Plant and Systems Engineering

Information modelling –

Getting started with EXPRESS–G

Siemens AG

Getting started with EXPRESS–G

1 Introduction

EXPRESS G is a diagrammatic modeling notation for the purpose of an object–oriented information modeling. This notation is based on the standardized EXPRESS-G notation, which is itself described in ISO 10303-11: Industrial Automation Systems – Product Data Representation and Exchange – Part 11: Description Methods: The EXPRESS Language Reference Manual.

2 Modeling and notation

2.1 *Entity*

An **entity** (or **entity type**) is a class (or set) objects. Each object in the class is somehow similar to every other object in that class, in that the object exhibit the same characteristics and/or behaviour.

In the diagrammatic modelling notation, an entity is shown as a rectangular box; the entity name is written inside the box. Examples of entities are: the set of all employees in a company (entity name **employee**); The set of all work projects (entity name **project**). The notation for these examples is shown in figure A.1

Figure A.1 – Entity

2.2 *Attributes*

An entity has attributes, describing the characteristics of this object. Attributes are defined as simple data types , e.g. integer, string, logical, boolean or more complex types as, e.g. other entities.Lines are used to connect the entity with its attributes. The names of the attributes are written above and along the lines. If the name of an attribute is composed of several terms, the underscore sign (_) is used as intermediate character. A solid line shows a mandatory attribute, whereas a dashed line shows an optional attribute.

Figure A.2 presents a person defined by certain characteristics including a first name, a last name, a birth date, a description of their hair and including the possibility to have a nickname.

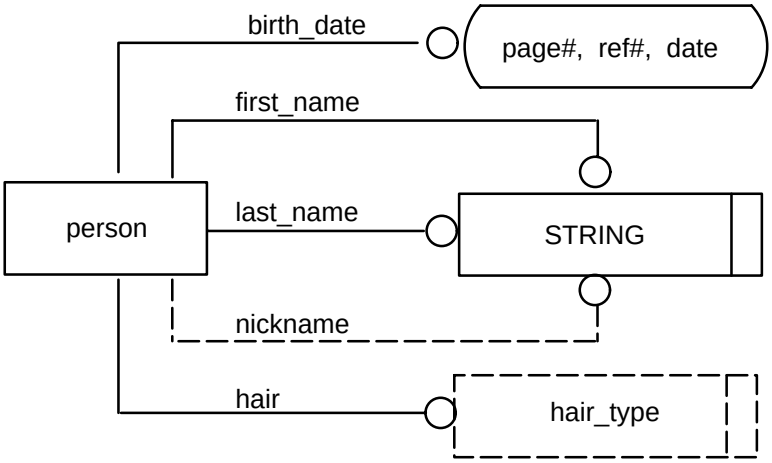

Figure A.2 – Attribute and data type

2.3 *Enumeration*

A data type **enumeration** expresses the existence of a range of values belonging to this attribute. Only one of the enumerated values may be chosen. Figure A.2 depicts the symbol, by which the data type is written within the

dashed box having a vertical bar at the right end of the box.

In figure A.2 different hair types are defined and exactly one value of the range has to be associated with the person.

2.4 *Select*

A data type **select** allows the possibility to choose exactly one of a defined variety of, e.g. other data types or entities. The name of the data type is written within a dashed box having a vertical bar at the left end of the box.

Figure A.3 shows an example for the use of the select data type. The data type has to be selected as one of the data types INTEGER, STRING, BOOLEAN, LOGICAL, REAL, BINARY or the entity measure_with_unit.

