### Process Overview

***Process:***  *title*

|  |
| --- |
| **Process Description** |
| This is a software product line re-engineering process. The main goal of this process is to extract features from a set/family of individual products. |
| **Owner** |
| Luciano Marchezan |
| **Actors/Roles** |
| * Domain Engineer: possesses valid knowledge related to an area of human endeavour, an autonomous computer activity, or other specialized discipline. * Architect: has valid knowledge related to software architecture, design and architectural patterns * Analyst: requirements specialist. Must know functional and nonfunctional requirements. * Developer: possesses programming languages and technologies knowledge. * Feature Tester: must know domain glossary, domain constraints, and requirements information. * Feature Retriever: must have knowledge about feature extraction strategies and techniques. |
| **I/O Artifacts**  For each activity/subprocess the inputs are classified as: mandatory (required), optional (can be used or not) or alternative (at least one must be used). Outputs are always mandatory |
| Input Artifacts:  Product Artifacts:   * Source Code * Requirements   + Requirements List   + Use Cases   + Business Rules * Design Models   + Class Diagram   + State Machines Diagrams   + Feature Models   + Activity diagrams * Domain Information   + High level descriptions   + Abstract Class Diagrams   + Reference Architecture   + Reference Requirements   Output Artifacts:   * Feature Artifacts * Reports |
| **Utilized Resources** |
| * Techniques   + Static Analysis     - Clustering     - Heuristics     - Overlaps     - Structural Similarity     - Model Transformation     - Dependency Analysis     - Rule-Based Techniques     - Data Flow Analysis   + Information Retrieval     - Formal Concept Analysis     - Latent Semantic Indexing     - Latent Semantic Analysis     - Vector Space Model   + Expert Driven |
| **Execution Time** |
| N/A |

### 

### Expanded Subprocess

|  |
| --- |
| **Title:** Planning |
| **Description** |
| During this subprocess team information is gathered. This information will be used during **Documentation Analysis** to make choices about artifacts collection. |
| **Activity Title:** Collect Team Information |
| **Description** |
| Information about the team which will execute the process is collected. This information includes experiences, skills, knowledge and preferences of each member. The information is registered in a document, and stored in a database. This document must have one section for each interviewed member and must contain identification information such as: Name, email, company role, etc. |
| **Actor** |
| Any |
| **Optional Inputs** |
| Project Plan  Business organization chart |
| **Outputs** |
| Team Information Report |
| **Activity Title:** Define Roles and Tasks |
| **Description** |
| Roles and Tasks are defined based on the information collected on the previous activity. The possible roles are: Domain Engineer, Architect, Analyst, Developer, Feature Tester and Feature Retriever; All those roles must be assigned to at least one team members and a team member can have more than one role. However, a person cannot be Feature Retriever and Feature Tester. Also, a role can be performed by more than one person. |
| **Actor** |
| All |
| **Mandatory Inputs** |
| Team Information Report |
| **Optional Inputs** |
| Project Plan  Business organization chart |
| **Outputs** |
| Team Information Report Updated |

