An Extended Meta-Model of Problem Frames for Enriching Environmental Descriptions

Hongbin Xiao†, Zhi Li†*, Yilong Yang‡, Jie Deng†, Shangfeng Wei†
†School of Computer Science and Engineering, Guangxi Normal University, Guilin, China
‡School of Software, Beihang University, Beijing, China
Email: 763567198@qq.com, zhili@gxnu.edu.cn, yilongyang@buaa.edu.cn

***Abstract*—Problem Frames (PF) have attracted extensive attention and research in the requirements engineering community, particularly in the field of modeling environment-based Cyber-Physical Systems (CPS). This paper proposes an approach based on the combination of PF and model-driven engineering (MDE). The approach builds an extended meta-model of the Problem Frames approach to support representing causal behaviours of the environment of CPS, which are essential domain knowledge of the environment for modelling CPS requirements and complexity analysis. EMF (Eclipse Model Framework) and the Sirius technology are used to create GUI items for the meta-model and provide a platform for modeling tools based on the Eclipse environment for PF modelers. An example case study is used to demonstrate the feasibility of our approach.**

***Index Terms*—Cyber-Physical Systems, Problem Frames, Requirements Engineering, Model-Driven Engineering**

I. Introduction

An outstanding characteristic of a Cyber-Physical System (CPS) [1] is that it includes a significant part of the physical world (e.g., physical devices or human beings), whose behaviors are controlled, coordinated or intervened by the software to be built. Modeling the complex behaviors of such systems and arguing that the deployed software satisfies system requirements poses significant challenges for model-driven engineering (MDE). We can observe that the above challenges center mostly on assuring requirements completeness, non-functional requirements (e.g., security), dealing with uncertainties in requirements and feature interactions, which we believe are caused by the inherent contextual complexities of the software. As Jackson argues in [2], Problem Frames (PF) [3] provide a suitable conceptual framework for modeling and validating the complex behaviors of CPS (see [4], [5]for recent work, case studies and support tools).

The Problem Frames (PF) approach [3] provides a way to describe the interaction between software and other systems. It helps developers to understand the context in which software problems reside and is relevant to the design of solutions in many ways. PF emphasises the decomposition of end-to-end software requirements into machine components and a set of application domains. The approach was first proposed by Jackson, a leading scholar in the field of software engineering, and has since been refined through continuous research and development of the PF modelling language. Compared with other software requirements analysis methods, PF fully considers the influence of the surrounding environment on the software system, describes the real world, and implements the denotative meaning of requirements into the description of related fields in the real world, so more and more researchers have started to pay attention to and study PF, and have made some progress. Several researchers [6], [7], [8], [9] including the authors of this paper have proposed Meta-models of Problem Frames. The contributions of this paper are summarized as follows:

- We have improved the *Phenomenon* and *Reference* sections of the Problem Frames meta-model to show more details of the Problem Frames theory. With these improvements, the environmental phenomena of CPS can be described in more detail and the requirements of CPS can be better expressed.
- We extend the meta-model of Problem Frames by introducing *Causality*, *RQReference*, *Property* and *Attribute* to support expression of causality. This allows our approach to model CPS requirements in a way that satisfies the causal relationship that the environment has on the system, as well as the ”include” and ”extend” relationships between requirements.
- A computer-aided support tool based on Problem Frames modelling is proposed. It can use the expertise and domain knowledge from various stakeholders to capture and model the contextual complexity of the CPS, and the expertise and domain knowledge of software developers (including software designers) to decide which part of the model to be implemented as software.

The rest of the paper is organised as follows. Section II presents details of related work and comparisons between other works and our proposed extended meta-model. Section III shows the details of the Problem Frames meta-model extension. Section IV presents the PF meta-model to the PF modelling tool. Section V validates the usability of our technique by a real problem. Finally, Section VI concludes the paper and proposes future work.

II. Related Work

For the past decade, Jackson’s Problem Frames have been used and extended in many directions. For instance, Choppy et al [10] have looked into the problem of deriving software architectures from Jackson’s basic frames; Bleistein et al [11]

*corresponding author: zhili@gxnu.edu.cn

978-1-6654-1898-0/21/$31.00 ©2021 IEEE
DOI 10.1109/REW53955.2021.00077

and Cox et al [12] have looked at modeling business problems with PF; Hatebur et al [13] have applied Problem Frames in the context of security engineering; Yu Y et al [14] [15]developed a meta-model of the Problem Frames and constructed the OpenPF tool; Hatebur et al [6], Chen et al [16] and Lavazza et al [17] have looked at meta-models and ontology for Problem Frames with a view to tool development; Haley et al [18] have applied Problem Frames for arguing that the system is adequately secure. Next, we highlight some of the researchers' studies of the Problem Frames meta-model.

