Systems Biology Graphical Notation: Process Description language Level 1

Version 2.0

Date: May 24, 2012

Disclaimer: This is a working draft of the SBGN Process Description Level 1 Version 2.0 specification. It is not a normative document.

Editors:

Stuart Moodie Nicolas Le Novère Emek Demir Huaiyu Mi Alice Viléger EMBL European Bioinformatics Institute, UK EMBL European Bioinformatics Institute, UK Sloan-Kettering Institute, USA University of Southern California, USA London, UK

To discuss any aspect of SBGN, please send your messages to the mailing list sbgn-discuss@sbgn.org. To get subscribed to the mailing list or to contact us directly, please write to sbgn-editors@lists.sourceforge.net. Bug reports and specific comments about the specification should be entered in the issue tracker http://p.sf.net/sbgn/pd_tracker.



Contents

1	Introduction					3.5.30 Annotation	47
	1.1	1.1 SBGN levels and versions				3.5.31 CrossReference	48
	1.2	Developments, discussions, and notifica-				3.5.32 SubmapTerminal	49
		tions of updates	2			3.5.33 Tag	50
_			_			3.5.34 FluxArc	50
2	Concepts		3			3.5.35 ModulationArc	53
	2.1		4			3.5.36 LogicArc	57
		2.1.1 Language versus notation	4			3.5.37 EquivalenceArc	58
		2.1.2 What are the languages?	4			3.5.38 CloneMarker	59
		2.1.3 Nomenclature	4			3.5.39 SimpleCloneMarker	60
		2.1.4 Graph, diagram or map?	5			3.5.40 LabelledClonerMarker	61
3	Lan	guage Flements	6		3.6	Controlled vocabularies	62
•	Language Elements 3.1 Introduction		6			3.6.1 Entity pool node material types	62
	3.2	Note on typographical convention	6			3.1	63
	3.3	How to read the Language Specification	7			3.6.3 Macromolecule covalent modifications	63
	0.0	3.3.1 An example definition	8				64
	3.4	Overview	9		3.7	Entity Pool Node Identity and Cloning	64
	3.5	Definitions	9		3.8	Map and Submap Linking	65
	0.0	3.5.1 SBGNElement	9			3.8.1 Namespace and the relationship	
		3.5.2 Map	11			•	
		3.5.3 SBGNGlyph	12			3.8.2 Compartment spanning	67
		3.5.4 AuxiliaryUnit	12	4	Lave	out Rules for a Process Description	69
		3.5.5 SBGNNode	13	7	4.1		69
		3.5.6 SBGNArc	14		4.2		
		3.5.7 EntityType	14		7.2	•	
		3.5.8 StateVariableDefinition	15			•	70
		3.5.9 EntityPoolNode	16				70
		3.5.10 Empty Set	17				70
		3.5.11 StatelessEPN	19				70
		3.5.12 Simple chemical	19				
		3.5.13 UnspecifiedEntity	21			4.2.7 Node labels	
		3.5.14 Perturbing Agent	22			4.2.8 Edge labels	
		3.5.15 StatefulEPN				4.2.9 Compartments	
		3.5.16 Macromolecule	23		4.3	Recommendations	
		3.5.17 NucleicAcidFeature	25			4.3.1 Node-edge crossing	
		3.5.18 Complex				4.3.2 Labels	
		3.5.19 Subunit	28			4.3.3 Avoid edge crossings	
		3.5.20 ProcessNode	-			4.3.4 Branching of association and disso-	
		3.5.21 NonStoichiometricProcess	30				72
		3.5.22 Phenotype	31			4.3.5 Units of information	72
		3.5.23 SubmapNode	32		4.4	Additional suggestions	72
		3.5.24 LogicalOperator	33				
		3.5.25 StoichiometricProcess	35	5		nowledgments	74
		3.5.26 Compartment	42				74
		3.5.27 Logical Identity	42			Level 1 Release 1.1	
		3.5.28 AttributeValue	44			Level 1 Release 1.2	
		3.5.29 StateVariable	45		5 4	Level 1 Release 1.3	74

	5.5 Comprehensive list of acknowledgements .	74	D	Issues postponed to future levels	83
	5.6 Financial Support	75		D.1 Multicompartment entities	83
	а така таруа			D.2 Logical combination of state variable values	83
Δ	A Complete examples of Process Description Maps			D.3 Non-chemical entity nodes	83
		76		D.4 Generics	84
	maps .	. •		D.5 State and transformation of compartments .	84
В	Reference card	80	Е	Revision History	85
				E.1 Version 1.0 to Version 1.1	85
C	Validation and consistency rules	82		E.2 Version 1.1 to Version 1.2	86
	C.1 EPN rules	82		E.3 Version 1.2 to Version 1.3	86

Chapter 1

Introduction

The goal of the **S**ystems **B**iology **G**raphical **N**otation (SBGN) is to standardize the graphical/visual representation of biochemical and cellular processes. SBGN defines comprehensive sets of symbols with precise semantics, together with detailed syntactic rules defining their use. It also describes the manner in which such graphical information should be interpreted. For a general description of SBGN, one can read:

Nicolas Le Novère, Michael Hucka, Huaiyu Mi, Stuart Moodie, Falk Schreiber, Anatoly Sorokin, Emek Demir, Katja Wegner, Mirit I Aladjem, Sarala M Wimalaratne, Frank T Bergman, Ralph Gauges, Peter Ghazal, Hideya Kawaji, Lu Li, Yukiko Matsuoka, Alice Villéger, Sarah E Boyd, Laurence Calzone, Melanie Courtot, Ugur Dogrusoz, Tom C Freeman, Akira Funahashi, Samik Ghosh, Akiya Jouraku, Sohyoung Kim, Fedor Kolpakov, Augustin Luna, Sven Sahle, Esther Schmidt, Steven Watterson, Guanming Wu, Igor Goryanin, Douglas B Kell, Chris Sander, Herbert Sauro, Jacky L Snoep, Kurt Kohn & Hiroaki Kitano. The Systems Biology Graphical Notation. *Nature Biotechnology* **27**, 735 - 741 (2009). http://dx.doi.org/10.1038/nbt.1558

This document defines the *Process Description* visual language of SBGN. Process Descriptions are one of three views of a biological process offered by SBGN. It is the product of many hours of discussion and development by many individuals and groups.

1.1 SBGN levels and versions

It was clear at the outset of SBGN development that it would be impossible to design a perfect and complete notation right from the beginning. Apart from the prescience this would require (which, sadly, none of the authors possess), it also would likely need a vast language that most newcomers would shun as being too complex. Thus, the SBGN community followed an idea used in the development of other standards, i.e. stratify language development into levels.

A *level* of one of the SBGN languages represents a set of features deemed to fit together cohesively, constituting a usable set of functionality that the user community agrees is sufficient for a reasonable set of tasks and goals. Within *levels*, *versions* represent small evolution of a language, that may involve new glyphs, refined semantics, but no fundamental change of the way maps are to be generated and interpreted. In addition new versions should be backwards compatible, i.e., Process Description maps that conform to an earlier version of the Process Description language within the same level should still be valid. This does not apply to a new levels.

Capabilities and features that cannot be agreed upon and are judged insufficiently critical to require inclusion in a given level, are postponed to a higher level or version. In this way, the development of SBGN languages is envisioned to proceed in stages, with each higher levels adding richness compared to the levels below it.

1.2 Developments, discussions, and notifications of updates

The SBGN website (http://sbgn.org/) is a portal for all things related to SBGN. It provides a web forum interface to the SBGN discussion list (sbgn-discuss@caltech.edu) and information about how anyone may subscribe to it. The easiest and best way to get involved in SBGN discussions is to join the mailing list and participate.

Face-to-face meetings of the SBGN community are announced on the website as well as the mailing list. Although no set schedule currently exists for workshops and other meetings, we envision holding at least one public workshop per year. As with other similar efforts, the workshops are likely to be held as satellite workshops of larger conferences, enabling attendees to use their international travel time and money more efficiently.

Notifications of updates to the SBGN specification are also broadcast on the mailing list and announced on the SBGN website.

Chapter 2

Concepts

To set the stage for what follows, we first give a brief overview of some of the concepts in the Process Description language with the help of an example shown in Figure 2.1.

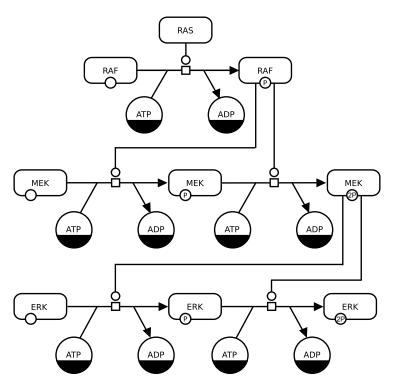


Figure 2.1: This example of a Process Description uses two kinds of entity pool nodes: one for pools of different macromolecules (Section 3.5.16) and another for pools of simple chemicals (Section 3.5.12). Most macromolecule nodes in this map are adorned with state variables (Section 3.5.29) representing phosphorylation states. This map uses one type of process node, the process node (Section 3.5.25), and three kind of connecting arc, consumption (Section 3.5.34), production (Section 3.5.34) and catalysis (Section 3.5.35). Finally, some entity pool nodes have dark bands along their bottoms; these are clone markers (Section 3.5.38) indicating that the same pool nodes appear multiple times in the map.

The map in Figure 2.1 is a simple map for part of a mitogen-activated protein kinase (MAPK) cascade. The larger nodes in the figure (some of which are in the shape of rounded rectangles and others in the shape of circles) represent biological materials—things like macromolecules and simple chemicals. The biological materials are altered via processes, which are indicated in Process Description language by lines with arrows and other decorations. In this particular map, all of the processes happen to be the same: processes catalyzed by biochemical entities. The directions of the arrows indicate the direction of the processes; for example, unphosphorylated RAF kinase processes to phosphory-

62

66

68

69

71

72

73

75

76

78

79

81

82

83

84

86

89

91

93

95

97

98

100

101

102

104

105

lated RAF kinase via a process catalyzed by RAS. Although ATP and ADP are shown as incidental to the phosphorylations on this particular graph, they are involved in the same process as the proteins getting phosphorylated. The small circles on the nodes for RAF and other entity pools represent state variables (in this case, phosphorylation sites).

The essence of the Process Descriptions is *change*: it shows how different entities in the system process from one form to another. The entities themselves can be many different things. In the example of Figure 2.1, they are either pools of macromolecules or pools of simple chemicals, but as will become clear later in this chapter, they can be other conceptual and material constructs as well. Note also that we speak of *entity pools* rather than individuals; this is because in biochemical network models, one does not focus on single molecules, but rather collections of molecules of the same kind. The molecules in a given pool are considered indistinguishable from each other. The way in which one type of entity is transformed into another is conveyed by a *process node* and links between entity pool nodes and process nodes indicate an influence by the entities on the processes. In the case of Figure 2.1, those links describe consumption Section 3.5.34, production Section 3.5.34 and catalysis Section 3.5.35, but others are possible. Finally, nodes in Process Descriptions are usually not repeated; if they do need to be repeated, they are marked with *clone markers*—specific modifications to the appearance of the node (Section 3.5.38). The details of this and other aspects of Process Description notation are explained in the following chapters.

2.1 Definitions and Nomenclature

2.1.1 Language versus notation

SBGN specifications propose symbols, ways to organise them, but also semantic rules to analyse the resulting representations. SBGN "drawings" can be translated into English, but also into computer readable formats. Those specifications really propose true languages. SBGN is therefore made up of three languages.

2.1.2 What are the languages?

PD is a language that permits the description of all the processes taking place in a biological system. The ensemble of all these processes constitute a Description. **ER** is a language that permits the description of all the relations involving the entities of a biological system. The ensemble of all these relations constitute a Relationship. **AF** is a language that permits the description of the flow of activity in a biological system.

2.1.3 Nomenclature

The three languages of SBGN should be referred to as:

- the Process Description language.
- the Entity Relationship language.
- the Activity Flow language.

Abbreviated as:

- the PD language.
- the ER language.
- the AF language.

A specific representation of a biological system in one of the SBGN languages should be referred to as:

- a Process Description map.
- an Entity Relationship map.
- an Activity Flow map.

123

	106		
Abbreviated as:	107		
• a PD map.	108		
• an ER map.	109		
• an AF map.	110		
	111		
The corpus of all SBGN representations should be referred to as:			
 Process Descriptions. 	113		
• Entity Relationships.	114		
Activity Flows.	115		
	116		
The capitalization is important. PD, ER and AF are names of languages. As such they must be capi-			
talized in English. This is not the case of the accompanying noun (language or map).			

2.1.4 Graph, diagram or map?

A graph is a very technical term that belongs to mathematics and is uncommon in biology. Diagram is a concept that encompasses more than just graph. Examples are Venn diagrams for instance. Therefore, we recommend using the term map for SBGN representations. Those representations effectively permit users to travel and orient themselves in a biological system. Map is also the term most frequently used by the different communities, whether in metabolism, signaling or genomics.

Chapter 3

126

130

138

140

141

142

146

147

149

151

152

155

Language Elements

3.1 Introduction

In this chapter we aim to describe the Process Description language by describing its elements and underlying concepts in detail. The challenge in doing this is to provide enough detail to minimise ambiguity, but to also make the rules understandable to the users of the specification. An additional goal is to minimise duplication or rules as much as possible, which makes the specification more difficult to update in the future. To achieve this we took our lead from other successful standards: the Unified Modelling Language (UML) [?] which is a graphical language that has been used by tens of thousands of software developers for over decade to describe and software system; and the Systems Biology Markup Language (SBML) Level 3 Core specification [?] which is the "original" computational systems biological standard and after 10 years is firmly embedded in the scientific community it serves. In both these specifications the language is modelled in UML where each language element is described as an UML class. The specification defines each class in turn and in so doing describes how the language elements fit together (syntax) and how they are applied and their rules (semantics). Since UML is a graphical language, the class definition also includes, where applicable to the language element, a description of description of the symbols used and guidelines about how the symbols should be drawn or laid out.

Based on these specifications we will follow the following conventions in this chapter:

- The language will be described by a UML class model, a language element of language concept being represented as a class.
- Each class will be defined in detail. The definition will describe parent classes, attributes, interactions with other classes and any rules the apply to that class.
- Where classes correspond to glyphs, or where appropriate, a detailed description of the glyph will be given.
- Any rules or concepts that apply to the language as a whole are defined after the individual class definitions.

3.2 Note on typographical convention

The concept represented by a glyph is written using a normal font, while a *glyph* means the SBGN visual representation of the concept. For instance "a biological process is encoded by the SBGN PD *process*". A UML class name is written in camel case and presented as CamelCaseClassName. Attributes and associations are written in lower case, using the underscore to separate words: an_attribute.

Note that in drafts of this document the margin-note is used to indicate that this part of the

See footnote.

¹The footnote contains a description of why this part of the specification is contentious, may require further discussion, or should be noted by the reviewer of the document.

160

165

168

169

170

171

172

175

176

177

178

181

182

183

186

187

191

192

193

194

196

197

document may require special attention from the reader as it may contain material that is new or potentially contentious.

3.3 How to read the Language Specification

Here will describe the elements of the class definitions that are the core of the language specification. Each definition starts with an introduction describing the purpose of the class and what it represents conceptually or physically. It's context within the UML Process Description language is then described by a detailed figure showing the class, its associations and its interaction with other key classes to help understand its context and any rules that are part of the definition.

A number of terms are used within the specification that for clarity we define here:

- **class** A class describes a set of objects that share the same specifications of features, constraints, and semantics (from the UML specification [?]).
- **subclass** A class that inherits attributes, behaviour and associations from another class. For example in figure 3.1 "class C is a subclass of class A".
- **superclass** A class that is an ancestor of another class. For example in figure 3.1 "class A is a superclass of class B".
- generalisation A relationship in UML that defines a subclass/superclass relationship.
- **association** A semantic relationship between two classes. Typically at least one, but often both classes require the other class to complete its definition.
- **instance** An instance is equivalent to an object a single example or realisation of a class.
- **role** The role describes the nature of an association from the perspective of one of the classes in that relationship.
- **cardinality** In the case of an attribute the number of separate values the attribute can hold and for an association the number of instances of each class that can be associated with each other. The permitted cardinality values are:
 - **R** Required (attributes only).
 - **O** Optional (attributes only).
 - 1 required only one instance is permitted (association only).
 - **0..1** optional zero or one instance ir permitted (association only).
 - 1..* at least 1 instance is required.
 - * any number of instances are permitted.
- **type** A type constrains the values represented by an attribute. For example an attribute of type int must have a value that is a integer.

The UML class diagrams used will look something like the examples in figure 3.1. The UML elements in this diagram are the only ones used in this specification and again as an aid to understanding their meanings are described below:

- **class** See above for definition. The class symbol displays each attribute and its type using the convention: "attribute: type".
- **generalisation** (**open triangle arrowhead**) Defines an inheritance relationship where class C inherits the attributes, associations and semantics of class A. This means that C also has attributes attrib1 and attrib2 as well as an association called container to class B.

201

202

203

204

207

208

209

210

213

215

216

217

222

224

225

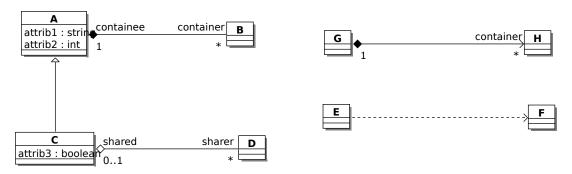


Figure 3.1: An example of the UML elements used in this specification. The diagram is explained more fully in the text.

composite aggregation (black diamond) The aggregating class (A) is the one adjacent to the diamond and at the other end is the contained class (B). This represents a whole/part relationship where class B is part of class A and B cannot exist independently of A. The cardinality and role of a given class are shown at the opposite end of the association to it.

shared aggregation (open diamond) The aggregating class (C) is adjacent to the diamond and is connected to the other class (D) via a solid line. Here there is a whole/part relationship, but there the part (class D) can be shared with another class and can exists independently of C.

navigability (black diamond anchor arrowhead) If the association ends in an arrow, this indicates the direction of the association. Here this should be taken to mean that only one aside of the association (G) is 'aware' of the relationship.

dependency Indicates that the dependent class (E) requires the other class adjacent to the arrow head (F) to satisfy its specification or implementation.

The specification uses a number of primitive types that are used in attribute definitions. These are:

int An integer.

string A string of UniCode characters.

boolean A Boolean value that can be either True or False.

object A type that can be any value.

cv A controlled vocabulary (see 3.6).

enum A value that must be chosen from on of an enumerated set of predefined values.

3.3.1 An example definition

Here we will provide an example specification. In this section is an overview of the class and its concepts.

Generalisation 221

This section defines the inheritance relationship(s) between this class and any other classes.

Attributes 223

Here any attributes specific to this class are defined and their meaning or purpose described. Attributes from superclasses are part of this classes definition but are not defined here explicitly.

232

233

235

239

240

241

242

243

244

248

251

252

253

256

259

260

261

Associations

Any associations are defined and described. Again associations from superclasses are not included here, but are part of this classes definition.

Notation 225

If the class corresponds to one or more glyphs then each glyph is described here. The glyph is described in words and graphically. In cases where several glyphs combine in complex ways usage examples are provided.

Layout Rules and Guidelines

In some cases the graphical layout of a glyph and is more complicated and requires some additional explanation. If this is the case this will be provided here.

