# Building Specifications in the Event-B Institution: A Summary

Marie Farrell[1(✉)], Rosemary Monahan[2], and James F. Power[2]

[1] Department of Computer Science, The University of Manchester, Manchester, UK
marie.farrell@manchester.ac.uk
[2] Department of Computer Science and Hamilton Institute, Maynooth University, Co. Kildare, Ireland
rosemary.monahan@mu.ie

**Abstract.** This "journal-first" paper summarises a publication by the same authors in the journal Logical Methods in Computer Science which describes a formal semantics for the Event-B specification language using the theory of institutions. It defines an institution for Event-B and shows how the constructs of the Event-B specification language can be mapped into our institution. This algebraic semantics distinguishes three constituent sub-languages of Event-B: the superstructure, infrastructure and mathematical languages. An important impact of this work is that our semantics provides access to the generic modularisation constructs available in institutions, including specification-building operators for parameterisation and refinement. We demonstrate how these features subsume and enhance the corresponding features already present in Event-B through a detailed study of their use in a worked example. Further benefits of the institutional approach are its provision for mathematically definable interoperability to facilitate heterogeneous specification.

**Keywords:** Event-B · Semantics · Modularisation · Interoperability

## 1 Introduction

Event-B is an industrial-strength formal specification language for the development of safety-critical systems [1], including applications in aerospace [3], rail [6],

This work was initially funded by a Government of Ireland Postgraduate Grant from the Irish Research Council. It has subsequently been supported by EPSRC Hubs for Robotics and AI in Hazardous Environments: EP/R026092 (FAIR-SPACE), and a Royal Academy of Engineering Research Fellowship.
Farrell and Monahan dedicate this paper to the memory of Dr. James F. Power who passed away before he could see this work accepted for publication. We thank him for his contributions and encouragement throughout this project.


U. Glässer et al. (Eds.): ABZ 2023, LNCS 14010, pp. 245–253, 2023.
https://doi.org/10.1007/978-3-031-33163-3_19

healthcare [4] and autonomous robotics [5]. To be capable of being adopted in the development of increasingly complex systems, formal methods must support standard software engineering practices such as modularity. Such formal methods should also be equipped with a detailed semantics so that the results are interpreted correctly by both software engineers as well as interoperable tools.

Although a mature formal specification language, Event-B has some limitations, particularly its lack of standardised modularisation constructs. While Event-B has been provided with a semantics in terms of proof obligations [12], the abstractness of this approach makes it difficult to formally deal with modularisation, or to define a concrete basis for interoperability with other formalisms.

Our paper, [10], provides an algebraic semantics for the Event-B language, generic modularisation constructs and pathways to interoperability with other logics. It does this by harnessing the benefits offered in the theory of institutions. Institutions are mathematical structures that are based in category theory and they provide a generic framework for formalising logics and formal languages [11,18]. We define the institution for Event-B, called $\mathcal{EVT}$, which we use to describe as a target for the semantics of the full Event-B specification language. In insitutions, specification-building operators are used to construct formal specifications of systems in a modular fashion. Further, institutions support the combination of different formal languages and logics in a way that preserves the properties of the individual languages while allowing for the expression of the system's behavior in a more powerful and expressive way. This is achieved by defining appropriate mappings between institutions for distinct formalisms.

In summary, the **principal contributions** of our paper [10] are:

1. We define a formal semantics for the Event-B formal specification language, as a series of functions from Event-B constructs to specifications over the $\mathcal{EVT}$ institution. This provides clarity on the meaning of the language elements and their interaction. To achieve this, we consider the constituent elements of the Event-B language as presented in our three-layer model shown in Fig. 1 (which we briefly describe later).
2. A well-defined set of generic modularisation constructs using the specification-building operators available through the theory of institutions. These are built-in to our semantics, they subsume and extend the existing Event-B modularisation constructs, and they provide a standardised approach to exploring new modularisation possibilities.
3. An explication of Event-B refinement in the $\mathcal{EVT}$ institution. Refinement in $\mathcal{EVT}$ incorporates and extends the Event-B refinement constructs.