- Hatebur was the first to propose a formal Problem Frames meta-model [6]. A UML class diagram is used to describe the meta-model of the Problem Frames and OCL is used to constrain the characteristics of the meta-model elements and the relationships with other elements. The meta-model divides Problem Frames into four parts: domain, interface, requirements and requirements references. The domain contains the designed domain, the given domain and the machine domain and is inherited by three subclasses, the biddable domain, the causal domain and the lexical domain. Interfaces contain multiple interface phenomenon, each of which can be associated with at least two domains. Phenomenon contain two subclasses, interface phenomenon and requirement phenomenon, and the phenomenon contains three types, i.e., event, symbolic phenomenon and causal phenomenon. Requirement references contain a requirement constraint subclass, which can only be associated with one requirement and one domain.
- Colombo constructed a Problem Frames meta-model [7] [8], which consists of a problem, a domain and an interface, where a problem can contain multiple sub-problems; the domain can be classified into three types: given, design and machine domains, and the domain also contains three types of behaviour: biddable, lexical and causal. The domain also contains two elements, problem statement and phenomenon, where the problem statement contains requirements, and there are constraints and reference associations between the requirements and the phenomenon. Phenomenon are of three types: entities, events and states; interfaces are used to describe the association relationships between domains and contain shared phenomenon elements, which are associated with phenomenon through observation and control relationships.
- Lavazza proposed a meta-model for problem decomposition of Problem Frames [9]. The model supports the splitting of complex problems into multiple sub-problems. The meta-model consists of three parts: requirements, domains and interfaces, where requirements contain multiple sub-requirements, each of which contains two sub-elements, reference interfaces and constraint interfaces, each of which are associated with the domain, and each of which corresponds to a textual description; domains are inherited by aggregated domains and simple domains, and aggregated domains can contain multiple sub-domains.
- Yu has constructed a Problem Frames meta-model based on the theory of Problem Frames and used a number of techniques to construct the OpenPF tool [19], an open-source EMF implementation, to address interaction problems [14]. The tool provides a graph editor for drawing problem diagrams and is tailored for the original Problem Frames.

The results shown in Table I are obtained by comparing the meta-model of other related works in ten aspects, e.g., domain, phenomenon, causality and attribute, etc. It can be seen that the meta-model proposed in this paper encompasses a broader and richer semantics of PF model.

In short, the above related works can 1) not support modelling of more details of the problem domain and 2) not support the representation of causal relationships between the domains in PF; our approach addresses all of these limitations.

TABLE I
COMPARISON OF META-MODELS IN RELATED WORK

	Hatebur	Lavazza	Colombo	OpenPF	This Paper
Problem Domain	✓	✓	✓	✓	✓
Machine Domain	✓	✓	✓	✓	✓
Phenomenon	✓	✓	✓	✓	✓
Requirement	✓	✓	✓	✓	✓
RequirementReference	✓	✓	✓	✓	✓
RequirementConstraint	✓	✓	✓	✓	✓
Interface	✓	✓	✓	✓	✓
SubProblem Domain			✓	✓	✓
Domain Property		✓			✓
RQReference					✓
PhenomenonSubcategory					✓
Causality					✓
Attribute					✓

III. META-MODEL OF PROBLEM FRAMES

MDA (Model Driven Architecture) defines a lot of model-related specifications, for which EMF (Eclipse Modeling Framework) provides Java implementations. EMF provides 4 types of technical support for modeling, namely Java comments, Rational Rose modeling, XML schema and Ecore modeling. Ecore provides a graph editor for creating meta-model elements, and it also supports third-party plugins *Ecore Tool* for creating diagrams. Figure 1 shows the complete meta-model that we create for Problem Frames, where those in yellow represent the original basic meta-model, those in green represent our improved section, those in red represent our extensions, and those in grey represent abstract meta-models, from which instances can not be created.

A. The original meta-model

The root of the meta-model is named *ProblemFramework*, which represents the most fundamental concepts, i.e., *Domain*, *Interface* and *Requirement*, whose details are described below (as shown in those yellow boxes in 1).