|  |
| --- |
| **Title:** Detection |
| **Description** |
| During this subprocess the features are extracted, categorized and grouped. The extraction is made based on the techniques which are also chosen during this subprocess. |
| **Activity Title:** Feature Search |
| **Description** |
| During this subprocess, techniques and strategies are applied to extract features. This subprocess will be more explained below |
| **Actor** |
| Feature Retriever |
| **Alternative Inputs** |
| Source Code  Requirements List  Use Cases  Business Rules  Class Diagram  State Machines Diagrams  Feature Models  Activity diagrams  High level descriptions  Abstract Class Diagrams  Reference Architecture  Reference Requirements |
| **Outputs** |
| Feature Artifacts  Modified Product Artifacts  Variability Reports |
| **Activity Title:** Categorize Features |
| **Description** |
| Once the features were retrieve, they must be categorize. The categories can be:mandatory or optional. Optional can also be divided in: alternative (XOR) and alternative (OR). However the use of alternative (OR) is optional. |
| **Actor** |
| Feature Retriever |
| **Mandatory Inputs** |
| Feature Artifacts  Modified Product Artifacts |
| **Optional Inputs** |
| Source Code  Requirements List  Use Cases  Business Rules  Class Diagram  State Machines Diagrams  Feature Models  Activity diagrams  High level descriptions  Abstract Class Diagrams  Reference Architecture  Reference Requirements |
| **Outputs** |
| Feature Artifacts Updated |
| **Activity Title:** Group Features |
| **Description** |
| Feature are grouped based on their dependencies in product artifacts. Feature model is not created yet, but similar artifacts will be. The artifacts type will depend on techniques selected during **Feature Search**. |
| **Actor** |
| Feature Retriever |
| **Mandatory Inputs** |
| Feature Artifacts  Modified Product Artifacts |
| **Optional Inputs** |
| Source Code  Requirements List  Use Cases  Business Rules  Class Diagram  State Machines Diagrams  Feature Models  Activity diagrams  High level descriptions  Abstract Class Diagrams  Reference Architecture  Reference Requirements |
| **Outputs** |
| Feature Artifacts Updated |
| **Activity Title:** Check Feature Artifacts |
| **Description** |
| Feature artifacts are checked to find missing features or problems with some features. This problems may include: feature names, wrong categories, wrong grouping choices. After the end of this step two results are possible: no problems found or problems are found. If problems are found, they can have two different types:features missing or problem with features. If no problems are found, this subprocess is finished. |
| **Actor** |
| Feature Checker |
| **Mandatory Inputs** |
| Feature Artifacts |
| **Alternative Inputs** |
| Source Code  Requirements List  Use Cases  Business Rules  Class Diagram  State Machines Diagrams  Feature Models  Activity diagrams  High level descriptions  Abstract Class Diagrams  Reference Architecture  Reference Requirements |
| **Outputs** |
| Feature Artifacts Updated |
| **Activity Title:** Edit Features |
| **Description** |
| This activity is performed if problems are found and they are related with features names, categories or grouping choices. Here, features are edited to fix those problems. After the end of this phase **Check Feature Artifacts** is performed again. |
| **Actor** |
| Feature Retriever |
| **Inputs** |
| Feature Artifacts |
| **Optional Inputs** |
| Source Code  Requirements List  Use Cases  Business Rules  Class Diagram  State Machines Diagrams  Feature Models  Activity diagrams  High level descriptions  Abstract Class Diagrams  Reference Architecture  Reference Requirements |
| **Outputs** |
| Feature Artifacts updated |
| **Activity Title:** Create Features |
| **Description** |
| This activity is performed if problems are found and they are related with missing features. New features will be created manually. This new features will be categorized and grouped. After the end of this phase **Check Feature Artifacts** is performed again. |
| **Actor** |
| Feature Retriever |
| **Mandatory Inputs** |
| Feature Artifacts |
| **Optional Inputs** |
| Source Code  Requirements List  Use Cases  Business Rules  Class Diagram  State Machines Diagrams  Feature Models  Activity diagrams  High level descriptions  Abstract Class Diagrams  Reference Architecture  Reference Requirements |
| **Outputs** |
| Feature Artifacts updated |

