Towards Increased Efficiency and Confidence in Process Compliance

Julieth Patricia Castellanos Ardila, Barbara Gallina

{julieth.castellanos, barbara.gallina}@mdh.se

This work is supported by the EU and VINNOVA via the **ECSEL JU project AMASS (No. 692474)**

24th EuroSPI Conference
VŠB – Technical University of Ostrava, Czech Republic
8 September 2017

Context and motivation

Process Reference Models

Safety Critical Systems → **Safety Standards Certification**

Evidence:
- **Arguments**
- **Proof of compliance**

Time consuming
Delicate
Costly

Compliance management

Automation
Systematic reuse

Confidence in the evidence

Logic-based approaches

Talk outline

- Background
- SoPLE&Logic-basedCM
- Applying SoPLE&Logic-basedCM
- Lessons learnt
- Related work
- Conclusion and future work

Background(1)

AUTOMOTIVE SPICE®

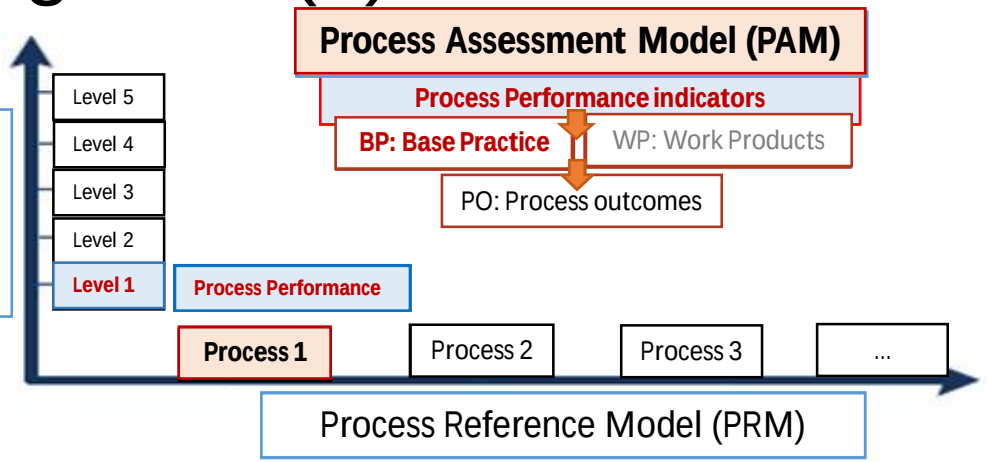
SWE.3

Software Detailed Design and Unit Construction

Base Practices (BP)	
BP1	Develop software detailed design
BP2	Define interfaces of software units
BP3	Describe dynamic behavior
BP4	Evaluate software detailed design
BP5	Establish bidirectional traceability
BP6	Ensure consistency
BP7	Communicate agreed software detailed design
BP8	Develop software units

Measurement framework:

- Capability Levels
- Process attributes

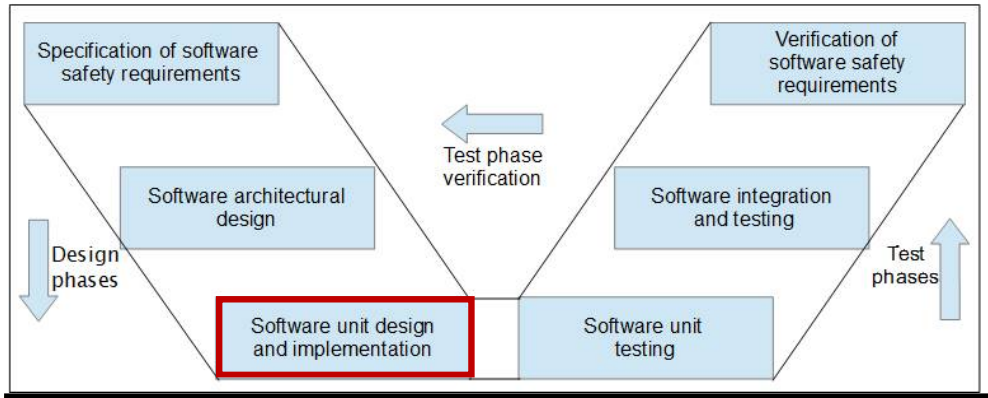


Process Outcome Description (PO)	
PO1	A detailed design is developed that describes software units
PO2	Interfaces of each software unit are defined
PO3	The dynamic behavior of the software units is defined. NOTE: Not all software units have dynamic behavior
PO4	Evaluate the software detailed design
PO5	Consistency and bidirectional traceability are established. NOTE: Consistency is supported by bidirectional traceability
PO6	The software detailed design and the relationship to the software architectural design is agreed and communicated to all affected parties
PO7	Software units defined by the software detailed design are produced.