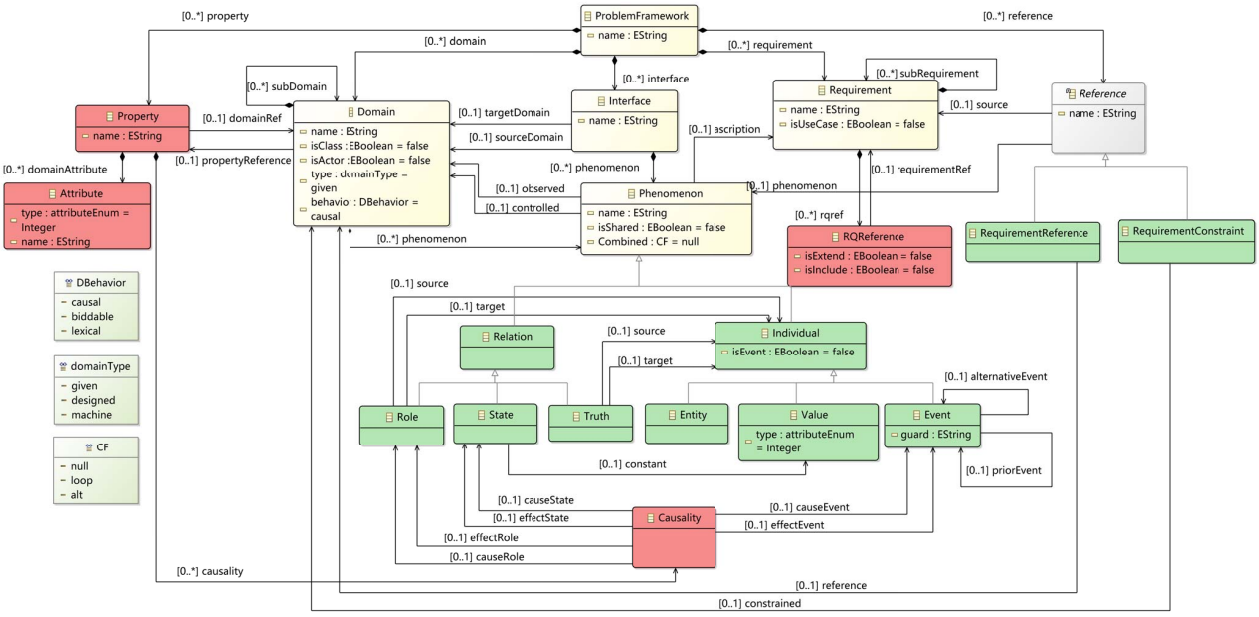

Fig. 1. The extended meta-model of Problem Frames

- *Domain Model* contains some internal and external attributes: *name* is of String type; These are important labels required for the model transformation; *type* consists of two kinds of types; the first kind *DBehavior* is of Enumeration type, which includes *causal*, *biddable* and *lexical*, corresponding to the domain behavior types in PF; the second kind *DomainType* is also of Enumeration type, which includes *given*, *designed* and *machine*, corresponding to those domain types in PF. According to the semantics of PF, a domain may include many subdomains, thus there is an "include" relationship between *Domain* and *subDomain*. In PF, a problem may contain many domains, thus there is a many-one relationship between *Domain* and *ProblemFramework*.
- *Phenomenon Model*: an interface class is created to be included by the root node *ProblemFramework*. It contains several attributes: *name* is of *String* type; *Combined* is of *Enumeration* type, with *null*, *loop* and *alt*; *isShared* is of *Boolean* type, representing if it is shared with other domains or not. According to PF semantics, the *Phenomenon* class includes 2 kinds of sub-classes, namely *Relation* and *Individual*, with the former containing *Role*, *State* and *Truth*, and the latter containing *Entity*, *Value* and *Event*; there is an association between *Role* and *Individual*, thus there are two kinds of dependencies between them, namely *target* and *source*; similarly, there are two types of dependencies between *Truth* and *Individual*, namely *source* and *target*, and one *Truth* corresponds to 0 or 1 *Individual*.
- *Requirement Model*: *Requirement* and *Reference* are created from the root node, with *name* attributes (of *String* type) added to uniquely identify them. In PF semantics, a requirement may contain many sub-requirements, thus *subRequirement* is added as a sub-class of *Requirement*; An association is added from *Reference* to *Phenomenon* so as to reduce the complexity of problem diagrams. Two sub-classes of *Reference*, namely *RequirementReference* and *RequirementConstraint* are created according to PF semantics.