Additionally, our EB2EVT translator transforms Event-B specifications that have been developed using Rodin into specifications over the $\mathcal{EVT}$ institution. We use EB2EVT to validate our semantic definitions, and to interact with the existing large corpus of Event-B specifications [9]. This paper summarises the main results and constructions from [10]. For the finer details including detailed descriptions, definitions, theorems, proofs and examples, we direct the interested reader to [10]. This work is beneficial to the Event-B community since it provides a template for defining extensions and modifications to the Event-B formalism.

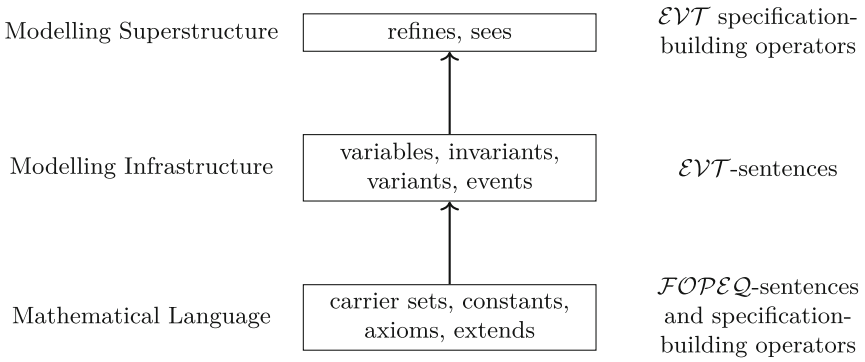



**Fig. 1.** For each of the Event-B sub-languages, we show their corresponding Event-B constructs, and their representation in our semantics.

## 2 The Institution for Event-B

Institutions have been devised for many logics and formalisms, we do not dwell on the mathematical definitions here since the detail can be found in [10]. An institution has four basic components: (1) *Signatures* determine the vocabulary of the language, (2) *Sentences* use the vocabularly to form statements, (3) *Models* are needed to give meaning to such sentences and, (4) a satisfaction relation determines satisfaction of sentences by models. These four aspects together form an institution if they are well-defined and preserve certain mathematical properties.

The institution for Event-B, called $\mathcal{EVT}$ is defined as follows:

Signatures: The vocabulary, $\langle S, \Omega, \Pi, E, V \rangle$, contains sets of sort names ($S$), arity-indexed operation names ($\Omega$), arity-indexed predicate names ($\Pi$), event-status pairs ($E$) and sort-indexed variable names ($V$).

Sentences: are of the form $\langle e, \phi(\overline{x}, \overline{x\prime}) \rangle$. Here, $e$ is an event name and $\phi(\overline{x}, \overline{x\prime})$ is an open first-order formula over the variables $\overline{x}$ from the signature and the primed versions, $\overline{x\prime}$, of the variables. Figure 2 shows the specific translations corresponding to the Event-B syntax.

Models: map event names to their corresponding set of variable-to-value mappings over the carriers corresponding to the sorts of each of the variables (and their primed versions).

Satisfaction Relation: the satisfaction relation in $\mathcal{EVT}$ devolves to mapping the $\mathcal{EVT}$ sentences to first-order logic and checking satisfaction of first-order sentences in the usual way. Full details are in [10].

Note that, since first-order logic is the foundational logic used in Event-B, it should be unsurprising that the $\mathcal{EVT}$ institution is also built on the institution for first-order predicate logic with equality (we refer to this as $\mathcal{FOPEQ}$). We relate $\mathcal{FOPEQ}$ and $\mathcal{EVT}$ using an institution comorphism which is a mapping

```
1  MACHINE m REFINES a SEES ctx
2    VARIABLES x̄
3    INVARIANTS I(x̄)
4    VARIANT n(x̄)
5    EVENTS
6    Initialisation ordinary
7      then  act-name: BA(x̄′)
       ⋮
8    ⋮
9    Event e_i ≙  convergent
10     any p̄
11     when  guard-name: G(x̄, p̄)
12     with  witness-name: W(x̄, p̄)
13     then  act-name: BA(x̄, p̄, x̄′)
       ⋮
14   ⋮
15 END
```

