Domain-Specific Language Design
Carnivals Management System

Participants: *S. Ishtiaq, S.K. Jain, N. Raj Pappuraj, A.Toloekashefpakdel*

Deadline: 10th April 2024 – Dr. Richard Paige

1 Introduction

Welcome to the world of the carnivals, a vibrant celebration that captivates hearts and sparks joy. Carnivals are festive gatherings that draw people from far and wide, offering a range of entertainment and excitement. In the spirit of embracing this joyful atmosphere, a domain–specific language using Epsilon has been designed to capture the essence of carnivals through meta–modeling.

Our journey begins with designing the ECore [4] meta–model for our carnival using Emfatic [3], carefully selecting classes, relationships, and multiplicities to capture the essence of our celebration. After ensuring the model's validity and registering it, we turn to Eugenia to generate the boilerplate code for the Graphical Modeling Framework (GMF) [6]. With a keen focus on clarity, we establish a direct correlation between model constructs and graphical symbols, ensuring semiotic clarity. With our newly minted GMF–based editor, we dive into creating our model - the McMaster carnival! Moving forward, we employ the Epsilon Validation Language to add robust constraints, ensuring its integrity and correctness. Finally, leveraging the power of Epsilon Generative Languages, we generate vibrant HTML pages.

Our goal is not only to develop a comprehensive meta-model for carnivals but also to provide insights into the practical applications of meta–modeling in software engineering contexts. By engaging in this project, you will gain hands-on experience with meta–modeling techniques and deepen your understanding of eclipse based development environments.

2 Meta-Modeling

To model the dynamic aspects of the carnival, Emfatic has been used to create the ECore model. The design encompasses fundamental elements that define the carnival experience, ranging from diverse activities to enthusiastic participants.

2.1 Class Diagram

- NamedElement: An abstract class serving as the foundation for naming various elements within the carnival.
- Activity: An abstract class capturing the diverse range of activities present at the carnival.
- Event: A type of activity open to all carnival attendees.
- Booth: A type of activity with limited resources, offering unique experiences.
- Participant: An abstract class encompassing the different individuals engaged in the carnival.
- Visitor: Class representing attendees who join the carnival as visitors, embracing the fun-filled spirit.

- Sponsor: Class representing folks who actively contribute as sponsors, enhancing the carnival experience.
- CarnivalDay: Class defining a specific day within the carnival, comprising a variety of engaging activities.
- Carnival: Base class encapsulating the entire carnival celebration, composed of carnival days and diverse participants.