B. Improvement to the original meta-model

Our improvement to the original meta-model lies in two main areas: on the one hand the phenomenon section is subdivided into individual and relations, and on the other hand the reference section is divided into requirement reference and requirement constraint, as shown in those green boxes in Figure 1).

- *Relation Model*: Phenomenon is not to be treated simply as streams of messages flowing between domains, so we have in the meta-model the *Relation Model* inheriting the *Phenomenon Model*, the *Relation Model* in turn being inherited by the *Role State Value*.
- *Individual Model*: For the same reason, the *Individual Model* inherits the *Phenomenon Model*. The *Individual Model* in turn is inherited by the *Entity Value Event*.
- *Reference Model*: Requirements interact with the problem domain during the modelling process and are identified as both RequirementReference and RequirementConstraint according to the definition of requirements in PF. Therefore, in the meta-model *Reference* two subclasses are created, *RequirementReference* and *RequirementConstraint*,

which are used to distinguish the reference and constraint relationships between requirements and domains.

C. The extended meta-model

We have customized our extensions of the meta-model of PF for modeling Cyber-Physical System. For instance, we have added "causality" as an important part of PF modeling to allow for reasoning about control behaviors in CPS. In addition, for the purpose of future work, we have also made the following extensions (those red boxes, as shown in Figure 1).

- *Property*: in PF modeling, a domain may contain a lot of detailed descriptions about its internal phenomenon, which are semantically consistent with Jackson's definition - "a set of related phenomenon that are usefully treated as a unit in problem analysis".
- *Attribute*: a subclass of *Property* is created to give more details; *RQReference*: an extension to *Requirements* to represent relationships among them.
- *Causality*: represents causal relationships among events or state-changing events so that control behaviors (an outstanding characteristic of CPS) can be reasoned based on the chains of causality.
- *RQReference*: In terms of requirements extension, *RQReference Model* is created in *Requirement Model* in order to determine the relationships between requirements.

IV. Mapping Meta-Model to Graphical Notation

EMF builds a bridge between modelers and Java programmers by allowing users to create and browse model instances with its tree-like editor. The Eclipse open-source community has provided many graphical tools and techniques, among which Sirius is better than many others. This paper uses Sirius technology to implement the model graphically.

Name Of Nodes	Graphical	Meta-Model Element	Descriptions
Machine	Machine	MachineType	The machine is where the software to be run.
Designed Domain	Model	LexicalType	Lexical domains provide physical space where the data is stored, thus with some causality in their storage behaviors.
Causal Domian	Causal C	CausalType	Causal domains contain predictability in their behaviors
Biddable Domian	Biddable B	BiddableType	biddable domains (usually human beings) do not contain predictability in their behaviors.
Domain Property	Property	Property	They describe internal or external causal relationships or other characteristics of the domain.
Requirement	Requirement	Requirement	They describe the behavioral references or constraints, prescribed by the problem owners.

Fig. 2. How meta-model elements are mapped PF Nodes

In the Problem Frames, the creation of a graphical interface for the metamodel is to provide the Problem Frames modeller with a platform of tools for drawing the problem diagram. Please note that a problem diagram consists of nodes and edges (relationships between nodes), therefore, the meta-model presented in Figure 1 is mapped to both nodes and edges. Figure 2 illustrates how meta-model elements are mapped to PF nodes, and Figure 3 illustrates how meta-model elements are mapped to PF edges. After this node mapping and relationship mapping, the Problem Frames modelling tool platform can be obtained as shown in Figure 4, with the property bar at the bottom, the drawing board area in the middle and the toolbar on the right.

Graphical Relationships (Edges)	Meta-Model Elements	Source	Target
Machine — Domain	Interface	sourceDomain	targetDomain
Domain — Domain	Interface	sourceDomain	targetDomain
Domain — Requirement	Requirement-Reference	source	target
Domain ← Requirement	Requirement-Constraint	source	target
Domain — Property			domainRef

Fig. 3. How meta-model elements are mapped PF Edges

V. Case Study

In this paper, we evaluate our extended PF meta-model and PF-GUI with a driverless system case study. The following is a rough sketch of the problem.