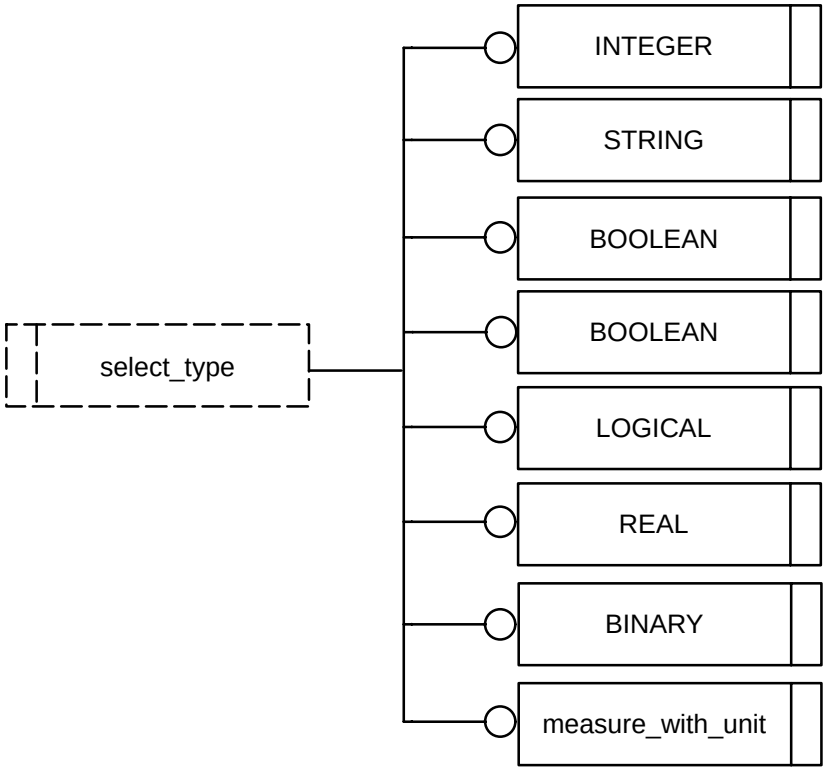

Figure A.3 – Select data type

2.5 *Cross references*

Graphical representations can span more than one page. If a relationship occurs between definitions on separate pages, the relationship line on each of the two pages is ended by a rounded box. It contains the page number, the reference number and the name of the entity referred to (see figure A.2). The page number is the number of that page where the referenced entity occurs. A reference number is used to distinguish between multiple references on a page.

2.6 *Relationship*

A **relationship** expresses a dependency or interaction between two entities. A relationship has cardinality, which indicates the number of objects in each of the entities at either end of the relationship that may be involved in a particular instance of that relationship. A relationship also has a name.

In the diagrammatic modelling notation, a relationship is shown as a solid or dashed thin line which is terminated by a circular arrowhead; the relationship name is written next to the line. The direction of a relationship is towards the arrowhead, which is important in that the name of the relationship must reflect its direction. A solid line indicates a compulsory relationship, whereas a dashed line indicates optional relationship (see figure A.4).

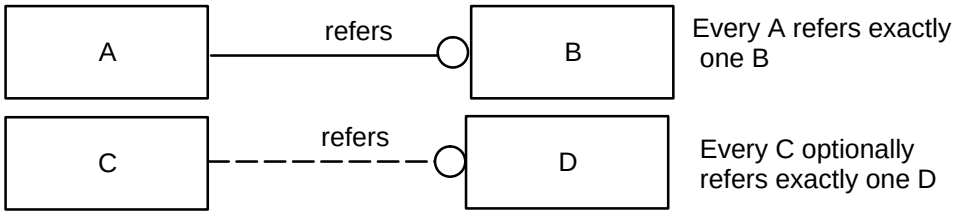

Figure A.4 – Relationships (1)

A cardinality string may be added to the relationship name, in which case it can take one of a number of forms (see below). If a relationship has no cardinality string included in is name, then the cardinality is assumed to be exactly one.

Cardinalities can be expressed in terms of sets, bags, lists and arrays. A **set** is an unordered variable length collection of unique items. A **bag** is an unordered variable length collection of not necessarily unique items A **list** is an ordered variable length collection of not necessarily unique items; however there are possibilities to constraint the list to a list of unique items. An **array** is a fixed size collection of not necessarily unique items which can be accessed by an index.