Rules and Constraints

The semantics of the class are defined here in the form of itemised rules and constraints on the behaviour of the class. The scope of this section is ideally restricted to rules and behaviours the related to the class itself or classes it has some form of immediate relationship with (an association or dependency). In the later case the rules should relate to that relationship.

Changes from Previous Version

In order to help track changes between versions of the specification this section documents where this class definition differs from that in previous versions. Where appropriate ticket numbers for bugs or issues addressed in this version should be included.

3.4 Overview

The UML model describing the Process Description language is summarised in figure 3.2. The model has a root class SBGNElement for all language elements and care has been taken keep the model as simple as possible. This includes minimising the use of multiple inheritance and reflecting the directed graph structure inherent in the language with the SBGNNode and SBGNArc classes. All graphical elements that can be drawn directly onto a Process Description map are glyphs (SBGNGlyph) and all those that decorate glyphs are auxiliary units (AuxiliaryUnit). There are a number of language rules that the model and the individual class specification cannot capture and these are dealt with later in the chapter after the class definitions in section 3.5.

3.5 Definitions 254

3.5.1 SBGNElement



All the glyphs in SBGN Process Description Level 1 inherit from SBGNElement². This is an abstract or conceptual class that helps organise Process Description conceptually. SBGNElement (figure 3.3) has a single attribute id that is an identifying attribute. This means that all SBGN elements defined here, which ultimately extend SBGNBase, can all be uniquely identified from each other. This makes sense if you think that a glyph drawn on a map is distinct from another glyph drawn on the map. The id attribute reflects this and is not shown explicitly in a Process Description map.

Generalisation 262

None 263

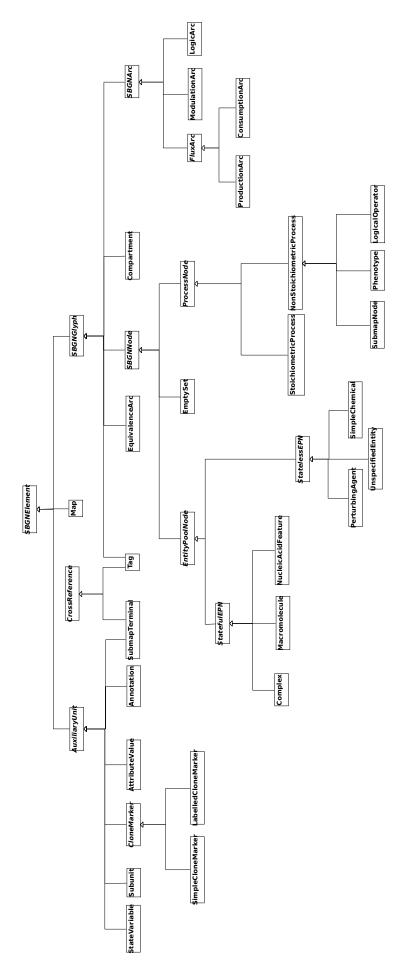


Figure 3.2: A view of the UML model describing Process Description language. This diagram shows the classes and their inheritance relationships. No attributes or associations between classes are shown to simplify the diagram. These details are provided in the UML class diagrams associated with most class definitions.

267

273

274

281

285

286

Attributes 264

id: identifier (R) uniquely identifies all SBGN elements in the same namespace³.

Changes from Previous Version

Not defined in the previous version.

3.5.2 Map 268

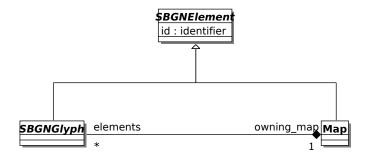


Figure 3.3: UML definition of Map and SBGNElement.

See footnote.

The Map⁴ (figure 3.3) is a container that holds all the glyphs (SBGNGlyph (see section 3.5.3)) drawn in a Process Description map. A map may represent a submap or a supermap and should comply with the rules set out in section 3.8. The elements held should be logically unique and conform to the identity rules in section 3.7.

Generalisation

• SBGNElement (see section 3.5.1)

Attributes 275

No additional attributes

Associations

elements:SBGNGlyph (*) The collection of glyphs held by the map.

Rules and Constraints 279

- A map is valid if it is empty (although not very useful).
- All instances of SBGNGlyph (see section 3.5.3) must be unique (see section 3.7).

Notation 28

The map is the canvas upon which the Process Description language is drawn. It's only visible feature is its colour. It can take any pattern or colour (or be transparent for that matter), but as SBGN is 'colour blind' this does not convey any meaning in itself.

Changes from Previous Version

Not defined explicitly in previous versions.

289

290

292

302

303

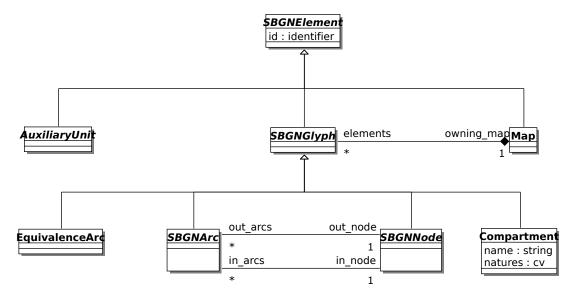


Figure 3.4: *UML definition of the SBGNGlyph and its subclasses.*

3.5.3 SBGNGlyph

The SBGNGlyph is the fundamental building blocks of the Process Description language. It is the only element that can be drawn directly on a map (Map (see section 3.5.2)).

Generalisation 291

• SBGNElement (see section 3.5.1)

Attributes 293

No additional attributes.

Associations 295

owning_map:Map (1) The map that contains this class.

Rules and Constraints

No additional rules and constraints.

Changes from Previous Version 299

Not defined in previous version.

3.5.4 AuxiliaryUnit

The AuxiliaryUnit (figure 3.5) represents symbols that may be used to adorn glyphs. In doing so they change the meaning of the glyph and/or provide additional information about it.

Generalisation 304

• SBGNElement (see section 3.5.1)

Attributes 306

No additional attributes.

²A new concept

³Not defined previously, but doesn't change Process Description semantics. reinforces the idea of instance identity that exists for all glyphs.

⁴Defining this explicitly is new, but the concept of the map as a container of glyphs has always existed in the Process Description language

317

318

319

320

321

322

323

324

325

328

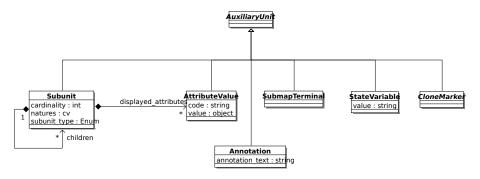


Figure 3.5: UML definition of the Auxiliary Unit and its subclasses.

Associations 308 No additional associations. 309 **Rules and Constraints** 310 No additional rules and constraints. 311 **Changes from Previous Version** 312 Not defined in previous version. 313 3.5.5 SBGNNode

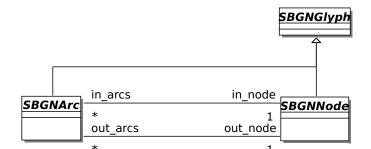


Figure 3.6: UML definition of the SBGNNode and SBGNArc classes.

The SBGNNode (figure 3.6) represents the nodes in the graph structure that is the core representation within the Process Description language. The nodes are connected to glyphs descended from SBGNArc for form a direct graph.

Generalisation

• SBGNGlyph (see section 3.5.3)

Attributes

No additional attributes.

Associations

out arcs:SBGNArc (*) The arcs that are leaving this node. in_arcs:SBGNArc (*) The arcs that are entering this node.

Rules and Constraints

• The set of SBGNNodes linked to this node via a SBGNArc (its adjacent nodes) must be all belong to different entity pools (as defined by EntityPoolNode) and cannot include more than one clone of the same entity pool.

333

334

335

336

342

343

345

350

351

352

353

Changes from Previous Version

Not defined in the previous version.

3.5.6 SBGNArc 331

The SBGNArc (figure 3.6) represents the directed arcs (also know as directed edges) in the directed graph structure that is the core representation within Process Description language. The arc is connected to two nodes descended from SBGNNode, one at each end. As the arc has a direction these nodes are by convention designated the out_node to indicate the nodes that the arc is leaving and in_node to indicate the node that it is entering.

Generalisation 337

• SBGNGlyph (see section 3.5.3)

Not defined in the previous version.

Attributes

No additional attributes.

Associations 341

out_node:SBGNNode (1) The node that this arc is leaving.
in_node:SBGNNode (1) The node that this arc is entering.

Rules and Constraints

No additional rules and constraints.

Changes from Previous Version

3.5.7 EntityType

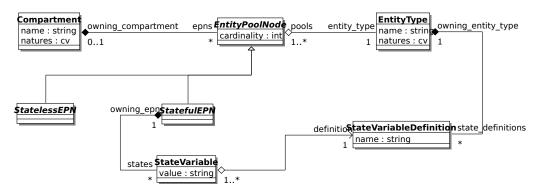


Figure 3.7: *UML definition of the entity type and the state variable definition. The diagram shows how these classes interact with the entity pool, state variable and so influence EntityPoolNode logical identity.*

<u>∧</u> See footnote. The EntityType⁵ is one of the code concepts in the Process Description language. It defines the type of entity that is instantiated by one or more entity pools in a Process Description map. The EntityType has associated state variable definitions (see figure 3.7) and this enforces one of the core rules in the Process Description language that once a state is associated with an entity type is must be used by all entity pools of that type.

⁵The concept of an entity pool's type has been there implicitly in previous versions and comes up in discussions. This class aims to formalise that concept and the rules associated with it and enable us to formalise rules associated with EPNs.

386

388

Generalisation 354 None. **Attributes** 356 name: string (R) The name that identifies the entity in the Process Description map. EPNs with the 357 same label should be from the same entity. the string cannot be empty and must start and end with a non-space character. Any Unicode character is acceptable⁶. 359 natures: cv(*) The nature of the entity pool node as defined by a controlled vocabulary. Zero, one or 360 more values may be set, but each one must belong to a different controlled vocabulary (see 361 section 3.6)⁷. **Associations** 363 state definitions: State Variable Definition (*) The state definitions associated with this type. 364 pools:EntityPoolNode (1..*) The entity pool nodes that used this type. 365 **Logical Identity** 366 Logical Key: 367 name 368 natures **Rule and Constraints** 370 All instances of EntityPoolNode associated with a particular EntityType must be of the same class. • If an instance of EntityType contains one or more instances of StateVariableDefinition then the 372 EntityPoolNodes associated with it must be subclasses of StatefulEPN. **Changes from Previous Version** Although not defined explicitly in the previous version, this concept and the associated rules did exist in the language. 376 3.5.8 **StateVariableDefinition** 377 The StateVariableDefinition⁸ defines the state variables used by an EntityType and therefore those state 378 variables that must exists in an EntityPoolNode (see figure 3.7). 379 Generalisation 380 None. 381 **Attributes** name: string (O) The name that of the state variable. This is optional, but if defined cannot be an

⚠ See footnote.

See footnote.

See footnote.

See footnote.

ter and end with a non-space character. It should not contain a '@' character 9.

Associations

owning_entity_type:EntityType (1) The EntityType that owns this definition.

Rule and Constraints

None.

empty string or just white space characters. It should also start with an alpha-numeric charac-

⁶Not discussed or defined anywhere, but would make sense to define this explicitly.

 $^{^{7}}$ This has been discussed on the mailing list where this seems to be the consensus solution

⁸As with EntityType this is new and aims to formalise the concept that an entity pool must preserve the same state variables whenever it is used.

⁹No rule defined previously, but this would seem to make sense.

392

393

397

398

400

402

403

404

405

406

407

408

409

410

414

415

416

417

418

Changes from Previous Version

Although not defined explicitly in the previous version, arguably this concept did exist in the language.

3.5.9 EntityPoolNode

An entity pool is a population of entities that cannot be distinguished from each other, when it comes to the SBGN Process Description Level 1 map. For instance all the molecular entities that fulfill the same role in a given process form an entity pool. As a result, an entity pool can represent different granularity levels, such as all the proteins, all the instances of a given protein, only certain forms of a given protein. To belong to a different compartment is sufficient to belong to different entity pools. Calcium ions in the endoplasmic reticulum and calcium ions in the cytosol belong to different entity pools when it comes to representing calcium release from the endoplasmic reticulum.

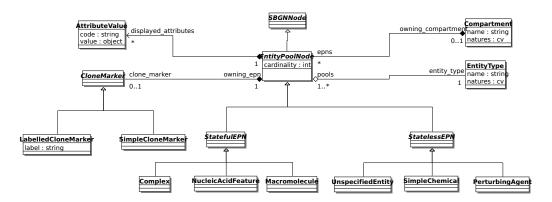


Figure 3.8: *UML definition of the entity pool node and its descendant glyphs.*

⚠ See footnote.

See footnote.

The EntityPoolNode (figure 3.8) is the definition of the entity pool and it shares an EntityType with other identical entities¹⁰. An instance of an entity pools is therefore distinguished from other pools with the same entity type by its cardinality, its owning_compartment and the values of its StateVariables (where appropriate). It must belong to a compartment or be associated with the map (c.f. section 3.5.27) and can contain a clone marker if it is cloned (see section 3.7)— note that not all EPNs can be cloned.

Generalisation

• SBGNNode (see section 3.5.5)

Attributes

cardinality: int (R) The number of copies of the entity. Must be a positive non-zero integer.

Associations

owning_compartment:Compartment (0..1) The compartment that this EPN belongs too. entity_type:EntityType (1) The type of this entity pool. clone_marker:CloneMarker (0..1) The clone marker decorator. See section 3.5.38 for its use.

clone_marker:CloneMarker (0..1) The clone marker decorator. See section 3.5.38 for its use. displayed_attributes:AttributeValue (*) One or more decorators used to display attribute values¹¹.

Logical Identity

Logical Key:

· owning compartment

 $^{^{10}}$ Although this concept is discussed it is not explicitly defined previously.

¹¹This is an alternate way of using the Unit of Information to display information, but to constrain it so that it presents attributes of the EPN not general annotation. See the AttributeValue class for more information.

- entity_typecardinality
- Rules and Constraints
 If cardinality > 1 then the descendant glyph must be displayed as a multimer.
 - If the EPN is drawn directly on a *Map* then owning_compartment is not set. We interpret this as belonging to an invisible default compartment.
 - natures can only use the material type (section 3.6.1), conceptual type (section 3.6.2) or physical characteristics (section 3.6.4) controlled vocabularies.
 - The appropriate subclass of CloneMarker must be used to distinguish logically identical instances of this class.
 - the EntityPoolNode must be associated with at least one SBGNArc (see section 3.5.6) (degree > 0).
 - All StateVariableDefinitions associated with the EntityType must have an associated StateVariable.

Notation

Although there is no direct graphical representation of this class the appearance of the AttributeValue and its associated glyph the *Unit of Information* us common to all subclasses so it is convenient to describe it here. The AttributeValue can be used to present the cardinality and natures of an EPN subclasses. These used the following codes to indicate which attribute is being presented:

- pc container physical characteristic
- mt entity pool material type
- ct entity pool conceptual type
- N multimer cardinality

Changes from Previous Version

Not defined explicitly in the previous version, but the concept of the EPN and its semantics existed. The main change to previous semantics is that of the natures, which didn't formally exist before, but which now must contain a unique set of controlled vocabularies and is part of the logical key of the EntityPoolNode.

3.5.10 Empty Set 446

It is useful to have the ability to represent the creation of an entity or a state from an unspecified source, that is, from something that one does not need or wish to make precise. For instance, in a model where the production of a protein is represented, it may not be desirable to represent all of the amino acids, sugars and other metabolites used, or the energy involved in the protein's creation. Similarly, we may not wish to bother representing the details of the destruction or decomposition of some biochemical species into a large number of more primitive entities, preferring instead to simply say that the species "disappears into a sink". Yet another example is that one may need to represent an input (respectively, output) into (resp. from) a compartment without explicitly representing a transport process from a source (resp. to a target).

For these and other situations, SBGN defines a single glyph to handle these situations representing the involvement of an external pool of entities. The symbol used in SBGN is borrowed from the mathematical symbol for "empty set", but it is important to note that it does not actually represent a true absence of everything or a physical void—it represents the absence of the corresponding structures in the model, that is, the fact that the external pool is conceptually outside the scope of the map.

464

465

466

468

469

470

471

472

475

476

477

478

479

481

484

485

487

A frequently asked question is, why bother having an explicit symbol at all? The reason is that one cannot simply use an arc that does not terminate on a node, because the dangling end could be mistaken to be pointing to another node in the map. This is specially true if the map is rescaled, causing the spacing of elements in the map to change. The availability and use of an explicit symbol for sources and sinks is critical.

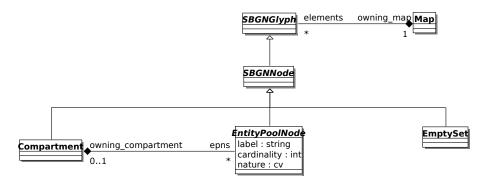


Figure 3.9: The UML definition of the EmptySet and its context in relation to other elements of the Process Description language.

The definition of the *Empty Set* is shown in figure 3.9. The empty set is not a subclass of EntityPoolNode as it does not represent a single pool of entities and does not share any of the other attributes of an EntityPoolNode, nor does it belong to a particular compartment 12.

See footnote.

Generalisation

• SBGNNode (see section 3.5.5)

Attributes

No additional attributes.

Associations

No additional associations.

Rules and Constraints

- All instances of *Empty Set* can be regarded as identical therefore not special decoration is used to indicate replication on the map.
- the EmptySet must be associated with at least one SBGNArc (see section 3.5.6) (degree > 0).

Notation 480

Glyph: Empty Set

Label: None

SBO Term: SBO:0000291! empty set

Container: Represented by the mathematical symbol for "empty set", that is, a circle crossed by a bar linking the upper-right and lower-left corners of an invisible square drawn around the circle (Ø). Figure 3.10 illustrates this. The symbol should be linked to one and only one edge in a map.

¹²This is a significant change to the semantics from v1.3 since it is no longer an EPN.

491

492

493

494

497

502

504

505



Figure 3.10: *The* empty set *glyph*.

Changes from Previous Version

The EmptySet and *Empty Set* glyph has replaced the *Source* and *Sink* glyphs. This symbols used remains the same, but the underlying concept has changed. The *Source* and *Sink* glyphs where types of EPN, representing single entity pools, while the EmptySet is not.

3.5.11 StatelessEPN

The StatelessEPN (figure 3.11) represents a pool where the entities do not change 'state'. In otherwords the entities do not undergo any physical change that is useful to record in a Process Description map. Therefore they cannot be assigned a state-variable.

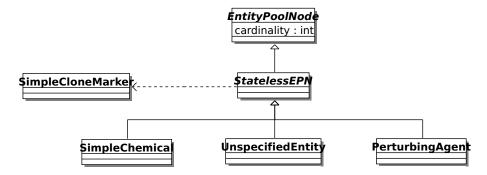


Figure 3.11: UML definition of the stateless entity pool node and its descendant glyphs.

Generalisation 496

• EntityPoolNode (see section 3.5.9)

Rules and Constraints

3.5.12 Simple chemical

Attributes 498

No additional attributes.

Associations 500

No additional associations. 501

• if a clone marker is used it must be of type SimpleCloneMarker.

a dione marker is used it must be of type simple of one warker.