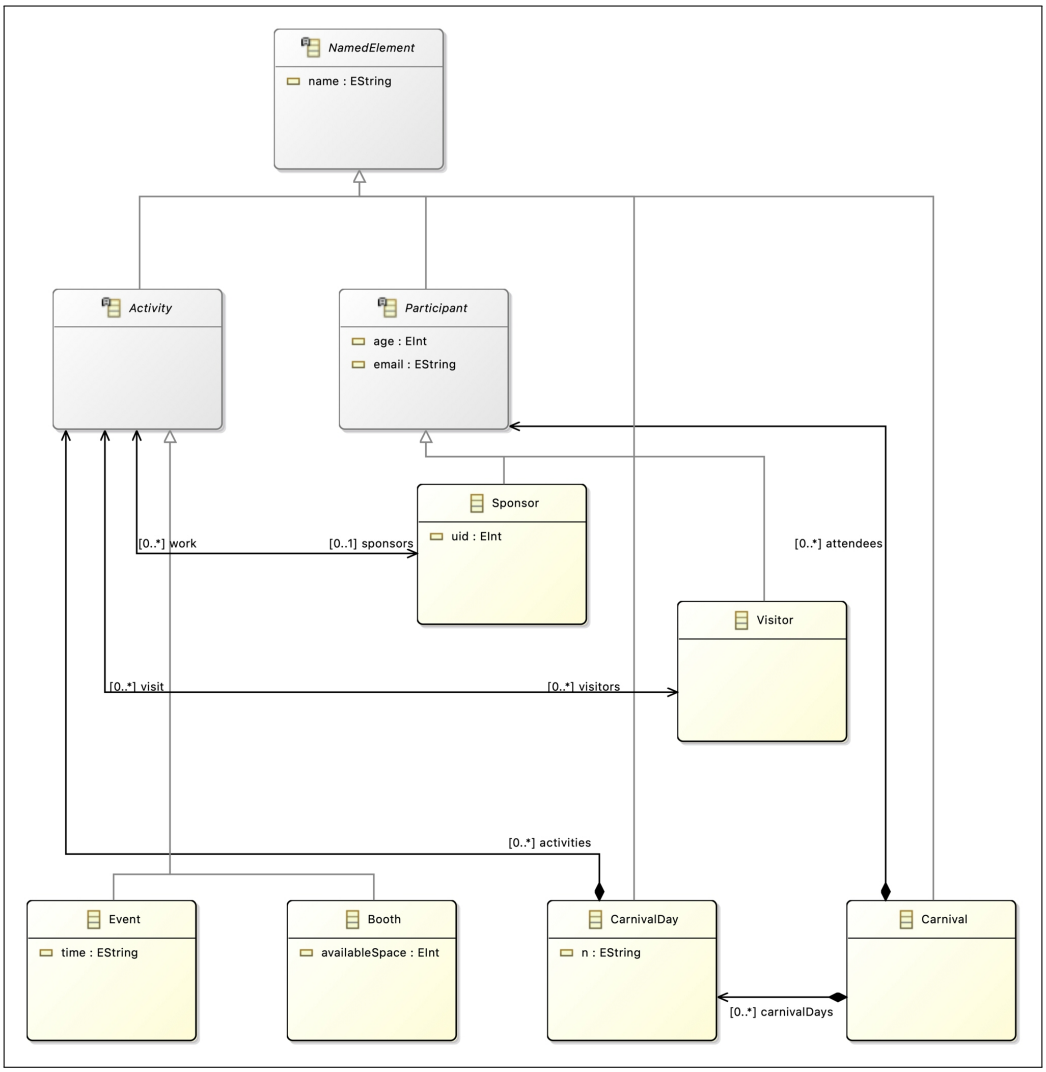


Figure 1: Illustrates the ECore meta-model for the carnival.

2.2 Assumptions

- The sponsors will assume responsibility for event and booth management, thereby eliminating the need for additional helpers or staff.
- Booths remain open continuously throughout the day or until their stocks are depleted.
- Each event takes place at a distinct time during the day, and no two events can happen simultaneously on the same day.

2.3 Alternatives

In the alternate design of the carnival meta model within the context of ECore modeling, the introduction of the "Helper" EClass would enhance the existing structure by providing a dedicated representation for staff or assistants involved in event and booth management tasks. This new EClass would establish a clear distinction between sponsors and the individuals responsible for executing various operational aspects of the carnival.

The "Helper" EClass would contain attributes defining the roles, skills, and availability of each staff member. Additionally, it would have references to other relevant classes within the meta model, such as events or booths, to indicate the specific areas where helpers are deployed. These references would enable efficient navigation and management of helpers within the carnival ecosystem.

By introducing the "Helper" EClass, the meta model gains added flexibility and granularity in representing the organizational structure and operational dynamics of the carnival. Sponsors can now have a dedicated team of helpers under their purview, allowing for more effective coordination and delegation of tasks. However, this addition may necessitate adjustments to the concrete syntax and navigation paths within the model to accommodate the new class and its relationships effectively. Nonetheless, the inclusion of helpers enhances the overall functionality and realism of the carnival meta model.

3 Concrete Syntax and Editor

3.1 Semiotic Clarity

In our carnival meta model design, achieving semiotic clarity was important to ensure that the graphical representations effectively conveyed the underlying model constructs. We first established a one-to-one mapping between model elements and their graphical symbols, ensuring each entity, such as sponsors, events, and helpers, was distinctly and clearly represented.

