Software Architecture Analysis Method (SAAM)

Lecture 7A

•This set of slides are provided for the information on the case study of applying Software Architecture Analysis Method (SAAM) to the evaluation of architectural designs of a software that extract keyword frequency vectors from text files.

•The information about architectural designs of the software can be found in the note “Using Architectural Styles in Design”.

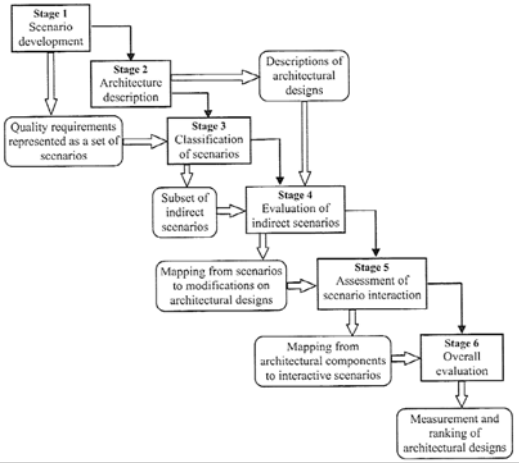
Where can SAAM apply?

- **Software Architecture Analysis Method (SAAM)** can be applied to two different analysis and evaluation tasks:
  - To compare two or more candidate design to see which one satisfies its quality requirement better
  - To evaluate a single design to point out the places where that architecture fails to meet its quality requirements and in some case to show obvious alternative designs that would work better.

The Input of SAAM

- The SAAM method require two types of input:
  - A description of the architectural design or a set of design that are under analysis and evaluation.
  - The quality requirements that the system is intended to achieve.

Activities in SAAM Analysis



Stage 1: Development of Scenarios

- •SAAM emphasizes the involvement of peoples who represent various stakeholders in scenario elicitation.
- •The identification of scenarios belongs to requirements engineering, rather than design.
- •The including of scenario elicitation is due to the current practice of requirements engineering not providing the required scenarios that represent quality requirement. If detailed quality requirements are available, scenarios should be identified and described according to the requirements

Stage 2: Description of Candidate Architecture

- •The candidate architecture or architectures should be described in an architectural notation that is well understood by the parties involved in the analysis.
- •These architectural descriptions must indicate the system's computation and data components as well as all relevant connections.
- •Accompanying this description of the architecture is a description of how the system behaves over time, or a more dynamic representation of the architecture. This may take the form of a natural language specification of the overall behavior or some other more formal and structured specification.

Stage 3: Classification of Scenarios

- •There is an important distinction between two types of scenarios.
  - •The system may directly support that scenario, meaning that anticipated use requires no modification to the system for the scenario to be performed.
  - •If a scenario is not directly supported, there must be some changes to the system that we would represent architecturally.

Stage 4: Scenario Evaluation

- •For each indirect scenario, the changes to the architecture that are necessary for it to support the scenario must be listed, and the cost of performing each change must be estimated.
- •A modification to an architecture means that either a new component or connection is introduced or an existing component or connection requires a change in its specification.

Stage 5: Revealing Scenario Interaction

- •When two or more indirect scenarios require changes to a single components of a system, they are said to Interact on that component.
- •Scenario interaction exposes the allocation of functionality to the product's design.
- •The interaction of semantically unrelated scenario explicitly shows which system modules are computing semantically unrelated functions.

Stage 6: Overall Evaluation

- •If architectures are being compared, a weight should be assigned to each Scenario and Scenario Interaction in terms of their relative importance.
- •The weighting should be used to determine an overall ranking of the candidate architectures.

Example: KWIC

- Function Requirement
  - The Keyword In Context (KWIC) index system accepts an ordered sequence of lines of text.
  - Each line is an ordered sequence of words, and each word is an ordered sequence of characters.
  - A line might be 'circularly shifted' by repeatedly removing the first word and appending it at the end of the line.
  - The KWIC index system outputs a listing of all circular shifts of all lines in alphabetical order.

The Problem:

- Input:
  - An ordered sequence of lines of text.
  - Each line is an ordered sequence of words
  - Each word is an ordered sequence of characters.
- Output:
  - Lines are 'circularly shifted' by repeatedly removing The first word and appending it at the end of the line.
  - Outputs a listing of all circular shifts of all lines in alphabetical order.