Changes from Previous Version

Not defined in the previous version.

A SimpleChemical is the 'opposite' of a macromolecule (Section 3.5.16): it is a chemical compound that is *not* formed by the covalent linking of pseudo-identical residues. Examples of simple chemicals are an atom, a monoatomic ion, a salt, a radical, a solid metal, a crystal, etc.

521

523

524

526

528

529

530

531

532

533

534

537

538

Generalisation 510

• StatelessEPN (see section 3.5.11)

Attributes 5

No additional attributes. 51

Associations

No additional associations.

514

Rules and Constraints 516

No additional rules and constraints.

Notation 518

There are two glyphs associated with SimpleChemical. The first *simple chemical monomer* is used when cardinality = 1 and the second *simple chemical multimer* is used when cardinality > 1.

Glyph: Simple chemical monomer

SBO Term: SBO:0000247! simple chemical

Container: A *simple chemical* is represented by a 'stadium' symbol: a circle split in two with a rectangle inserted between them (see figure 3.12). If desired the rectangle can have zero length and the symbol is then identical to a circle (Figure 3.12). To avoid confusion with the Unspecified Entity (3.5.13), this form of the glyph must remain a circle and cannot be deformed into an eclipse.

Label: The identification of the *simple chemical* is carried by an unbordered box containing a string of characters. The characters may be distributed on several lines to improve readability, although this is not mandatory. The label box has to be attached to the center of the circular container. The label is permitted to spill outside the container.



Figure 3.12: The Process Description glyph for simple chemical monomer. The stadium form and circular forms are shown, as are the cloned forms of the glyph.

Glyph: Simple chemical multimer

SBO Term: SBO:0000421! multimer of simple chemicals

Container: A *simple chemical multimer* is represented by two identical containers shifted horizontally and vertically and stacked one on top of the other. Figure 3.13 illustrates the glyph.

Label: The multimer carries an identifying label. The label is placed in an unbordered box containing a string of characters. The characters can be distributed on several lines to improve readability, although this is not mandatory. The label box must be attached to the center of the top monomer's container. The label may spill outside of the container.

542

543

544

546

547

548

550

552

553

554

555

556

557

567

568

569



Figure 3.13: *The Process Description glyph for* simple chemical multimer. *The figures show the stadium and circular forms, and their cloned variants.*

Changes from Previous Version

The glyphs used for the SimpleChemical have been changed to the stadium glyph. Previously the glyph was a circle. To maintain compatibility with previous versions the stadium symbol can be drawn without straight horizontal elements so that it becomes a circle.

3.5.13 UnspecifiedEntity

The simplest type of EntityPoolNode is the UnspecifiedEntity — one whose type is unknown or simply not relevant to the purposes of the map. This arises, for example, when the existence of the entity has been inferred indirectly, or when the entity is merely a construct introduced for the needs of a map, without direct biological relevance. These are examples of situations where the UnspecifiedEntity is appropriate. (Conversely, for cases where the identity of the entities composing the pool *is* known, there exist other, more specific glyphs described elsewhere in the specification.)

Generalisation

• StatelessEPN (see section 3.5.11)

Attributes

No additional attributes.

Associations

No additional associations.

Rules and Constraints

- The UnspecifiedEntity cannot have cardinality > 1. This means there is no multimer glyph.
- It cannot have a natures ¹³, therefore natures must **not** be set.

See footnote.

Notation

Glyph: Unspecified entity

SBO Term: SBO:0000285! material entity of unspecified nature

Container: An *unspecified entity* is represented by an elliptic container, as shown in 3.14. Note that this must remain an ellipse to avoid confusion with the Simple Chemical glyph, which is a circle (c.f. 3.5.12).

Label: An *unspecified entity* is identified by a label placed in an unbordered box containing a string of characters. The characters can be distributed on several lines to improve readability, although this is not mandatory. The label box must be attached to the center of the container. The label may spill outside of the container.

 $^{^{13}}$ Not explicitly discussed, but natures don't make sense here.

Changes from Previous Version

577

581

587

588

592

593

594

595

596



Figure 3.14: The Process Description glyph for unspecified entity.

No changes from the previous version. 3.5.14 Perturbing Agent 572

Biochemical networks can be affected by external influences. Those influences can be the effect of well-defined physical perturbing agents, such as a light pulse or a change in temperature; they can also be more complex and not well-defined phenomena, for instance the outcome of a biological process, an experimental setup, or a mutation. For these situations, SBGN provides the perturbing agent glyph. It is an EPN, and represents the amount to perturbing agent applied to a process.

Generalisation 578

• StatelessEPN (see section 3.5.11) 579

Attributes No additional attributes.

Associations 582

No additional attributes. 583

Rules and Constraints

• The PerturbingAgent cannot have cardinality > 1. This means there is no multimer glyph.

Notation

Glyph: Perturbing agent **SBO Term:** SBO:0000405! perturbing agent

Container: A *perturbing agent* is represented by a modified hexagon having two opposite concave 589

faces, as illustrated in Figure 3.15.

Label: A perturbing agent is identified by a label placed in an unbordered box containing a string of characters. The characters can be distributed on several lines to improve readability, although this is not mandatory. The label box must be attached to the center of the perturbing agent container. The label may spill outside of the container.



Figure 3.15: The Process Description glyph for perturbing agent.

Changes from Previous Version

No changes from pervious version.

601

603

608

609

610

611

612

613

617

618

619

620

621

625

3.5.15 StatefulEPN 597

Stateful entity pools can undergo physical changes, for example chemical modification or conformational change, which we wish to record in a Process Description map. This information is captured via the StateVariable (as can be seen in figure 3.16). The LabellecCloneMarker must be used to indicated that the StatefulEPN is cloned.

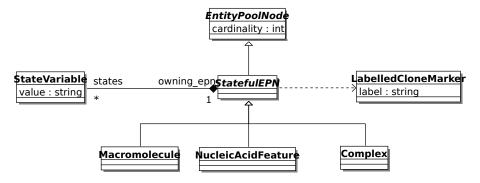


Figure 3.16: *UML definition of the stateful entity pool node: showing its descendants and its association with state variables.*

Generalisation 602

• EntityPoolNode (see section 3.5.9)

Attributes 604

No additional attributes.

Associations

states:StateVariable (*) The state variables that belong to this class.

Rules and Constraints

• State variables do not need to be logically unique, therefore two or more state variables with the same name are permitted.

• The LabelledCloneMarker must be used to indicate cloning for instances of StatefulEPN and its subclasses, with a must use the same

Changes from Previous Version

Not defined explicitly in the previous version.

3.5.16 Macromolecule

Many biological processes involve *macromolecules*: biochemical substances that are built up from the covalent linking of pseudo-identical units. Examples of macromolecules include proteins, nucleic acids (RNA, DNA), and polysaccharides (glycogen, cellulose, starch, etc.). Attempting to define a separate glyph for all of these different molecules would lead to an explosion of symbols in SBGN, so instead, SBGN Process Description Level 1 defines only one glyph for all macromolecules. The same glyph is to be used for a protein, a nucleic acid, a complex sugar, and so on. The exact nature of a particular macromolecule in a map is then clarified using its label and decorations, as will become clear below.

Generalisation

• StatefulEPN (see section 3.5.15)

635

637

638

640

641

642

643

646

649

650

651

652

Attributes

No additional attributes.

626

Associations

No additional associations. 629

Rules and Constraints

No additional rules and constraints.

Notation 632

There are two glyphs associated with Macromolecule. The first $Macromolecule\ monomer$ is used when cardinality = 1 and the second $Macromolecule\ multimer$ is used when cardinality > 1.

Glyph: Macromolecule monomer

SBO Term: SBO:0000245! macromolecule

Container: A macromolecule is represented by a rectangular container with rounded corners, as illustrated in Figure 3.17.

Label: A *macromolecule* is identified by a label placed in an unbordered box containing a string of characters. The characters can be distributed on several lines to improve readability, although this is not mandatory. The label box must be attached to the center of the container. The label may spill outside of the container.



Figure 3.17: The Process Description glyph for macromolecule, shown plain and unadorned on the left, and with an additional state variable and a unit of information in the middle and the cloned form on the right.

Glyph: Macromolecule multimer

SBO Term: SBO:0000420! multimer of macromolecules

Container: A *multimer* is represented by two identical containers shifted horizontally and vertically and stacked one on top of the other. Figure 3.18 illustrates the glyph.

Label: As monomer

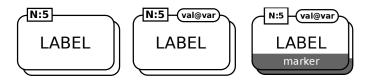


Figure 3.18: The Process Description glyph for macromolecule multimer, shown plain and unadorned on the left, and with an additional state variable and a unit of information in the right and the cloned form on the right.

Usage Examples In this section, we provide examples of Entity Pool Node representations drawn using the SBGN Process Description Level 1 glyphs described above.

Figure 3.19 represents calcium/calmodulin kinase II, with phosphorylation on the sites threonine 286 and 306, as well as catalytic and autoinhibitory domains. Note the use of *units of information* and *state variables*.

655

656

658

660

661

662

663

665

668

669

672

673

674

675

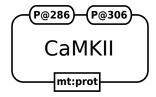


Figure 3.19: An example representation of calcium/calmodulin kinase II.

Figure 3.20 represents the glutamate receptor in the open state, with both phosphorylation and glycosylation. The entity carries two functional domains, the ligand-binding domain and the ion pore, and its chemical nature is precided.

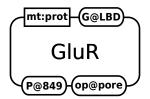


Figure 3.20: An example of a glutamate receptor in the open state.

Changes from Previous Version

No changes from the previous version.

3.5.17 NucleicAcidFeature

The NucleicAcidFeature represents a fragment of a macromolecule carrying genetic information. A common use for this construct is to represent a gene or transcript. The label of this EPN and its natures are often important for making the purpose clear to the reader of a map.

Generalisation

• StatefulEPN (see section 3.5.15)

Attributes 664

No additional attributes.

Associations

No additional associations

Rules and Constraints

No additional associations.

No additional rules and constraints.

Notation 670

The NucleicAcidFeature has two associated glyphs. The first $Nucleic\ acid\ feature\ monomer$ is used when cardinality = 1 and the second, $Nucleic\ acid\ feature\ multimer$ is used when cardinality > 1.

when cardinality = 1 and the second, Nucleic acia feature multimer is used when cardinality > 1.

Glyph: *Nucleic acid feature monomer* This glyphs represents a monomeric macromolecule. **SBO Term:** SBO:0000354! informational molecule segment

Container: A *nucleic acid feature* is represented by a rectangular container whose bottom half has rounded corners, as shown in Figure 3.21.

681

683

684

685

686

687

691

692

Label: The identity of a particular *Nucleic acid feature* is established by a label placed in an unordered box containing a string of characters. The characters may be distributed on several lines to improve readability, although this is not mandatory. The label box must be attached to the center of the container. The label may spill outside of the container.



Figure 3.21: The Process Description glyph for nucleic acid feature monomer, shown plain and unadorned on the left and with an additional state variable and a unit of information in the middle and the cloned form on the right.

Glyph: *Nucleic acid feature multimer* This glyphs represents a multimeric macromolecule.
 SBO Term: SBO:0000419! multimer of informational molecule segments
 Container: A *Nucleic acid feature multimer* is represented by two identical containers shifted horizontally and vertically and stacked one on top of the other. Figure 3.22 illustrates the glyph.
 Label: As monomer glyph.



Figure 3.22: The Process Description glyph for nucleic acid feature multimer, shown plain and unadorned on the left and with an additional state variable and a unit of information in the middle and the cloned form on the right.

Changes from Previous Version

No changes from the previous version.

3.5.18 Complex 688

A Complex represents a biochemical entity composed of other biochemical entities, whether macromolecules, simple chemicals, multimers, or other complexes (figure 3.23). The Complex can described its composition by the set of Subunits it contains (see figure 3.5.19). This description is entirely optional and is their to assist the user with a visual shorthand about the composition of the complex.

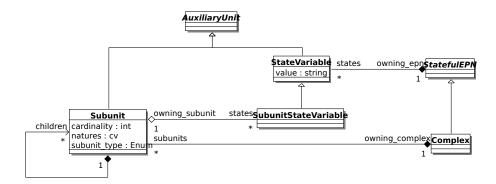


Figure 3.23: The UML definition of the Complex and its associated subunits. In particular this describes organisation of the state variables that belong to both the subunit, but also the complex.

696

699

701

702

703

705

707

708

709

710

711

713

714

717

718

721

723

724

Generalisations

EntityPoolNode (see section 3.5.9)

Attributes

Associations 697

subunits:Subunit (*) The subunits that describe the composition of this complex. 698

Special Rules and Constraints

No additional attributes

See footnote.

- Once a set of subunits are defined for an Complex with a given EntityType, then they must be used by every instance using that entity type. 14.
- The set of subunits in the Complex does not identify it. One or more Complexes that contain the same set of subunits, but have different labels are **not** identical.

Notation 704

The Complex is represented by two glyphs, the *Complex Monomer* which represents a Complex where the cardinality is one and the Complex Multimer where the cardinality is greater than that.

Complex Monomer

SBO Term: SBO:0000253! non-covalent complex

Container: A *complex* possesses its own container box surrounding the juxtaposed container boxes of its components. This container box is a rectangle with cut-corners (an octagonal box with sides of two different lengths). The size of the cut-corners are adjusted so that there is no overlap between the container and the components. The container boxes of the components must not overlap.

Label: The identification of a *named complex* is carried by an unbordered box containing a string of characters. The characters may be distributed on several lines to improve readability, although this is not mandatory. Ideally the label box should be attached to the midway between the border of the complex's container box and the border of the components' container boxes. However, if the Complex contains Subunit glyphs then the label may be positions to optimise the clarity and avoid overlapping.

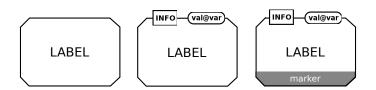


Figure 3.24: *The* complex *glyph*.

Complex Multimer 720 **SBO Term:** SBO:0000418! multimer of complexes

Container: A *Complex Multimer* is represented by two identical *Complex* containers shifted hori-

zontally and vertically and stacked one on top of the other. Figure 3.25 illustrates the glyph.

Label: As monomer

¹⁴New rule.

729

731

732

737

738

739

741

743

744

746

747

750

751

752

753

754

755 756

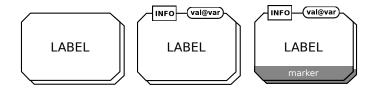


Figure 3.25: *The* Complex Multimer *glyph*.

Layout Rules and Guidelines

- The subunits inside the complex must not overlap.
- The subunits should sit above the clone marker so that they are not obscured by it.
- The label should not be obscured by subunits or obscure them.

Changes from Previous Version

- Clarified that complex must have a label and the label identifies the complex irrespective of its subunit composition.
- The label positioning does not need to be at the centre of the Complex glyph.

3.5.19 Subunit 733

<u>∠!\</u> See footnote. The Subunit¹⁵ is used to describe the composition of the Complex (see section 3.5.18). A complex can optionally be decorated with one or more subunits, which represent the types of EntityPoolNode (see section 3.5.9) that may aggregate to form a complex. As we can see from the UML representation (figure 3.23) the Subunit is an auxiliary unit that decorates the Complex and does not represent an entity pool directly. In addition it does not mimic the EntityPoolNode class hierarchy (Subunit, but rather uses the subunit_type attribute to indicate the type of subunit.

Generalisation 740

• EntityPoolNode (see section 3.5.9)

Attributes 742

cardinality: int $(\mbox{\bf R})~$ The number of copies of the subunit.

name: string (**0**) The name of the subunit.

subunit_type: enum (R) The type of the subunit. It can have one of the following values that correspond to the equivalent EPN class: SimpleChemical, UnspecifiedEntity, PerturbingAgent, Macromolecule, NucleicAcidFeature, Complex.

Associations 748

owning_complex:Complex (1) The complex that owns the subunit. states:SubunitStateVariable (*) The state variables assigned to this subunit. children:Subunit (*) Subunits that are contained by this subunit.

Rules and Constraints

- Two or more state variables with the same name are permitted.
- State variables with no name set are permitted.
- Subunits can also contain subunits. There is no limit on such nesting. The namespace rules below apply.

¹⁵The principles of the subunit have been agreed, but the details of this implementation should be reviewed. In particular the rules about state variables.

759

760

761

762

763

767

769

776

777

778

780

781

- The subunit defines a namespace for its state variables, e.g. subunit "A" assigned a state variable "P@Ser202" and a subunit "B" assigned the same state variable can be distinguised as A:P@Ser202 and B:P@Ser202.
- If the subunit is of type Complex then children can contain one or more Subunit instances.
- If the subunit has a cardinality > 1 then this should be displayed by the AttributeValue (see section 3.5.28).
- If natures contains one or more instances then these must be displayed via an Attribute Value.

Notation

The subunit symbol used for the *subunit* glyph varies depending on the subunit_type and cardinality. The symbols available are equivalent to those used by the EPN glyphs including the *complex*. Therefore it is possible to describe complexes within complexes. The mapping between these and the symbol used is shown table 3.1. Not that subunits may contain labels corresponding to their name.

Table 3.1: Mapping between the subunit_type, cardinality values of Subunit and the glyphs used to represent it. These are essentially the EPN glyphs described in this document.

subunit_type	cardinality = 0	cardinality > 0
SimpleChemical	Simple Chemical Monomer	Simple Chemical Multimer
UnspecifiedEntity	Unspecificed Entity	None
PerturbingAgent	Perturbing Agent	None
Macromolecule	Macromolecule Monomer	Macromolecule Multimer
NucleicAcidFeature	Nucleic Acid Feature Monomer	Nucleic Acid Feature Multimer
Complex	Complex Monomer	Complex Multimer

The example in figure 3.26 illustrates the use of subunits in a complex. It also shows an equivalent compex without subunits. This is an import point. For every *Complex* drawn with subunits it will always be possible to drawn an equivalent version that does not use contains subunits ¹⁶.



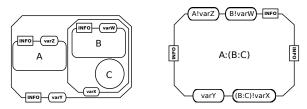


Figure 3.26: Both these complex glyphs are equivalent. The one on the left is described using sub-unit decorators, the one on the right describes the same thing without them.

Changes from Previous Version

In previous version of the spec the subunits of a Complex were regarded as an EPN. This however, is incorrect as it implies there are pools within pools, which breaks one of the fundamental paradigms of the Process Description language. This is corrected in the current version and subunits are now adornments of the Complex.

3.5.20 ProcessNode

The Process (figure 3.27) represents a process that transforms one or more entity pools into one or more entity pools, that are identical or different. A process may be used to represent or summarise more than one known process.

¹⁶Not discussed in detail. This must be true if states can be drawn on subunits, but actually belong to the complex. Either this or we enforce a rule that all state vars must be named uniquely.

785

790

791

792

793

795

796

797

800

801

803

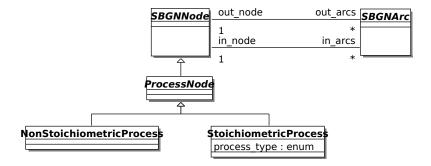


Figure 3.27: The UML definition of the Process and its associated subclasses. Note that the Process extends SBGNNode so all its descendants can potentially be nodes in a directed graph.

Generalisation 782

• SBGNNode (see section 3.5.5)

Attributes 784

No additional attributes.