"*We consider the development of driverless systems for automobiles. The driverless system will be used in the field of autonomous driving for cars. The system consists of a software controller to be designed and some hardware to be purchased from a third party. The hardware includes sensors and bodywork, with sensors including LIDAR, cameras, millimetre wave radar etc. The problem is to develop the software for the driverless system and guide the trend towards urban rail transport.*"

This is a typical control problem which is representative of a cyber-physical system. It includes the cyber part (i.e., the software controller to be built), and the complex physical part: the sensor part (e.g., LiDAR, Camera, Radar), the actuator part (e.g., GPS, 5G, BaseStation), the human operator (e.g., Passenger), and potential feature interactions.

The core of PF is the *problem diagram*, which is a graphical model of the scope and structure of a software problem. Figure 4 shows a problem diagram for the Driverless System. Some basic elements of a typical CPS case of an driverless system for modeling requirements to construct a problem diagram are explained below:

- The *machine* - the *Driverless* - the box with double stripes, representing that it is a hardware with programmed software running on it;
- The *problem domain* - *Passenger*, *Sensor*, *GPS* et al - boxes without double stripes (where B in the bottom right corner is human beings and C in the bottom right corner

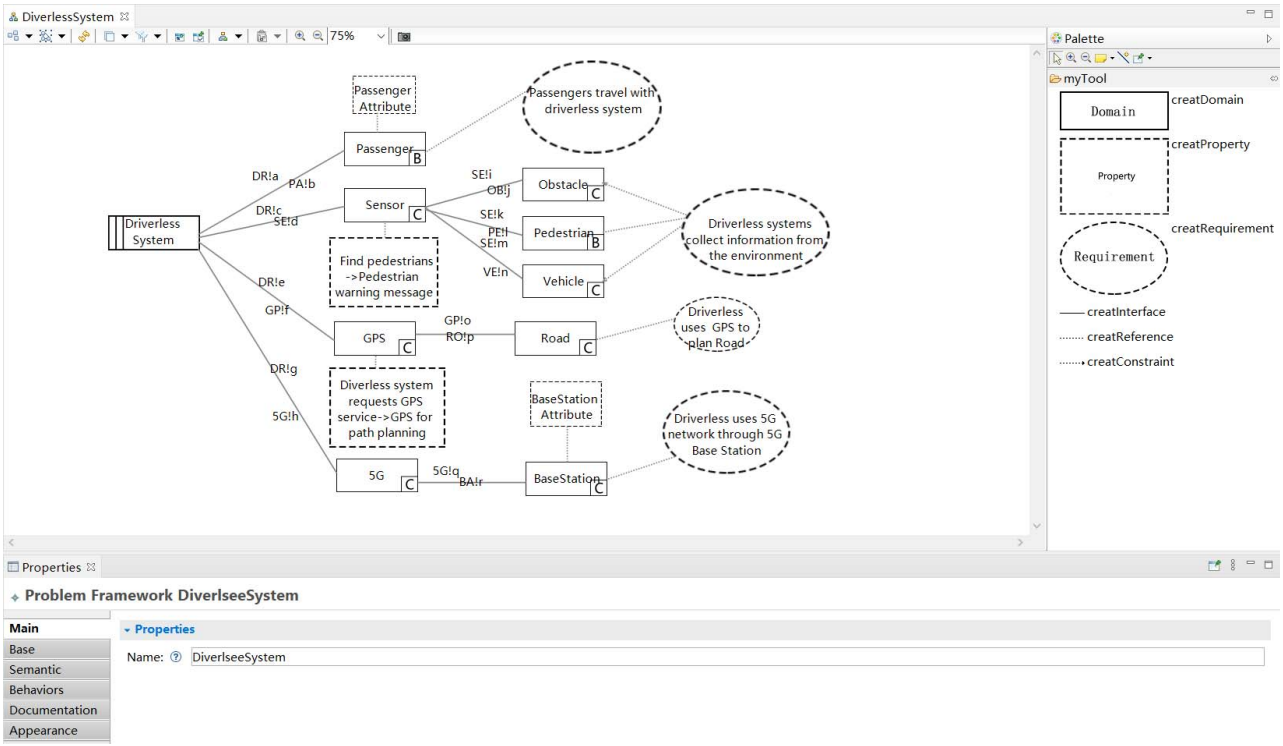

Fig. 4. Computer-aided support tools based on PF model

is physical device). Table II shows the identified domains and their descriptions.