A cardinality can be shown as a string in the form **C[m:n]**, where **C** is one of **S**(set), **B**(bag), **L**(list) or **A**(array), where **m** is the lowest number of items allowed in the aggregation, and where **n** is the highest number of items allowed. If the cardinality is shown as **m:n**, then it is assumed to be as set (S). If the cardinality is shown as a single number (n), then it is assumed to be **S[n:n]**. See figure A.5. Note that an upper limit of **?** indicates ”unbounded”.

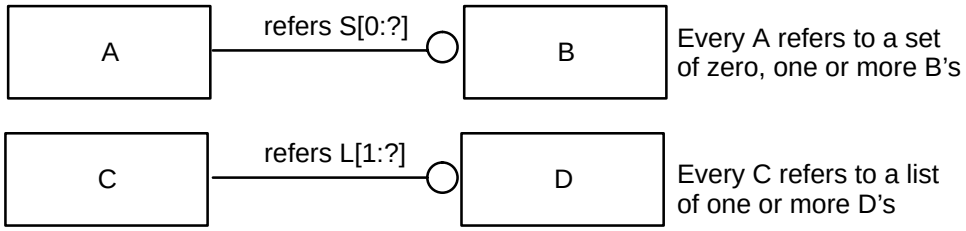

Figure A.5 – Relationships (2)

Examples of relationships are: *manages* (a **manager** manages exactly one **project**); *is_worked_on_by* (a **project** is worked on by zero, one ore more **non–managers**). The notation for these examples is shown in figure A.6.

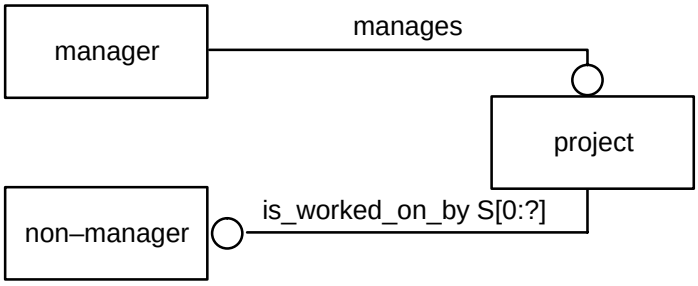

Figure A.6 – Relationship (3)

NOTE – A containment relationship is a special type of relationship (also called ownership). Such relationships are specially identified because of the special behaviour of the relationship.

Consider the two entities A and B, which have a containment relationship *owns* between them (every A owns zero, one ore more B's). See figure A.7.

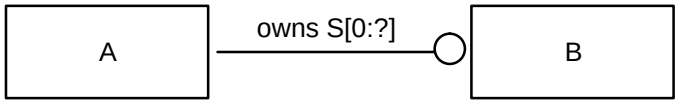

Figure A.7 – Relationships (4)

NOTE – If a database were constructed according to this model, then the database would not be in a valid state unless every instance of B were owned by one and only one instance of A. If an instance of A were deleted, then all the instances of B owned by that A would need to be deleted as well. (This is the special nature of the containment relationship). Note that this type of relationship is not specially indicated in the diagrammatic notation; however, any relationship like that shown in figure A.7 must be considered to be a possible containment relationship.

2.6.1 *Inverse Relationship*

In most cases, an inverse relationship can be inferred directly from the original relationship. For example, in figure A.7, the relationship *owns* has inferred inverse relationship *is_owned_by*, directed from entity **B** towards entity **A**, and having in this example the cardinality "exactly one". Strictly speaking, this should be shown as in figure A.8. (An inverse relationship is indicated by writing (INV) at the front of the name of the relationship).

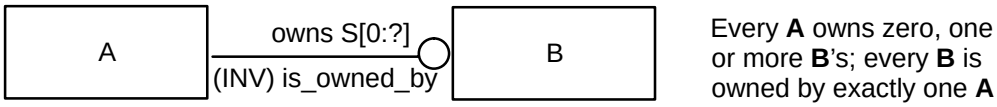

Figure A.8 – Relationships (5)

2.6.2 *Entity generalisation and specialisation*

Two or more entities which have some (but not all) characteristics and/or behaviour in common may be generalised into a **supertype**. Each of the entities that the supertype generalises is known as a **subtype**.