Background(2)

ISO 26262

ASIL: Automotive Safety Integrity Levels



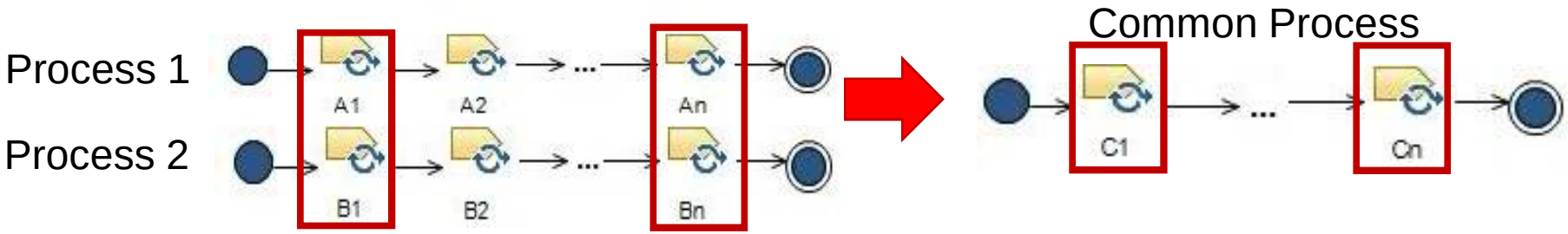
Software Unit Design and Inplementation

Activities(A)	
A1	Specify the software units
A2	Verify the software unit desing
A3	Implement the software units
A4	Verify the software unit implementation

Requirements (R)	
R1	The requirements of this subclase shall be complied with if the software unit is safety-related
R2	Software units are designed by using a notation that depends on the ASIL and the recommedation levels
R3	The specification of the software units shall describe the functional behavior and the internal design to the level of detail necesary for their implementation
R4	Design principles for software unit design and implementation shall be applied depending on the ASIL and the recommendation levels
R5	Software unit design and implementation are verified by applying verification methods accordin to the ASIL and recomendations levels.
R6	When ASIL and recommendation levels are not followed, a rationale that explain the reasons for this behavior must be provided

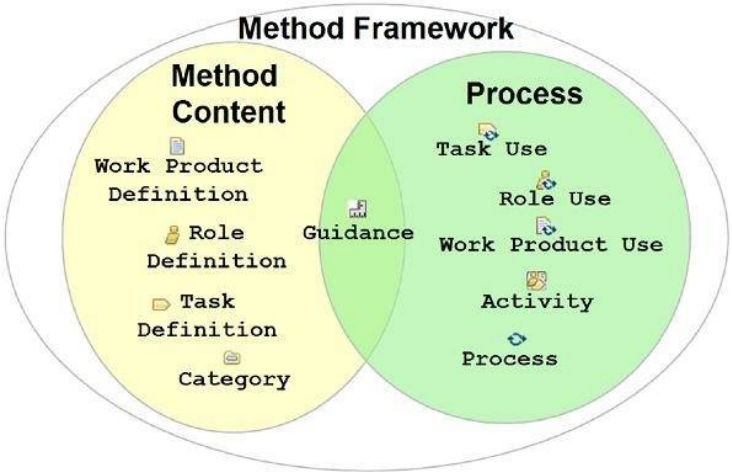
Background(3)

SoPLE: Safety-**o**riented **P**rocess **L**ine **E**ngineering [Gallina, 2012] [Gallina, 2014]



SPEM 2.0

- **Method content variability**
- It allows adaptation of created process content without affecting the original content, e.g., ***contributes***



[Gallina, 2012] Gallina, B., Sljivo, I., Jaradat, O.: **Towards a safety-oriented process line for enabling reuse in safety critical systems development and certication**. In: 35th Annual IEEE Software Engineering Workshop. (2012) 148-157