Example:

- Input: Sequence of lines
  - An introduction to software architecture
  - Key word in context
- Output: Circularly shifted, alphabetically ordered lines
  - An introduction to software architecture
  - Architecture an introduction to software
  - Context key word in
  - In context key word
  - Introduction to software architecture an
  - Key word in context
  - Software architecture an introduction to
  - To software architecture an introduction
  - Word in context key

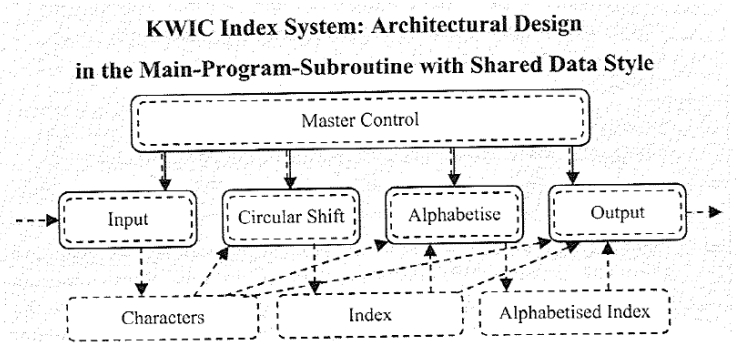
Stage 1: Development of Scenarios

Scenario	Definition
1	To operate in the batch fashion: line shifting is to be performed on all lines after they are read.
2	To operate in the incremental fashion: line shifting is to be performed on each line as it is read from the input device.
3	To eliminate noise words in the shifted lines.
4	To change the internal representation of the lines.
5	To change the internal representation of intermediate data structure.
6	To operate in the on demand fashion: line shifting is to be performed when alphabetization requires a new set of shifted lines.
7	To operate interactively on the input: to allow the user to inset and delete lines from the original list of lines.
8	To operate interactively on the output: to allow the user to inset and delete lines from the circular shifted list.

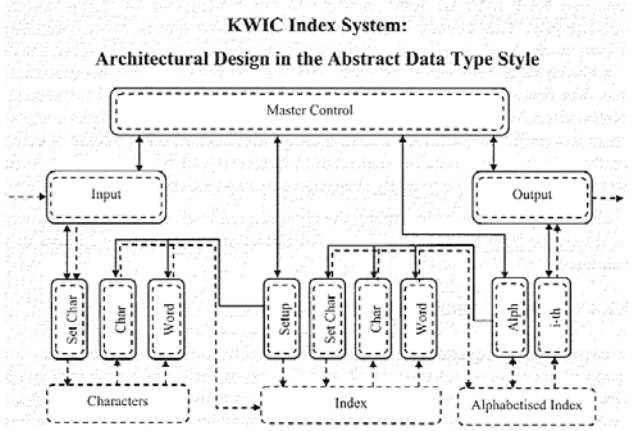
The software that calculates keywords in context should also be efficient in both space and time. Its components should serve as reusable entities. It should be able to support the following changes in the future.

- ***Changes in the processing algorithm:*** For example, line shifting can be performed on each line as it is read from the input device, on all the lines after they are read, or on demand when the alphabetization requires a new set of shifted lines.
- ***Change in data representation:*** For example, lines, words, and characters can be stored in various ways. Similarly, circular shifts can be stored explicitly or implicitly (as parts of index and offset).
- ***Enhancement to system function:*** For example, modify the system to eliminate circular shifts that start with certain noise words (such as a, an, and, etc.), change the system to be inactive, and allow the user to delete lines from the original lists. (or from the circular shifted lists).

Stage 2: Description of Candidate Architecture



Stage 2: Description of Candidate Architecture



Stage 3: Classification of Scenarios

- For this KWIC example, among the right scenarios given above, only Scenario 1 is direct for both of two candidates.
- All others are indirect with respect to the candidates.

Stage 4: Scenario Evaluation

Table 10.1 Evaluation of shared data architecture for KWIC

Scenario			Modification	
No	Description	Type	Component	Change
1	To operate in the batch fashion	Direct		
2	To operate in the incremental fashion	Indirect	Input	To yield control after reading each line
			Master Control	To call subroutines repetitively rather than just once
			Alphabetiser	To use an incremental sorting algorithm
3	To eliminate noise words in shifted lines	Indirect	Circular shift	To eliminate shifted sentences beginning with a noise word
4	To change the internal representation of lines	Indirect	Input	To modify the implementation according to the new data representation
			Circular shift	
			Alphabetiser	
			Output	
5	To change the internal representation of intermediate data structure	Indirect	Circular shift	To modify the implementation according to the new data representation
			Alphabetiser	
			Output	

For the scenario 2, the scenario of operation in an incremental fashion, the following modifications on the shared data architecture must be made to support it.