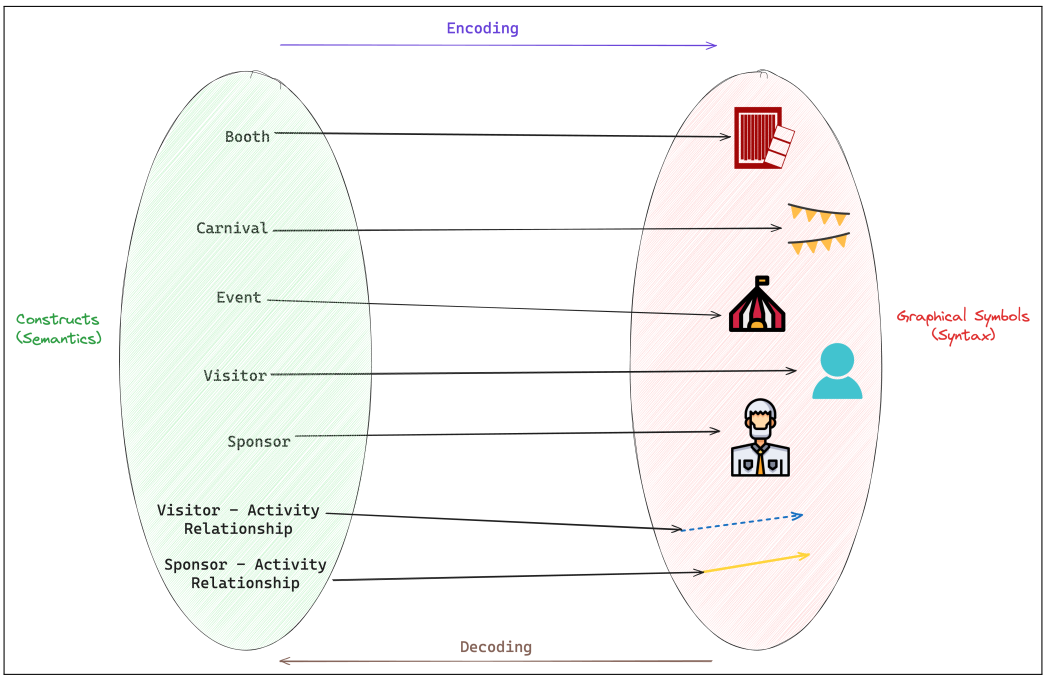


Figure 2: Semiotic Clarity for an illustration of the mapping.

3.2 The GMF Editor

We employed Eugenia, an Eclipse-based tool built on the Graphical Modeling Framework (GMF), to create the Concrete Syntax and Editor The Eugenia generated editor was run as a plugin on Eclipse, using which we created a sample carnival model illustrated in Figure 3 below. The resulting graphical editor provides stakeholders with an intuitive and easily interpretable visual representation of the meta model's structure and relationships.