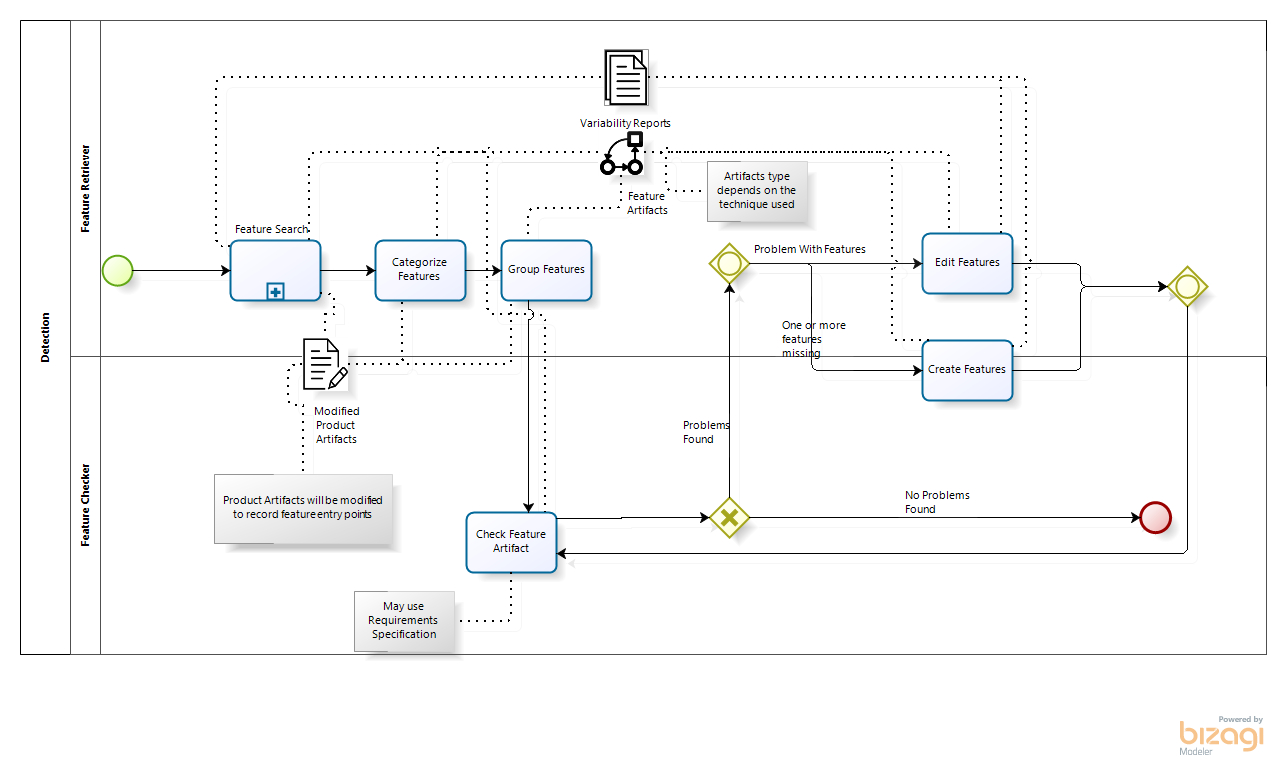
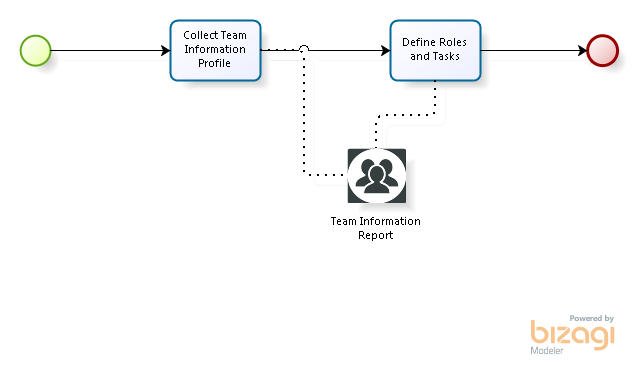
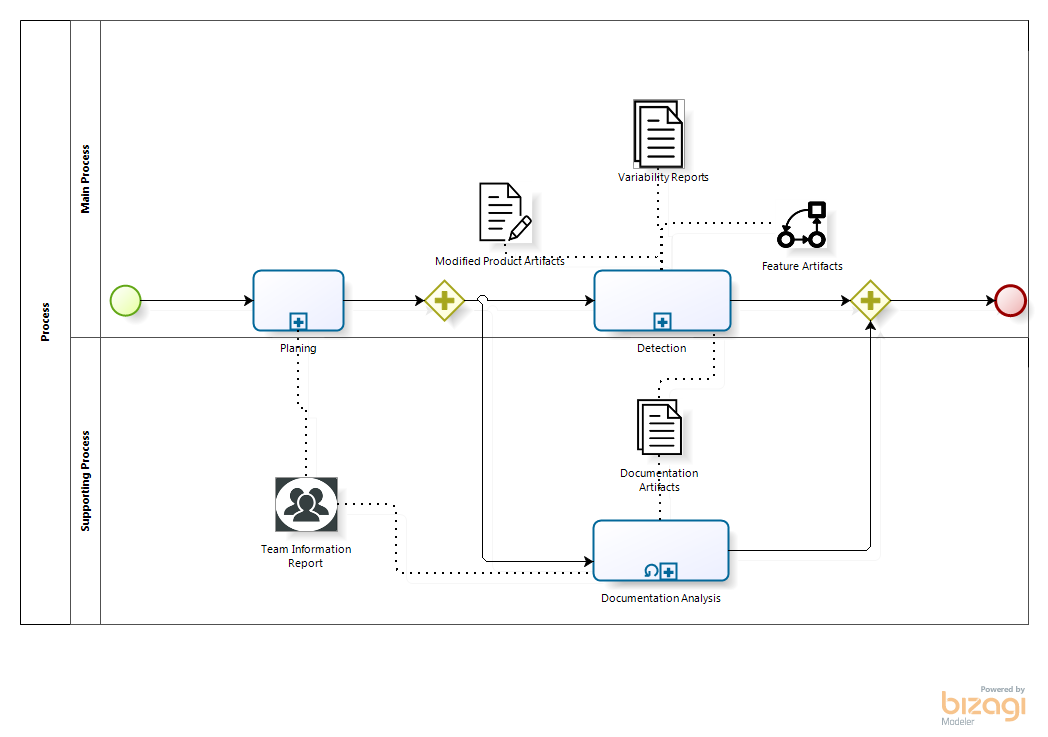
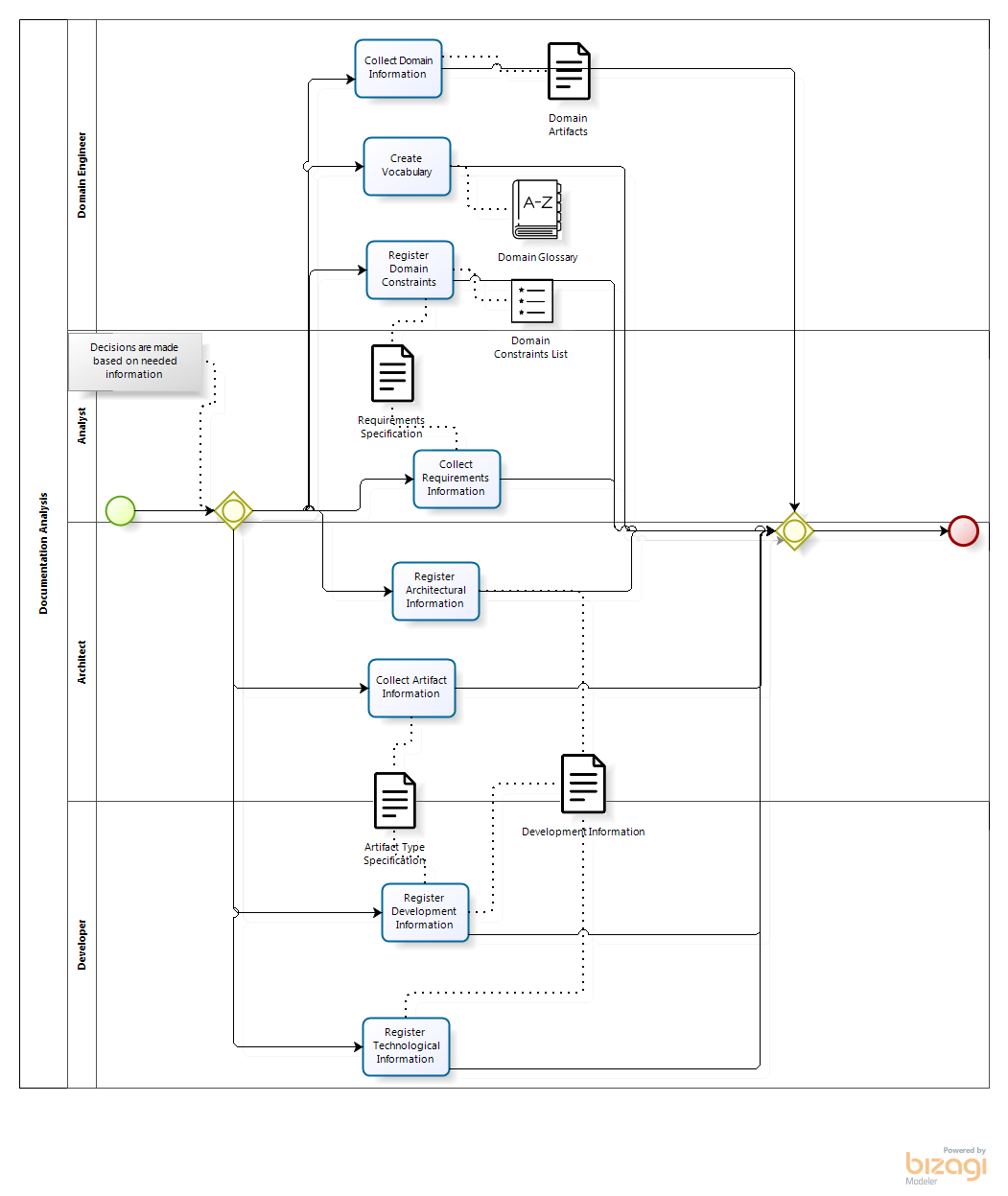
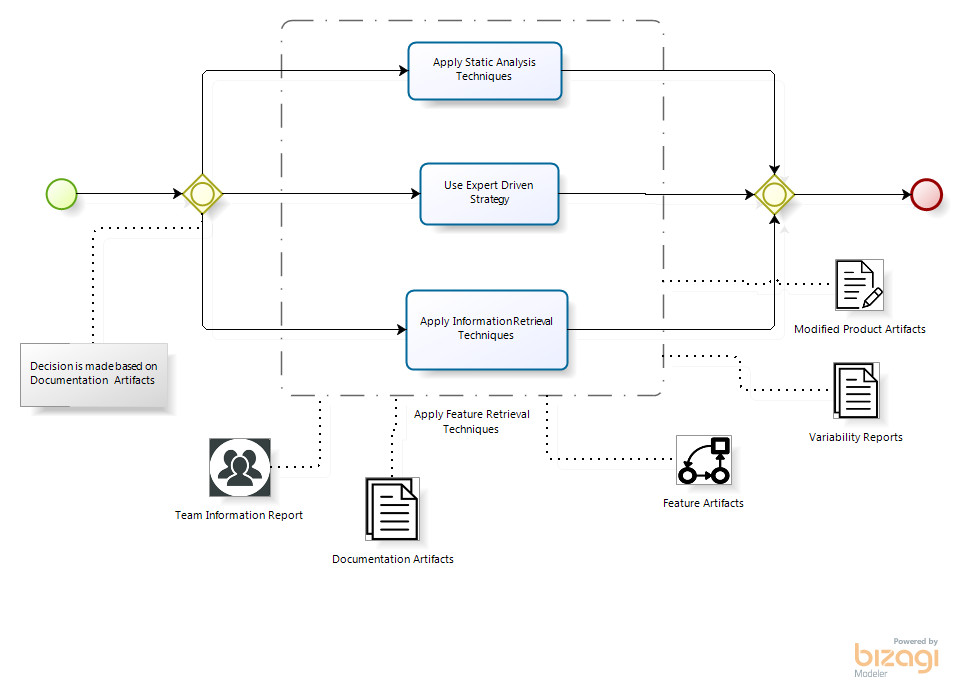
|  |
| --- |
| **Title:** Feature Search |
| **Description** |
| This is a subprocess of **Detection**.  During this subprocess strategies and techniques are chosen and applied in product artifacts to find and extract features. |
| **Activity Title:** Apply Static Analysis Techniques (Apply Feature Retrieval Techniques Group) |
| **Description** |
| This activity is chosen or not based on Artifacts from **Documentation Analysis** and **Planning**. Here static analysis techniques are selected. The selection is based on team member experiences, product and domain artifacts. According to [[1]](#_ltbr30r4srbs), possible techniques are: Clustering, Heuristics, Overlaps, Structural Similarity, Model Transformation, Dependency Analysis, Rule-Based Techniques and Data Flow Analysis. At least one technique must be selected. The applying process will depend on techniques selected. More information can be seen in [Section 3](#_dt1ozxxi0ndi). |