$$\{\langle e,\ I(\overline{x})\ \wedge\ I(\overline{x}\prime)\rangle\ \mid\ e\ \in\ dom(\Sigma.E)\}$$

$$\langle \texttt{Init},\ BA(\overline{x}')\rangle$$

$$\langle \texttt{e}_i,\ n(\overline{x}\prime)\ <\ n(\overline{x})\rangle$$

$$\langle \texttt{e},\ \exists \overline{p}\ \cdot\ G(\overline{x},\overline{p})\ \wedge\ W(\overline{x},\overline{p})\ \wedge\ BA(\overline{x},\overline{p},\overline{x}')\rangle$$

**Fig. 2.** The elements of an Event-B machine specification as presented in Rodin (left) and the corresponding sentences in the $\mathcal{EVT}$ institution (right).

that allows us to write first-order logic sentences in $\mathcal{EVT}$. This captures the way that first-order formulae can be written in Event-B, as shown in Fig. 2.

**The Three-Layer Model:** Taking inspiration from an early version of the specification of UML [16,17], we split the Event-B language into three constituent layers. Each layer corresponds to a sub-language of Event-B as shown in Fig. 1. This three-layer model plays a key role in structuring the definitions of the semantic functions given in [10]. Specifically, the institutional constructs that we use to define the semantics of each of the sub-languages are listed on the right of Fig. 1. We use this model to structure our translation from Event-B to $\mathcal{EVT}$ as follows:

- The Event-B mathematical language (base of Fig. 1) is captured using the institution of first-order predicate logic with equality, $\mathcal{FOPEQ}$, which is embedded via an institution comorphism into $\mathcal{EVT}$ [8]. Our semantics translates the constructs of this sub-language into corresponding $\mathcal{FOPEQ}$ constructs.
- Event-B infrastructure comprises the elements used to define variables, invariants, variants and events. These are translated into $\mathcal{EVT}$ sentences.
- Event-B superstructure deals with the definition of Event-B machines, contexts and their relationships (`refines`, `sees`, `extends`). These are translated into $\mathcal{EVT}$ structured specifications using the specification-building operators.

**Building Specifications:** The specification-building operators in the $\mathcal{EVT}$ institution are, used at multiple levels and are, essentially generic modularisation constructs. These are breifly summarised in Table 1 (full descriptions are shown in Table 1 of [10]). For example, the `and` specification-building operator provides a straightforward way of combining specifications. If we consider two

**Table 1.** A brief summary of the institution-theoretic specification-building operators that can be used to modularise specifications. Here $SP_1$ and $SP_2$ denote specifications written over some institution, and $\sigma$ is a signature morphism in the same institution.