Associations 786

No additional associations.

Rules and Constraints

No additional rules and constraints.

Changes from Previous Version

- This was not explicitly defined in the previous version, but this version did define a glyph called *Process*. To avoid ambiguity this glyph has now been renamed *Stoichiometric Process* (see section 3.5.25).
- Previous specifications stated that processed could be duplicated when all associated EPNs were cloned. This behaviour has been changed the current status where all processes are unique in a Process Description map.

3.5.21 NonStoichiometricProcess

See footnote.

The NonStoichiometricProcess¹⁷ (figure 3.28) is a type of process. It does not necessarily result in a measurable change of entity pools, nor does it necessarily have a defined start and end point. In many cases the process is not well defined. This may because it is not well understood or because the detail is not important or is being summarised.

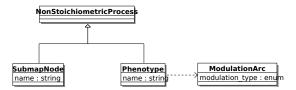


Figure 3.28: The UML definition of the NonStoichiometricProcess and its associated subclasses.

Generalisation 802

ProcessNode (see section 3.5.20)

 $^{^{17}}$ This has been discussed and agreed in past meetings

	Attributes	804		
	No additional attributes.	805		
	Associations	806		
	No additional associations.	807		
	Rules and Constraints	808		
	No additional rules and constraints.	809		
	Changes from Previous Version	810		
	Not defined in the previous version.	811		
3.5.22	Phenotype	812		
	A biochemical network can generate phenotypes or affect biological processes. Such processes can			
	take place at different levels and are independent of the biochemical network itself. To represent			
	these processes in a map, SBGN defines the Phenotype (figure 3.28).			
	Generalisation			
	NonStoichiometricProcess (see section 3.5.21)	817		
	Attributes	818		
	name: string (R) The name of the phenotype.	819		
	Associations	820		
	No additional associations.	821		
	Logical Identity	822		
	Logical Key:	823		
	• owning_map	824		
	• name	825		
	Rules and Constraints	826		
	• The number of in_arcs must be > 0.	827		
	• in_arc can only contain instances of ModulationArc (see section 3.5.35) and its subclasses.	828		
	• out_arcs must be empty.	829		
	Notation	830		
	Glyph: Phenotype			
	SBO Term: SBO:0000358! phenotype			
	Container: A <i>phenotype</i> is represented by an elongated hexagon, as illustrated in Figure 3.29.			
	Label: A <i>phenotype</i> is identified by a label placed in an unbordered box containing a string of char-			
	acters. The characters can be distributed on several lines to improve readability, although this	835		
	is not mandatory. The label box must be attached to the center of the <i>phenotype</i> container. The	836		
	label may spill outside of the container.	837		



 $\textbf{Figure 3.29:} \ \textit{The Process Description glyph for } phenotype.$

840

845

846

849

851

852

854

855

856

857

858

859

860

861

862

864

865

Changes from Previous Version

This definition clarifies that the Phenotype cannot be cloned as it is now a subclass of Process, which is always unique.

3.5.23 SubmapNode

See footnote.

The SubmapNode¹⁸ (figure 3.30) is a placeholder for another process and is used when one wishes to hide the detail of this process from the Process Description map, but make it available to the reader as a separate related map. The Submap is not equivalent to an OmittedProcess (section 3.5.25). The Submap allows the detail of section of the Process Description map to be exported to another Process Description map and replaced by the SubmapNode, which acts as a place-holder. This is described in section 3.5.2 and the semantics of submap linking is defined in section 3.8.

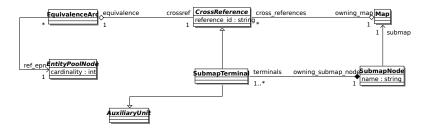


Figure 3.30: The UML definition of the SubmapNode and its relationship to its submap, tags etc.

Generalisation 848

• NonStoichiometricProcess (see section 3.5.21)

Attributes 850

name: string **(R)** The name of the submap that is being summarised. Note that this name ideally will indicate the function or the processes that are being summarised.

Associations 853

terminals:SubmapTerminal (1..*) The terminals provide a reference between the EPNs in the Main Map and those in the submap, which are identified by a Tag.

Logical Identity

Logical Key:

- · owning map
- name

Rules and Constraints

- All instances of SubmapTerminal (see section 3.5.32) held by this class must be logically unique.
- attribin arcs and out arcs must be empty (i.e.,, degree = 0).

Notation 863

Glyph: Submap Node

SBO Term: SBO:0000395! encapsulating process

Container: The *submap* is represented as a square box to remind the viewer that it is fundamentally a process.

 $^{^{18}}$ This name change has not been discussed at the time of writing. The aim is to provide clarity between the submap and this glyph.

870

873

874

Label: The identification of the *submap* is carried by an unbordered box containing a string of characters. The characters may be distributed on several lines to improve readability, although this is not mandatory. The label box has to be attached to the center of the container box.

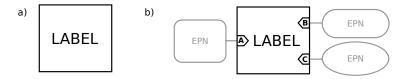


Figure 3.31: The Process Description glyph for submap. (a) the basic glyph symbol, without the submap terminal auxiliary units that would normally be associated with it. (b) The glyph as it would typically be used within a map — associated with EPN glyphs and containing submap

Changes from Previous Version

This glyph was called *Submap* in previous version of the Process Description specification. This is confusing when talking about the Submap itself so this glyph is now referred to as the SubmapNode to distinguish it.

3.5.24 LogicalOperator

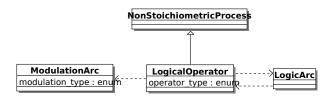


Figure 3.32: The UML definition of the LogicalOperator.

The LogicalOperator (figure 3.32 performs a Boolean operation on one or more inputs to give a binary output. The input must be a Boolean value, and are obtained from the LogicArc (see section 3.5.36) connected to the LogicOperator. The output a two-value quantity, 0 for False and positive nonzero for True. This is required because the output of the LogicOperator must be connected to either a LogicArc or a ModulationArc (see section 3.5.35) both of which require their out node to provide a quality. The behaviour of the logical operator for each type of operator type is shown in the following table:

AND All inputs must be True for output to be True, otherwise output is false.

OR At least one input must be True for output to be True. If all inputs are False then output is False.

NOT Only one input is permitted and the output is the inversion of the input. Therefore True gives False and False gives True.

Generalisation

NonStoichiometricProcess (see section 3.5.21)

Attributes

operator_type: enum (R) The operator type must be one of the following enumerations: AND, OR, NOT.

881

882 883

884

885

886

887

888

Associations 889 No additional associations. **Rules and Constraints** 891 • in_arc can only contain one or more instances of LogicArc. 892 • out_arc can only contain one or more instances of LogicArc or ModulationArc. 893 • if operator_type is AND or OR, then in_arc must contain two or more arcs. • if operator_type is NOT then in_arc must contain only one arc. 895 out_arc can contain only one arc. 896 **Notation** Three glyphs are used to represent the different operator types. The glyphs are names after the corresponding type. 899 Glyph: And 900 SBO Term: SBO:0000173! and. 901 **Node:** *And* is represented by a circle carrying the word "AND". 902



Figure 3.33: The Process Description glyph for and. Only two inputs are represented, but more would be allowed.

Glyph: Or 903 SBO Term: SBO:0000174! or. 904 Node: Or is represented by a circle carrying the word "OR". 905

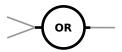


Figure 3.34: The Process Description glyph for or. Only two inputs are represented, but more would be allowed.

Glyph: Not
SBO Term: SBO:0000238! not.
907
Node: Not is represented by a circle carrying the word "NOT".
908



Figure 3.35: *The Process Description glyph for* not.

Changes from Previous Version

Although the LogicOperator was not explicitly defined in the previous version the semantics and glyphs are unchanged. 910

915

916

918

919

924

925

3.5.25 StoichiometricProcess

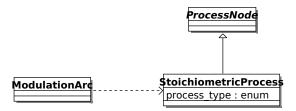


Figure 3.36: The UML definition of the StiochiometricProcess. The class interacts with subclasses of FluxArc and ModulationArc.



A stoichiometric process¹⁹ produces a measurable change in the quantities of entity pools consumed and produced. This might imply modification of covalent bonds (conversion), modification of the relative position of constituents (conformational process) or movement from one compartment to another (translocation). Such a process will have a basal rate at which this change occurs, which can be affected positively or negatively by the other entity pools, which 'modulate' the process. Examples of this include stimulation, inhibition and catalysis. In an irreversible process the entity pools interacting with it can be grouped into inputs and outputs. However, a stoichiometric process can also be reversible and so for convenience we refer to these groupings as the "left-hand-side" (LHS) and "right-hand-side" (RHS) of the process²⁰ (figure 3.37).

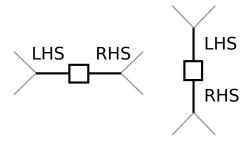


Figure 3.37: An illustration of the "sidedness" of a process. The designation of LHS and RHS is essentially arbitrary.

In the Process Description language this is represented by the Stoichiometric Process (figure 3.36). It can be one of several different types, which indicate the amount that is known about the process or in some cases the nature of the process, for example association and dissociation. The permitted values for process_type are described in the following table:

¹⁹New concept, but discussed in previous meetings. The semantics of the process being stoichiometrically balanced has not been discussed in detail for the stoichiometric process, and this is the subject of a tracker query. The spec previously stated that the process should be balanced and this is therefore consistent with that.

²⁰Note this designation is purely for grouping and is used even then the sides of the reaction are above and below the process.

generic	A generic stoichiometric process that transforms a set of entity pools into another set of entity pools.
omitted	Omitted processes are processes that are known to exist, but are omitted from the map for the sake of clarity or parsimony. A single <i>omitted process</i> can represent any number of actual processes. The <i>omitted process</i> is different from a <i>submap</i> . While a <i>submap</i> references to an explicit content, that is hidden in the main map, the <i>omitted process</i> does not "hide" anything
	within the context of the map, and cannot be "unfolded".
uncertain	Uncertain processes are processes that may not exist. A single <i>uncertain process</i> can represent any number of actual processes.
association	The association between one or more <i>EPNs</i> represents the non-covalent binding of the biological objects represented by those <i>EPNs</i> into a larger complex.
dissociation	The dissociation of an <i>EPN</i> into one or more <i>EPNs</i> represents the rupture of a non-covalent binding between the biological entities represented by those <i>EPNs</i> .

Since this process is stoichiometric the relative quantities of the entity pools participating the process must be specified. For this reason the FluxArc (see section 3.5.34) has an stoichiometry attribute and each EntityPoolNode (see section 3.5.9) has a cardinality, which should be balanced in a valid Process Description map. This is especially important where there is potential ambiguity in the stoichiometry of the process (figure 3.38).

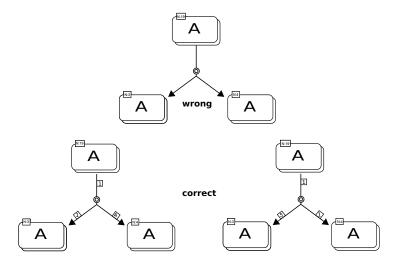


Figure 3.38: The figure illustrates why for the stoichiometry label is required to clarify potentially ambiguous stoichiometry. In the top example there is more than one possible solution, which can only be made clear using the stoichiometry labels in the bottom examples.

A stoichiometric process is deemed to be reversible its in_arcs are FluxArcs of type 'reversible' (see figure 3.39). Semantically, this permits a reversible flow of entities through the process. Modulation of a reversible process affects the rate of flux through the process, but does not directly affect the direction of that flow.

939

940

948

949

950

952

953

955

956

958

960

961

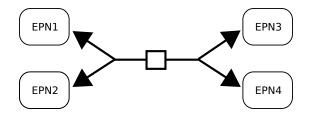


Figure 3.39: A valid reversible process. A process is reversible if its LHS and RHS contain only production arcs.

Generalisation	936

• ProcessNode (see section 3.5.20)

Attributes 938

process_type: enum (R) This must be one of the following enumerations: generic, omitted, uncertain, association, dissociation.

Associations 941

No additional associations.

Rules and Constraints

General 94

- The in_arc must contain one or more FluxArcs containing the same flux_type value.
- The in_arc may only contains FluxArc instances with a flux_type of 'consumption', or 'reversible'.
- In addition the in arc may contain zero, one or more instances of ModulationArc.
- The out_arc must contain one or more instances of FluxArc with a flux_type or 'production'.
- If in arcs contains one or more FluxArcs of type 'reversible' this process reversible.
- The EntityPoolNodes that make up the LHS of the process should be consistent with the RHS, i.e. the process should be stoichiometrically balanced.²¹
- If at least one FluxArc associated with a StoichiometricProcess displays its stoichiometry via a *stoichiometry label* then all must.²²
- If more than one set of stoichiometries can be applied to the flux arcs of the process then the stoichiometry of the flux arcs must be displayed.

Association These rules apply if the process_type is 'association'.

- The process must be irreversible.
- There can only be one 'production' FluxArc, with stoichiometry = 1.
- If a Complex is on the RHS of the association then there must be at least 2 EPNS on the LHS. Is this too restrictive? It prevents multimers being represented as a complex of 2 identical subunits. It is taken from v1.0 of the spec and got lost in later versions.





 $^{^{21}}$ Tracker issue 329060. If the process is stoichiometric this must make sense. The previous spec states this so this is consistent with it.

²²Take from previous spec, but that said if one displays stoichiometry in a map which is too restrictive.

964

965

966

972

975

976

Dissociation These rules apply if the process_type is 'dissociation'.

- The process must be irreversible.
- There can only be one 'consumption FluxArc, with stoichiometry = 1.
- If a Complex is on the LHS of the dissociation then there must be at least 2 EPNS on the RHS. see comment in association rules.

Notation 967

Glyph: Process

SBO Term: SBO:0000375! process

Node: A process is represented by a square box linked to two connectors: small arcs attached to the centers of opposite sides and referred to here as 'lugs'²³. The flux arcs arcs are linked to the ends of the lugs as shown in figure 3.40. The lug's purpose is to 'gather' the flux arcs together before meeting the process node proper and in doing so they emphasis the 'sides' of the reaction. Therefore the lug must have a visually appreciable length²⁴ and must be placed on opposite sides of the process square. The modulatory arcs (section 3.5.35) point to the other two sides of the box.

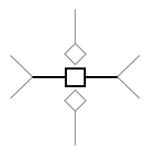


Figure 3.40: The Process Description glyph for process.

The example in Figure 3.41 illustrates the use of a *process* node to represent the phosphorylation of a protein in a Process Description.

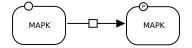


Figure 3.41: Phosphorylation of the protein MAP kinase.

The example in Figure 3.42 illustrates the use of a *process* node to represent a reaction between two reactants that generates three products.







978

²³The term lugs is used in discussion. We haven't discussed this in detail or at least come to a consensus on it. In particular does the lug need to be perpendicular to the process and does it need to be a straight line? How should it be used when the arc connecting to it is curved.

²⁴Undefined previously, but if we define it then it should be visible.

988

989

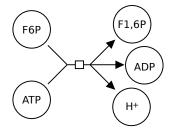


Figure 3.42: Reaction between ATP and fructose-6-phosphate to produce fructose-1,6-biphosphate, ADP and a proton.

The example in Figure 3.43 illustrates the use of a *process* node to represent a translocation. The large round-cornered rectangle represents a compartment border (see Section 3.5.27).

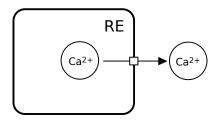


Figure 3.43: *Translocation of calcium ion out of the endoplasmic reticulum. Note that the* process *does not have to be located on the boundary of the* compartment. *A* process *is not attached to any* compartment.

The example in Figure 3.44 illustrates the use of a *process* node to represent the reversible opening and closing of an ionic channel in a Process Description.

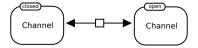


Figure 3.44: Reversible opening and closing of an ionic channel.

When such a reversible process is asymmetrically modulated, it must be represented by two different processes in a Process Description. Figure 3.45 illustrates the use of two *process* nodes to represent the reversible activation of a G-protein coupled receptor. In the absence of any effector, an equilibrium exists between the inactive and active forms. The agonist stabilises the active form, while the inverse agonist stabilises the inactive form.

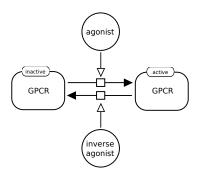


Figure 3.45: The reversible activation of a G-protein coupled receptor.

993

996

999

1000

1003

The example in Figure 3.46 presents the conversion of two galactoses into a lactose. Galactoses are represented by only one *simple chemical*, the cardinality being carried by the *consumption* arc.

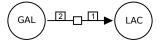


Figure 3.46: Conversion of two galactoses into a lactose.

Glyph: Omitted process

SBO Term: SBO:0000397 - omitted process.

Node: An *omitted process* is represented by a *process* in which the square box contains a two parallel slanted lines oriented northwest-to-southeast and separated by an empty space.

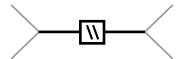


Figure 3.47: The Process Description glyph for omitted process.

Glyph: Uncertain process

SBO Term: SBO:0000396! uncertain process.

Node: An *uncertain process* is represented by a *process* which square box contains a question mark.

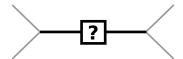


Figure 3.48: *The Process Description glyph for an* uncertain process.

Glyph: *Association* **SBO Term:** SBO:0000177! non-covalent binding.

Node: An *association* between several entities is represented by a filled disc linked to two connectors, small arcs attached on point separated by 180 degrees. The consumption (Section 3.5.34) and production (Section 3.5.34) arcs are linked to the extremities of those connectors.

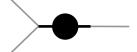


Figure 3.49: The Process Description glyph for association.

The example in Figure 3.50 illustrates the association of cyclin and CDC2 kinase into the Maturation Promoting Factor.

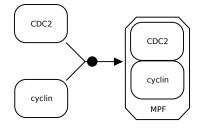


Figure 3.50: Association of cyclin and CDC2 kinase into the Maturation Promoting Factor.

Figure 3.51 gives an example illustrating the association of a pentameric macromolecule (a nicotinic acetylcholine receptor) with a simple chemical (the local anesthetic chlorpromazin) in an unnamed complex.

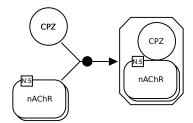


Figure 3.51: The association of a pentameric macromolecule with a simple chemical in an unnamed complex.

Glyph: Dissociation

SBO Term: SBO:0000180! dissociation.

Node: A *dissociation* between several entities is represented by two concentric circles. A simple empty disc could be, in some cases, confused with the *catalysis* (section Section 3.5.35). Moreover, the existence of two circles reminds the dissociation, by contrast with the filled disc of the association (Section 3.5.25).

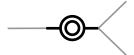


Figure 3.52: *The Process Description glyph for* dissociation.

The example in Figure 3.53 illustrates the dissociation of the small and large ribosomal subunits from a messenger RNA.

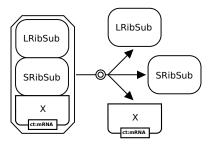


Figure 3.53: Dissociation of the small and large ribosomal subunits from a messenger RNA.

1019

1020

1025

1027

1030

1032

1033

1036

1037

1040

1042

Changes from Previous Version

Although the NonStoichiometricProcess was not explicitly defined in the previous version the semantics and glyphs are unchanged.

