\Rightarrow$ deal with degraded behaviors.

- Control:
  1. number of fault cases;
  2. domino behavior of cascading faults.

## Compositional Certfication: conclusions

Compositional certification framework that builds system-level assurance cases from component-level subcases is urgently required;

- big challenge for regulatory agencies.

**Future systems will:**

- exist in many configurations,
- be reconfigurable,
- undergo major evolution during their lifetimes;
  e.g. NASA's Project Constellation;

$\Rightarrow$ Compositional certification should extend to incremental certification, reuse, and modification.

## Compositional Certfication: conclusions

Compositional certification framework that builds system-level assurance cases from component-level subcases is urgently required;

- big challenge for regulatory agencies.

**Future systems will:**

- exist in many configurations,
- be reconfigurable,
- undergo major evolution during their lifetimes;

  e.g. NASA's Project Constellation;

$\Rightarrow$ Compositional certification should extend to incremental certification, reuse, and modification.

## Just-In-Time Certification: motivation

Final configuration of many systems is now determined later than the time of design and certification:

- Configuration determined at load time (e.g., Kernel configuration);
- Self-assembly Service Oriented Architecture;
- Artificial Intelligence planning;
- Runtime adaptation/reconfiguration and learning.

$\Rightarrow$ How can these be certified?

## Just-In-Time Certification: principles

**"Just in time certification":**

At installation time, a single configuration (of many)

- certification consider that configuration,
- directly checking its attributes against requirements.

$\Rightarrow$
- mechanize this check,
- transfer it to load time or runtime.

$\Rightarrow$ Certain elements (automated calculations rather than human judgments) of a certification can be transferred to runtime.

## Just-In-Time Certification: principles

$\Rightarrow$ Certain elements (automated calculations rather than human judgments) of a certification can be transferred to runtime.

Scientific certification uses formal methods to calculate and analyze reachable states at design time.

**Just-In-Time Certification:**

use these methods to synthesize:

- monitors that check behavior at runtime ("runtime verification");
- the component interactions themselves;
- controllers to generate safe behavior ("controller synthesis").

## Just-In-Time Certification: synthesis of monitors

**Software function that uses a critical resource:**

Safety requirement: interactions with the resource must follow a certain protocol.

Design-time certification approach:
formally verify that the component follows the protocol.

Just-in-time certification approach:
generate a runtime monitor:

- synthesized from the model that specifies the protocol (using similar techniques as those used in formal verification);
- that blocks any interactions that violate the protocol.

## Just-In-Time Certification: synthesis of monitors

**Software function that uses a critical resource:**

Safety requirement: interactions with the resource must follow a certain protocol.

Design-time certification approach:
formally verify that the component follows the protocol.

Just-in-time certification approach:
generate a runtime monitor:

- synthesized from the model that specifies the protocol (using similar techniques as those used in formal verification);
- that blocks any interactions that violate the protocol.

## Just-In-Time Certification: synthesis of the component interactions

Design-time composition:
each component has knowledge about the components with which it interacts,

$\Rightarrow$ monitor are synthesized at design time and applied at runtime.

Runtime composition (adaptive systems):
knowledge about the components must be acquired dynamically,

$\Rightarrow$ Components publish a model of their capabilities and requirements, these can be used to check or synthesize component interactions.

- Components can adapt their behavior $\Rightarrow$ publish a dynamic model: a component that is operating in a degraded mode will publish a different model than one that is functioning normally.

## Just-In-Time Certification: synthesis of the component interactions

Design-time composition:
each component has knowledge about the components with which it interacts,

⇒ monitor are synthesized at design time and applied at runtime.

Runtime composition (adaptive systems):
knowledge about the components must be acquired dynamically,

⇒ Components publish a model of their capabilities and requirements, these can be used to check or synthesize component interactions.

- Components can adapt their behavior ⇒ publish a dynamic model: a component that is operating in a degraded mode will publish a different model than one that is functioning normally.

## Just-In-Time Certification: synthesis of the component interactions

Design-time composition:
each component has knowledge about the components with which it interacts,

$\Rightarrow$ monitor are synthesized at design time and applied at runtime.

Runtime composition (adaptive systems):
knowledge about the components must be acquired dynamically,

$\Rightarrow$ Components publish a model of their capabilities and requirements, these can be used to check or synthesize component interactions.

- Components can adapt their behavior $\Rightarrow$ publish a dynamic model: a component that is operating in a degraded mode will publish a different model than one that is functioning normally.

## Just-In-Time Certification: full behavioral synthesis

Full behavioral synthesis = "Controller synthesis problem" specialization of

**"General synthesis problem":**

- environment of a component: sum of the models published by the components with which it interacts;
- problem: constructing a transition system satisfying a given temporal logic property $P$;
- game theoretic solution: synthesized system has a strategy for satisfying $P$ no matter what inputs are given by the environment.

## Just-In-Time Certification: full behavioral synthesis

Full behavioral synthesis = "Controller synthesis problem"

**Controller synthesis:**

- the plant $T(\vec{r}, \vec{y})$, is fixed,
- the task is to construct the controller $R(\vec{y}, \vec{r})$ such that
- $R \parallel T \models P$: for any transition of $T$, the controller generates a control input $\vec{r}$ that maintains the temporal property $P$.