- ***Modification of the input component:*** The input component must be modified so that after reading each line from the input device, it must pass the control back to the master control component.
- ***Modification of the Master Control component:*** The control component must be modified so that it repetitively calls the subroutines Input, Circular Shift and Alphabetize rather than just once.
- ***Modification of the Alphabetizer component:*** The Alphabetize must now use an incremental sorting algorithm so that circular shifted lines can be added into sorted lines incrementally.

Stage 5: Revealing Scenario Interaction

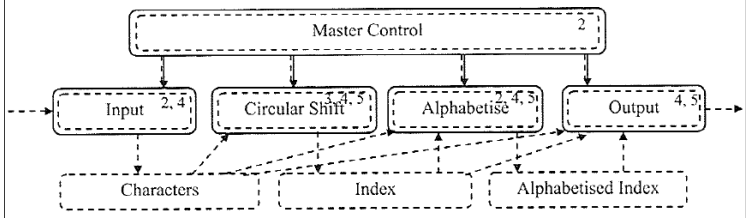


Figure 10.6 Interactions between scenarios on shared data store architecture

Stage 5: Revealing Scenario Interaction

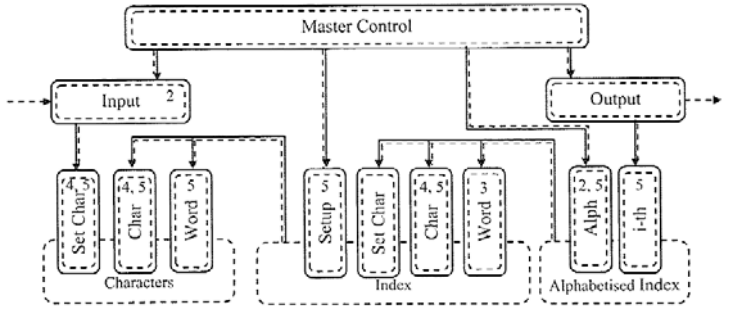


Figure 10.7 Interactions between scenarios on abstract data type architecture

Stage 6: Overall Evaluation

Table 10.2 Evaluation of KWIC architectures

Scenario			Modification Cost / Effort	
No.	Description	Weight	Shared data	Abstract data type
2	To operate in an incremental fashion	45%	3/5	2/12
3	To eliminate noise words in shifted lines	35%	1/5	1/12
4	To change the internal representation of lines	10%	4/5	3/12
5	To change the internal representation of intermediate data structure	10%	3/5	6/12
Overall			48	17.9

Case Study: KFV

- **The Problem**: Keyword Frequency Vector
  - The keyword frequency vector (KFV) of a text file is a sequence of pairs of keywords and their frequency of appearance in the text
    - A good representation of the contents of a text
    - Widely used in information retrieval
    - Can be extracted from texts automatically
    - Small words (such as ‘a’, ‘the’, ‘is’, ‘it’) are removed from the vector
    - The same word of different forms should be treated as one

Example of KFV

- **Input**

The keyword frequency vector of a text file is a sequence of pairs of keywords and their frequency of appearance in the text.
It is a good representation of the contents of the text.
Keyword frequency vectors are widely used in information retrieval.
For example the following is the keyword frequency vector of this paragraph.

- **Output**

Word	Frequency
keyword	4
frequency	4
text	4
vector	4
appearance	1
content	1
example	1
file	1
follow	1
good	1
information	1
pair	1
paragraph	1
representation	1
retrieval	1
sequence	1
use	1
widely	1

The Quality Concerns

- Quality Attributes to be Considered:
  - Modifiability
    - Modifiability with regard to changes in the processing algorithm
    - Modifiability with regard to changes in data representation
    - Modifiability with regard to enhancement to system function
  - Performance
  - Reusability

Modifiability with regard to changes in the processing algorithm:

- To extract KFV incrementally paragraph by paragraph as it is read from the input device
- To extract KFV on the whole text file after they are read
- To extract on demand when the KFV is required

Modifiability with regard to changes in data representation:

- To store text, words and characters in various ways
- To store the KFV explicitly or implicitly