Another interpretation of subtypes and supertypes is to consider that a supertype entity is specialised into a series of subtypes. Each of the subtypes is a specialisation inherits all of the characteristics and behaviour of the supertype, but adds new characteristics and/or behaviour of its own.

Note that it is possible for an entity to be both a supertype and a subtype simultaneously.

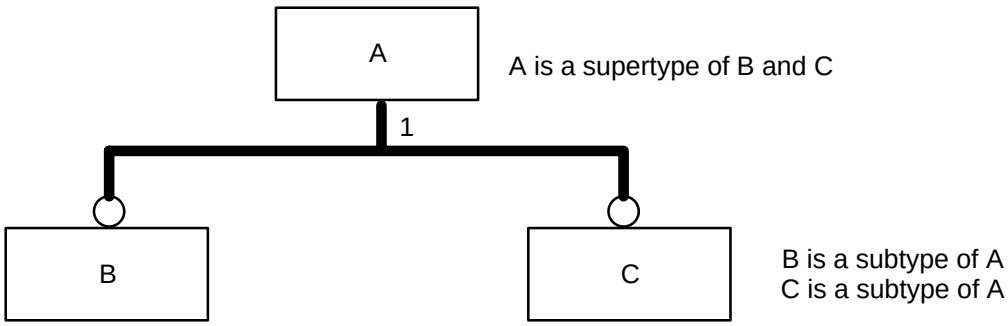

Figure A.9 – Subtypes and supertypes (1)

In the diagrammatic modelling notation, the relationship between a supertype and its subtypes is shown by using a series of thick lines, each one terminated at the subtype end with an open circle. See figure A.9.

Figure A.9 shows the notation for a **non–abstract supertype**. A non–abstract supertype is an entity which can be instanced as an object, but which is nonetheless a supertype. An **abstract supertype**, on the other hand, is a supertype entity which cannot be instanced. Figure A.10 shows the notation for an abstract supertype.

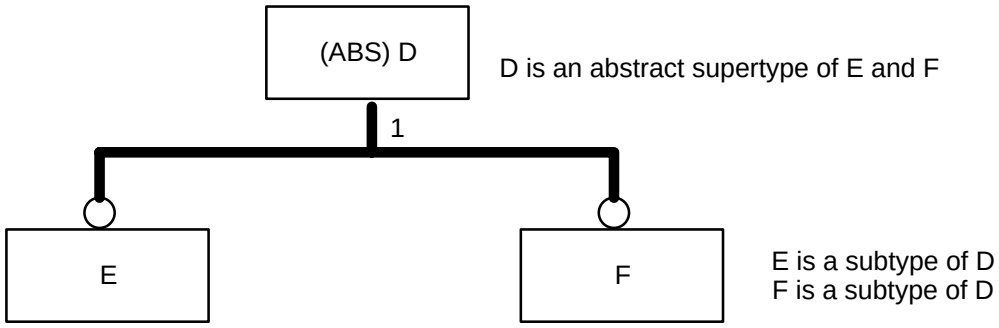

Figure A.10 – Subtypes and supertypes (2)

In figure A.9, an object of type **A** exhibits the characteristics and behaviour of the entity **A**, whereas an object of type **B** exhibits the characteristics and behaviour of entities **A** and **B** simultaneously. In figure A.10, it is not possible for objects of type **D** to be instanced. In other words, figure A.10 shows a "complete specialisation" (all objects of type **D** are also of type **E** or type **F**), whereas figure A.9 shows an "incomplete specialisation" (there are some objects of type **A** which are neither of type **B** nor of type **C**).

An entity can be a subtype of more than one supertype simultaneously. This is known as "multiple inheritance". Figure A.11 shows the notation for an entity **A** which is a subtype of two other entities **B** and **C**.