| **Actor** |
| Feature Retriever |
| **Alternative Inputs** |
| Team Information Report  Domain Artifacts  Domain Glossary  Domain Constraints List  Requirements Specification  Artifact Type Specification  Development Information |
| **Outputs** |
| Feature Artifacts  Variability Reports  Modified Product Artifacts |
| **Activity Title:** Use Expert Driven Strategy (Apply Feature Retrieval Techniques Group) |
| **Description** |
| This activity is chosen or not based on Artifacts from **Documentation Analysis**, team experience and domain knowledge. Here, the domain experts are selected. Later, they will analysed product artifacts to identify and extract features. This particular strategy is used along other techniques. |
| **Actor** |
| Feature Retriever |
| **Alternative Inputs** |
| Domain Artifacts  Domain Glossary  Domain Constraints List  Requirements Specification  Artifact Type Specification  Development Information |
| **Outputs** |
| Feature Artifacts  Variability Reports  Modified Product Artifacts |
| **Activity Title:** Apply Information Retrieval Techniques (Apply Feature Retrieval Techniques Group) |
| **Description** |
| This activity is chosen or not based on Artifacts from **Documentation Analysis**. Here, information retrieval techniques are selected. The selection is based on team member experiences, product artifacts and domain artifacts. According to [[1]](#_ltbr30r4srbs), possible techniques are: Formal Concept Analysis, Latent Semantic Indexing, Latent Semantic Analysis and Vector Space Model. At least one technique must be selected. The applying process will depend on techniques selected. More information can be seen in [Section 3](#_dt1ozxxi0ndi). |
| **Actor** |
| Feature Retriever |
| **Alternative Inputs** |
| Domain Artifacts  Domain Glossary  Domain Constraints List  Requirements Specification  Artifact Type Specification  Development Information |
| **Outputs** |
| Feature Artifacts  Variability Reports  Modified Product Artifacts |