| Operation | Format | Description |
|---|---|---|
| Translation | $SP_1$ `with` $\sigma$ | Renames the signature components of $SP_1$ (e.g. sort, operation and predicate names in $\mathcal{FOPEQ}$) using the signature morphism $\sigma : \Sigma_{SP_1} \to \Sigma'$. $Sig[SP_1 \texttt{ with } \sigma] = \Sigma'$ $Mod[SP_1 \texttt{ with } \sigma] = \{M' \in \lvert\mathbf{Mod}(\Sigma')\rvert \mid M'\lvert_\sigma \in Mod[SP_1]\}$. |
| Sum | $SP_1$ `and` $SP_2$ | Combines the specifications $SP_1$ and $SP_2$. It is the most straightforward way of combining specifications with different signatures. $SP_1 \texttt{ and } SP_2 = (SP_1 \texttt{ with } \iota) \cup (SP_2 \texttt{ with } \iota')$ where $Sig[SP_1] = \Sigma$, $Sig[SP_2] = \Sigma'$, $\iota : \Sigma \hookrightarrow \Sigma \cup \Sigma'$, $\iota' : \Sigma' \hookrightarrow \Sigma \cup \Sigma'$ and $\cup$ is applied to specifications ($SP_3$ and $SP_4$) over the same signature ($\Sigma''$) as follows $Sig[SP_3 \cup SP_4] = \Sigma''$ $Mod[SP_3 \cup SP_4] = Mod[SP_3] \cap Mod[SP_4]$. |
| Enrichment | $SP_1$ `then` ... | Extends the specification $SP_1$ by adding new sentences after the `then` specification-building operator. This operator can be used to represent superposition refinement of Event-B specifications by adding new variables and events. |
| Hiding | $SP_1$ `hide via` $\sigma$ | Hiding via the signature morphism $\sigma$ allows viewing a specification, $SP_1$, as a specification restricted to the signature components of another specified by the signature morphism $\sigma : \Sigma \to \Sigma_{SP_1}$. $Sig[SP_1 \texttt{ hide via } \sigma] = \Sigma$ $Mod[SP_1 \texttt{ hide via } \sigma] = \{ M\lvert_\sigma \mid M \in Mod[SP_1]\}$. |

specifications, $SP1$ and $SP2$, the specification $SP1$ `and` $SP2$ represents their combination. It has a signature that is the union of the signatures of $SP1$ and $SP2$, valid models of this specification are captured as the intersection of the valid models of the individual specifications. This can be understood as a generalisation of the `SEES` construct in Event-B (line 1 of Fig. 2). In this way, `and` can be used to incorporate both machines and contexts into a given specification.

We use these specification-building operators throughout our semantics for Event-B. Figure 3 shows an example of the semantic functions used for the superstructure language which uses the `with` (translation via signature morphism) and `then` (specification enrichment/extension) operators. We provide the full semantic functions and a worked example using EB2EVT in [10].

## 3 Refinement, Modularisation and Interoperability

We briefly summarise the enhancements that our semantics offers in terms of refinement, modularisation and interoperability for Event-B.

**Representing Refinement:** No semantics of Event-B would be complete without reference to refinement. The institution framework allows us to capture refinement as model-class inclusion where the class of models of a specification comprises the models satisfying that specification. In [10], we consider two

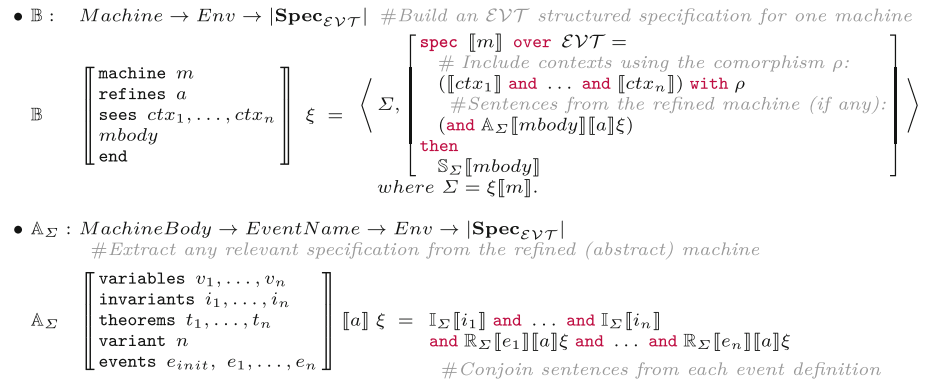



**Fig. 3.** The semantics for the Event-B superstructure sub-language is defined by translating Event-B specifications into structured specifications over $\mathcal{EVT}$ using the function $\mathbb{B}$ and the specification-building operators defined in the theory of institutions.

cases of refinement for an abstract specification $SP_A$ and concrete specification $SP_C$:

1. When the signatures are the same, we capture refinement as

$$SP_A \sqsubseteq SP_C \quad \iff \quad Mod(SP_C) \subseteq Mod(SP_A)$$

   This corresponds to superposition refinement in Event-B.
2. When the signatures are different, we capture refinement as