3.5.26 Compartment

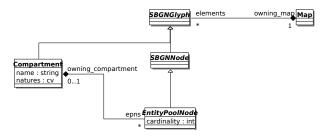


Figure 3.54: The UML definition of the Compartment showing how it containment of EntityPoolNode.

The Compartment is a logical or physical structure that contains entity pool nodes. An EntityPoolNode (see section 3.5.9) can only belong to one compartment. Therefore, the "same" biochemical species located in two different compartments are in fact two different pools.

Generalisation 1024

SBGNGlyph (see section 3.5.3)

Attributes 1026

name: string (R) The name of the compartment.

natures: cv(*) A set of controlled vocabularies²⁵ that describes a characteristic of the compartment. Zero, one or more values may be set, but each one must belong to a different controlled vocabulary.

Associations 1031

epns:EntityPoolNode (*) The EntityPoolNodes contained by this compartment.

3.5.27 **Logical Identity**

name

Logical Key: 1034

• owning_map 1035

Rules and Constraints

- name must not be used by another instance of Container contained by the same instance of Map. 1038
- epns must contain a unique set of EntityPoolNodes. See section 3.7 for the definition of Entity-1039 PoolNode uniqueness.

Notation 1041

Glyph: Compartment

SBO Term: SBO:0000290! physical compartment



 $^{^{25}}$ This reconciles the use of the Unit of Information to represent the nature of an EPN by using to present similar information. mation for the compartment. This is consistent with previous usage, but not with the usage of the UofI for annotation.

Container: A compartment is represented by a surface enclosed in a continuous border or located 1044 between continuous borders. These borders should be noticeably thicker than the borders of 1045 the EPNs. A compartment can take **any** geometry. A compartment must always be entirely 1046 enclosed.

Label: The identification of the compartment is carried by an unbordered box containing a string of 1048 characters. The characters can be distributed on several lines to improve readability, although 1049 this is not mandatory. The label box can be attached anywhere in the container box. Note that 1050 the label can spill-over from the container box.

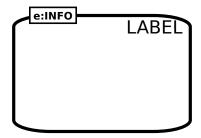


Figure 3.55: The Process Description glyph for compartment.

To allow more aesthetically pleasing and understandable maps, compartments are allowed to 1052 overlap each other visually, but it must be kept in mind that this does not mean the top compartment contains part of the bottom compartment. Figure 3.56 shows two semantically equivalent placement of compartments:

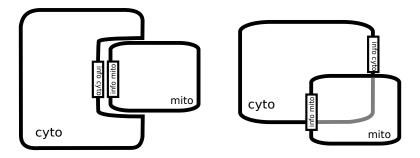


Figure 3.56: Overlapped compartments are permitted, but the overlap does not imply containment.

Overlapped (hidden) part of the compartment should not contain any object which could be 1056 covered by an overlapping compartment. Figure 3.57 illustrates the problem using an incorrect map. 1057

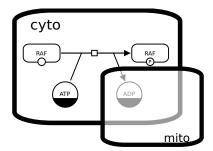


Figure 3.57: Example of an incorrect map. Overlapped compartments must not obscure other objects.

1061

1062

1063

1071

1073

1074

1089

Changes from Previous Version

The use Compartment has a set of natures, which previously were less well specified and handled as notation provided by the *unit of information*. In some cases, where the CVs used are not distinct or if the *unit of information* contains arbitrary text as annotation then maps containing these features will be invalid according to the current specification.

3.5.28 AttributeValue

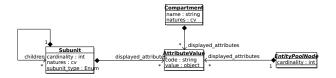


Figure 3.58: The UML definition of the Attribute Value and its usage by other classes.

<u>↑</u> See footnote. The AttributeValue²⁶ is used to present the values of certain attributes held by other SBGN elements. It is typically contained and owned by the class containing the attribute (or its descendants). It contains two values, one is a code to indicate the attribute that is defined and the other is the value itself. The code and the presentation format of the value are defined by the SBGN element that contains the AttributeValue, currently Compartment (see section 3.5.26), EntityPoolNode (see section 3.5.9), and Subunit (see section 3.5.19).

Generalisation 1070

AuxiliaryUnit (see section 3.5.4)

Attributes 1072

code: string **(R)** The code indicating the attribute that is being presented.

value: object (R) The value of the attribute. The format of the value is determined by the class holding the attribute.

Associations 1076

No additional associations.

Rules and Constraints

No additional rules and constraints.

Notation 1080

For historical reasons the Attribute Value is represented graphically by the glyph *Unit of Information*.

Glyph: *Unit of information* When representing biological entities, it is often necessary to convey some abstract information about the entity's function that cannot (or does not need to) be easily related to its structure. The *unit of information* is a decoration that can be used in this situation to add information to a glyph. Some example uses include: characterizing a logical part of an entity such as a functional domain (a binding domain, a catalytic site, a promoter, etc.), or the information encoded in the entity (an exon, an open reading frame, etc.). A *unit of information* can also convey information about the physical environment, or the specific type of biological entity it is decorating. **SBO Term:** Not applicable.

 $^{^{26}}$ A new concept, that modifies the behaviour of the Unit of Information in previous versions with the need to use it to present the nature and cardinality of an EPN. The glyph retains its original name, but the class has been names to reflect it purpose.

1103

1105

1109

1117

Container: A unit of information is represented by a rectangle. The long side of the rectangle should be oriented parallel to the border of the EPN being annotated by the unit of information. The center of the bounding box of a state of information should be located on the mid-line of the border of the *EPN*.

Label: A *unit of information* is identified by a label placed in an unbordered box containing a string of characters. The characters can be distributed on several lines to improve readability, although this is not mandatory. The label box must be attached to the center of the container. The label may spill outside of the container.

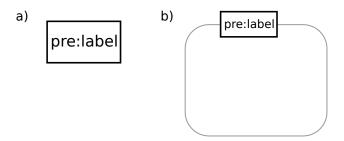


Figure 3.59: The Process Description glyph for unit of information. (a) The glyph. (b) An example of its usage with a macromolecule.

Changes from Previous Version

There was no definition of the AttributeValue in the previous version of this specification. However, the *Unit of Information* did exist although its semantics have been changed. It no longer can hold arbitrary annotation but must display an attribute value and observe the constraints set out by the 1101 definition of the class owning the attribute.

Since the use of the *Unit of Information* has been deprecated, it is recommended that Annotation (see section 3.5.30) and the Annotation glyph is used instead.

3.5.29 **StateVariable**

Many biological entities such as molecules can exist in different states, meaning different physical or 1106 informational configurations. These states can arise for a variety of reasons. For example, macromolecules can be subject to post-synthesis modifications, wherein residues of the macromolecules (amino acids, nucleosides, or glucid residues) are modified through covalent linkage to other chemicals. Other examples of states are alternative conformations as in the closed/open/desensitized 1110 conformations of a transmembrane channel, and the active/inactive forms of an enzyme.

In the Process Description language these states are defined by the StateVariableDefinitions associated with the EntityType, but the specific values of the variables are define by the StateVariable (figure 3.60) associated with the EntityPoolNode. For every StateVariableDefinition associated with an Entity-Type there should be a corresponding StateVariable associated with the instance of EntityPoolNode using that type. This enforces one of the fundamental rules of the language that once a state variable has been displayed for a given entity type, then it must always be displayed.



Figure 3.60: The UML definition of the StateVariable showing its relationship to StatefulEPN and StateVariableDefinition.

1125

1126

1138

1148

Generalisation 1118

AuxiliaryUnit (see section 3.5.4)

See footnote.

Attributes

value: string (R) The value of the state variable. This is optional, but cannot be an empty string, should start with a non-space character and end with a non-space character. It should also not include the '@' character²⁷.

Associations 1124

owning epn:StatefulEPN (1) The stateful EPN that owns the state variable. definition:State Variable Definition (1) The definition of this state variable ²⁸.

Rules and Constraints 1127

No additional rules and constraints. 1128

Notation 1129

Glyph: State variable 1130 **SBO Term:** Not applicable. 1131

Container: A *state variable* is represented by a "stadium" container, that is two hemicercles of same radius joined by parallel segments, as shown in Figure 3.61. The parallel segment axis should be tangent to the border of the glyph of the EPN being modified by the state variable. The center of the bounding box of a state variable should be located on the mid-line of the border of the EPN. In previous versions of this specification the state variable was represented by an 1136 ellipse. This symbols is now **deprecated** in favour of the stadium symbol described above. New 1137 Process Description maps should not use the ellipse symbol.

Label: An unbordered box containing a string indicating the contents of the StateVariable. The style 1139 of labeling of State Variables encouraged by SBGN Process Description Level 1 is to combine 1140 a prefix representing the value of the variable with a suffix representing the variable's name. Prefix and suffix should be separated by the symbol '@', X@Y thus meaning value X AT variable 1142 Y. If name is undefined then only the value should be displayed and the '@' character omitted. 1143 If both the name and value are undefined then the label should be empty (i.e., an empty string). The label of a *state variable* should, if possible, be displayed within the boundary of the glyph. In earlier versions of the SBGN specification it was permitted to separate the name and value into two unlabelled boxes and display the name box outside the state variable glyph. This is now **deprecated** and new Process Description maps should not use this notation.

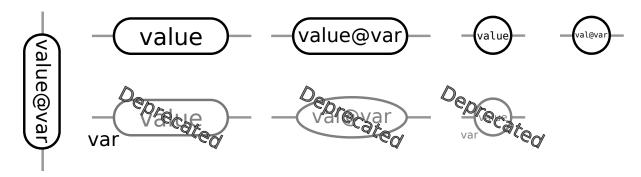


Figure 3.61: Examples of the Process Description glyph for state variable. Included are the older forms of glyph, which are now deprecated.

 $^{^{27}}$ Not specified is previous spec, but seems essential if we are to avoid ambiguity.

²⁸New concept. It aims to ensure that all state variables are defined in all EPNS of the same type.

1160

1161

1165

1166

1168

1171

1173

1175

1179

1180

1182

1185

A state variable does not necessarily have to be Boolean-valued. For example, an ion channel can 1149 possess several conductance states; a receptor can be inactive, active and desensitized; and so on. As another example, a *state variable* "ubiquitin" could also carry numerical values corresponding to the number of ubiquitin molecules present in the tail. However, in all cases, a state variable on an 1152 EPN can only take *one* defined value. Further, an EPN's *state variable* should always be displayed and always set to a value. An "empty" state variable is a state variable that is set to the value "unset", it is not a state variable with no value. Note that the value "unset" is not synonymous to "any value" or "unknown value".

Changes from Previous Version

The StateVariable class was not explicitly defined in previous versions of the specification, however 1158 the state variable was. Some aspects of its notation have been deprecated and these are detailed above (section 3.5.29).

3.5.30 **Annotation**

In SBGN Process Description Level 1 there are cases where the language does not capture everything 1162 the author wishes to convey. This may be additional experimental detail or descriptions of mechanisms that cannot be described full by the Process Description language. In this case the language provides the Annotation. This contains text and is associated with a particular glyph in a map. Importantly, it is purely "decoration" and does alter the meaning the map.

Generalisation 1167

• AuxiliaryUnit (see section 3.5.4)

Attributes 1169

annotation_text: string (R) The text of the annotation. The text is mandatory and cannot be empty or just spaces.

Associations

annotated_glyph:SBGNGlyph (1) The instance of SBGNGlyph that is being annotated²⁹.

Rules and Constraints 1174

No additional rules and constraints.

Notation 1176

Glyph: Annotation 1177

SBO Term: SBO:NEW

Container: An *annotation* is represented by a rectangular container with a folded corner, as illustrated in Figure 3.62. This container is linked to the annotated element via a callout (see figure 3.63. The callout should overlap with the object it is annotating.

Label: An *annotation* contains information placed in an unbordered box containing a string of characters. The characters can be distributed on several lines to improve readability, although this is not mandatory. The label box must be attached to the center of the container. The label may spill outside of the container.

 $^{^{29}}$ Note that as a result of this association only glyphs and ${f not}$ auxiliary items may be annotated by instances of Annotation

1187

1188

1191



Figure 3.62: *The Process Description glyph for* annotation.

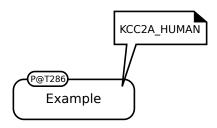


Figure 3.63: Example of annotations adding information to the description of the transphosphorylation of CaMKII. Note that three different types of links are used between annotation nodes and annotated elements. However, it is recommended to use a consistent scheme whithin a

Changes from Previous Version

This is a new language element and an not previous versions of the Process Description language.

3.5.31 CrossReference

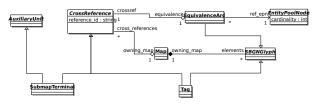


Figure 3.64: The UML definition of the Crossreference showing its subclasses Tag and SubmapTerminal and its association with other elements in the Process Description language.

See footnote.

CrossReference³⁰ handles links or relationships between elements of a map and sub-map. At 1189 present there is only one reference glyph, tag, which can be used in a map refered to by a submap 1190 (Section 3.5.23) or as an auxiliary unit on the *submap*. The *clone marker* can also provide additional reference mechanisms and is discussed below (Section 3.5.38).

Generalisation 1193

None 1194

Attributes 1195

reference id: string (R) a string that identifies the cross-reference. The string cannot start and end in 1196 white space and cannot be empty. 1197

 $^{^{30}\}mathrm{A}$ new concept required for consistency between tags ans submap terminals.

1201

1203

1204

1205

1207

1210

1215

1219

1223

1226

Associations 1198 equivalence: Equivalence Arc (1) The equivalence arc that links this class to the referenced element.

Rules and Constraints

- Two or more instances of CrossReference with the same reference id value are pointing to the same element.
- The above rules applies within a Process Description map's namespace (see section 3.8).

Changes from Previous Version

Not defined in the previous version.

3.5.32 SubmapTerminal

A SubmapTerminal³¹ (figure 3.30) is a named reference that is part of a SubmapNode (see section 3.5.23). It provides the reference that is the link to a tag in the submap that the SubmapNode refers to.

Generalisation 1209

- AuxiliaryUnit (see section 3.5.4)
- CrossReference (see section 3.5.31) 1211

Attributes

No additional attributes. 1213

Associations 1214 No additional associations.

Rules and Constraints 1216

No additional rules and constraints.

Notation 1218

Glyph: Submap Terminal **SBO Term:** Not applicable.

Container: A *tag* is represented by a rectangle fused to an empty arrowhead. The flat edge opposite the arrowhead should be aligned to the edge of the *Submap* glyph and the connecting should connect to the middle of this face (see figure 3.65).

Label: A *tag* is identified by a label placed in an unbordered box containing a string of characters. The characters can be distributed on several lines to improve readability, although this is not mandatory. The label box must be attached to the center of the container. The label may spill outside of the container.

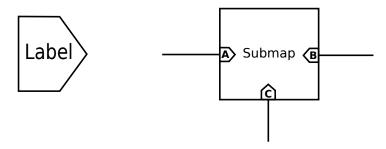


Figure 3.65: The Process Description glyph for Submap Terminal. This shows the basic glyph and its correct usage within a Submap glyph.

 $^{^{31}}$ Not properly defined in previous versions of the spec. The rules described here are consistent with accepted usage.

1261

	Changes from Previous Version	1228
	Clarified that the tag does not link a Compartment, but only instances of EntityPoolNode.	1229
3.5.33	Tag	1230
	A Tag is a named handle, or reference, to another EntityPoolNode. <i>Tags</i> are used to identify those elements in <i>submaps</i> (Section 3.5.23).	1231 1232
	Generalisation	1233
	• SBGNGlyph (see section 3.5.3)	1234
	• CrossReference (see section 3.5.31)	1235
	Attributes	1236
	No additional attributes.	1237
	Associations	1238
	No additional associations.	1239
	Rules and Constraints	1240
	 All values of reference_id must be unique within an instance of Map. 	1241
	Notation	1242
	Glyph: Tag	1243
	SBO Term: Not applicable.	1244
	Container: A tag is represented by a rectangle fused to an empty arrowhead, as illustrated in Fig-	1245
	ure 3.66. The symbol should be linked to one and only one edge (i.e., it should reference only	1246
	one EPN or compartment).	1247
	Label: A <i>tag</i> is identified by a label placed in an unbordered box containing a string of characters.	1248
	The characters can be distributed on several lines to improve readability, although this is not	1249
	mandatory. The label box must be attached to the center of the container. The label may spill	1250
	outside of the container.	1251



Figure 3.66: The Process Description glyph for tag.

Changes from Previous Version

Clarified that the tag does not link a Compartment, but only instances of EntityPoolNode.

3.5.34 **FluxArc** 1254

The FluxArc permits a quantity of entities to flow through the arc and in doing so connects a stoichio- 1255 metric process (StoichiometricProcess (see section 3.5.25)) and an EPN (EntityPoolNode (see section 1256 3.5.9)). The FluxArc has a stoichiometry which is used to indicate the stoichiometry of a process. It is 1257 required to eliminate ambiguity when the exact composition, or the number of copies, of the inputs or outputs to a reaction are ambiguous from the map (see figure 3.71 for an example). The FluxArc 1259 has three forms determined by its flux_type:

consumption Links an entity pool to a process that will consumed it (an input to the process).

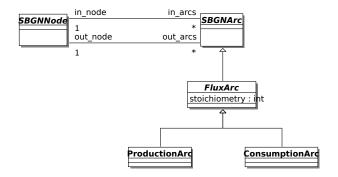


Figure 3.67: The UML definition of the FluxArc and its subclasses.

production Links an entity pool to a process that will be produced by a process (an output from the 1262 process). It may also links entities on the "right-hand-side" of a reversible process (see section 3.5.25 for more details about the reversible process). reversible Links a process to an entity pool that is the "left-hand-side" of a process (see section 1265 3.5.25). Generalisation 1267 • SBGNArc (see section 3.5.6) **Attributes** 1269 stoichiometry: int (R) The stoichiometry of this FluxArc (see section 3.5.34). This must be a non-zero positive integer. flux_typeenumR The type of the flux arc. One of the following: consumption, 1271 production and reversible. 1272 No additional attributes. 1273 **Associations** 1274 No additional associations. 1275 **Rules and Constraints** 1276 • if the stoichiometry is not displayed then it is assumes to be 1³² • If the stoichiometry > 1 then the stoichiometry must be displayed.³³ • if flux_type = 'consumption' or 'reversible' then: 1279 - The in_node must be an instance of EntityPoolNode (see section 3.5.9). 1280 - The out node must be an instance of Stoichiometric Process (see section 3.5.25). • if flux_type = 'production' then: 1282 - The in node must be an instance of Stoichiometric Process (see section 3.5.25). 1283 - The out_node must be an instance of EntityPoolNode (see section 3.5.9). 1284 **Notation** 1285

The FluxArc is represented by three glyphs depending on its flux_type:

consumption consumption arc.

production production arc.

See footnote.

See footnote.

See footnote.

reversible reversible arc^{34} .

These are defined below. In addition the stoichiometry is displayed by the *stoichiometry label*. Its appearance and layout in relation to the flux arc is also described here.

Glyph: Consumption

SBO Term: SBO:0000394! consumption.

End point: No particular symbol is used to represent a consumption.

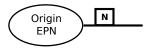


Figure 3.68: *The Process Description glyph for* consumption.

Glyph: Production

SBO:0000393! production.

End point: The target extremity of a production carries a filled arrowhead.

1295



Figure 3.69: The Process Description glyph for production.