Modifiability with regard to enhancement to system function:

- To treat synonyms as the same word
- To change the systems to be interactive, and allow the user to delete and insert words from the original text

Performance:

- The performance of the system in terms of space used and the time needed to execute the program

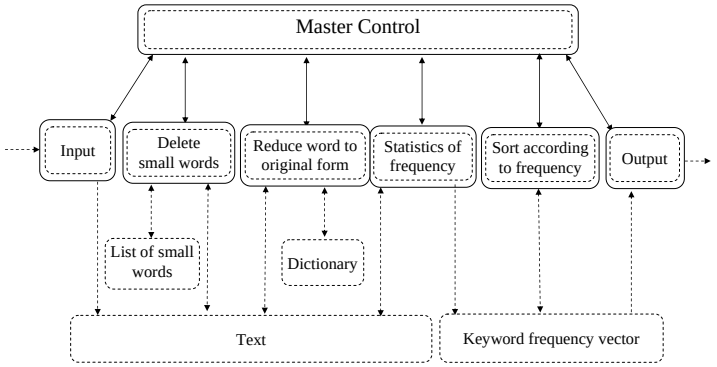
Reusability:

- To what extend can the components are reusable

Definition of Scenarios

Scenario	Definition
1	Extract keyword frequency vector incrementally as a paragraph is read from the input device;
2	Extract keyword frequency vector on the whole text file after they are read
3	Extract keyword frequency vector on demand when the keyword frequency vector is required
4	Change the data representation of text
5	Change the data representation of words
6	Change the data representation of characters
7	Change the data representation of keyword frequency vector
8	Treat synonyms as the same word in the extraction of keyword frequency vectors
9	Delete and insert words from the original text

Design 1: Main Program/Subroutines with Shared Data



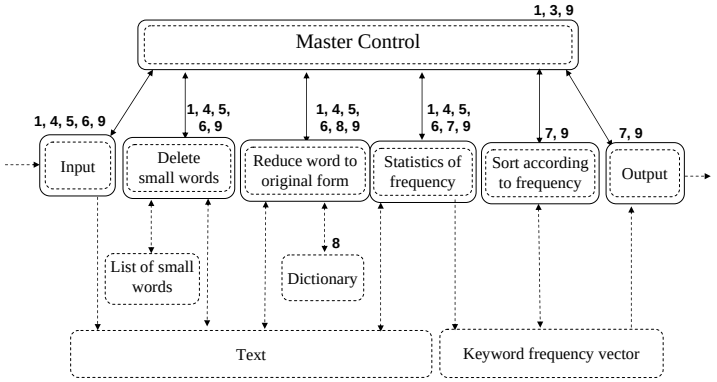
There are 6 major components:

1. Input
   - To get the input text from input device or any other source of information
   - To store the text into internal memory in an appropriate format
   - The design of internal format will be determined by detailed design
2. Delete small words
   - The small words contained in the text are deleted from the text as it is stored in the internal memory
   - It will use a list of small words
3. Reduce word to its original form
   - Each word left in the text are then reduced to its original form
     - 'Architectures' ➔ 'architecture'
     - 'Calculi' ➔'calculus'
     - 'Followed' ➔'follow'
   - A dictionary will be used
4. Statistics of frequency
   - To count the occurrences of a word in the text to generate a sequence of pairs comprising the word and its frequency
   - This sequence of pairs is not necessarily ordered according to the frequency, but may be in the alphabetic order of keywords
   - The result will be stored in another memory storage
5. Sort according to the frequency
   - Sort the sequence of pairs of keywords and their frequencies into an order according to the frequency
6. Output
   - Translate the keyword frequency vector into required output format
   - Output to the device

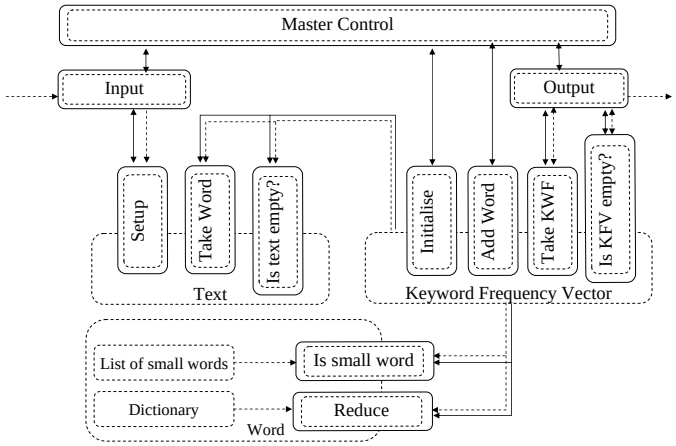
Evaluation of the Main Program/ Subroutines with Shared Data