|  |
| --- |
| **Title:** Documentation Analysis |
| **Description** |
| During this phase the re-engineering documentation is collected and compiled. All the activities of this subprocess are optional and they can be performed at the same time of any activity from **Detection.** |
| **Activity Title:** Collect Domain Information |
| **Description** |
| This is an optional activity and will or will not be performed based on the need of its outputs artifacts. During this activity, domain information is collected and registered. This information can be used as an input for some extraction techniques. According to [[1]](#_x34c6bklay5), domain artifacts may contain informations such as products description, user comments, documentation of systems in specific domain, and domain analysis. |
| **Actor** |
| Domain Engineering |
| **Alternative Inputs** |
| High level descriptions  Abstract Class Diagrams  Reference Architecture  Reference Requirements |
| **Outputs** |
| Domain Artifacts |
| **Activity Title:** Create Vocabulary |
| **Description** |
| This is an optional activity and will or will not be performed based on the need of its outputs artifacts. During this activity a domain glossary is created. This domain glossary will contain names, terms, synonyms and any kind of terminology specific for the system domain. |
| **Actor** |
| Domain Engineering |
| **Alternative Inputs** |
| High level descriptions  Abstract Class Diagrams  Reference Architecture  Reference Requirements |
| **Outputs** |
| Domain Glossary |
| **Activity Title:** Register Domain Constraints |
| **Description** |
| This is an optional activity and will or will not be performed based on the need of its outputs artifacts. During this activity a list of constraints related to the system domain. This constraints can be collected in the system business rules or even non-functional requirements. |
| **Actor** |
| Domain Engineering |
| **Alternative Inputs** |
| High level descriptions  Abstract Class Diagrams  Reference Architecture  Reference Requirements |
| **Outputs** |
| Domain Constraints List  Requirements Specification |
| **Activity Title:** Collect Requirements Information |
| **Description** |
| This is an optional activity and will or will not be performed based on the need of its outputs artifacts. During this activity the requirement information is collected and registered. Requirements artifacts may be Requirements List, Use Cases, User Stories or any kind of requirements specification. |
| **Actor** |
| Analyst |
| **Alternative Inputs** |
| Requirements List  Use Cases  Business Rules |
| **Outputs** |
| Requirements Specification |
| **Activity Title:** Register Architectural Information |
| **Description** |
| This is an optional activity and will or will not be performed based on the need of its outputs artifacts. During this activity architectural information is collected and registered. This information may include: design patterns, architectural patterns. The artifacts used to register these can be class diagrams, state machine diagrams or even activity diagrams. |
| **Actor** |
| Architect |
| **Alternative Inputs** |
| Class Diagram  State Machines Diagrams  Feature Models  Activity diagrams |
| **Outputs** |
| Development Information |
| **Activity Title:** Collect Artifact Information |
| **Description** |
| This is an optional activity and will or will not be performed based on the need of its outputs artifacts. During this activity the information about artifacts types (extensions, formats, structures, etc) is collected and registered. This information can be used to decide which extraction techniques can be used. |
| **Actor** |
| Architect |
| **Alternative Inputs** |
| Source Code  Requirements List  Use Cases  Business Rules  Class Diagram  State Machines Diagrams  Feature Models  Activity diagrams  High level descriptions  Abstract Class Diagrams  Reference Architecture  Reference Requirements |
| **Outputs** |
| Artifact Type Specification |
| **Activity Title:** Register Development Information |
| **Description** |
| This is an optional activity and will or will not be performed based on the need of its outputs artifacts. During this activity the information about the developed products will be collected and registered. This information may include programming patterns, programming and development paradigms |
| **Actor** |
| Developer |
| **Mandatory Inputs** |
| Source Code |
| **Optional Inputs** |
| Class Diagram  State Machines Diagrams  Activity diagrams |
| **Outputs** |
| Artifact Type Specification  Development Information |
| **Activity Title:** Register Technological Information |
| **Description** |
| This is an optional activity and will or will not be performed based on the need of its outputs artifacts. During this activity information about technologies used in each product are collected and registered. This information can be used to decide which is the best extraction technique or exclude the use of some techniques as well. |
| **Actor** |
| Developer |
| **Optional Inputs** |
| Source Code |
| **Outputs** |
| Development Information |