- Can be solved using techniques based on model checking.
- Calculation of a winning strategy requires a search over a large space.

## Just-In-Time Certification: runtime assurance

- Use run-time monitoring or synthesis of behavior from models (instead of design-time analysis of implementation):
  - only consider the current state and reachable states.
- Synthesis driven by model published by interacting components; their methods of construction analyzed at design time to demonstrate that:
  - each component publishes models that are consistent with its actual behavior,
  - models are consistent across interacting components,
  - composite behavior of the interacting models satisfies the required safety (or other certification) claims.
- Instead of using model checking and other formal methods for analysis, we use them for monitoring and synthesis.

## Just-In-Time Certification: runtime assurance

- Use run-time monitoring or synthesis of behavior from models (instead of design-time analysis of implementation):
  - only consider the current state and reachable states.
- Synthesis driven by model published by interacting components; their methods of construction analyzed at design time to demonstrate that:
  - each component publishes models that are consistent with its actual behavior,
  - models are consistent across interacting components,
  - composite behavior of the interacting models satisfies the required safety (or other certification) claims.
- Instead of using model checking and other formal methods for analysis, we use them for monitoring and synthesis.

## Just-In-Time Certification: runtime assurance

- Use run-time monitoring or synthesis of behavior from models (instead of design-time analysis of implementation):
  - only consider the current state and reachable states.
- Synthesis driven by model published by interacting components; their methods of construction analyzed at design time to demonstrate that:
  - each component publishes models that are consistent with its actual behavior,
  - models are consistent across interacting components,
  - composite behavior of the interacting models satisfies the required safety (or other certification) claims.
- Instead of using model checking and other formal methods for analysis, we use them for monitoring and synthesis.

## Just-In-Time Certification: conclusion

- Certification examines the models, trusts the synthesis of monitor and controller.

*We could imagine the synthesis generating a certificate, rather as some theorem proving techniques can generate an independently checkable "proof object". Such a certificate would truly be "just-in-time certification".*

## Conclusions: paper achievements and future works

New methods are needed for effective and cost-effective certification of modern systems.

1. Outlined a "scientific" approach to certification;

requires: more powerful formal analysis tools for an industrial application.

2. Suggested how that "scientific" approach can be extended to compositional certification;

requires: some worked examples and powerful automation for compositional verification over hybrid systems.

3. argued that monitoring or synthesis from verified models can support "just-in-time" certification;

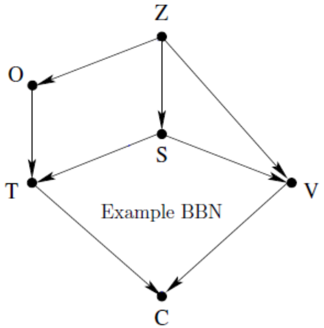
requires: more scalable methods for controller synthesis.

## Bibliography

- John Rushby, "Just-in-Time Certification", *12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS)*, 2007.
- John Rushby, "The Interpretation and Evaluation of Assurance Cases", *Technical Report SRI International*, 2015.

Grazie per l'attenzione.

## A "Science of Certification": BBN example



**Z:** System Specification
**O:** Test Oracle
**S:** System's true quality
**T:** Test results
**V:** Verification outcome
**C:** Conclusion

T is a Boolean variable that represents the outcome of testing. It depends on both the oracle O and the true quality of the system S. Its probability distribution over these is represented by a joint probability table such as the following, which gives the probabilities that the test outcome is judged successful.

| Correct System | | Incorrect System | |
|---|---|---|---|
| Correct Oracle | Bad Oracle | Correct Oracle | Bad Oracle |
| 100% | 50% | 5% | 30% |

## A "Science of Certification": use of unconditional claims derived by automated formal methods

**Unconditional claims:**

of the form "$x$% confidence that this property holds unconditionally" rather than "$y$% confidence that this property holds with probability $z$%".

Automated formal methods (e.g., static analysis) can provide evidence for many unconditional properties.

**Using Multi-legged arguments based on this kind of evidence:**

1. "add up" easily, they deliver the conjunction of their individual properties as an unconditional claim;
2. BBNs analysis is simplified, and anomalies disappear.

## Compositional Certification: partitioning mechanism

Avoiding unintended interaction through shared resources:

Partitioning mechanism ensures composability of components:

- properties of an application subsystem are unchanged when it is composed with a partitioning mechanism;
- components cannot interfere with each other nor with the partitioning mechanisms;

Partitioning mechanism ensures compositionality of system:

- partitioning mechanisms compose with each other additively:
  - e.g. partitioning(kernel) + partitioning(network) provides partitioning(kernel + network);

## Compositional Certification: partitioning mechanism

Avoiding unintended interaction through shared resources:

Partitioning mechanism ensures composability of components:

- properties of an application subsystem are unchanged when it is composed with a partitioning mechanism;
- components cannot interfere with each other nor with the partitioning mechanisms;

Partitioning mechanism ensures compositionality of system:

- partitioning mechanisms compose with each other additively:
  - e.g. partitioning(kernel) + partitioning(network) provides partitioning(kernel + network);