**2/4/2020**

A concept can be modeled as:

1. Class: exist independently, have properties (attributes), form relationship,..
2. Attribute: has only value; simplicity
3. Association: ask: what are the objects in the association? Mostly between two objects.
4. No modeling

**SCMS**

Create a data model by constructing an UML class diagram to support a portion of a drastically simplified part of a conference management database: SCMS. It only supports a *very* limited set of functions. Make reasonable assumptions.

SCMS supports hosting Web-based conference systems to manage conferences.

Concepts:

1. Conference: class
2. Management: relationship, not for the moment; no modeling.
3. Conference system: class? (definition: a software managing conferences. E.g. SCMS.) Do we need to store data about the concept? (Do you have many instances of the concept?) no.
4. Host: no, or later.

noun likely -> class (e.g. conference)  
verb likely -> relationship Mr. A (object) chairs (verb -> relationship) conference B (object)

There are many real world conferences and you can browse a few to get some ideas. However, keep in mind that we are modeling a drastically simplified part only.

SCMS allows members to set up conferences for people to submit papers, review papers, and participate in the conferences.

1. Member/People: more specific term -> member (naming is important). Assume people is a synonym of member.
   1. Authors submit papers
   2. Organizers organizes conference

As a class: def: A member can submit a paper to a conference.

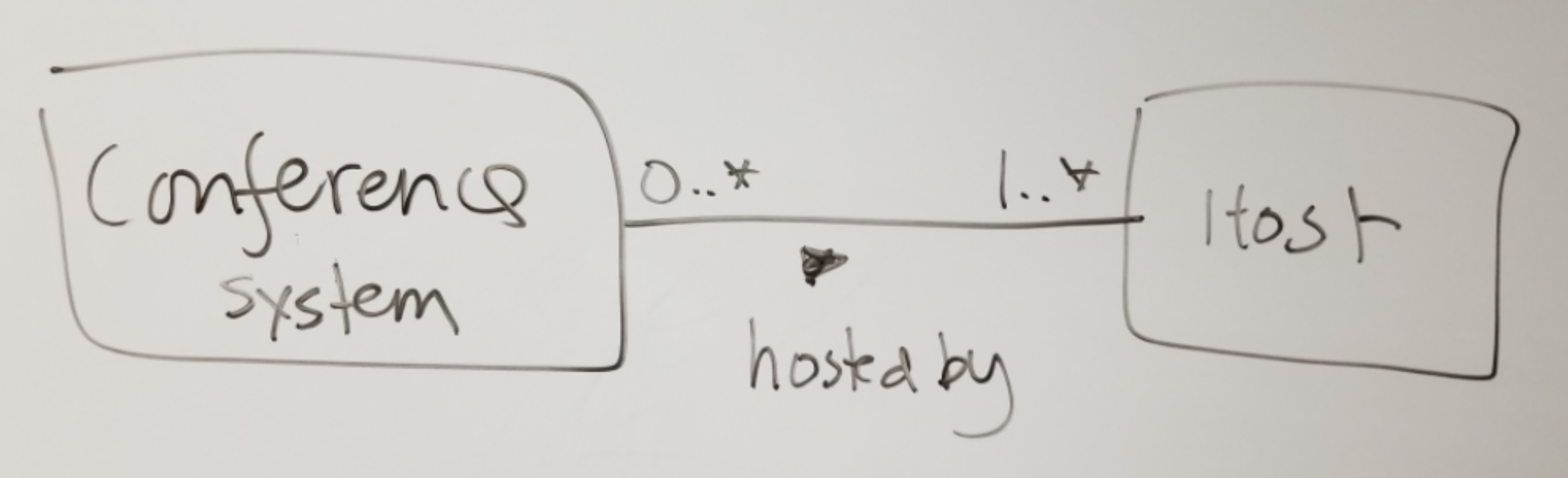
1. Paper: class: properties: title, authors, keywords, …
2. Submit: association. Promote to a class