- The *shared phenomenon- DR!a*, *PA!b*, *DR!c* et al - annotating the solid lines, representing interfaces between domains. Table III shows the shared phenomenon and their designations.
- The *requirements* dashed oval with arrows linked to the domain indicating that the requirement is a constraint on the domain, and dashed lines without arrows indicating a dependency.
- The *domain properties* - *PassengerAttribute* and *BaseStation Attribute* - dashed boxes, representing given characteristics of domains (e.g., internal states, causal relationships).

Our problem diagram tool has been constructed by improving and extending the meta-model of the Problem Frames and is more suitable for CPS requirements modelling for the following main reasons.

- **RE1:** Our problem diagram tool allows for the representation of six shared phenomena categories between domains, allowing for a clearer representation of the complexity of the demand environment in CPS. For example, Figure 5 depicts several different kinds of phenomena between the sensor domain and the pedestrian domain in driverless system, and for space reasons only two of the phenomena categories are represented in Table III.
- **RE2:** Our problem diagram tool can represent its causal relationships in the domain so that we can describe the causal relationships that exist between complex requirement environments in CPS. For example, Figure 6 represents the causal relationships that occur in the sensor domain in driverless system.
- **RE3:** Our problem diagram tool shows the relationship to another requirement within a requirement, so that we are more aware of the associations that exist between the many requirements in the CPS. For example, Figure 7 represents Requirement (Driverless uses GPS to plan

TABLE II
DOMAINS, DOMAIN PROPERTIES AND DESCRIPTIONS

Name	Description
Driverless	A terminal controller for driverless systems.
Passenger	A passenger who wants to use the driverless system.
Sensor	A device that receives a signal or stimulus (LIDAR, cameras, millimeter wave).
GPS	A navigation system.
5G	A technology of fifth generation mobile communications.
Obstacle	A barrier that stands in the way.
Pedestrian	A person who travels by foot on the road.
Vehicle	A conveyance that transports people or objects. (bicycles, cars,trams, etc)
Road	A long, hard surface built for vehicles to travel along.
Base Station	A platform for transmitting and receiving 5G.
Passenger Attribute	A characteristic that the passenger has. (Name, Phone, Age, ID)
Base Station Attribute	A characteristic that base station has. (ID, Hight, Structure)

TABLE III
SHARED PHENOMENON AND THEIR DESIGNATIONS

Name	Type	Designation
DR!a	phenomenon	The action of the driverless system returning information to the passenger.
PA!b	phenomenon	The action of passengers sending commands to the driverless systems.
DR!c	phenomenon	The action of driverless systems analysing environmental information sensed by sensors.
SE!d	phenomenon	The action of sensors sending environmental information to the driverless system for processing.
DR!e	phenomenon	The action of driverless systems sending commands to BeiDou navigation.
GP!f	phenomenon	The action of BeiDou Navigation returning navigation information to driverless systems.
DR!g	phenomenon	The action of driverless systems uploading information to cloud platform via 5G.
5G!h	phenomenon	The action of downloading data from the cloud to driverless systems via 5G.
SE!i	phenomenon	The action of sensors sensing if there are obstacles nearby.
OB!j	phenomenon	The action of obstacles being sensed by sensors.
SE!k	phenomenon	The action of sensors scanning for pedestrians in the vicinity
PE!l	phenomenon	The action of pedestrians being scanned by sensors.
SE!m	phenomenon	The action of Sensors obtaining information about nearby vehicle.
VE!n	phenomenon	The action of nearby vehicles being perceived by sensors.
GP!o	phenomenon	The action of Beidou navigation system locating the road.
RO!p	phenomenon	The action of information about the road being analysed by .
5G!q	phenomenon	The action of 5G signals being transmitted via base stations.
BA!r	phenomenon	The action of base stations transmitting 5G signals.
SE!k	event	Find pedestrians.
SE!d	event	Pedestrian warning message.
DR!e	event	Diverless system requests GPS service.
GP!o	event	GPS for path planning.

Road) in driverless system as an extended requirement of Requirement (Passengers travel with driverless system).

- **RE4:** Our problem diagram tool allows us to add attribute descriptions to the domain, and we can portray the attributes of people, machines and objects in the problem diagram. For example, Figure 8 represents the attributes of the passenger domain in driverless system.