[Gallina, 2014] Gallina, B., Kashiyarandi, S., Martin, H., Bramberger, R.**: Modeling a safety and automotive-oriented process line to enable reuse and flexible process derivation**. In: IEEE 38th International Computer Software and Applications Conference Workshops. (2014) 504-509

Background (4)

DL: Defeasible Logic [Antoniu, 2000]

Defeasible theory: (F,R,>)

Fact: "Tweety is an emu" ***emu(twety)***

Strict rules: "Emus are birds" $emu(x) \rightarrow birds(x)$

Defeasible rules: "Birds typically flies"
$bird(x) \Rightarrow flies(x)$

Defeater: "If something is heavy then it may not be able to fly" $heavy(x) \rightsquigarrow \neg flies(x)$

Superiority relations:

r: $bird(x) \Rightarrow flies(x)$
r': $brokenWing(x) \Rightarrow \neg flies(x)$
$r' > r$

Formally: $r: A(c) \hookrightarrow C(r)$

Where: $r = \{\rightarrow, \Rightarrow, \rightsquigarrow\}$

Compliance by design [Hashmi, 2016]

Deontic effects
- **Obligations**
- Prohibitions
- Permission

Compliance checking
1. Determine the obligations of the rules.
2. Determine the state of each task in a process.
3. Determine the obligations in force for each task.
4. Check that the obligation in force have been fulfilled or violated.

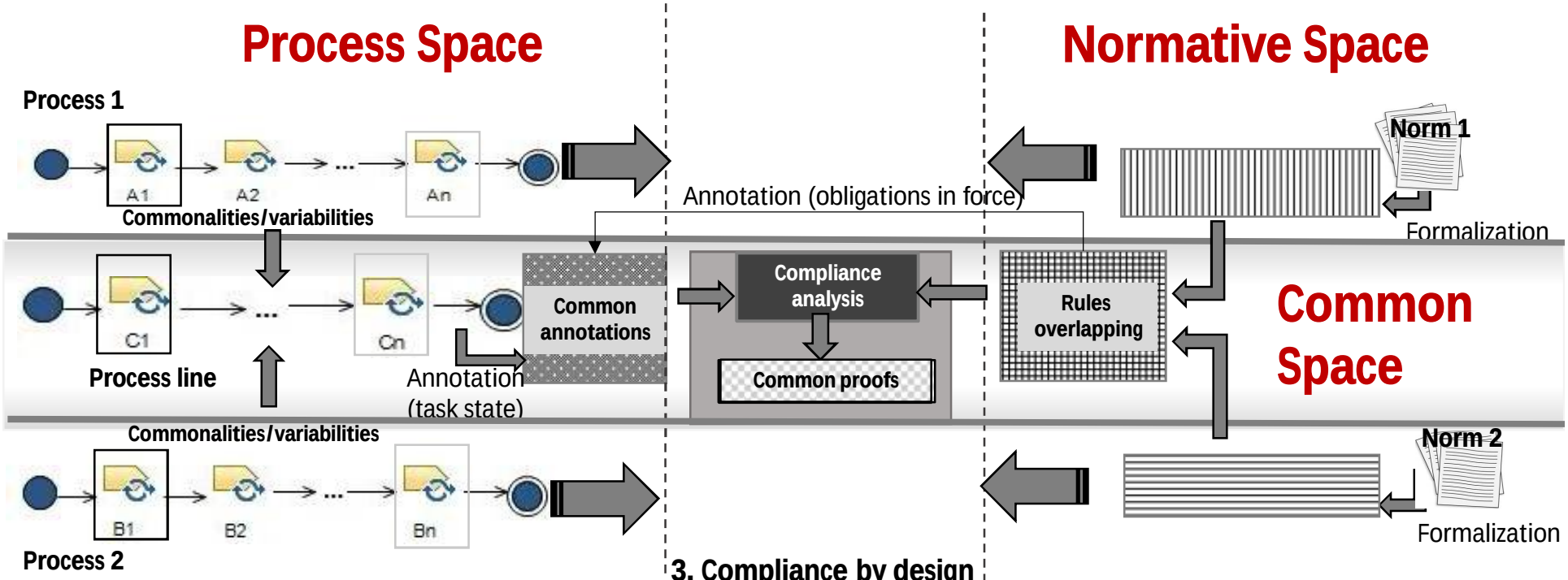
Semantic annotations

$Ann(n, t, i)$ = **s**
returns the state (s) of a trace n after a task t, in the step i