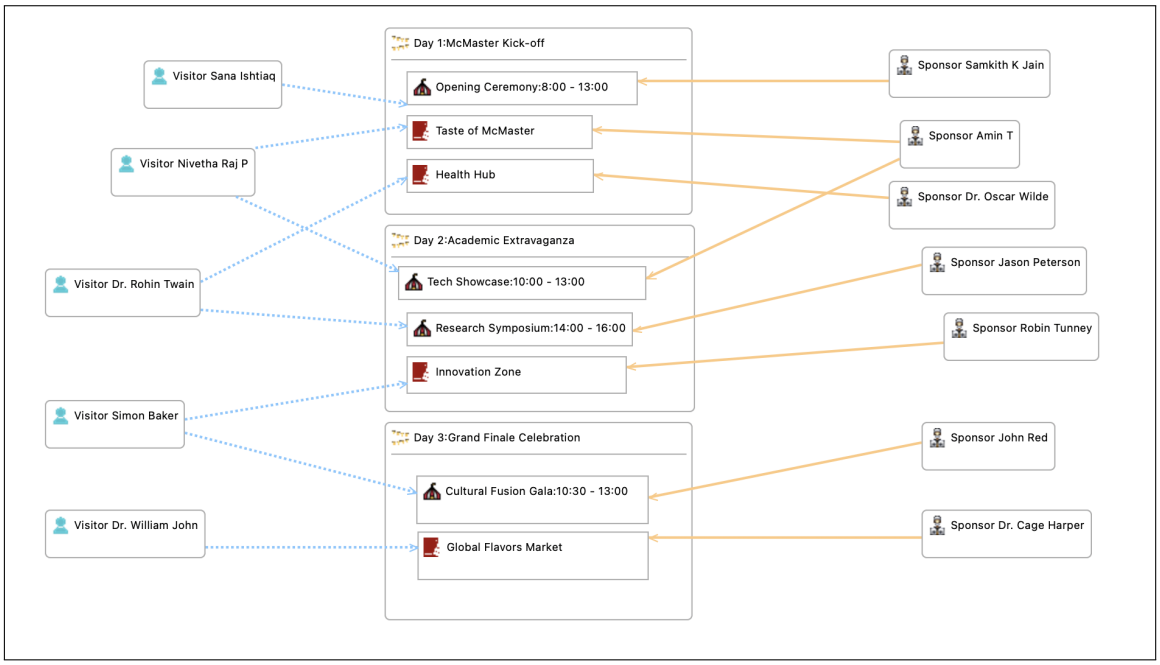


Figure 3: GMF Editor / Model Instance.

3.3 Advantages

Using Graphical Modeling Framework (GMF) for the carnival meta-model offers several advantages. One major advantage is its ability to provide a customizable and visually appealing graphical editor tailored specifically to the domain of the carnival meta-model. This enables stakeholders to intuitively create, edit, and visualize the model's elements and relationships, enhancing understanding and collaboration. Additionally, GMF automates much of the underlying infrastructure required for creating graphical editors, saving time and effort in development.

3.4 Disadvantages

The alternative design incorporating a set of helpers/staff, while adding functionality, poses challenges in terms of clutter and symbol flooding. Introducing additional elements like helpers/staff may lead to a denser diagram with an increased number of graphical symbols, potentially making it more difficult to discern and interpret the model's structure. This issue can diminish the clarity and usability of the graphical representation, counteracting the benefits of using GMF for visual modeling. Thus, careful consideration and balancing of functionality and clarity are essential when incorporating such complex features into the graphical representation of the carnival meta-model using GMF.

4 Validation

Epsilon Validation Language (EVL) [5] has played a crucial role in ensuring the integrity and correctness of our carnival meta-model. Leveraging EVL, we added validation constraints to the meta-model, ensuring that it adheres to the specified criteria. Different critiques and constraints have been strategically applied to various classes within the meta-model to produce human-readable error messages, aiding in the identification and resolution of potential issues. Figure 4 below, provides an example of EVL validation in action, demonstrating how it helps detect and report violations of defined constraints.