Glyph: Reversible

SBO:0000393! production.

End point: The origin extremity of a reversible carries a filled arrowhead.



Figure 3.70: The Process Description glyph for reversible.

Glyph: *Stoichiometry Label* The stoichiometry label is part of the *consumption arc* and *production*arc glyphs see below (sections 3.5.34 and 3.5.34). However, as their use is common to all subclasses of FluxArc their presentation is described here.

³²The previous version said that the stoichiometry was unspecified, but this was before the stoichiometric process was specified. In addition since cardinality is defined explicitly the stoichiometry can often be implied.

³³See above comment.

³⁴The reversible arc is introduced here to fix syntax problems in previous spec. The change is visually neutral on previous Process Description maps.

1306

1307

1309

1310

1311

1312

1314

1315

1319

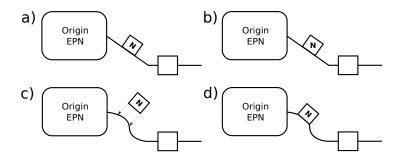


Figure 3.71: Examples of stoichiometry label layout. In figure (a) the label is aligned with the stoichiometry box, while in (b) the label is aligned with the orientation of the map: these are both simple cases where the arc is a straight line. In cases where the arc is curved, the corners at the base of the label are anchored to point on the arc (c) and the label is drawn over the arc (d). Note that in (d) the covered part of the arc is shown for clarity, but normally the box is opaque and so the arc is not visible.

The label is a node that must be drawn above the flux arc. This node is attached to the arc where it intersects the arc with its bottom corners (see figure 3.71.)³⁵.

See footnote.

SBO Term: None

Container: A rectangle with a draw edge.

Label: A number that should remain within the container and be of a normal font, i.e., not bold or 1308 italic.

Changes from Previous Version

See footnote.

The reversible arc has been added to ensure the correct syntax is observed for a reversible process³⁶ — using the *production arc* for this was syntactically forbidden in the previous version of the specification. There is no change to semantics from previous version, but layout rules for placement of stoichiometry label have been clarified.

3.5.35 **ModulationArc**

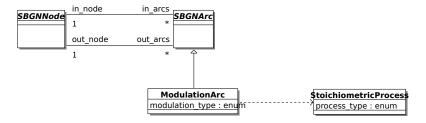


Figure 3.72: The UML definition of the ModulationArc. The class interacts with subclasses of StoichiometricProcess.

The ModulationArc (figure 3.72) affects the flux of a process represented by the target process. Such a modulation can affect the process positively or negatively, or even both ways depending on the 1317 conditions, for instance the concentration of the intervening participants. The permitted values for 1318 process type are described in the following table:

 $^{^{35}}$ These layout rules were agreed at COMBINE 2010, but have not been voted on.

 $^{^{36}}$ New glyph. We need it to ensure the correct syntax as we have a directed graph and we need to prevent the production arc pointing towards the process. It looks the same as the previous situation.

1324

1326

1328

1329

1330

1331

1332

1334

1335

1336

1337

1340

1344

modulation	A general modulation where the exact nature of the modulation is not specified or not known. Modulation can be used when one does not know the precise direction of the effect.
stimulation	A stimulation affects positively the flux of a process represented by the target process. This stimulation can be for instance a catalysis or a positive allosteric regulation. Note that <i>catalysis</i> exists independently in SBGN, see Section 3.5.35.
catalysis	A particular case of stimulation, where the effector affects positively the flux of a process represented by the target process. The positive effect on the process is due to the lowering of the activation energy of a reaction.
inhibition	An inhibition negatively affects the flux of a process represented by the target process. This inhibition can be for instance a competitive inhibition or an allosteric inhibition.
necessary_stim	A necessary stimulation, is one that is necessary for a process to take place. A process modulated by a necessary stimulation can only occur when this necessary stimulation is active.

As discussed in Chapter 2, it is implied, but not defined explicitly that the process has a rate at 1321 which it converts its LHS EPNs to its RHS EPNs (and vice-versa in the case of a reversible process). This concept is important in understanding how the Process Description language describes process modulation³⁷.

See footnote.

- 1. A process with no modulations has an underlying "basal rate" which describes the rate at which it converts inputs to outputs.
- 2. A modulation changes the basal rate in an unspecified fashion.
- 3. A *stimulation* is a modulation that increases the basal rate.
- 4. An *inhibition* is a modulation that decreases the basal rate.
- 5. The above types of modulation, when assigned to the same process, are combined and have a multiplicative effect on the basal rate of the process.
- 6. Modulators that do not interact with each other in the above manner, should be drawn as modulating different process nodes. Their effect is therefore additive.

Generalisation

• EntityPoolNode (see section 3.5.9)

Attributes

No additional attributes.

Associations

states:StateVariable (*) The state variables associated with this EPN.

Rules and Constraints

• At most one necessary stimulation can be assigned to a process node. Two necessary stimulations would imply an implicit AND or OR operator. For clarity only one necessary stimulation can be assigned to a process, and such combinations must be explicitly expressed using logical 1943 operators³⁸.

See footnote.

³⁷This is taken from the grammar section of the previous specification.

³⁸Taken from the grammar section of the previous specification.

1352

1353

1354

 At most one catalysis can be assigned to a process. Modulation by a catalysis arc implies that the exact biochemical mechanism underlying the process is known. In this context two catal*ysis* cannot be assigned to the same process node as they represent independent reactions. Other EPNs can be assigned to the same process as a catalysis, such as modulators, stimulators, and inhibitors, and will have a multiplicative modulation on the reaction rate defined by the catalysis³⁹.

See footnote.

Notation 1351

The ModulationArc is represented by a number of glyphs depending on its modulation_type. The table below defines what glyph is used for each type.

Type	Glyph
modulation	Modulation
stimulation	Stimulation
inhibition	Inhibition
necessary_stim	Necessary Stimulation

Glyph: Modulation 1355 SBO Term: SBO:0000168! control. 1356 **End point:** The target extremity of a *modulation* carries an empty diamond.

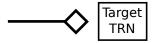


Figure 3.73: The Process Description glyph for modulation.

Figure 3.74 represents the effect of nicotine on the process between closed and open states of a 1358 nicotinic acetylcholine receptor. High concentrations of nicotine open the receptor while low concentrations can desensitize it without opening. 1360

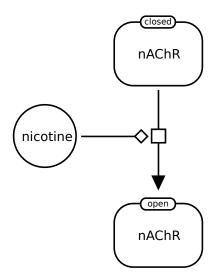


Figure 3.74: Modulation of nicotinic receptor opening by nicotine.

 $^{^{\}rm 39}{\rm Taken}$ from the grammar section of the previous specification.

Glyph: Stimulation

SBO Term: SBO:0000170! stimulation.

End point: The target extremity of a stimulation carries an empty arrowhead.

1361

1362



Figure 3.75: *The Process Description glyph for* stimulation.

Glyph: Catalysis

SBO:0000172! catalysis.

Node: The target extremity of a catalysis carries an empty circle.

1365

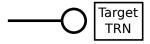


Figure 3.76: *The Process Description glyph for* catalysis.

Glyph: Inhibition

SBO Term: SBO:0000169! inhibition.

Node: The target extremity of an inhibition carries a bar perpendicular to the arc.

1368

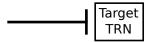


Figure 3.77: *The Process Description glyph for* inhibition.

Glyph: Necessary stimulation

SBO Term: SBO:0000171! necessary stimulation.

Node: The target extremity of a necessary stimulation carries an open arrow (to remind that it is a stimulation) coming after a larger vertical bar.



 $\textbf{Figure 3.78:} \ \textit{The Process Description glyph for Necessary Stimulation}.$

Examples The example in Figure 3.79 below describes the transcription of a gene X, that is the creation of a messenger RNA X triggered by the gene X. The creation of the protein X is then triggered by the mRNA X. (Note that the same example could be represented using the gene as reactant and product, although it is semantically different.)

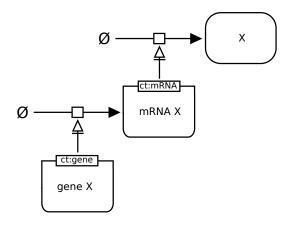


Figure 3.79: The creation of a messenger RNA X triggered by the gene X.

The example in Figure 3.80 below describes the transport of calcium ions out of the endoplasmic reticulum. Without IP3 receptor, there is not calcium flux, therefore, one cannot use a stimulation. The Necessary Stimulation instead represents this absolute stimulation.

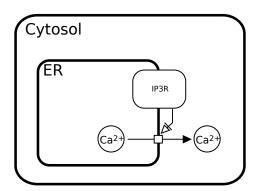


Figure 3.80: The transport of calcium ions out of the endoplasmic reticulum into the cytosol. Note that IP3R crosses both compartment boundaries. This is allowed, but the Macromolecule should only belong to one of the compartments see section D.1 for more discussion of this issue.

Changes from Previous Version

The definition of ModulationArc did not exist in the previous version but there has been no changes to the glyphs and glyph semantics in this version.

3.5.36 LogicArc

The LogicArc (figure 3.81) takes a quantity from either a LogicalOperator (see section 3.5.24) or an 1385 EntityPoolNode (see section 3.5.9) and converts it into a Boolean output, which serves as an input for a LogicalOperator (see section 3.5.24). How this is done is not defined, but one could imagine that when a threshold value of the quantity is exceeded the output is True.

Generalisation

• SBGNArc (see section 3.5.6)

Attributes

No additional attributes.

Associations

No additional associations.

1380

1381

1383

1384

1388

1389

1390

1391

1392

1393

1394

1401

1402

1403

1404

1405

1407

1408

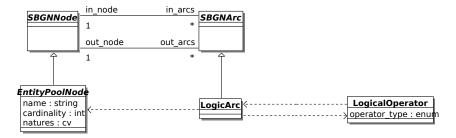


Figure 3.81: The UML definition of the LogicArc and its context.

Rules and Constraints
 The in_node must be an instance of EntityPoolNode or LogicalOperator.

• The out_node must be an instance of LogicalOperator.

Notation 1398

Glyph: *Logic arc Logic arc* is used to represent the fact that an entity influences the outcome of a logic operator.

SBO Term: SBO:0000398! logical relationship.

End point: No particular symbol is used to represent a logic arc.



Figure 3.82: *The Process Description glyph for* logic arc.

Changes from Previous Version

No changes from the previous version.

3.5.37 EquivalenceArc

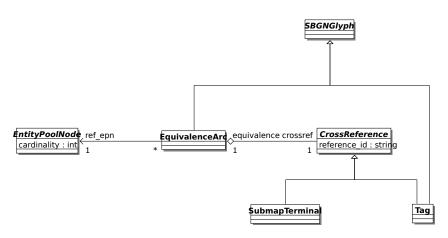


Figure 3.83: The UML definition of the EquivalenceArc and its context.

EquivalenceArc (figure 3.83) is the arc used to link a cross-reference to an EPN in another Process Description map (represented by CrossReference (see section 3.5.31)) with an EPN (EntityPoolNode (see section 3.5.9)) in this map.

1435

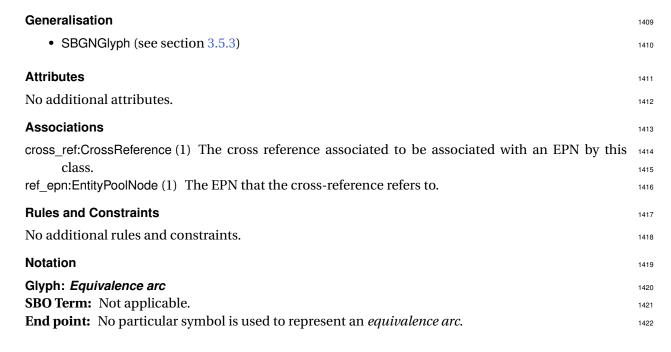




Figure 3.84: *The Process Description glyph for* Equivalence arc.

Changes from Previous Version

The relationship of EquivalenceArc to the SubmapTerminal (see section 3.5.32) was unclear in previous 1424 versions of the specification and has been clarified here.

3.5.38 CloneMarker 1426

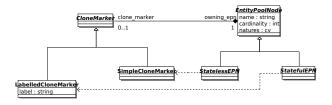


Figure 3.85: The UML definition of the StateVariable showing its relationship to StatefulEPN, Complex and Subunit.

If an EntityPoolNode (see section 3.5.9) is duplicated on a map, it is necessary to indicate this fact 1427 by the CloneMarker auxiliary unit (figure 3.85). The purpose of this marker is to provide the reader 1428 with a visual indication that this node has been cloned, and that at least one other occurrence of the 1429 EntityPoolNode can be found in the map (or in a submap; see Section 3.5.23). The clone marker takes 1430 two forms, simple and labeled, depending on whether the node being cloned can carry state variables 1431 (i.e., whether it is a stateful EPN). Note that an EntityPoolNode belongs to a single compartment. If 1432 two classes named "X" are located in two different compartments, such as ATP in cytosol and ATP in mitochondrial lumen, they represent different Entity Pools, and therefore do not need to be marked as cloned.

	Generalisation	1436
	• AuxiliaryUnit (see section 3.5.4)	1437
	Attributes	1438
	No additional attributes.	1439
	Associations	1440
	owning_epn:EntityPoolNode (1) The EPN that holds this clone marker.	1441
	Rules and Constraints	1442
	No additional rules and constraints.	1443
	Changes from Previous Version	1444
	Not defined in previous version.	1445
3.5.39	SimpleCloneMarker	1446
	The SimpleCloneMarker (figure 3.85) is the unlabelled subclass CloneMarker. All duplicated instances of StatelessEPN must contain an instance of this class.	1447 1448
	Generalisation	1449
	• CloneMarker (see section 3.5.38)	1450
	Attributes	1451
	No additional attributes.	1452
	Associations	1453
	No additional associations.	1454
	Rules and Constraints	1455
	• Only subclasses of StatefulEPN (see section 3.5.15) can contain labelled clone markers.	1456
	Notation	1457
	Simple clone marker	1458
	SBO Term: Not applicable.	1459
	Container: The simple (unlabeled) <i>clone marker</i> is a portion of the surface of an <i>EPN</i> that has been	1460
	modified visually through the use of a different shade, texture, or color. Figure 3.86 illustrates	1461
	this. The <i>clone marker</i> occupies the lower part of the <i>EPN</i> . The filled area must be smaller than	1462
	the unfilled one.	1463
	Label: Not applicable.	1464

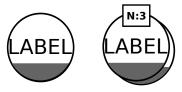


Figure 3.86: *The Process Description glyph for* simple clone marker *applied to a* simple chemical *and a* multimer *of* simple chemicals.

Figure 3.87 contains an example in which we illustrate the use of *clone markers* to clone the species ATP and ADP participating in different reactions. This example also demonstrates the chief drawbacks of using clones: it leads to a kind of dissociation of the overall network and multiplies 1467

1472

1473

1476

1479

1480

1481

1482

1483

1485

1486

1488

1489

1491

1492

the number of nodes required, requiring more work on the part of the reader to interpret the result. 1468 Sometimes these disadvantages are offset in larger maps by a reduction in the overall number of line crossings, but not always. In general, we advise that cloning should be used sparingly.

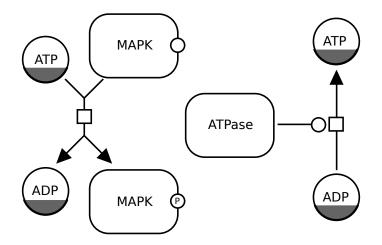


Figure 3.87: An example of using cloning, here for the species ATP and ADP.

Changes from Previous Version

No change from previous version.

3.5.40 LabelledClonerMarker

Unlike the classrefSimpleCloneMarker, the LabeledCloneMarker (figure 3.85) includes (unsurprisingly, given its name) an identifying label that can be used to identify equivalent clones elsewhere in the 1475 map. This is particularly useful for subclasses of StatefulEPN (see section 3.5.15), because these can have a large number of state variables displayed and therefore may be difficult to visually identify as being identical.

Generalisation

• CloneMarker (see section 3.5.38)

Attributes

label: string (R) The label that identified the clone. This label must start and end with an alphanumeric character, and cannot contain white space.

Associations 1484

No additional associations.

Rules and Constraints

- At least two or more instanced of a LabelledCloneMarker with the same label must exist in this same in a given Map (see section 3.5.2).
- Only subclasses of Stateful EPN (see section 3.5.15) can contain labelled clone markers.

Notation 1490

Labeled clone marker

SBO Term: Not applicable.

Container: The labeled *clone marker* is a portion of the surface of an *EPN* that has been modified 1493 visually through the use of a different shade, texture, or color. The clone marker occupies the 1494

1500

1501

1503

1504

1505

1506

1509

1510

1513

1514

1517

1520

1521

1522

1523

1526

1528

lower part of the EPN glyph. The filled area must be smaller than the unfilled one, but the be large enough to have a height larger than the clone marker's label (cf below).

Label: A clone marker is identified by a label placed in an unbordered box containing a string of 1497 characters. The characters can be distributed on several lines to improve readability, although 1498 this is not mandatory. The label box must be attached to the center of the container. The label may spill outside of the container (the portion of the surface of the EPN that has been modified visually). The font color of the label and the color of the clone marker should contrast with one another. The label on a *labeled clone marker* is mandatory.



Figure 3.88: The Process Description glyph for labeled clone marker applied to a macromolecule, a nucleic acid feature and a multimer of macromolecules.

Changes from Previous Version

No changes from previous version.

3.6 Controlled vocabularies

Some classes in the SBGN Process Description language can contain particular kinds of textual annotation conveying information relevant to the class. Examples are the natures of an EntityPoolNode (see section 3.5.9) or Compartment (see section 3.5.26) or the value of the StateVariable (see section 3.5.29). The values held by these attributes can be taken from controlled vocabularies defined below. When displayed in some cases is mandatory to prefix a code indicating the type of controlled vocabulary used. This is in order to make it clear what the information is that the value referes to: for example 1511 'mt' indicates that a value 'rna' is describing the material RNA.

In the rest of this section, we describe the controlled vocabularies (CVs) used in SBGN Process Description Level 1. They cover the following categories of information: an EPN's material type, an EPN's conceptual type, covalent modifications on macromolecules and the physical characteristics. These controlled vocabularies are *closed* in the sense that only the values defined for each CV can 1516 be used in a valid Process Description map and also closed because only the controlled vocabularies defined in this specification can be used in a valid Process Description map ⁴⁰. We understand that this is of necessity restrictive, but in closing these definitions it means we can be clear about the meaning of all CV terms used in the specification. Updates to the CV terms and the CVs used are welcome and we encourage any changes or additions to be submitted as a tracker item at the address given on the front page of this specification.

3.6.1 **Entity pool node material types**

The material type of an EPN indicates its chemical structure. A list of common material types is 1524 shown in Table 3.2, but others are possible. The values are to be taken from the Systems Biology Ontology (http://www.ebi.ac.uk/sbo/), specifically from the branch having identifier SBO:0000240 (material entity under physical entity representation). The labels are defined by SBGN Process Description Level 1.



 $^{^{}m 40}$ The previous version was a bit vaque but implied that the CVs were open. This has been changed after discussion on the list.