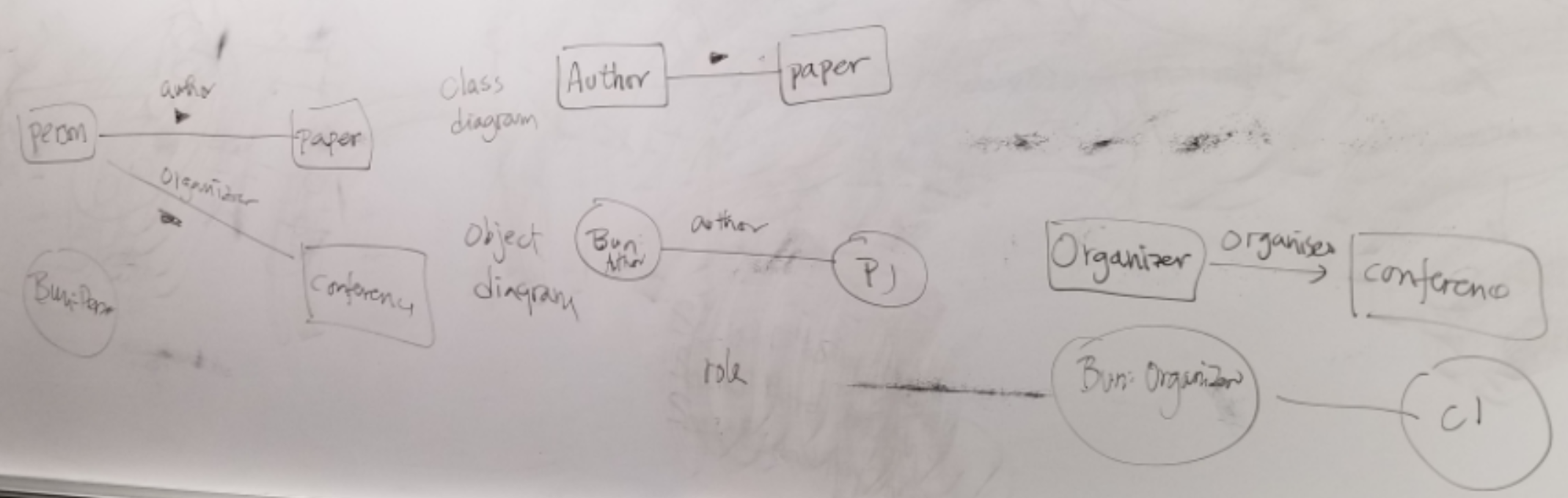
E.g. A member X submits (verb -> association) a paper Y.

A member X submits (verb -> association) a paper Y to a conference Z. (tertiary association)

A member P reviews a paper Y of a conference Z.

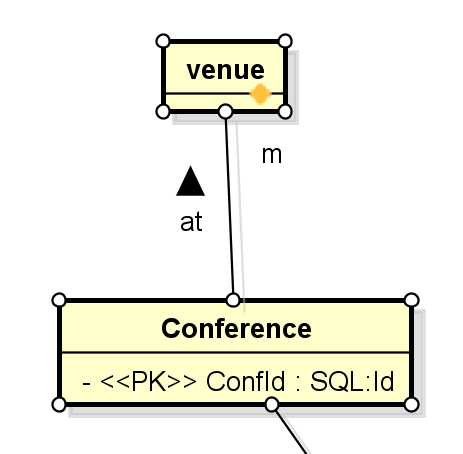
1. Conference type



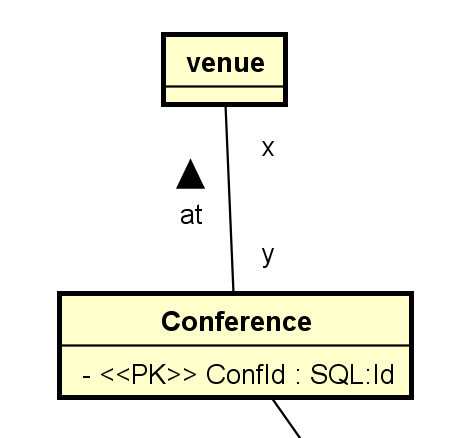


A conference has an unique Id (attribute, PK) (all Id should be of the type SQL:Id). It must have a name, start date, end date (attribute) and a venue (class) name. A venue must have an address. Every address must have an unique Id, a street, a city, and a country. It may or may not have a state or a zip code since come countries do not have them. A country has an unique country code and an unique name (e.g. 'US' and 'United States of America').