VI. CONCLUSION AND FUTURE WORK

There are several issues which we plan to address in future work. Firstly, an operational semantics will be developed for PF so that more rigorous reasoning and safety and security argument can be formulated for safety-critical systems. Both logic-based semantics and statistics-based approach to the semantics will be considered so that both rigorous reasoning and practical reasoning are supported by our methods and tools. Secondly, a new approach will be proposed and developed to automatically transform problem diagrams into UML use case diagrams, sequence diagrams, and conceptual class

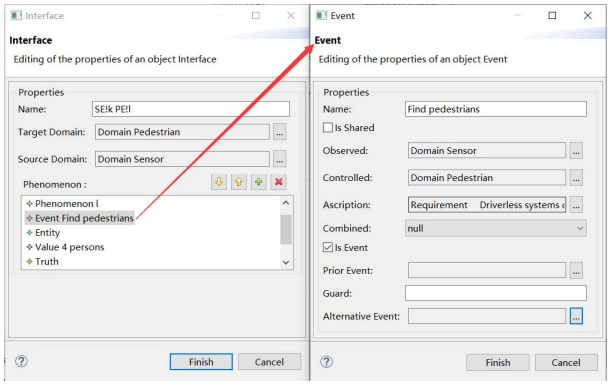

Fig. 5. Representation of inter-domain phenomena

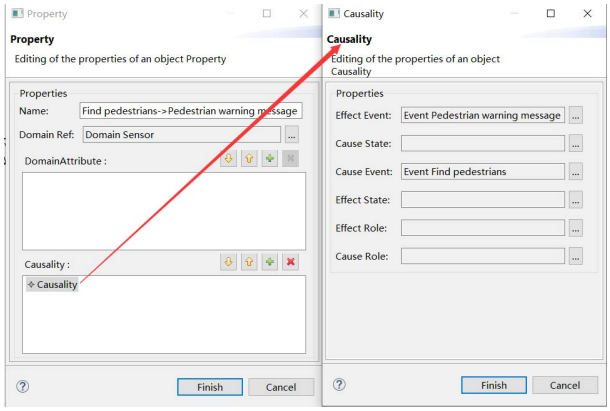

Fig. 6. Representation of causality

diagrams, which can directly guide downstream designs and implementations.

In conclusion, our improved and extended meta-model of Problem Frames can be applied to address both functional and have the potential to address complex nonfunctional requirements which are typical of CPS, due to its ability to support reasoning about complex causal behaviours, which are usually caused by global non-functional quality requirements,

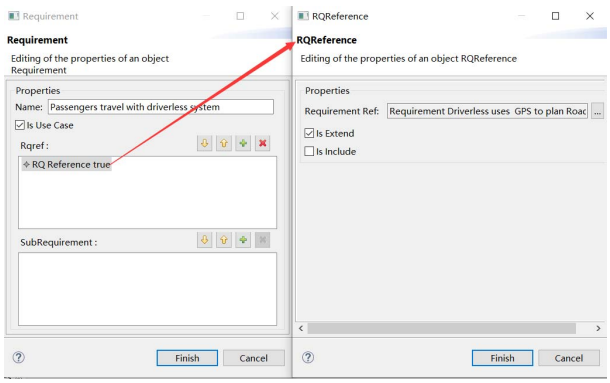

Fig. 7. Representation of the relationship between demand

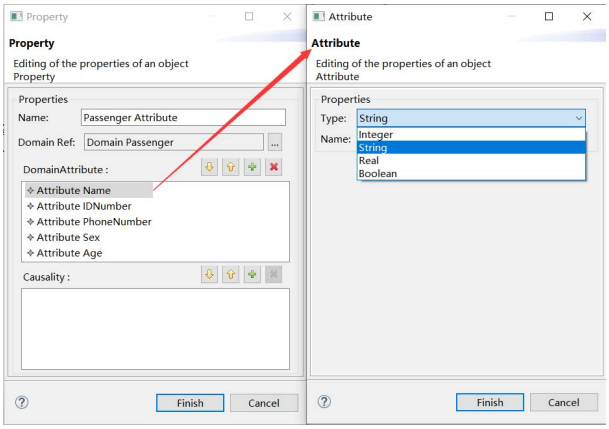

Fig. 8. Attributes representing domains

and its flexibility - allowing both application stakeholders and developers to identify problem concerns and deciding what part of the problem is to be implemented in the software.