### Guidelines

1. **Feature Retrieval Techniques** 
   1. **Static Analysis Techniques**
      1. **Clustering:** 
         1. Definition: group features based on their dependencies.
         2. Variations:
            1. Agglomerative Hierarchical Clustering (AHC): a "bottom up" approach where each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
            2. Divisive Hierarchical Clustering (DHC): a "top down" approach where all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.
         3. Inputs: source code
         4. Outputs: feature tree, feature clusters, dendrogram tree.
         5. Examples: [[4]](#_x34c6bklay5) , [[9]](#_x34c6bklay5), [[20]](#_x34c6bklay5)
      2. **Heuristics**
         1. Definition:.
         2. Variations:
            1. Inputs:
            2. Outputs:
         3. Examples:
      3. **Overlaps**
         1. Definition:.
         2. Variations:
            1. Inputs:
            2. Outputs:
         3. Examples:
      4. **Structural Similarity**
         1. Definition:.
         2. Variations:
            1. Inputs:
            2. Outputs:
         3. Examples:
      5. **Model Transformation**
         1. Definition:.
         2. Variations:
            1. Inputs:
            2. Outputs:
         3. Examples:
      6. **Dependency Analysis**
         1. Definition:.
         2. Variations:
            1. Inputs:
            2. Outputs:
         3. Examples:
      7. **Rule-Based Techniques**
         1. Definition:.
         2. Variations:
            1. Inputs:
            2. Outputs:
         3. Examples:
      8. **Data Flow Analysis**
         1. Definition:.
         2. Variations:
            1. Inputs:
            2. Outputs:
         3. Examples:
2. **Guideline 2**
   1. **Guideline 2.1**

### Diagrams

]

### Bibliografy

1. ASSUNCÃO, Wesley KG, LOPEZ-HERREJON, Roberto E, LINSBAUER, Lukas, VERGILIO, Silvia R, EGYED, Alex y ER. "Reengineering legacy applications into software product lines: a systematic mapping". *Empirical Software Engineering*. 2017, p. 1–45.
2. SEIDL, Christoph, WINKELMANN, Tim y SCHAEFER, Ina. *A Software Product Line of Feature Modeling Notations and Cross-Tree Constraint Languages.*. 2016.p. 157–172.