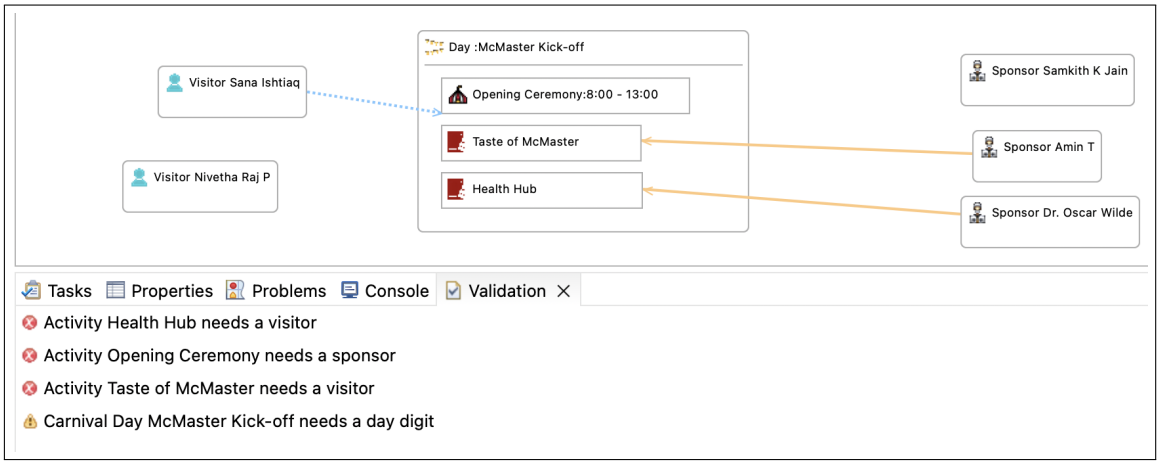


Figure 4: Validation using EVL .

4.1 Validation Constraints

1. Carnival root class Constraints:
   - HasAtleastOneDay:
     - Checks if the carnival has at least one day.
     - Message: carnival needs at least one day.
   - HasAtleastOnePerson:
     - Checks if the carnival has at least one person.
     - Message: carnival needs at least one person.
2. Carnival Day Constraints:
   - HasId:
     - Critique ensuring each carnival day has a day digit.
     - Message: carnival Day needs a day digit.
   - HasAtleastOneActivity:
     - Checks if the carnival day has at least one activity.
     - Message: carnival Day needs at least one activity.

3. Activity Constraints:
    - HasVolunteer:
        - Checks if the activity has a sponsor.
        - Message: Activity needs a sponsor.
    - AllEventsHaveDurations:
        - Checks if all events have durations.
        - Message: Event must have a duration.
4. Booth Constraints:
    - HasAvailableSpace:
        - Checks if the booth provides available space.
        - Message: Booth must provide space available.
5. Person Constraints:
    - HasAge:
        - Checks if the person has an age.
        - Message: Person must have an age.
    - HasEmail:
        - Critique ensuring each person has an email.
        - Message: Person should have an email.
    - AllPersonsHaveUniqueEmails:
        - Checks if all persons have unique email addresses.
        - Message: All persons must have unique email addresses.
6. Sponsor Constraints:
    - HasPartnerId:
        - Checks if the sponsor has a partner ID.
        - Message: Sponsor must have a partner ID.
    - AllSponsorIdsUnique:
        - Checks if all sponsor IDs are unique.
        - Message: Sponsor IDs must be unique.
7. Named Element Constraints:
    - HasName:
        - Checks if the element has a name.
        - Message: Element needs a name.

4.2 Integration

Leveraging the EVL documentation and insights from course slide decks, we implemented custom EVL constraints to validate aspects of the model that couldn't be expressed in the metamodel itself. Through careful integration with the editor's codebase, we enabled validation checks, allowing users to receive feedback on their model edits. By highlighting errors and warnings directly within the editor interface, we empowered users to identify and address validation issues efficiently. This seamless integration enhances the usability and reliability of our editor, providing a more robust modeling experience.

For more details on the EVL validation process and specific constraints code, please refer to the validation section of our GitHub Repository - carnival.evl.

5 Model to Text Transformation

Model to text transformation is a crucial aspect of the carnival project, facilitating the generation of human-readable documents from the underlying model. In this project, we utilized Epsilon Generation Language (EGL)[1] and EGL Co–Ordination Language (EGX)[2] to transform the carnival meta-model into HTML documents for displaying the sponsors, visitors, and schedule of activities during the carnival.