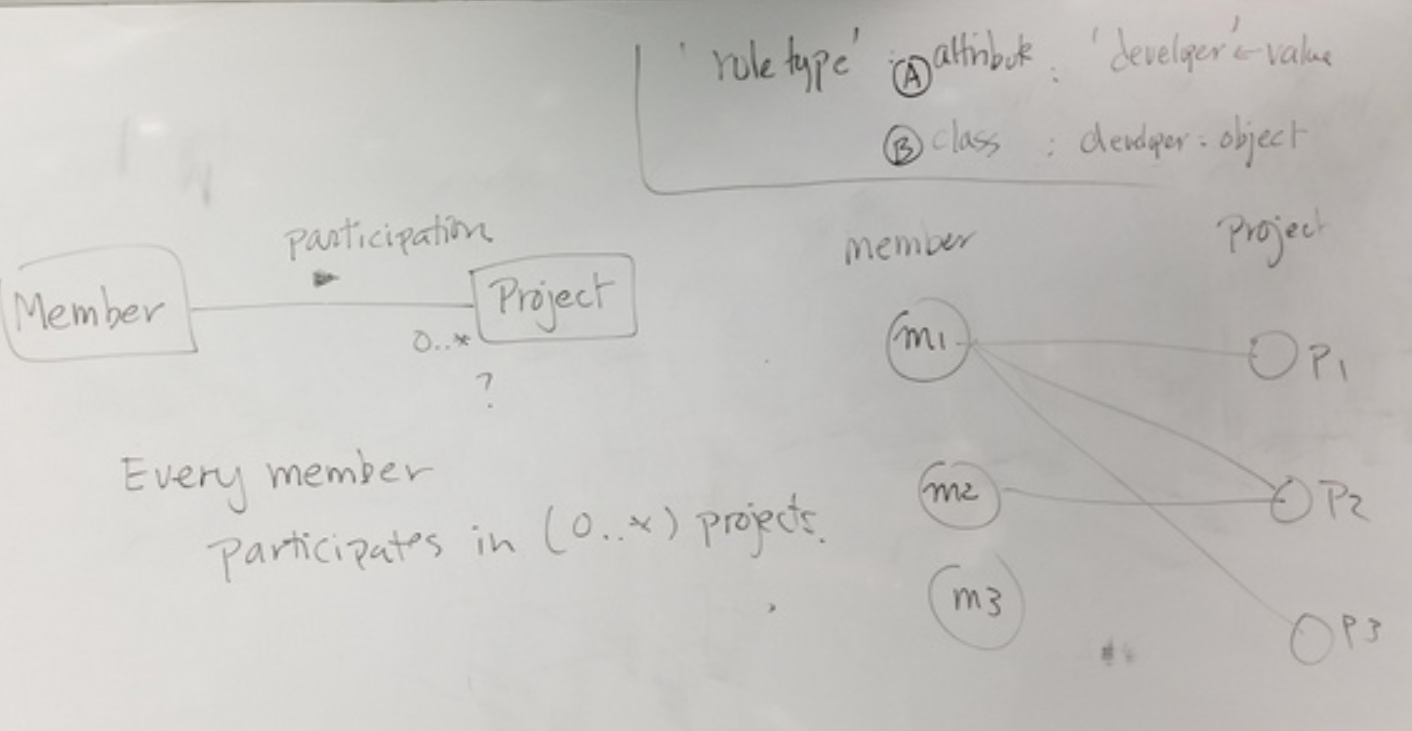
What is m below?