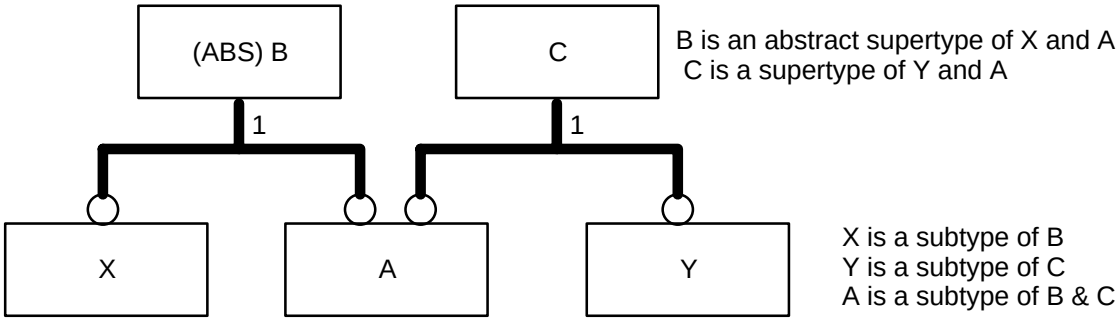

Figure A.11 – Subtypes and supertypes (3)

An example of a generalisation is that which generalises the **manager** and **non–manager** entities into the **employee** supertype (see figure A.12). The objects in the **manager** entity are slightly different to those in the **non–manager** entity (managers manage a project and supervise other employees, whereas non–managers share a lot of common characteristics (e.g. both appear on a payroll), there are occasions when it is convenient to refer to such objects as simply an employee. This implies the existence of an abstract supertype called **employee**.

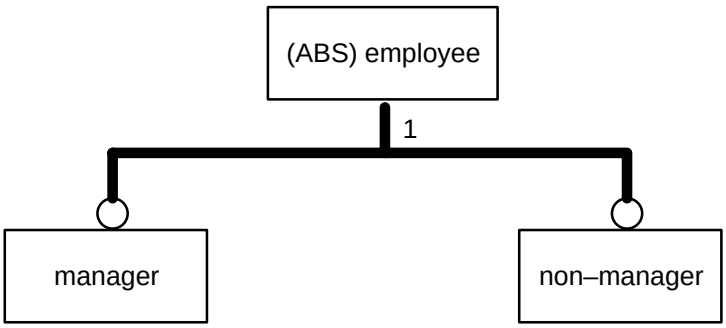

Figure A.12 – Subtypes and supertypes (4)

An example of a specialisation is that which specialises the **mammal** entity into the **cat** and **dog** subtypes (see figure A.13). All objects in the **mammal** entity share common characteristics and behaviour (e.g. they all have hair, and suckle their young). The objects in the **cat** entity are slightly different to those in the **dog** entity in a number of respects. Note that this is an example of an incomplete specialisation, because there are objects in the **mammal** entity which neither cats nor dogs (e.g. whales, horses, humans).

As a general rule, the process of generalisation will often lead to the definition of an abstract supertype, whereas the process of specialisation will often lead to the conversion of an existing entity into a non–abstract supertype (unless the specialisation is complete, in which case the entity is converted into an abstract supertype).

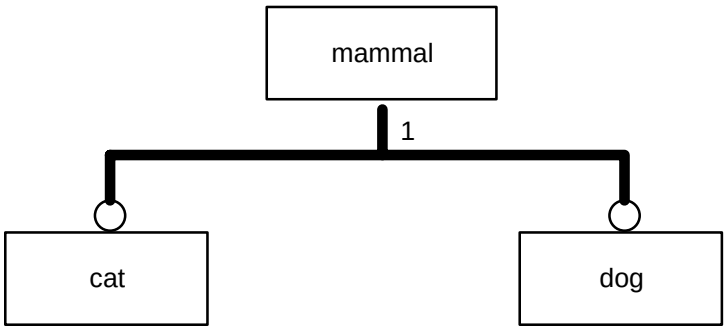

Figure A.13 – Subtypes and supertypes (5)

Many models specifically avoid the use of non–abstract supertypes. This is achieved by the addition of an extra catch–all subtype. For example, in the specialisation of the **mammal** entity shown in figure A.13, a new subtype entity called **other_mammal** could be added (as in figure A.14), making the specialisation complete, and converting the **mammal** entity into an abstract supertype.