VII. Acknowledgements

This work is partially supported by the National Natural Science Foundation of China (61862009), Guangxi Natural Science Foundation (2012GXNSFCA053010), Guangxi "Bagui Scholar" Teams for Innovation and Research, the Project of the Guangxi Key Lab of Multi-source Information Mining & Security (Director's grant 19-A-01-02), Guangxi Collaborative Innovation Center of Multi-source Information Integration and Intelligent Processing.

References

[1] Gill, H. 2008. Cyber-Physical Systems. Program Announcements & Information. The National Science Foundation, 4201 Wilson Boulevard, Arlington, Virginia 22230, USA. (Sept. 2008), retrieved 2021-05-08 [], URL: http://www.nsf.gov/pubs/2008/nsf08611/nsf08611.pdf.

[2] M. Jackson. 2015. System Behaviours and Problem Frames: Concepts, Concerns and the Role of Formalisms in the Development of Cyber-physical Systems. Dependable Software Systems Engineering, 79-104, 2015.

[3] M. Jackson. Problem frames: analyzing and structuring software development problems. Addison-Wesley Publishing Company, Boston, 2001.

[10] C. Choppy, D. Hatebur and M. Heisel. 2005. Architectural patterns for problem frames. IEE Proceedings-Software, 152(4):198–208

[4] Z.Jin, Xiaohong Chen, Z. Li, and Yijun Yu. RE4CPS: Requirements engineering for cyber-physical systems. In Proceedings of the 27th IEEE International Conference on Requirements Engineering (Vol. 2019-September, pp.496–497). IEEE Computer Society,Jeju Island, South Korea. https://doi.org/10.1109/RE.2019.00072

[5] Website: http://www.re4cps.org/home, containing case studies, tutorials and tool support, etc

[6] Hatebur D, Heisel M, Schmidt H. A formal meta-model for problem frames[C]//International Conference on Model Driven Engineering Languages and Systems. Springer, Berlin, Heidelberg, 2008: 68-82.

[7] Colombo P, Lavazza L, Coen-Porisini A, et al. Towards a meta-model for problem frames: conceptual issues and tool building support[C]//2009 Fourth International Conference on Software Engineering Advances. IEEE, 2009: 339-345.

[8] Colombo P, Lavazza L, Coen-Porisini A, et al. A Meta-model for Problem Frames: Conceptual Issues and Tool Building Support[J]. International Journal On Advances in Software, 2010, 3(2).

[9] Lavazza L, Coen-Porisini A, Colombo P, et al. A meta-model supporting the decomposition of problem descriptions[C]//2010 Fifth International Conference on Software Engineering Advances. IEEE, 2010: 50-57.

[11] S. Bleistein, K. Cox, J. Verner. 2004. Requirements engineering for e-Business systems: integrating Jackson context diagrams with goal modelling and BPM. In: Proceedings of the 11th international Asia-Pacific software engineering conference (APSEC 2004), pp 410–417. IEEE, Busan, Korea. 30th November– 3rd December 2004

[12] K. Cox, J.G. Hall, L. Rapanotti. 2005. Editorial: a roadmap of problem frames research. Information and Software Technology 47(14):891–902

[13] D. Hatebur, M. Heisel, H. Schmidt. 2006. Security engineering using problem frames. In: Emerging trends in information and communication security, lecture notes in computer science, vol 3995/2006. Springer, New York, pp 238–253

[14] Tun T T , Yu Y , Laney R , et al. Early identification of problem interactions: A tool-supported approach.[J]. Lecture Notes in Computer Science, 2009, 5512:74-88.

[15] Yu Y , Tun T T , Tedeschi A , et al. OpenArgue: Supporting argumentation to evolve secure software systems[C]// RE 2011, 19th IEEE International Requirements Engineering Conference, Trento, Italy, August 29 2011 - September 2, 2011. IEEE, 2011.

[16] X. Chen, Z. Jin, L. Yi. 2007. An ontology of problem frames for guiding problem frame specification. In: Knowledge schience, engineering and management, lecture notes in computer science, vol 4798/2007, pp 384–395

[17] L. Lavazza, A. Coen-Porisini, V. Del Bianco. 2009. Towards a meta-model for problem frames: conceptual issues and tool building support. In: 4th International conference on software engineering advances (ICSEA '09). IEEE Press, New York, pp 339–345

[18] B. H. Haley, R. Laney et al. Security Requirements Engineering: A Framework for Representation and Analysis.[J]. IEEE Transactions on Software Engineering, 2008.

[19] OpenPF website (open-source). https://github.com/problem-frames/openpf, retrieved on August 1, 2021.