1537

1538

1544

1547

1551

Name	Label	SBO term
Non-macromolecular ion	mt:ion	SBO:0000327
Non-macromolecular radical	mt:rad	SBO:0000328
Ribonucleic acid	mt:rna	SBO:0000250
Deoxribonucleic acid	mt:dna	SBO:0000251
Protein	mt:prot	SBO:0000297
Polysaccharide	mt:psac	SBO:0000249

Table 3.2: A sample of values from the material types controlled vocabulary (Section 3.6.1).

The material types are in contrast to the conceptual types (see below). The distinction is that 1529 material types are about physical composition, while conceptual types are about roles. For example, 1530 a strand of RNA is a physical artefact, but its use as messenger RNA is a role.

3.6.2 Entity pool node conceptual types

An EPN's conceptual type indicates its function within the context of a given Process Description. A 1533 list of common conceptual types is shown in Table 3.3, but others are possible. The values are to be taken from the Systems Biology Ontology (http://www.ebi.ac.uk/sbo/), specifically from the branch having identifier SB0:0000241 (functional entity under physical entity representation). The 1536 labels are defined by SBGN Process Description Level 1.

Name	Label	SBO term
Gene	ct:gene	SBO:0000243
Transcription start site	ct:tss	SBO:0000329
Gene coding region	ct:coding	SBO:0000335
Gene regulatory region	ct:grr	SBO:0000369
Messenger RNA	ct:mRNA	SBO:0000278

Table 3.3: A sample of values from the conceptual types vocabulary (Section 3.6.2).

Macromolecule covalent modifications

A common reason for the introduction of state variables (Section 3.5.29) on an entity is to allow access to the configuration of possible covalent modification sites on that entity. For instance, a macromolecule may have one or more sites where a phosphate group may be attached; this change in the 1541 site's configuration (i.e., being either phosphorylated or not) may factor into whether, and how, the entity can participate in different processes. Being able to describe such modifications in a consistent 1543 fashion is the motivation for the existence of SBGN's covalent modifications controlled vocabulary.

Table 3.4 lists a number of common types of covalent modifications. The most common values 1545 are defined by the Systems Biology Ontology in the branch having identifier SBO: 0000210 (addition 1546 of a chemical group under interaction \rightarrow process \rightarrow biochemical or transport reaction \rightarrow

biochemical reaction→conversion). The labels shown in Table 3.4 are defined by SBGN Process Description Level 1; for all other kinds of modifications not listed here, the author of a Process Description must create a new label (and should also describe the meaning of the label in a legend or 1550 text accompanying the map).

1562

1565

1566

1568

Name	Label	SBO term
Acetylation	Ac	SBO:0000215
Glycosylation	G	SBO:0000217
Hydroxylation	OH	SBO:0000233
Methylation	Me	SBO:0000214
Myristoylation	Мy	SBO:0000219
Palmytoylation	Pa	SBO:0000218
Phosphorylation	P	SBO:0000216
Prenylation	Pr	SBO:0000221
Protonation	Н	SBO:0000212
Sulfation	S	SBO:0000220
Ubiquitination	Ub	SBO:0000224

Table 3.4: A sample of values from the covalent modifications vocabulary (Section 3.6.3).

3.6.4 Physical characteristics

SBGN Process Description Level 1 defines a special unit of information for describing certain common physical characteristics. Table 3.5 lists the particular values defined by SBGN Process Description Level 1.

Name	Label	SBO term
Temperature	pc:T	SBO:0000147
Voltage	pc:V	SBO:0000259
рН	pc:pH	SBO:0000304

Table 3.5: A sample of values from the physical characteristics vocabulary (Section 3.6.4).

3.7 **Entity Pool Node Identity and Cloning**

All elements in an SBGN Process Description map have an implicit identify defined by the id attribute 1557 in SBGNElement (see section 3.5.1) so in that respect all elements drawn are unique. However, in 1558 some cases it is possible to draw two or elements that define identical information in the map. In 1559 some clases, such as the Compartment this is prohibited, but in others (EntityPoolNode) it is not, but 1560 requires special decoration (the clone marker) to indicate that the information is replicated on the 1561 map. We define this type of identity and "logical identity" and the attributes that so discriminate between elements as the "logical key" 41.

See footnote.

Clearly it is therefore important that we define the logical key for all the elements in the map and 1564 that we do this for the class definitions (section 3.5). The conventions used for this is as follows:

- If no logical key is specified then instance identity applies (see above).
- If the logical key is defined then this should be applied when determining equality between two elements.
- If an element can be replicated and two elements are determined to be logically identical, then instance identity should be used to determine uniqueness.

 $^{^{41}}$ This is recapitulation of the rules as defined in the previous version of the specification and as it is understood in discussions.

1577

• If an element can be replicated if marked as a clone, then the above rule applies. If the element is not marked as cloned, then the only the logical key can be used when determining uniqueness.

Figure 3.89 illustrates how the above rules are applied in practise in the Process Description map. 1574 They also affect our understanding when reading a map. For example since all processes are unique, two Process glyphs connected to the same input and output indicates that the same "reaction" can be carried out by two distinct processes, with potentially different mechanisms.

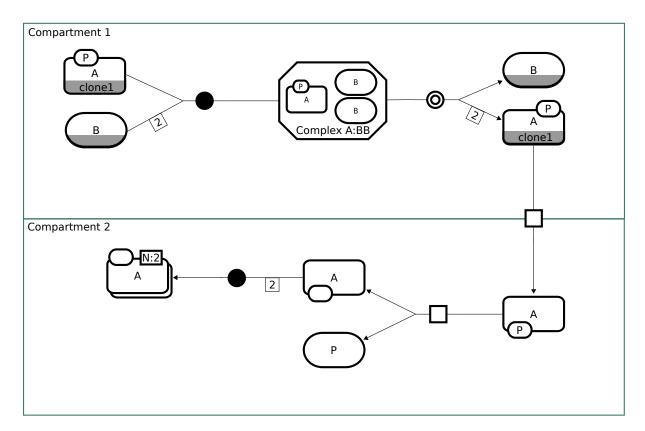


Figure 3.89: This fictitious example illustrates the correct cloning semantics in the Process Description language. The diagram shows the association of the phosphorylated macromolecule A (A[P]) and simple chemical B into a complex. Two copies of B are consumed and these are shown as separate subunits in the complex (note that no clone marker is used). The complex then dissociates into the same components that formed it which requires an appropriate clone marker to be applied. A[P] is translocated into compartment 2 where A[P] is not cloned as it does not have a duplicate. The unphosphorylated from of A dimerised, but again this does not require a clone marker as these are regarded as separate entity pools.

3.8 Map and Submap Linking

The submap is a visual device that allows the detail of an Process Description map to be exported 1579 into another Process Description map and replaced by a SubmapNode (see section 3.5.23), which acts as a place-holder. In the case of an SBGN description that is made available through a software tool, the content of a SubmapNode may be available to the tool. A user could then ask the tool to expand it, for instance by clicking on the *submap node* glyph. The tool might then expand and show its submap within the same map, or it might display it as a separate map. In the case of an SBGN description made available in a book or a website, the content of the *submap node* may be available on another page, possibly accessible via a hyperlink.⁴²

1578

1584

See footnote.

⁴²Currently we can't name a map so linking between supermaps and submaps is problematic.

1590

1591

1592

1593

1598

As you can see the rules relating to the submap concept range wider than those applying to the SubmapNode so we will detail these rules here. The concepts and nomenclature here can get confusing, so we will use the following nomenclature in the rest of this section⁴³: See footnote.

supermap The map that owns the submap and so contains the submap glyph referring to it.

submap The map that contains the detail summarised by the submap glyph in the super-map.

reference id the value displayed by the tag and submap terminal. These must be the same to indicate equivalence between map and submap.

Figure 3.90 represents a submap node that transforms glucose into fructose-6-phosphate. The 1594 submap node carries five terminals, four linked to EPNs and one linked to a compartment. The latter is particularly important in the case of EPNs present only in a *compartment* enclosed in a *submap* node, and that are not linked to terminals themselves. Note that the terminals do not define a "direction", such as input or output. The flux of the reactions is determined by the context.

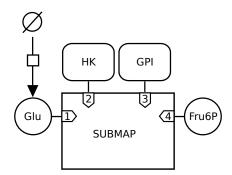


Figure 3.90: Example of a submap with contents elided.

The map in Figure 3.91 represents an unfolded version of a *submap*. Here, anything outside the submap node has disappeared, and the internal tags are not linked to the corresponding external submap terminals.44

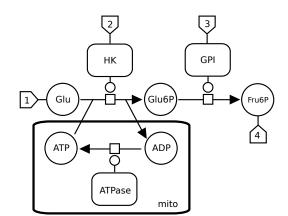


Figure 3.91: Example of an unfolded submap. The unfolded submap corresponds to the folded submap of Figure 3.90.



⁴³This nomenclature has not been used previously. The previous nomenclature was a bit confusing and didn't capture the hierarchy of maps well.

⁴⁴This example in the previous version fo the spec included compartment tagging. This has been removed, but the example may need refinement as a result.

1606

1607

1608

1613

1616

1617

1619

1622

1623

1624

1625

1626

1630

1640

3.8.1 Namespace and the relationship between maps

See footnote.

The following rules apply to the use of the SubmapNode and submaps in the Process Description language 45.

- Submaps can contain SubmapNode instances so submaps can contain submaps there is no limit on the depth of this.
- Submaps share the same namespace as their supermap. Therefore submaps of submaps share the top-most map's namespace.
- Compartment's can be replicated across submaps and they all refer to the same compartment if they share the same name.
- Subclasses of EntityPoolNode replicated in both map and its immediate submap must be associated with a subclass of CrossReference (SubmapTerminal for the supermap and Tag for the submap).
- Subclasses of EntityPoolNode may be replicated between submaps separated by one or more other submaps, i.e. those not affected by the above rule.
- All reference_ids used must be unique within a given map namespace.
- submap folding: all tagged nodes in submap merge with those linked to terminals in supermap. If either node is cloned then all equivalent nodes in merged map must share the appropriate clone marker.
- For a valid mapping between subclasses of EntityPoolNode in a supermap and submap to exist, the EntityPoolNode subclasses must be logically identical and the reference id in their associated CrossReference must be identical, identical.
- A SubmapNode which do not have a corresponding submap is valid.
- Submaps without a corresponding supermap are invalid⁴⁶.
- Subclasses of EntityPoolNode with the same name must share the same EntityType.

See footnote.

3.8.2 Compartment spanning

An EPN cannot belong to more than one compartment. However, an EPN can be drawn over more than one *compartment*. In such cases, the decision on which is the owning *compartment* is deferred to the drawing tool or the author. A complex may contain EPNs which belong to different compartments and in this way a complex can be used to describe entities that span more than one compartment.

This restriction makes it impossible to represent in a semantically correct way a macromolecule 1652 that spans more then one compartment — for example a receptor protein. It is clearly desirable to be able to show a macromolecule in a manner that the biologist expects (i.e. spanning from the outside through the membrane to the inside). Therefore, the author is recommended to draw the macromolecule across compartment boundaries, but the underlying SBGN semantic model will assign it to only one. The assignment to a *compartment* may be decided by the software drawing tool or the author. Note that this has implications for auto-layout algorithms as they will only be able to treat such *entity pool nodes* as contained within a *compartment* and will have no way of knowing a macromolecule spans a compartment.

 $^{^{}m 45}$ These rules are taken from the previous version of the spec, but rules about the depth of submap 'recursion', the uniqueness of reference_id values and cloning between maps are not

 $^{^{46}}$ We need to sort out mapping between super and sub maps to get this working. I propose naming the maps

The current solution is consistent with other Systems Biology representations such as SBML and $_{1641}$ BioPAX. For more information about the problems representing membrane spanning proteins and $_{1642}$ the rationale behind the current solution see Section D.

Chapter 4

Layout Rules for a Process Description

4.1 Introduction 1646

1644

1645

1653

1657

1658

1659

1661

1665

1666

1670

1671

1672

1674

1675

The previous chapters describe the appearance and meaning of SBGN Process Description Level 1 components. Here we describe rules governing the visual appearence and asthetics of the Process Description language. The components of a Process Description have to be placed in a meaningful way – a random distribution with spaghetti-like connections will most likely hide the information encoded in the underlying model, whereas an elegant placement of the objects, giving a congenial appearance of the maps, may reveal new insights. The arrangement of components in a map is called 1652 a *layout*.

SBGN Process Descriptions should be easily recognisable not only by the glyphs used, but also by the general style of the layout. However, the arrangement of the components is a complex art in itself, and there is no simple rule which can be applied to all cases. Therefore this section provides rules for the layout of process description maps, divided into two categories:

- 1. requirements, i. e. rules which **must** be fulfilled by a layout, and
- 2. recommendations, i. e. rules which **should** be followed if possible.

In addition, we provide a list of additional suggestions which may help in producing aesthetically more pleasant layouts, possibly easier to understand.

Those layout rules are independent of the method used to produce the map, and apply to both manually drawn maps as well as maps produced by an automatic layout algorithm. The rules do not deal with interactive aspects (e.g. the effect of zooming). Further information about automatic network layout (graph drawing) can be found, for example, in the books of Di Battista and co-authors [13] and Kaufmann and Wagner [14].

Please note that the color of objects do not carry any meaning in SBGN. Although one can use colors to emphasize part of a map or encode additional information, the meaning of the map should not depend on the colors. Furthermore, objects can have different sizes and size is also meaningless in SBGN. For example, a process node may be larger than a protein node. Also the meaning of a graph should be conserved upon scaling as far as possible.

4.2 Requirements

Requirements are rules which **must** be fulfilled by a layout to produce a valid Process Description 1673 map.

4.2.1 **Node-node overlaps**

Nodes are only allowed to overlap in two cases when they are allowed to contain other nodes — as described in Chapter 3. Otherwise, nodes are not allowed to overlap (Figure 4.1). This includes the

1680

1682

1683

1687

1688

1690

1691

1693

touching of nodes. Touching is not allowed apart from the case where it has a specific meaning, e.g. two macromolecules touching each other within a complex because they form the complex.

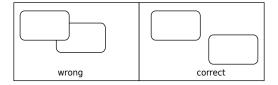


Figure 4.1: Nodes must not overlap.

4.2.2 Node-edge crossing

Edges must be drawn on the top of a the node (Figure 4.2). See also recommendation Section 4.3.1.

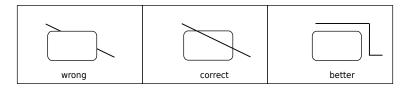


Figure 4.2: *If an edge crosses a node, the edge must be drawn on top of the node.*

4.2.3 Node border-edge overlaps

Edges are not allowed to overlap the border lines of nodes (Figure 4.3).

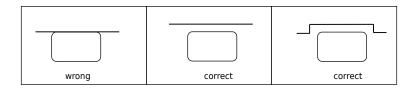


Figure 4.3: *Edges must not overlap node borders.*

4.2.4 Edge-edge overlaps

Edges are not allowed to overlap (Figure 4.4). This includes touching of edges. Furthermore, an edge is neither allowed to cross itself nor to cross a boundary of node more than twice or other edges more than once.

4.2.5 Node orientation

Nodes have to be drawn horizontally or vertically, any other rotation of elements is not allowed (Figure 4.5).

4.2.6 Node-edge connection

1. The arcs linking the square glyph of a *process* to the *consumption* and *production arcs* are attached to the center of opposite sides (Figure 4.6).

1699

1701

1706

1707

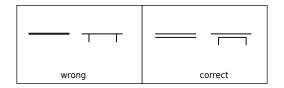


Figure 4.4: Edges must not overlap.

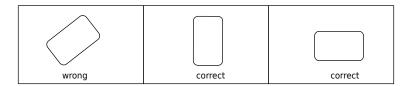


Figure 4.5: The node orientation must be horizontally or vertically.

2. The modulatory arcs are attached to the other two sides, but not necessarily all to the center, 1694 as several modifiers can affect the same process node.

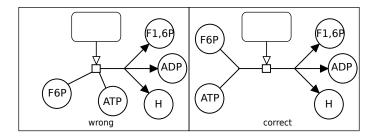


Figure 4.6: Arcs between a process and the consumption and production arcs must be attached to the center of opposite sides, modulatory arcs must be attached to the other two sides.

4.2.7 Node labels 1696

At least a part of the label (unbordered box containing a string of characters) has to be placed inside the node it belongs to. Node labels are not allowed to overlap other nodes or other labels (this includes touching of other nodes or labels).

4.2.8 Edge labels 1700

Edge labels are not allowed to overlap nodes. This includes touching of nodes.

4.2.9 Compartments 1702

If a process has all participants in the same compartment the process node and all edges/arcs should 1703 be drawn in this compartment. If a process has participants in at least two different compartments, the process node has to be either in a compartment where the process has at least one participant or 1705 in the empty space.

4.3 Recommendations

Recommendations are rules which should be followed if possible and generally should improve the clarity of the diagram. 1709

Node-edge crossing 4.3.1

Situations where edges and nodes cross should be avoided. Note that some crossings may be unavoidable, e.g., the crossing between an edge and a compartment border or an edge and a complex 1712 (if the edge connects an element inside the complex with something outside).

4.3.2 Labels 1714

Labels should be horizontal. Node labels should be placed completely inside the node if possible. 1715 Edge labels should be placed close to the edge and avoid overlapping the edge as well as other edge labels.

4.3.3 Avoid edge crossings

The amount of crossings between edges should be minimized.

4.3.4 Branching of association and dissociation

The branching points of association and dissociation nodes should be placed closed to the symbol 1721 of the process, if possible at a distance comparable than, or smaller to, the diameter of the symbol 1722 defining the process (Figure 4.7).

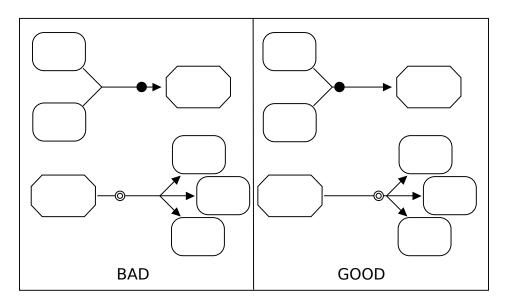


Figure 4.7: *Branching points should be close to association and dissociation symbols.*

4.3.5 Units of information

Units of information should not hide the structure of the corresponding node and should not overlap 1725 other elements (Figure 4.8). 1726

4.4 Additional suggestions

Here is a list of additional layout suggestions which may help improve the asthetics and clarity of 1728 Process Description maps.

 Angle of edge crossings: If edge crossing cannot be avoided then the edges should cross with an angle close to 90 degrees.

1710

1713

1717

1720

1719

1723

1724

1727

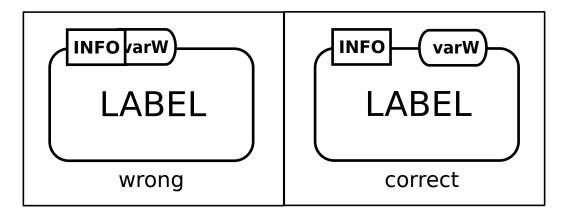


Figure 4.8: *Units of information should not overlap with any other element.*

- Drawing area and width/height ratio: The drawing should be compact and the ratio between 1732 the width and the height of the drawing should be close to 1. • Edge length: Long edges should be avoided. 1734 • Number of edge bends: Edges should be drawn with as few bends as possible. 1735 • Similar and symmetric parts: Similar parts of a map should be drawn in a similar way, and 1736 symmetric parts should be drawn symmetrically. 1737 • Proximity information: Related elements (e.g. nodes connected by a process or all elements 1738 within a compartment) should be drawn close together. Directional information: Subsequent processes (e. g. a sequence of reactions) should be drawn
- Compartments: It can help clarity to use a different background shade or color for each compartment.

in one direction (e.g. from top to bottom or from left to right).

