Method parts / Fragments construction for business process modelling

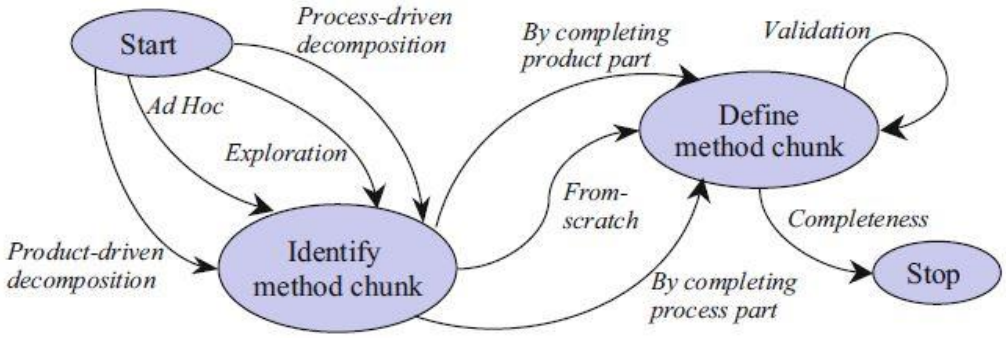


Fig. 5.4 Process model for construction of method chunks (modified from Ralyté 2004)

For the construction of method parts (fragments or chunks), the SME book recommends the map-based approach as shown in figure 5.4. For our application domain of business process modelling (BPM), we would like to use a fine granular approach since we believe an application made from fine-grained components can be better tailored then a coarse-grained application. Method tailoring plays an important role in the situational method engineering (SME). For these reasons, we will focus mainly in the construction of fragments as method parts. As per the book of SME, a map consists of sections of intentions and strategy as follows:

Map = sections consisting of **<Source intention, target intention, strategy>**

A strategy says how to go from source intention to target intention. But the question here is how we can know that a particular goal/idea can be a source intention or target intention. To extend the question, we can also ask how to choose the best strategy to move from source to target intention. To answer all these three questions, the SME book provided 3 guidelines as follows:

1. ISG (Intention selection guideline): Associated with a target intention
2. SSG (Strategy selection guideline): Associated with a source and target intention
3. IAG (Intention achievement guideline): Associated with all three (source and target and some selection of strategy). IAG can also be considered as a map.

The authors in the SME book has conformed that the map shown in figure 5.4 can also be used for construction of fragments. In this figure, we have two **main intentions / Sections:**

(i) Identify a method fragment: To achieve this intention from our application domain perspective, the authors have suggested to take the "Ad-hoc" strategy.

(ii) Define a method fragment: To achieve this intention, the authors have suggested to follow the strategy of "From scratch".

As per the authors, strategies such as "Product driven decomposition" or "Process driven decomposition" are unnecessary for our application domain since they deal with construction of method parts from existing non-modular methodology. To apply the above-mentioned guidelines to construct new fragment, first we need to elicit requirements for our business process application. The section 6.3 of the SME book gives an idea about requirement gathering before construction of any method parts.

Requirement gathering:

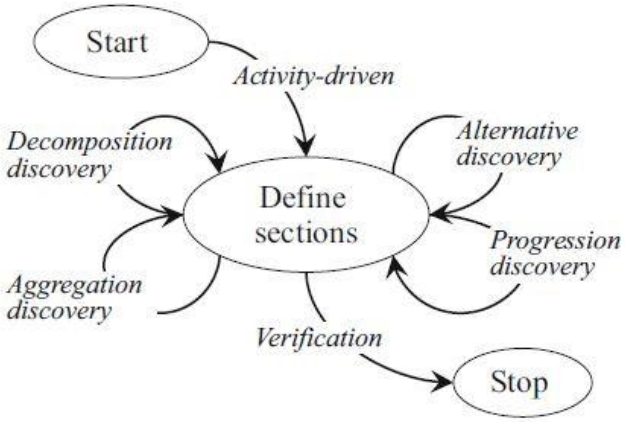


Fig. 6.13 The process-driven requirements elicitation model (after Ralyté, figure 2, 2002) (With kind permission of Springer Science + Business Media)

To elicit requirements for business process modelling, we would refer a requirement elicitation model as shown in figure 6.13. The figure shows only one main intention i.e. "Define Sections". To achieve these sections, only two strategies are relevant for our application domain:

1. **Activity driven strategy**: This strategy says what are different engineering activities that are going to be provided by a constructed method.
2. **Decomposition strategy**: This strategy is just a revision of different sections generated by activity driven strategy, to avoid duplicate sections and find any possible alternative to postulated sections.