1. HJØRLAND, Birger; ALBRECHTSEN, Hanne. Toward a new horizon in information science: domain‐analysis. Journal of the Association for Information Science and Technology, v. 46, n. 6, p. 400-425, 1995.
2. EYAL-SALMAN, Hamzeh, SERIAI, Abdelhak-Djamel y DONY, Christophe. *Feature location in a collection of product variants: Combining information retrieval and hierarchical clustering*. 2014.p. 426–430.
3. MEFTEH, Mariem, BOUASSIDA, Nadia y BEN-ABDALLAH, Hanкne. "Feature model extraction from documented UML use case diagrams". *ADA USER*. 2014, vol 35, nГєm. 2, p. 107.
4. LAGUNA, Miguel A y CRESPO, Yania. "A systematic mapping study on software product line evolution: From legacy system reengineering to product line refactoring". *Science of Computer Programming*. 2013, vol 78, nГєm. 8, p. 1010–1034.
5. AL-MSIE’DEEN, Abdelhak, SERIAI, Djamel, HUCHARD, Marianne, URTADO, Christelle y VAUTTIER, Sylvain. *Mining features from the object-oriented source code of software variants by combining lexical and structural similarity*. 2013.p. 586–593.
6. AL-MSIE’DEEN, R, SERIAI, A Djamel, HUCHARD, M, URTADO, C, VAUTTIER, S y SALMAN, H Eyal. "An approach to recover feature models from object-oriented source code". *Actes de la Journйe Lignes de Produits*. 2012, p. 15–26.
7. DAMASEVICIUS, Robertas, PASKEVICIUS, Paulius, KARCIAUSKAS, Eimutis y MARCINKEVICIUS, Romas. "Automatic Extraction of Features and Generation of Feature Models from Java Programs". *Information Technology And Control*. 2012, vol 41, nГєm. 4, p. 376–384.
8. EYAL-SALMAN, Hamzeh, SERIAI, Abdelhak-Djamel, DONY, Christophe y AL-MSIE'DEEN, Ra'fat. *Recovering traceability links between feature models and source code of product variants*. 2012.p. 21–25.
9. OL, Andrzej y JØRGENSEN, Bo Nørregaard. "Remodularizing Java programs for improved locality of feature implementations in source code". *Science of Computer Programming*. 2012, vol 77, nГєm. 3, p. 131–151.

1. CZARNECKI, Krzysztof, HELSEN, Simon y EISENECKER, Ulrich. "Formalizing cardinality-based feature models and their specialization". *Software process: Improvement and practice*. 2005, vol 10, nГєm. 1, p. 7–29.

1. POHL, Klaus, BЦCKLE, Günter y VAN DER LINDEN, Frank J. *Software product line engineering: foundations, principles and techniques*. Springer Science & Business Media, 2005.

1. CLEMENTS, Paul y NORTHROP, Linda. *Software product lines*. Addison-Wesley,, 2002.

1. VAN GURP, Jilles, BOSCH, Jan y SVAHNBERG, Mikael. *On the notion of variability in software product lines*. 2001.p. 45–54.

1. CZARNECKI, Krzysztof. "Generative programming: Principles and techniques of software engineering based on automated configuration and fragment-based component models". 1998,

1. CHIKOFSKY, Elliot J. y CROSS, James H. "Reverse engineering and design recovery: A taxonomy". *IEEE software*. 1990, vol 7, nГєm. 1, p. 13–17.

1. DAVIS, Stanley M. "From “future perfect”: Mass customizing". *Planning review*. 1989, vol 17, nГєm. 2, p. 16–21.
2. ROKACH, Lior; MAIMON, Oded. Clustering methods. In: Data mining and knowledge discovery handbook. Springer US, 2005. p. 321-352.
3. WESTON, Nathan; CHITCHYAN, Ruzanna; RASHID, Awais. A framework for constructing semantically composable feature models from natural language requirements. In: Proceedings of the 13th International Software Product Line Conference. Carnegie Mellon University, 2009. p. 211-220.