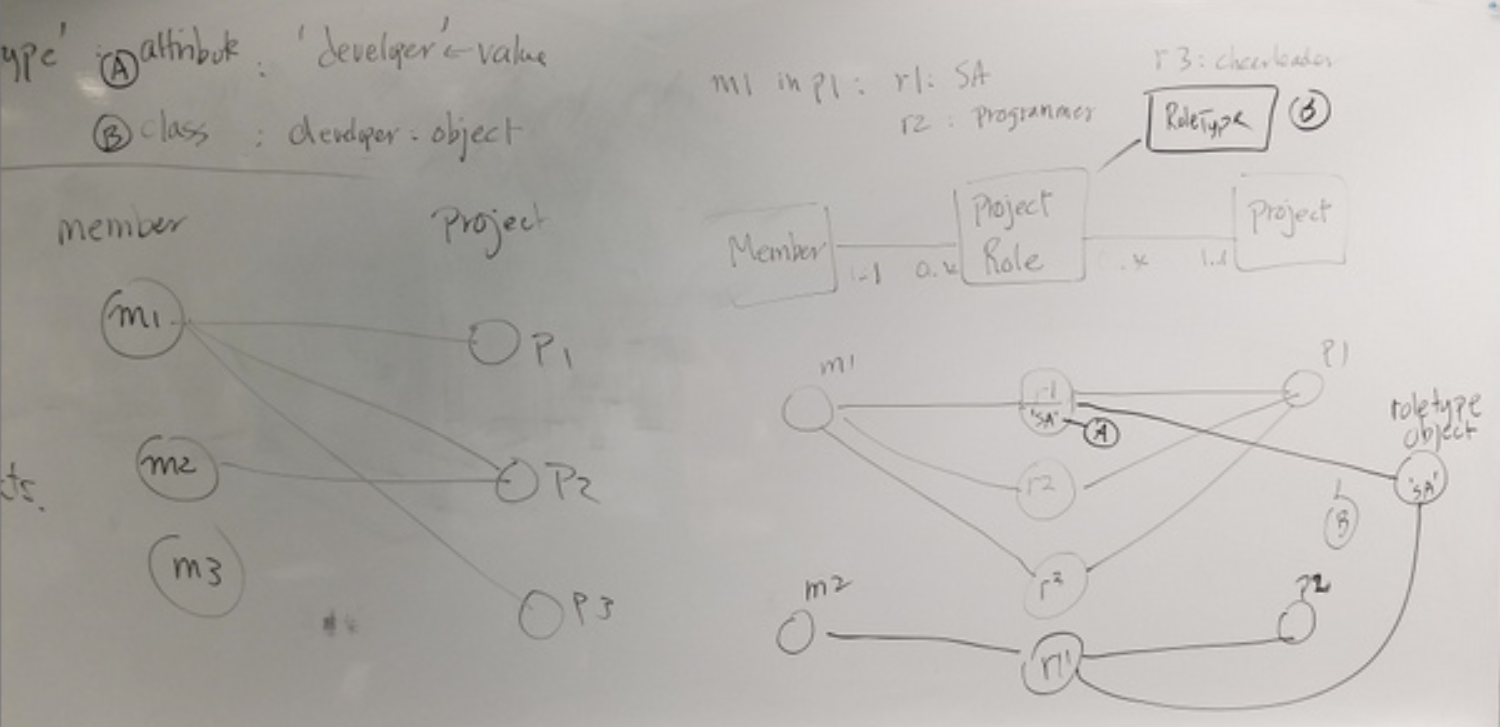


m:

1. 1: every conference has one and only one venue.
2. 0..\* : every conference has zero or more venue.
3. 0..1: every conference has no or one venue.
4. 1..\* : every conference has one or more venue. 

Y?

1. 1: every venue has one and only one conference.
2. 0..\* : every venue has zero or more conference.
3. 0..1: every venue has no or one conference.
4. 1..\* : every venue has one or more conference. (information about the venue cannot be stored in the database unless it is used for at least one conference
5. 



A conference has a lead organization and any number of support organizations. An organization has an unique Id, a name, and an optional phone (all Phone should be of the type SQL:PhoneType). An organization may have an address and/or a country of origination. For example, the organization 'BASG Houston' may have the address '2909, Bay Area Boulevard, Houston, TX 77058, USA' and a country of origin of 'Germany.'

A conference may have many tracks. For example, the conference 'Artificial Intelligence Application' (AIA) may have the tracks 'Machine Learning', 'Deep Learning', 'Genetic Algorithms', etc. Papers can be submitted to one of these tracks for publications. If a conference does not have any naturally tracks, a generic track, such as 'general', is created. All papers will then be submitted to this 'general' track. A track has an unique id and a track name. Note that a track name is not unique, as, for example, more than one conferences may have the same track names of 'Deep Learning.'

A paper has an unique Id, a title, a submission date (of type SQL:Datetime), and one or more authors. Any person stored in SCMS has a last name, a first name, an email, and an optional phone. Any person can be an author of multiple papers. The position of the authorship of a paper should be recorded. For example, the paper:

Jane Smith, Jade Johnson, Karl Eastmond, A mid-summer night dream of deep learning.

may be submitted to the track 'Deep Learning' of the conference AIA. The authorship information:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Author** | **Position** | **EMail** | **Phone** | **Organization** |
| Jane Smith | 1 | smithJ111@uhcl.edu | NA | UHCL |
| Jade Johnson | 2 | JJohnson2088@gmail.com | NA | NA |
| Karl Eastmond | 3 | eastmondk1@uhcl.edu | 281-283-1000 | UHCL |

A person can also review any number of papers. The review comment and outcome must be recorded. Outcomes are selected from a list of defined choices for a conference. For example, for AIA, the possible outcomes may be:

|  |  |  |
| --- | --- | --- |
| **Outcome Id** | **Outcome #** | **Outcome** |
| 1290 | 1 | Accept with no modification |
| 1291 | 2 | Accept with minor modifications |
| 1292 | 3 | Accept with major modifications |
| 1293 | 4 | Conditional Accept with major modifications |
| 1294 | 5 | Reject |

Note that Outcome # is meaningfully used by AIA only whereas Outcome Id is an unique Id across all conferences.

Thus, a person 'Beto Jones' may review the paper "A mid-summer night dream of deep learning" withe result:

* OutcomeId = 1290; 1, Accept with no modification.
* Comment = 'Excellent work.'

A paper can have any number of reviewers.

A paper may have many keywords, which can be shared among papers.

A person needs to be a registered member to participate in a conference. The account name and password of a member should be stored. The address of a member must also be recorded. A member may have an organization associated with him.

A conference must have a member as its chair. It can have any number of members as its managers. A member can attend many conferences.

A member may have many roles in a project. Some of these role types are predefined by Taskster, for example, librarian, developer, team leader, supervisor, consultant, etc. They are standard roles. Some roles are project specific and can be defined by a manager of the project. For example, a software project may include roles such as modeler, programmer, technical writer, etc. When a project-specific role type is defined, a description and a creation time may be stored together with the role name. The manager who defines the role should also be noted.

Member M1 works on project p1 as a developer, …..

Member M1 is a developer on project p1.

Member M1 is a developer.