$Force(n, t, i) = p$
Associate to each task t, in a trace n, in the step i, the obligation p

[Antoniou. 2000] Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: **Representation results for defeasible logic.** ACM Transactions on Computational Logic 2(2) (2000) 255-287
[Hashmi, 2016] Hashmi, M., Governatori, G., Wynn, M.T.: **Normative requirements for regulatory compliance: An abstract formal framework.** Information Systems Frontiers (2016) 429-455

SoPLE&Logic-basedCM



1. SOPLE

- Scope: selection of the processes.
- Definition of commonalities/variabilities.
- Process line modeling.

3. Compliance by design

- Process annotation.
  - ✓ Tasks state.
  - ✓ Oblig. in force
- Compliance analysis.
- Reuse of proofs.

2. Defeasible logics

- Rules formalization: Defeasible theories.
- Overlapping rules discovery.

Applying SoPLE&Logic-basedCM (1)

ASPICE SWE.3
Software Detailed Design and Unit Construction

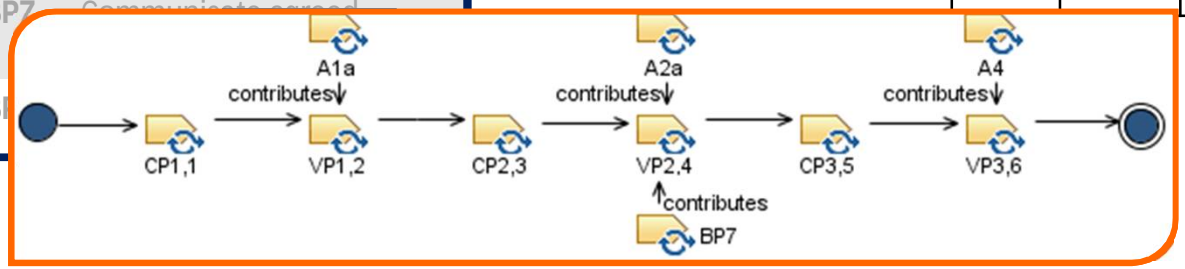
Base Practices (BP)	
BP1	Develop software detailed design
BP2	Define interfaces of software units
BP3	Describe dynamic behavior
BP4	Evaluate software detailed design
BP5	Establish bidirectional traceability
BP6	Ensure consistency
BP7	Communicate agreed ...
BI	

ISO 26262
Software Unit Design and Specification

Activities(A)	
A1	Specify the software units
A2	Verify the software unit desing
A3	Implement the software units
A4	Verify the software unit implementation

Commonalities/Variabilites

ID	SUDI	SWE.3	Common Name
CP1	A1	BP1, BP2, BP3	Define Software Unit Design
VP1	A1a		Define Software Unit Design Concerning Safety
CP2	A2	BP4, BP5, BP6	Verify Software Unit Design
VP2	A2a		Verify Software Unit Design concerning safety
		BP7	Communicate agreed software detailed design
CP3	A3	BP8	Implement Software Units
VP3	A4	BP8	Verify Software Developed Units



ISO 26262	ASPICE	SPEM2.0/EPF
Activity	Base Practice	TaskUse/

Applying SoPLE&Logic-basedCM (2)

ASPICE SWE.3

Software Detailed Design and Unit Construction

Process Outcome Description (PO)	
PO1	A detailed design is developed that describes software units
PO2	Interfaces of each software unit are defined
PO3	The dynamic behavior of the software units is defined. NOTE: Not all software units have dynamic behavior
PO4	Evaluate the software detailed design
PO5	Consistency and bidirectional traceability are established. NOTE: Consistency is supported by bidirectional traceability
PO6	The software detailed design and the relationship to the software architectural design is agreed and communicated to all affected parties
PO7	Software units defined by the software detailed design are produced.