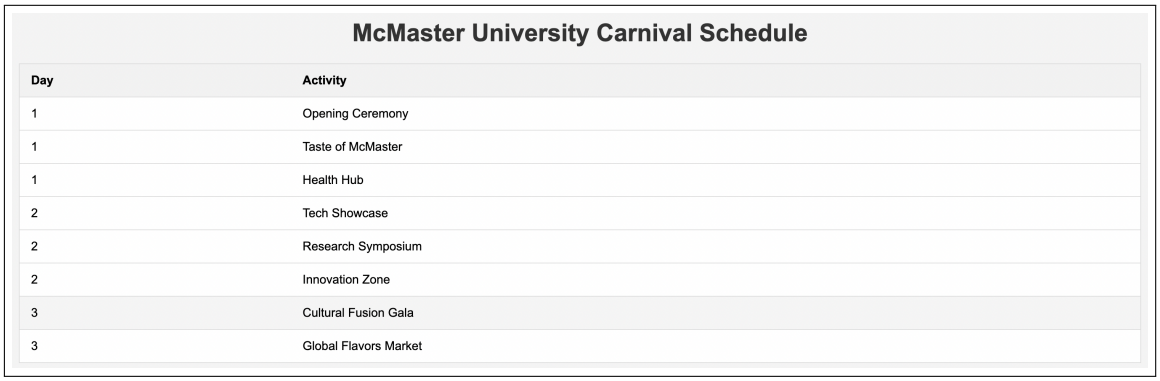
5.1 Summary of Transformation Files

1. **Visitors List HTML Template:** The Visitors List HTML template generates a document displaying information about sponsors and visitors attending the carnival. It presents a tabular format with columns for Serial Number, UID, Name, Email, and Activity. Using EGL syntax, the template iterates through carnival days and activities to populate the table with relevant data.

2. **Carnival Schedule HTML Template:** The carnival Schedule HTML template generates a document presenting the schedule of activities for each day of the carnival. It organizes the information in a table format with columns for Day and Activity. Similar to the Visitors List template, EGL is used to iterate through carnival days and activities, filling the table with the schedule details.

3. **Transformation Rules:** Two transformation rules are defined to apply the templates to the carnival meta-model:

   - *carnivalModelToSchedule*: Applies the carnival Schedule template to generate the carnivalSchedule.html document, Figure 5 shows the output for the same.



McMaster University Carnival Schedule

Day	Activity
1	Opening Ceremony
1	Taste of McMaster
1	Health Hub
2	Tech Showcase
2	Research Symposium
2	Innovation Zone
3	Cultural Fusion Gala
3	Global Flavors Market

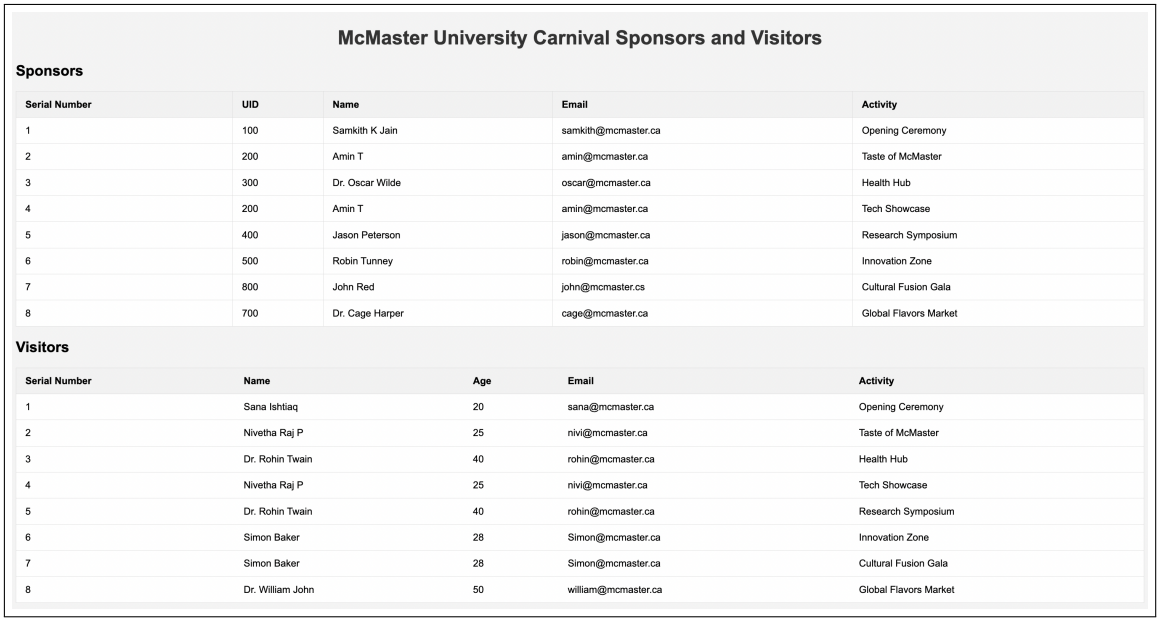
Figure 5: Schedule of carnival.