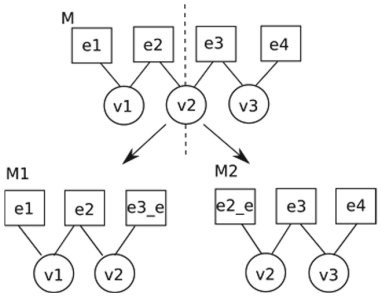
$$SP_A \sqsubseteq SP_C \quad \iff \quad Mod(\sigma)(SP_C) \subseteq Mod(SP_A)$$

   This captures data refinement in Event-B, where the signature morphism $\sigma$ corresponds to the relevant gluing invariant. We can also use the `hide via` specification-building operator to capture this refinement. Related work on a CSP semantics for Event-B refinement used a similar notion of hiding [19].

More details are described in [10] including a worked example.

**Modularisation via Specification-Building:** It has been shown that Event-B lacks a unified set of modularisation constructs [9]. Current approaches to modularisation in Event-B consist of a suite of Rodin plugins that each support a specific approach to modular specification. Decomposition-style modularisation was first proposed by Abrial where larger systems could be decomposed into smaller ones and independently refined [2]. Ultimately, these smaller specifications could be recombined to construct a specification that could have been devised without the use of decomposition techniques from the outset.

In [10], we describe the evolution of modularisation constructs for Event-B and Rodin, and show how the specification-building operators in $\mathcal{EVT}$ can be




```
1  spec  M1 =
2    (M hide via  σ1)
3      with {e3 ↦ e3_e}
4  end
5    where σ1 = {v1 ↦ v1, v2 ↦ v2, e1 ↦ e1,
6                e2 ↦ e2, e3 ↦ e3}

7  spec  M2 =
8    (M hide via  σ2)
9      with {e2 ↦ e2_e}
10 end
11   where σ2 = {v2 ↦ v2, v3 ↦ v3, e2 ↦ e2,
12               e3 ↦ e3, e4 ↦ e4}
```

**Fig. 4.** We represent the *shared variable* decomposition of machine `M` into sub-machines `M1` and `M2` (on the left) using specification-building operators (on the right).

used to capture current modularisation approaches. We provide a snapshot of the classical shared variable approach on the left of Fig. 4, the right of Fig. 4 illustrates this kind of modularisation using specification-building operators in $\mathcal{EVT}$. Specifically, we use `hide via` to split the signatures and `with` to rename the events in the individual machines. In [10], we demonstrate how our semantics defines a theoretical foundation for the current Rodin modularisation plugins.

**Interoperability:** The theory of institutions provides a framework for combining different logical systems in a consistent and meaningful way [11]. Institution (co)morphisms specify how the elements of one institution relate to the elements of another. By correctly defining these mappings, we can formally reason across different formal languages. In fact, $\mathcal{EVT}$ already uses an institution comorphism to capture the mathematical layer ($\mathcal{FOPEQ}$) of Event-B. We are actively exploring how we can write heterogeneous specifications using these mappings.

## 4 Conclusions and Future Work

Our paper [10] contributes a formal semantics for the Event-B specification language. To this end, we distilled a three-layer model for the Event-B language. The semantics for each of these distinct layers is grounded in our institution for Event-B, $\mathcal{EVT}$, and the institution for first-order predicate logic with equality, $\mathcal{FOPEQ}$. We show how this semantics supports the restructuring and modularisation of Event-B specifications using the specification-building operators. We focused on Event-B but our work demonstrates, more generally, how such modularisation capabilities can be added to a formal specification language using the theory of institutions. Future work examines how this approach can be applied to other similar formal languages that are also represented as institutions, for example UML [13], CASL [15] and CSP [14]. Through the theory of institutions, we have provided scope for interoperability between Event-B and other formal languages. Support for heterogeneous specification is desired in the development of complex safety-critical systems (e.g. robotics [7]) and we will explore this in future work.