Chapter 5 1744 **Acknowledgments** 1745 Here we acknowledge those people and organisations the assisted in the development of this and 1746 previous releases of the SBGN Process Description language specification. First we specifically acknowledge those who contributed directly to each revision of the specification document, followed 1748 by a comprehensive acknowledgement of contributors that attended workshops and forum meetings or in some other way provided input to the standard. Finally, we acknowledge the bodies that 1750 provided financial support for the development of the standard. 1751 Level 1 Release 1.0 1752 The specification of was written by Nicolas Le Novère, Stuart Moodie, Anatoly Sorokin, Michael 1753 Hucka, Falk Schreiber, Emek Demir, Huaiyu Mi, Yukiko Matsuoka, Katja Wegner and Hiroaki Kitano. In addition, the specification benefited much from the help of (in alphabetical order) Frank 1755 Bergmann, Sarala Dissanayake, Ralph Gauges, Peter Ghazal, Lu Li, and Steven Watterson. 1756 Level 1 Release 1.1 1757 The specification of SBGN PD Level 1.1 was written by Stuart Moodie and Nicolas Le Novère, with 1758 contributions from (in alphabetical order) Frank Bergmann, Sarah Boyd, Emek Demir, Sarala Wimalaratme, Yukiko Matsuoka, Huaiyu Mi, Falk Schreiber, Anatoly Sorokin, Alice Villéger. Level 1 Release 1.2 The specification of SBGN PD Level 1.2 was modified by Stuart Moodie, with contributions from (in 1762 alphabetical order) Sarah Boyd, Nicolas Le Novère, Huaiyu Mi. 1763 Level 1 Release 1.3 1764 The specification of SBGN PD Level 1.3 was modified by Stuart Moodie, with contributions from (in 1765 alphabetical order), Tobias Czauderna, Nicolas Le Novère, Anatoly Sorokin. 1766 Comprehensive list of acknowledgements 1767 Here is a more comprehensive list of people who have been actively involved in SBGN development, 1768

5.1

5.2

5.3

5.4

5.5

if we forgot someone, and would be grateful if you could notify us of any omission.

either by their help designing the languages, their comments on the specification, help with development infrastructure or any other useful input. We intend this list to be complete. We are very sorry

1780

1781

1793

1794

Mirit Aladjemm, Frank Bergmann, Sarah Boyd, Laurence Calzone, Melanie Courtot, Emek Demir, 1772 Ugur Dogrusoz, Tom Freeman, Akira Funahashi, Ralph Gauges, Peter Ghazal, Samik Ghosh, Igor Goryanin, Michael Hucka, Akiya Jouraku, Hideya Kawaji, Douglas Kell, Sohyoung Kim, Hiroaki Kitano, Kurt Kohn, Fedor Kolpakov, Nicolas Le Novère, Lu Li, Augustin Luna, Yukiko Matsuoka, Huaiyu 1775 Mi, Stuart Moodie, Sven Sahle, Chris Sander, Herbert Sauro, Esther Schmidt, Falk Schreiber, Jacky Snoep, Anatoly Sorokin, Jessica Stephens, Linda Taddeo, Steven Watterson, Alice Villéger, Katja Wegner, Sarala Wimalaratne, Guanming Wu.

The authors are also grateful to all the attendees of the SBGN meetings, as well as to the subscribers of the sbgn-discuss@sbgn.org mailing list.

5.6 Financial Support

The development of SBGN was mainly supported by a grant from the Japanese New Energy and Industrial Technology Development Organization (NEDO, http://www.nedo.go.jp/). The Okinawa Institute of Science and Technology (OIST, http://www.oist.jp/), the AIST Computational Biology Research Center (AIST CBRC, http://www.cbrc.jp/index.eng.html) the British Biotechnology and Biological Sciences Research Council (BBSRC, http://www.bbsrc.ac.uk/) through a Japan 1786 Partnering Award, the European Media Laboratory (EML Research gGmbH, http://www.eml-r. org/), and the Beckman Institute at the California Institute of Technology (http://bnmc.caltech. edu) provided additional support for SBGN workshops. Some help was provided by the Japan Science and Technology Agency (JST, http://www.jst.go.jp/) and the Genome Network Project of 1790 the Japanese Ministry of Education, Sports, Culture, Science, and Technology (MEXT, http://www. mext.go.jp/) for the development of the gene regulation network aspect of SBGN, and from the 1792 Engineering and Physical Sciences Research Council (EPSRC, http://www.epsrc.ac.uk) during the redaction of the specification.

Appendix A

Complete examples of Process Description **Maps**

1795

1797

1799

1800

1801

1807

The following maps present complete examples of SBGN Process Descriptions representing Biological processes. They by no mean exhaust the possibilities of SBGN Process Description Level 1.

Figure A.1 presents an example of metabolic pathway, that examplifies the use of the EPNs simple chemical, macromolecule, and clone marker, the PNs process, and the connecting arcs consumption, production and catalysis.

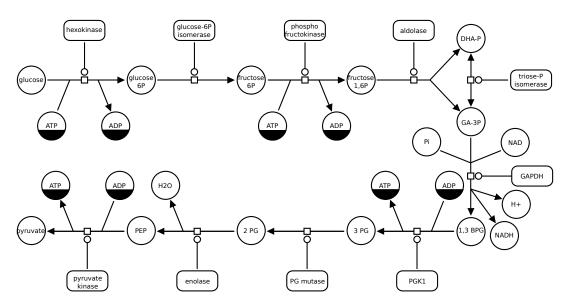


Figure A.1: Glycolysis. This example illustrates how SBGN can be used to describe metabolic pathways.

Figure A.2 presents an example of signalling pathway, that examplifies in addition the use of the EPNs phenotype, and state variable, the containers complex, compartment and submap, the PNs association, and the connecting arcs stimulation. Note the complex IGF and IGF receptor, located on 1805 the boundary of the compartment. This position is only for user convenience. The complex has to 1806 belong to a given compartment in SBGN Process Description Level 1.

Figure A.3 is an expanded version of the submap present on the map present in Figure A.2. It shows the use of tag.

Figure A.4 introduces an SBGN Process Description that spans several compartments. Note that the compartment "synaptic vesicle" is not contained in the compartment "synaptic button" but overlaps it. The simple chemical "ACh" of the "synaptic vesicle" is not the same EPN than the "ACh" of the

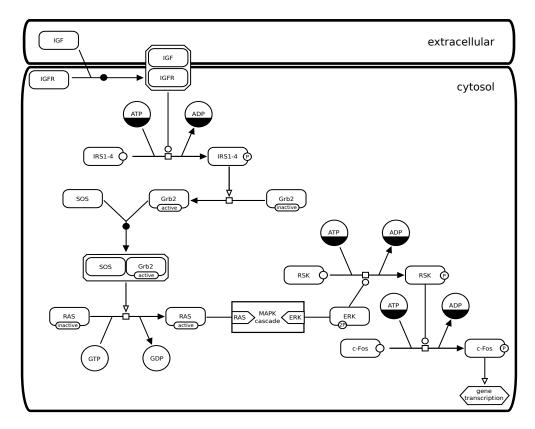


Figure A.2: Insulin-like Growth Factor (IGF) signalling. This example shows the use of compartments and how details can be hidden by using a submap. The submap is shown on Figure A.3.

"synaptic button" and of "synaptic cleft". The situation is similar with the compartments "ER" and 1813 "muscle cytosol". The map exemplifies the use of the *PN omitted* and *dissociation*, and the *connecting* 1814

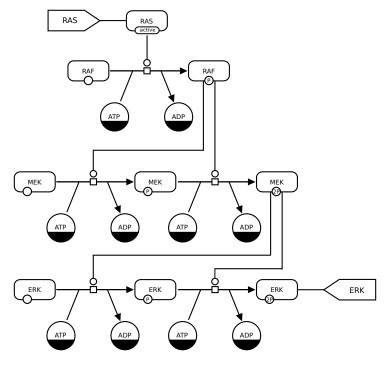


Figure A.3: A submap of the previous map showing the MAPK cascade.

arc necessary stimulation.

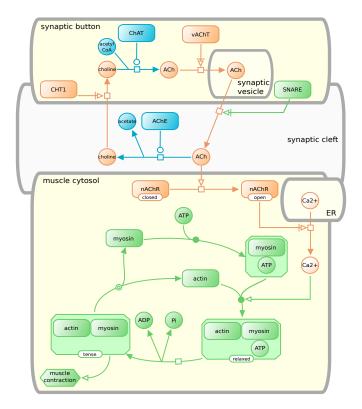


Figure A.4: Neuronal/Muscle signalling. A description of inter-cellular signalling using SBGN.

Figure A.5 introduces the use of SBGN Process Description Level 1 to encode gene regulatory networks. It also show the use of the EPNs Source and the logical operator and.

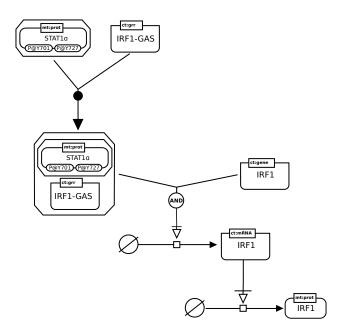
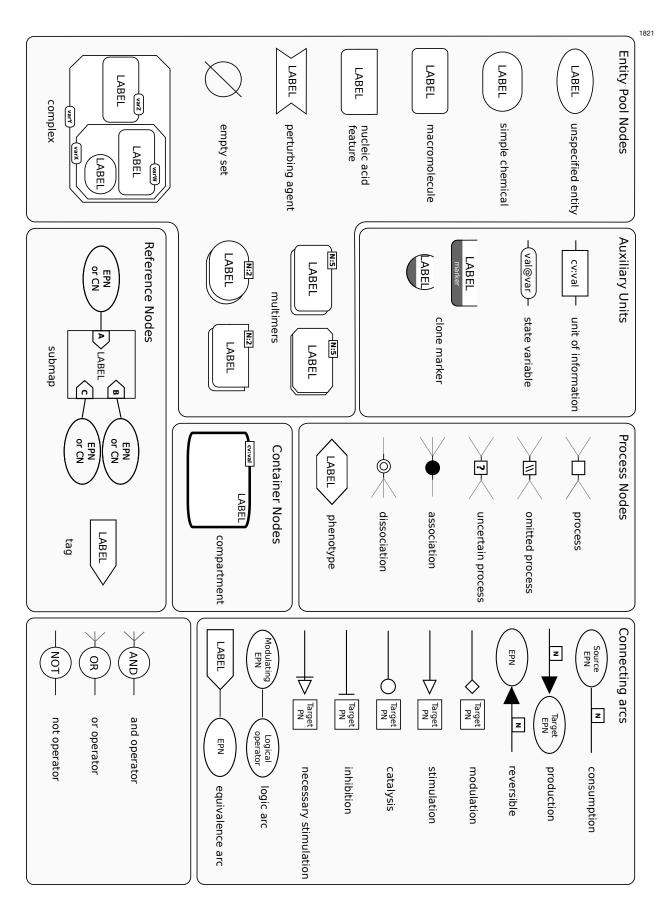


Figure A.5: Activated STAT1 α induction of the IRF1 gene. An example of gene regulation using logical operators.

Appendix B	1818
Reference card	1819
Print the summary of SBGN symbols on the next page for a quick reference.	1820



Appendix C 1822 Validation and consistency rules 1823 This section describes criteria that must be met by the diagram (or SBGN-ML diagram description) 1824 to conform SBGN Process Description Level 1 specification. All criteria has unique and perennial numbering. In the case when modification of the specification makes a rule irrelevant it will be deleted but the number of the rule will not be taken, so there will be gaps in rule numbering. 1828 **C.1 EPN rules** 1829 000001. Elements that do not comply SBGN Process Description Level 1 specification are not allowed 1830 on SBGN Process Description Level 1 map. 1831 000002. All instances of SBGNGlyph (see section 3.5.3) must be unique (see section 3.7). 1832 000003. The set of SBGNNodes linked to this node via a SBGNArc (its adjacent nodes) must be all belong to different entity pools (as defined by EntityPoolNode) and cannot include more than one clone of the same entity pool. 1835

Appendix D

Issues postponed to future levels

D.1 Multicompartment entities

The problem of entities, such as macromolecules, spanning several compartments proved to be a 1839 challenge for the community involved in the development of SBGN Process Description Level 1. It 1840 was thus decided to leave it for a future Level. It turns out there is at the moment no obvious solution 1841 satisfactory for everyone. Three broad classes of solutions have been identified so far:

1836

1837

1838

1846

1850

1854

1855

1860

1861

1864

- One can systematically locate an EPN in a given compartment, for instance a transmembrane receptor in a membrane. However, the reactions of this entity with entities represented by EPN in other compartments, such as extracellular ligands and second messenger systems, will 1845 create artificial transport reactions.
- One can represent the domains of proteins in different compartments by macromolecules, and link all those macromolecules in a *complex* spanning several compartments. However, such 1848 a representation would be very confusing, implying that the domains are actually different 1849 molecules linked through non-covalent bonds.
- On can accept *macromolecules* that span several compartments, and represent domains as 1851 units of information. Those units of information should then be located in given compartments. To make a full use of such a representation, one should then start and end connecting arcs on given units of information, something prohibited by the current specification.

D.2 Logical combination of state variable values

The value of a *state variable* has to be perfectly defined in SBGN Process Description Level 1. If a state variable can take the alternative values 'A', 'B' and 'C', one cannot attribute it values such as 'non-A', 'A or B', 'any' or 'none'. As a consequence some biochemical processes cannot be easily represented because of the very large number of state to enumerate. The decision to forbid such a Boolean logic lies in the necessity of maintaining truth path all over an SBGN map.

D.3 Non-chemical entity nodes

The current specification cannot represent combinations of events and entities. For instance a variable "voltage" cannot be controlled by a difference of concentration between different entities, such as a given ion in both sides of a membrane.

1870

1875

D.4 Generics 1865

SBGN Process Description Level 1 does not provide mechanisms to sub-class EPNs. There is no specific means of specifying that macromolecules or nucleic acid features X1, X2 and X2 are subclasses of X. Therefore, any process that applies to all the subtypes of X has to be triplicated. That situation can 1868 easily generate combinatorial explosions of the number of EPNs or PNs.

D.5 State and transformation of compartments

In SBGN Process Description Level 1 a compartment is a stateless entity. It cannot carry state variables, and cannot be subjected to process modifying a state. As a recult, a compartment cannot be 1872 transformed, moved, split or merged with another. If one want to represent the transformation of a 1873 compartment, one has to create the start and end compartments, and represent the transport of all 1874 the EPNs from one to the other. This is not satisfactory, and should be addressed in the future.

Appendix E

Revision History

1876

1877

1878

1881

E.1 Version 1.0 to Version 1.1

Below are the changes incorporated into Version 1.1 of the SBGN Process Description Level 1 specification. 1879

Description	Tracker ID	
Regarding modulation of reversible processes, changed "should" to "must"		
be represented by two <i>process</i>		
Removed "The connectors and the box move as a rigid entity" in the defini-		
tion of process		
Changed the definition of process node to "represent processes that trans-		
form one or several EPNs into one or several EPNs, identical or different"		
Changed SBO term of compartment From SBO:0000289 (functional com-		
partment) to SBO:0000290 (physical compartment)		
Reoganised classification of glyphs		
Reoganised glyph section to reflect the above changes		
Revised reference card to reflect changes in glyph organisation		
Revised logic operators throughout spec to make sure input and output arcs		
meet before attaching to the glyph - as with processes.		
Added enumerated rules to grammar section. This is probably not com-		
plete, but should help the implementation of semantc validation by soft-		
ware tools. The hope is this will be refined as tools start validating maps.		
Updated UML maps and data dictionary to be consistent with rest of		
changes to spec.		
Definition of cardinality is ambiguous	2840996	
Sink and source are lumped together	2726435	
SBO terms are incorrect or missing.	2841261	
Compartment description is confusing and contradictory.	2841122	
Clone marker fill percentages unhelpful.	2841114	
Use of CV for physical charactetistic not clear.	2841085	
Definition of Cardinality is ambiguous.	2840996	
input to AND on IFN example.	2804326	
more SBO terms for <i>multimers</i>	2803593	
Legend of figure 2.20 is incorrect	2803537	
legend of figure 3.2	2802990	
continued on next page		

continued from previous page	
Description	Tracker ID
Compartment colouring	2745703
Errors in diag a4.	2664912
Change name of trigger glyph.	2664908
Transition should be renamed process.	2664862
Converting arcs tautological.	2664843
Example invalid.	2545870
consumption and production.	2388317
Should require circles to be distinguishable from ellipses	2219388
Figure 2.53	2162619
Reference card: production	2104471
Figure 2.42 is wrong	2104465
Mistake in the multi-cellular example	2395488
Should not prevent processes having identical in and out	2664933
No description of linking to subunit rules.	2545810
Extensively revised the grammar section. The UML diagrams have been	
simplified to show glyph taxonomy, and the data dictionary has been	
pruned to just show glyph identity. The some syntax rules have been moved	
into semantics and the rules reformulated to make them easier to under-	
stand.	
Eliminated duplicate rules in layout section and revised text slightly.	
Phenotype cloning?	2989007
Perturbing agent description	2940021

E.2 Version 1.1 to Version 1.2

Below are the changes incorporated into Version 1.2 of the SBGN Process Description Level 1 specification.

Description	Tracker ID
Perturbing agent description	2940021
Members of complex touching	2849273
PD Reference card error for submap glyph	3029242

E.3 Version 1.2 to Version 1.3

Below are the changes incorporated into Version 1.3 of the SBGN Process Description Level 1 specification.

Description	Tracker ID
Incorrect editor on title page	
Typos in acknowledgements	
Fixed typo in item on catalysis in section on modulation semantics.	
State variables figure 2.6 V1.2	3090543

1882

1883

1885

1887

1886

1889

Bibliography

[1]	Object management Group. OMG Unified Modeling Language (OMG UML), Superstruc-	1892
	ture, V2.1.2. Available via the World Wide Web at http://www.omg.org/spec/UML/2.1.2/	1893
	Superstructure/PDF/, 2010.	1894
[2]	Michael Hucka, Frank T. Bergmann, Stefan Hoops, Sarah M. Keating, Sven Sahle, James C. Schaff,	1895
	Lucian P. Smith, and Darren J. Wilkinson. The Systems Biology Markup Language (SBML): Lan-	1896
	guage Specification for Level 3 Version 1 Core. Available via the World Wide Web at http:	1897
	//sbml.org/Documents/Specifications, November 2007.	1898
[3]	G. Di Battista, P. Eades, R. Tamassia, and I.G. Tollis. <i>Graph Drawing: Algorithms for the Visualization of Graphs</i> . Prentice Hall, New Jersey, 1998.	1899 1900
	tion of druphs. I felicide Hall, New Jersey, 1990.	1900
[4]	M. Kaufmann and D. Wagner. <i>Drawing Graphs: Methods and Models</i> , volume 2025 of <i>Lecture</i>	1901
	Notes in Computer Science Tutorial. Springer, 2001.	1902