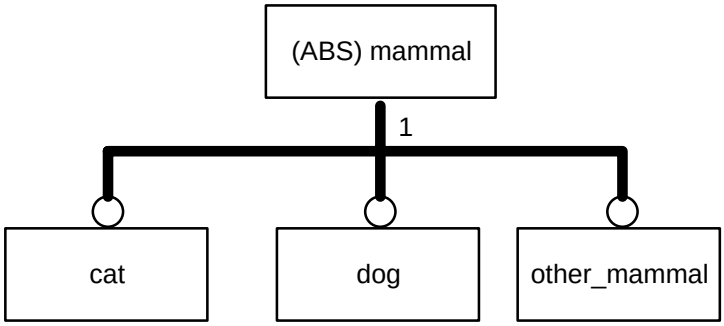

Figure A.14 – Subtypes and supertypes (6)

3 *Example Model*

Figure A.15 shows a simple example of an information model drawn using the notation described above.

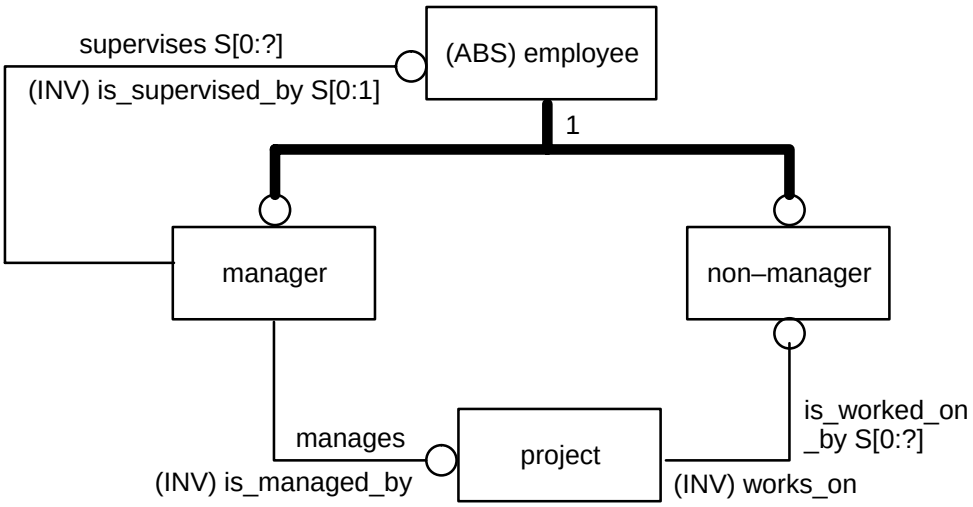

Figure A.15 – Example Model

The model should be read as follows:

1. Every manager and non–manager is also an employee; every employee is either a manager or a non–manager.

2. Each manager manages exactly one project; each project is managed by exactly one manager.

3. Each manager supervises zero, one or more employees; each employee is supervised by zero or one manager. Note that there may be some employees who are not supervised (e.g. the General manager).

4. Each project is worked on by zero, one or more non–managers; each non–manager works on exactly one project.

Note that each entity describes a complete set of objects in a particular domain of interest. A relationship is between one group of objects and another group of objects; each group of objects involved in such a relationship may be a subset of the whole group like objects in the domain of interest. A common mistake that is made while reading an information model is to assume that a relationship is between all of the objects in one entity and all of the objects in another entity.

———————————

Published by Siemens AG
Industrial Projects and Technical Services
ATD TD5 SP
P.O. Box 3240, D–91050 Erlangen

Contact: Fritz Reuter
Tel +49 (0)9131/7–43825
Fax +49 (0)9131/7–20249
email:pdm@erl9.siemens.de

Know–how in
Systems Integration.
Siemens

Siemens Aktiengesellschaft

Germany