   - *carnivalModelToList*: Applies the Visitors List template to generate the carnivalList.html document, Figure 6 shows the output for the same.

These transformation rules define the templates to be used and specify the target HTML files where the generated content will be stored. Additionally, the EVL and EGX code used for validation constraints and model-to-text transformation rules can be found in the project's GitHub Repository - EVL and EGX files.

5.2 Challenges

We wanted to transform the model into an innovative webpage, however, one rule in EGX allows us to write to one target, we could not find a way to append the transformation to one target HTML. We decided to produce multiple HTML files. These files could later be referenced in a aggregate HTML file along with other web files for styling, markup and



McMaster University Carnival Sponsors and Visitors

Sponsors

Serial Number	UID	Name	Email	Activity
1	100	Samkith K Jain	samkith@mcmaster.ca	Opening Ceremony
2	200	Amin T	amin@mcmaster.ca	Taste of McMaster
3	300	Dr. Oscar Wilde	oscar@mcmaster.ca	Health Hub
4	200	Amin T	amin@mcmaster.ca	Tech Showcase
5	400	Jason Peterson	jason@mcmaster.ca	Research Symposium
6	500	Robin Tunney	robin@mcmaster.ca	Innovation Zone
7	800	John Red	john@mcmaster.cs	Cultural Fusion Gala
8	700	Dr. Cage Harper	cage@mcmaster.ca	Global Flavors Market

Visitors

Serial Number	Name	Age	Email	Activity
1	Sana Ishtiaq	20	sana@mcmaster.ca	Opening Ceremony
2	Nivetha Raj P	25	nivi@mcmaster.ca	Taste of McMaster
3	Dr. Rohin Twain	40	rohin@mcmaster.ca	Health Hub
4	Nivetha Raj P	25	nivi@mcmaster.ca	Tech Showcase
5	Dr. Rohin Twain	40	rohin@mcmaster.ca	Research Symposium
6	Simon Baker	28	Simon@mcmaster.ca	Innovation Zone
7	Simon Baker	28	Simon@mcmaster.ca	Cultural Fusion Gala
8	Dr. William John	50	william@mcmaster.ca	Global Flavors Market

Figure 6: Visitor and Sponsor List.

more.

Moreover, we attempted to come up with graphs in HTML for visualizing the sponsors and visitor distribution across different days and acivities but it was not straightforward to do so. Alternatively, we could have transformation rules to dump the data to a CSV file and produce visuals as necessary.

6 Conclusion

Our project delves into the creation of a carnival meta-model using Epsilon and its various aspects. Meta-modeling is extensively covered, showcasing the ECore meta-model for the carnival, along with assumptions and alternatives considered. Next, we look into the concrete syntax & editor details the Graphical Modeling Framework (GMF) and its advantages, followed by a discussion on validation using EVL and integration with the editor. Model to text transformation is explored through EGL and EGX, demonstrating the generation of HTML documents. Overall, we have captured the comprehensive journey of designing and implementing the carnival meta-model, providing valuable insights into each phase of the project.

7 Software and Dependencies

1. Eclipse Modeling Tools Version - 2023-12 (4.30.0): Download.
2. Graphical Modeling Framework (GMF) - Tooling (3.2.1): Download.
3. Epsilon (2.1): Download.
4. Emfatic (1.0.0): Download.

References

[1] EGL: Epsilon Generation Language.

[2] EGX: Epsilon Generation Language Documentation.

[3] EMF: Eclipse Modeling Framework Documentation.

[4] ECore: Eclipse Modeling Framework Core Tutorial.

[5] EVL: Epsilon Validation Language Documentation.

[6] GMF: Graphical Modeling Framework Tutorial.

[7] OCL: Object Constraint Language Documentation.