We will have to keep in mind that the generated sections will contribute to the construction of a requirement map. For requirement gathering for a BPM, we would consider example situations from a standard BPM such as "Business model canvas (Alexander Ostwalder, Yves Pigneur)".

Using the activity driven strategy and from the viewpoint of business model canvas, we could find four sections as follows:

1. Section (Start, show view of organization, Read existing organization from database)
   Section (Start, show view of organization, Create new organization)
   Section (Start, show view of organization, Update an organization)

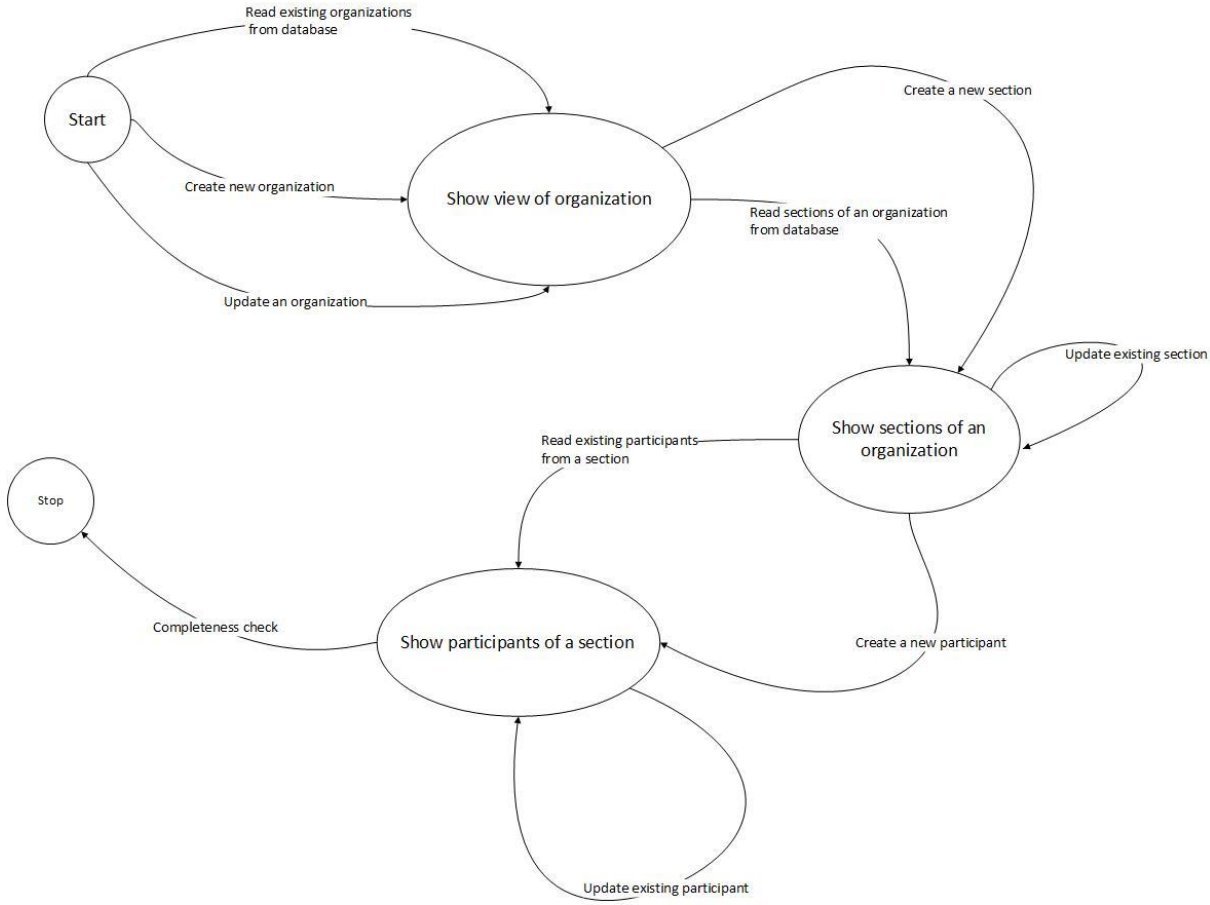
2. Section (Show view of organization, show sections of an organization, Read sections of an organization)
   Section (Show view of organization, show sections of an organization, Update existing section)
   Section (Show view of organization, show sections of an organization, Create new section)