Scenario		Modification	
No.	**Type**	**Component**	**Change**
1	Indirect	Input	To yield control after read each paragraph
		Delete words	To change the algorithm to be incremental
		Reduce	To change the algorithm to be incremental
		Statistics	To change the algorithm to be incremental
		Master control	To call the subroutines repetitively
2	Direct		
3	Indirect	Master control	To change the condition of call the subroutines
4	Indirect	Input	To change the implementation according to the new data representation of text
		Delete words	
		Reduce	
		Statistics	
5		Input	To change the implementation according to the new data representation of word
		Delete words	
		Reduce	
		Statistics	
6		Input	To change the implementation according to the new data representation of characters
		Delete words	
		Reduce	
		Statistics	
7		Statistics	To change the implementation according to the new data representation of KFV
		Sort KFV	
		Output	
8	Indirect	Reduce / Dictionary	Change the dictionary and/or the reduce algorithm
9	Indirect	All	A major re-development may be necessary.

Reveal Scenario Interaction for Main Program/Subroutines with Shared Data Architecture



Design 2: Abstract Data Type



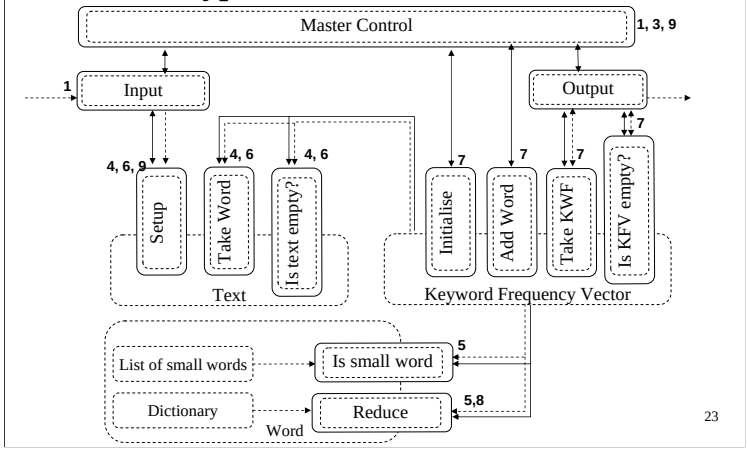
There are three Abstract Data Type in this design:

1. Word ADT
   - Is-small-word:
     - A Boolean function that checks if the parameter is a small word.
     - It returns TRUE if the word is listed in a list of small words, otherwise it returns FALSE.
   - Reduce:
     - A function on words
     - It changes a word to its original form according to a dictionary of words and returns back.
2. Text ADT
   - Setup:
     - get the text from the input component
     - translate the text into an internal format
     - stores the text in its internal data storage
   - Take-word:
     - a function that returns one word in the text and deletes it from its internal data storage.
   - Is-text-empty:
     - A Boolean function that returns TRUE if the internal storage is empty, otherwise returns FALSE when it contains at least one word.
3. Keyword Frequency Vector ADT
   - Initialise:
     - It initialises the internal representation of the vector
   - Add-word: adds a word to the keyword frequency vector
     - Calls is-small-word function of the word ADT
     - If the function returns TRUE, then do nothing
     - ELSE calls the reduce function of the word ADT and searches the keyword frequency vector
     - If the vector already contains the word, then its frequency is added by 1, else the keyword is added into the vector with frequency 1
   - Take-KWF:
     - Return the frequency and keyword of highest frequency
     - Delete the keyword from the vector
   - Is-KFV-empty:
     - A Boolean function that returns TRUE if the vector is empty, otherwise, it returns FALSE