ID	Rule	Description
RA1	$sud \longrightarrow d$	Software unit design (sud) is developed (d).
RA2	$sud \longrightarrow su$	(sud) describe software units.
RA3	$su \longrightarrow i$	(su) has defined interfaces
RA4	$su \Longrightarrow db$	(su) has usually described dynamic behavior
RA5	$sud \longrightarrow v$	(sud) is verified
RA6	$su \longrightarrow tc$	(su) has established traceability and consistency
RA7	$sud \longrightarrow ac$	(sud) is agreed and communicated
RA8	$sud \longrightarrow sui$	(sud) is used to implement the software units
RA9	$sui \longrightarrow i$	(sui) is implemented

Applying SoPLE&Logic-basedCM (3)

ISO 26262 SUDI

Requirements (R)	
R1	The requirements of this subclase shall be complied with if the software unit is safety-related.
R2	Software units are designed by using a notation that depends on the ASIL and the recommedation levels.
R3	The specification of the software units shall describe the functional behavior and the internal design to the level of detail necesary for their implementation.
R4	Design principles for software unit design and implementation shall be applied depending on the ASIL and the recommendation levels.
R5	Software unit design and implementation are verified by applying verification methods accordin to the ASIL and recomendations levels.
R6	When ASIL and recommendation levels are not followed, a rationale that explain the reasons for this behavior must be provided.

ID	Rule	Description
RI1	$sud \longrightarrow sr$	Software unit design (sud) is safety-related.
RI2	$sud \longrightarrow d$	(sud) is developed.
RI3	$d \Longrightarrow n$	(d) Is usually implemented by using a notation (n) that depends on the ASIL and the recommendation levels.
RI4	$sud \longrightarrow su$	(sud) describes software units.
RI5	$sud \longrightarrow fb$	(sud) has described functional behavior (fb).
RI6	$sud \longrightarrow id$	(sud) has described internal design (id).
RI7	$sud \Longrightarrow dp$	(sud) is implemented by using design principles (dp) that depends on the ASIL and the recommendation level.
RI8	$sud \longrightarrow sui$	(sud) is used to implement the software units.
RI9	$sui \longrightarrow i$	sui is implemented (i).
RI10	$sud, sui \longrightarrow v$	sud, sui are verified.
RI11	$v \Longrightarrow vm$	(v) is usually done by using a method that depends on the ASIL and the recommendation level (m).
RI12 RI13	$\neg n \longrightarrow rn$ $\neg dp \longrightarrow rdp$	If (n) is not provided then rationale (rn) is required. If (dp) is not provided then rationale (rdp) is required.

Applying SoPLE&Logic-basedCM (4)

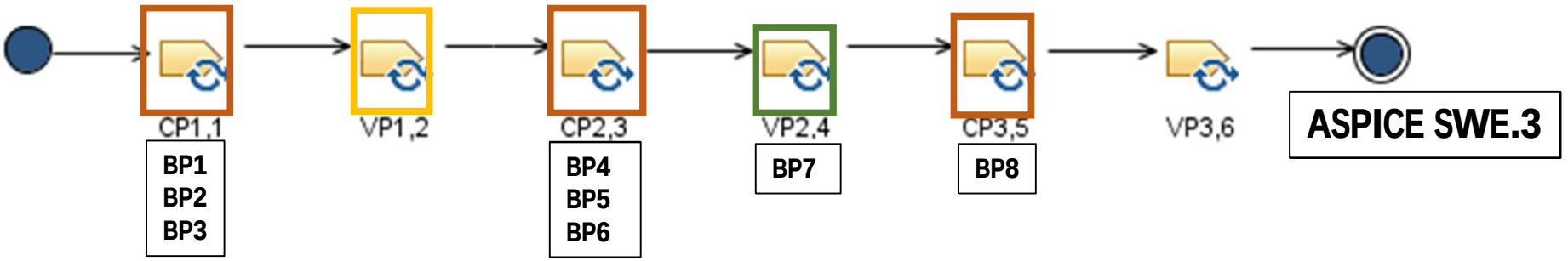
SPICE SWE.3		ISO 26262 SUDI		Common Rule	Description
ID	Rule	ID	Rule		
RA1	$sud \longrightarrow d$	RI2	$sud \longrightarrow d$	CR1	Software unit design (sud) is developed (d).
RA2	$sud \longrightarrow su$	RI4	$sud \longrightarrow su$	CR2	(sud) describes software units.
RA3	$su \longrightarrow i$	RI6	$sud \longrightarrow id$	CR3	Internal design (id) is described, including interfaces and usually dynamic behavior.
RA4	$su \Longrightarrow db$				
RA5	$sud \longrightarrow v$	RI10	$sud \longrightarrow v$	CR4	(su) is verified and traceability is demonstrated.
RA6	$su \longrightarrow$tc				
RA8	$sud \longrightarrow$ sui	RI8	$sud \longrightarrow$ sui	CR5	(sud) is used to implement the software units.
RA9	$sui \longrightarrow$ i	RI9	$sui \longrightarrow$ i	CR6	(sui) is implemented.