3. Section (Show sections of an organization, show participants of a section, Read existing participants of a section)
   Section (Show sections of an organization, show participants of a section, Create new participant)

Show (Show sections of an organization, show participants of a section, Update existing participant)

4. Section (Show participants of a section, Stop, Completeness check)

Requirement map of business process modelling



Above figure shows a requirement map constructed with sections derived from “Activity-driven” strategy. People associated with this thesis topic can refine the map using “Decomposition strategy”. Since we have a concrete idea about the requirements of a BPM application, we can use the guidelines provided for the map fig 5.4 to construct several fragments as shown in the table below. Based on the requirements of a BPM, we could define eight fragments as below:

1. AbstractController (Work-unit)
2. AbstractDatabaseManager (Work-unit)
3. AbstractRequest (Work-product)
4. BaseTemplate (Work-product)
5. OrganizationTemplate (Work-product)
6. DepartmentTemplate (Work-product)

Intention selection guideline (ISG)	**Relevant preconditions for BPM**	**Result (Guided by IAG and SSG table of the SME book)**
ISG0: Start	None	-
ISG1: Identify method fragment	Only if there are some requirements which needs to be addressed.	*Requirements* 1. Template to show organizations e.g. baseTemplate 2. Template to show sections of an organization e.g. OrganizationTemplate 3. Template to show contents of a section/department e.g. departmentTemplate 4. A database manager for CRUD operations 5. A request class to read user requests 6. A controller class to handle all backend operations and render a template. 7. Functionality for file upload. 8. Functionality to send mail.
ISG2: Define method fragment	Only if method fragment candidates have been identified.	1- **baseTemplate**: This is a work-product with variable placeholders to render view for organizations. This will be used by controller which will fill the placeholders with data from database. 2- **organizationTemplate**: This is same as baseTemplate. The only difference is that a controller will use this template to show information about different departments of an organization. 3- **departmentTemplate**: Same as both baseTemplate and organizationTemplate except it will show information about various participants of a department/section of an organization. 4- **abstractDatabaseManager**: As the name suggest, this a work-unit fragment and it will be responsible for CRUD operation. This fragment will indirectly use Request object (through a controller) to provide deliverable. 5- **abstractRequest**: This is a work product. It will contain user request which will be intercepted by a controller object to be passed to other fragments. 6- **abstractController**: This fragment will act as a middleman between database and view rendering. Controller can also be used to send emails using a mailer fragment or upload files using a FileUploader fragment. This is a work-unit which is dependent on other work-units to provide an output work-product e.g. baseTemplate or organizationTemplate or departmentTemplate. 7- **abstractUploader**: This is a work-unit responsible for storing files in the server. This fragment will take an input work-product (e.g. File) from a user request intercepted through a controller. As an output work product, an uploader will serialize the file name and return the address of the stored file. 8- AbstractMailer: This is a work-unit responsible for sending emails. A controller fragment will extract user-information from an input work-product e.g. user request and pass it to a mailer object to send an email.

ISG3: Validation	When method fragments have been defined.	Quality check of the method fragments. Each fragment must conform to at least a process-part or product-part and must be associated with a required deliverable mentioned in the requirements sections.
ISG4: Stop	When method fragments have been defined.	-