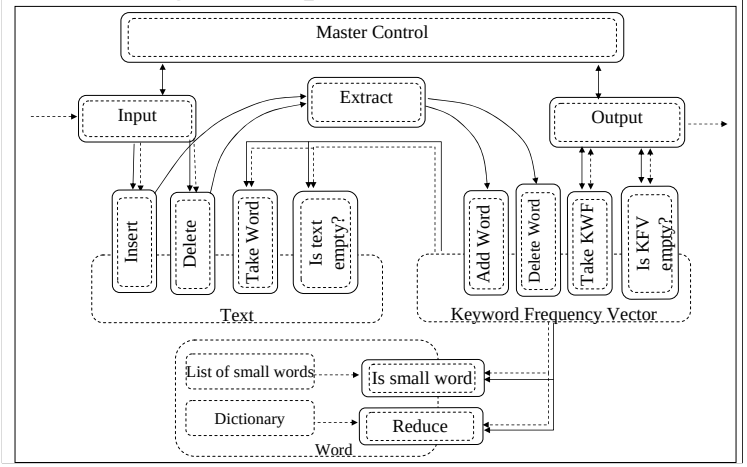
Evaluation of the Abstract Data Type Architecture

Scenario		Modification	
No.	**Type**	**Component**	**Change**
1	Indirect	Input	To yield control after read each paragraph
		Master control	To call the subroutines repetitively
2	Direct		
3	Indirect	Master control	To change the condition of calling subroutines
4	Indirect	Setup	To change the implementation according to the new data representation of text
		Take word	
		Is text empty	
5	Indirect	Is small word	To change the implementation according to the new data representation of word
		Reduce	
6	Indirect	Setup	To change the implementation according to the new data representation of characters
		Take word	
		Is text empty	
7	Indirect	Initialise	To change the implementation according to the new data representation of KFV
		Add word	
		Take KFV	
		Is KFV empty	
8	Indirect	Reduce	Change the reduce algorithm and the dictionary used by the function
9	Indirect	Master control	Add user interface and corresponding control to allow user to insert or delete text
		Insert word	Additional component as a part of text ADT to allow user to insert word into text
		Delete word	Additional component as a part of text ADT to allow user to delete text
		Delete from KFV	Additional component as a part of KFV ADT to update the KFV after delete a word from the text

Reveal Scenario Interaction for Abstract Data Type Architecture



Design 3: Implicit Invocation

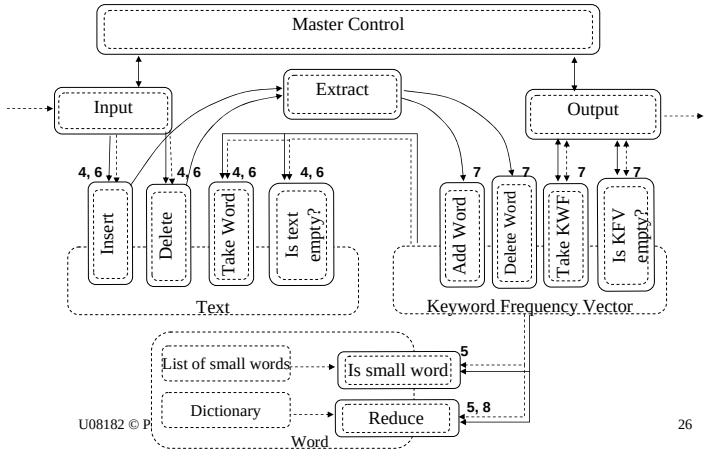


- The implicit invocation architecture also use three abstract data types to access the data abstractly.
- The computations are invoked implicitly when data is modified.
- Each time when the data is modified, an event is generated and the event drives a corresponding event handling function to execute.
- Interactions are based on an active data model.
- The act of inserting or deleting a word from the text will cause the extract component to call the add-word or delete-word operation on the keyword frequency vector, which consequently change the vector's value.
- This allows the system to produce keyword frequency vector interactively and keep the stored vector consistent with the text while the user is editing the text.

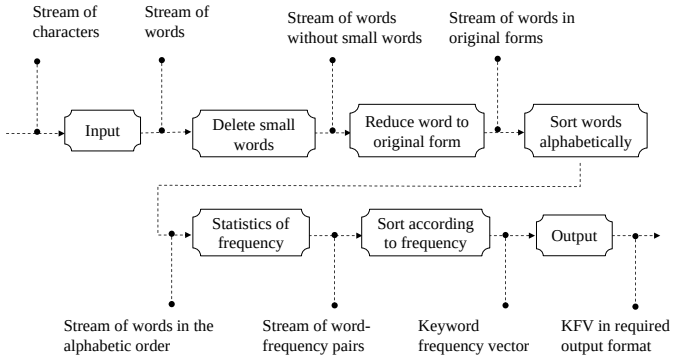
Evaluation of the Implicit Invocation Architecture