## References


1. Abrial, J.R.: Modeling in Event-B: System and Software Engineering, 1st edn. Cambridge University Press, Cambridge (2010)
2. Abrial, J.R., Hallerstede, S.: Refinement, decomposition, and instantiation of discrete models: application to event-B. Fund. Inform. **77**(1–2), 1–28 (2007)
3. Banach, R.: The landing gear case study in hybrid event-B. In: Boniol, F., Wiels, V., Ait Ameur, Y., Schewe, K.-D. (eds.) ABZ 2014. CCIS, vol. 433, pp. 126–141. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07512-9_9
4. Banach, R.: Hemodialysis machine in hybrid event-B. In: Butler, M., Schewe, K.-D., Mashkoor, A., Biro, M. (eds.) ABZ 2016. LNCS, vol. 9675, pp. 376–393. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33600-8_32
5. Bourbouh, H., et al.: Integrating formal verification and assurance: an inspection rover case study. In: Dutle, A., Moscato, M.M., Titolo, L., Muñoz, C.A., Perez, I. (eds.) NFM 2021. LNCS, vol. 12673, pp. 53–71. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-76384-8_4
6. Dghaym, D., Poppleton, M., Snook, C.: Diagram-led formal modelling using iUML-B for hybrid ERTMS level 3. In: Butler, M., Raschke, A., Hoang, T.S., Reichl, K. (eds.) ABZ 2018. LNCS, vol. 10817, pp. 338–352. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91271-4_23
7. Farrell, M., Luckcuck, M., Fisher, M.: Robotics and integrated formal methods: necessity meets opportunity. In: Furia, C.A., Winter, K. (eds.) IFM 2018. LNCS, vol. 11023, pp. 161–171. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98938-9_10, http://arxiv.org/abs/1805.11996
8. Farrell, M., Monahan, R., Power, J.F.: An institution for event-B. In: James, P., Roggenbach, M. (eds.) WADT 2016. LNCS, vol. 10644, pp. 104–119. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-72044-9_8
9. Farrell, M., Monahan, R., Power, J.F.: Specification clones: an empirical study of the structure of event-B specifications. In: Cimatti, A., Sirjani, M. (eds.) SEFM 2017. LNCS, vol. 10469, pp. 152–167. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66197-1_10
10. Farrell, M., Monahan, R., Power, J.F.: Building specifications in the event-B institution. Log. Methods Comput. Sci. **18** (2022). https://doi.org/10.46298/lmcs-18(4:4)2022
11. Goguen, J.A., Burstall, R.M.: Institutions: abstract model theory for specification and programming. J. ACM **39**(1), 95–146 (1992)
12. Hallerstede, S.: On the purpose of event-B proof obligations. In: Börger, E., Butler, M., Bowen, J.P., Boca, P. (eds.) ABZ 2008. LNCS, vol. 5238, pp. 125–138. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87603-8_11
13. Knapp, A., Mossakowski, T., Roggenbach, M., Glauer, M.: An institution for simple UML state machines. In: Egyed, A., Schaefer, I. (eds.) FASE 2015. LNCS, vol. 9033, pp. 3–18. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46675-9_1
14. Mossakowski, T., Roggenbach, M.: Structured CSP – a process algebra as an institution. In: Fiadeiro, J.L., Schobbens, P.-Y. (eds.) WADT 2006. LNCS, vol. 4409, pp. 92–110. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71998-4_6
15. Mosses, P.D. (ed.): Springer, Heidelberg (2004). https://doi.org/10.1007/b96103
16. OMG: UML Infrastructure Specification, v2.4.1. Specification formal/2011-08-05, Object Management Group (2011)

17. OMG: UML Superstructure Specification, v2.4.1. Specification formal/2011-08-06, Object Management Group (2011)
18. Sannella, D., Tarlecki, A.: Foundations of Algebraic Specification and Formal Software Development. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-17336-3
19. Schneider, S., Treharne, H., Wehrheim, H.: The behavioural semantics of event-B refinement. Formal Aspects Comput. **26**, 251–280 (2014)