Applying SoPLE&Logic-basedCM (6)

Fully reused

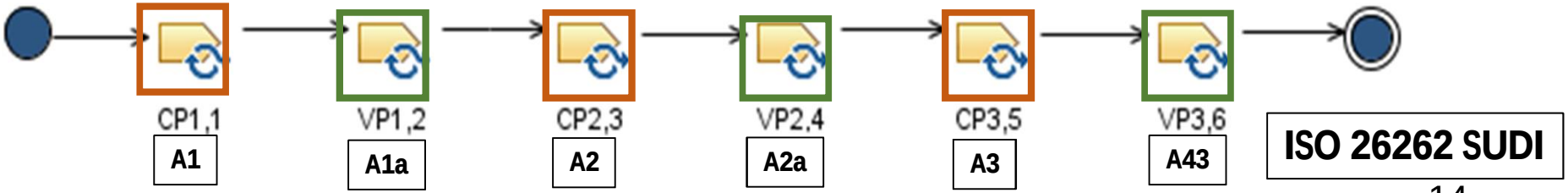
- CPs are not preceded by a VP.
- VP option is not selected.

Proof:
The software unit design is developed
The software unit design describes software units
The software unit design describes internal design



Partially reused

- CPs preceded by a VP that is contributed with standard specific rules that spread out their influence



Lessons Learnt

- We have, manually, determine proofs of compliance for our annotated automotive SoPL.

A proof can increase confidence

- Proofs of compliance obtained can be fully or partically reused in the derivation of standard-specific processes.

Reuse can increase efficiency

Related work

Business Process compliance

1. Awad, A., Decker, G., & Weske, M. (2008). **Efficient Compliance Checking Using BPMN-Q and Temporal Logic**. *International Conference on Business Process Management (BPM)*, 326–341.

2. Schumm, D., Leymann, F., Ma, Z., Scheibler, T., & Strauch, S. (2010). **Integrating Compliance into Business Processes: Process Fragments as Reusable Compliance Controls**. In *Multikonferenz Wirtschaftsinformatik* (p. 421).

Reuse of proofs for verification tasks

Reif, W., & Stenzel, K. (1993). **Reuse of Proofs in software verification**. In *International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)* (pp. 284–293). Lecture Notes in Computer Science.

Beckert, B., Bormer, T., & Klebanov, V. (2005). **Reusing Proofs when Program Verification Systems are Modified**. *Long Beach, California, USA*, 41.

Conclusion and future work

We:

- introduced an approach for process compliance, combining SoPLE, defeasible logic and compliance by design.
- illustrated the potential or our approach in term of reuse in the automotive domain.

We plan to:

- address additional process elements beyond tasks (i.e., work products, roles, guidelines…).
- investigate deontic notions beyond obligations (prohibitions, permissions…).
- explore tools that have the potential to support our work.

MÄLARDALEN UNIVERSITY
SWEDEN

AMASS

Thank you for your attention!

Discussion time...

References

1. Gallina, B., Sljivo, I., Jaradat, O.: **Towards a safety-oriented process line for enabling reuse in safety critical systems development and certication**. In: 35th Annual IEEE Software Engineering Workshop. (2012) 148-157

2. Gallina, B., Kashiyarandi, S., Martin, H., Bramberger, R**.: Modeling a safety and automotive-oriented process line to enable reuse and flexible process derivation**. In: IEEE 38th International Computer Software and Applications Conference Workshops. (2014) 504-509

3. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: **Representation results for defeasible logic.** ACM Transactions on Computational Logic 2(2) (2000) 255-287

4. Hashmi, M., Governatori, G., Wynn, M.T.: **Normative requirements for regulatory compliance: An abstract formal framework.** Information Systems Frontiers (2016) 429-455