Scenario		Modification	
No.	Type	Component	Change
1	Direct		
2	Direct		
3	Direct		
4	Indirect	Insert	To change the implementation according to the new data representation of text
		Delete	
		Take Word	
		Is text empty	
5	Indirect	Is small word	To change the implementation according to the new data representation of word
		Reduce	
6	Indirect	Insert	To change the implementation according to the new data representation of characters
		Delete	
		Take Word	
		Is text empty	
7	Indirect	Add word	To change the implementation according to the new data representation of KFV
		Delete Word	
		Take KFV	
		IS KFV empty	
8	Indirect	Reduce	Change the reduce algorithm and the dictionary used by the function
9	Direct		

Reveal Scenario Interaction for Implicit Invocation Architecture



Design 4: Pipe-and-Filter



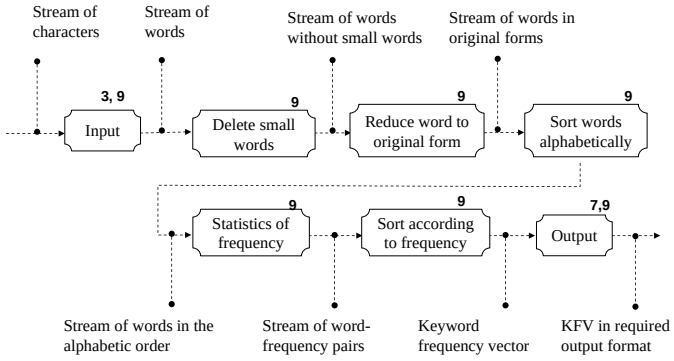
There are 7 major components:

1. Input
   - Takes the stream of characters and breaks it down to a stream of words.
2. Delete small words
   - Removes the small words in the input stream of words
3. Reduce words to original forms
   - Changes each word in the stream of words into their original forms
4. Sort words alphabetically
   - Takes the stream of words and sort it into alphabetical order
5. Count the frequency
   - Count the occurrences of each word in the stream and generates a stream of keyword-frequency pairs
6. Sort vector according to frequency
   - Sort the stream of keyword-frequency pairs according to frequency
7. Output
   - Takes a stream of keyword-frequency pairs that is sorted according to the frequency and generates a keyword frequency vector in the required output format

Evaluation of the Pipe-and-Filter Architecture

Scenario		Modification	
No.	Type	Component	Change
1	Direct		
2	Direct		
3	Indirect	Input/Output filters	To change the condition executing the input filter or output filter
4	Not applicable	1 or 2 components	Change the format of the output streams of a filter will force the down-stream filter to be modified
5	Not applicable	1 or 2 components	Change the format of the output streams of a filter will force the down-stream filter to be modified
6	Not applicable	1 or 2 components	Change the format of the output streams of a filter will force the down-stream filter to be modified
7	Indirect	Output	To change the implementation according to the new data representation of KFV
8	Indirect	Reduce	Change the reduce algorithm and the dictionary used by the function
9	Indirect	The whole system	No easy way the modify the system to support this scenario. Major re-development is required

Reveal Scenario Interaction for Pipe-and-Filter Architecture



Evaluation of KFV Architectures

Scenarios		Architectures			
No.	Weight	Shared data	Abstract data type	Implicit invocation	Pipe-and-filter
1	20	5/7	2/12	0	0
2	5	0	0	0	0
3	15	1/7	1/12	0	1/7
4	5	4/7	3/12	4/14	2/7
5	5	4/7	2/12	2/14	2/7
6	5	4/7	3/12	4/14	2/7
7	10	3/7	4/12	4/14	1/7
8	15	1/7	1/12	1/14	1/7
9	20	7/7	4/12	0	7/7
Overall		51.43	19.15	7.49	30.00

Further Readings

- Zhu, H., Software Design methodology. Chapter 10, pp249-276.
- Bass, L., Clements, P. and Kazman, R., Software Architecture in Practice, Addison Wesley, 1998. Chapter 9: Analyzing Development Qualities at the Architectural Level: The Software Architecture Analysis Method, pp189~1220
- Clements, P., Kazman, R. and Klien, M., Evaluating Software Architectures: Methods and Case Studies, Addison Wesley, 2002.