SFWR ENG 3A04: Software Design II

Dr. Ridha Khedri

Department of Computing and Software, McMaster University
Canada L8S 4L7, Hamilton, Ontario

Term 1, 2008–2009

Acknowledgments: Material based on *Software Architecture Design* by Tao et al. (Chapter 5)

Outline of Part I

1. Principle of Low Coupling and High Cohesion
2. Open-Closed Principle
3. Liskov substitution principle
4. Dependency Inversion Principle
5. Law of Demeter
6. Other Design Principles for Security
7. Questions???

Outline of Part II

Part I

Review of Previous Lecture

Part II

Today's Lecture

Data Flow Architecture Overview

- The data flow software architecture presents a system as a series of transformations on successive sets of data

Data Flow Architecture Overview

- The data flow software architecture presents a system as a series of transformations on successive sets of data
- The software system is decomposed into functional process modules (sub-systems) where data directs and controls the order of data computation processing

Data Flow Architecture Overview

- The data flow software architecture presents a system as a series of transformations on successive sets of data
- The software system is decomposed into functional process modules (sub-systems) where data directs and controls the order of data computation processing
- Each sub-system (module) component in this architecture transforms its input data to its output data

Data Flow Architecture Overview

- The data flow software architecture presents a system as a series of transformations on successive sets of data
- The software system is decomposed into functional process modules (sub-systems) where data directs and controls the order of data computation processing
- Each sub-system (module) component in this architecture transforms its input data to its output data
- The connection between the module components =

Data Flow Architecture Overview

- The data flow software architecture presents a system as a series of transformations on successive sets of data
- The software system is decomposed into functional process modules (sub-systems) where data directs and controls the order of data computation processing
- Each sub-system (module) component in this architecture transforms its input data to its output data
- The connection between the module components =
  - Input-output streams

Data Flow Architecture Overview

- The data flow software architecture presents a system as a series of transformations on successive sets of data
- The software system is decomposed into functional process modules (sub-systems) where data directs and controls the order of data computation processing
- Each sub-system (module) component in this architecture transforms its input data to its output data
- The connection between the module components =
  - Input-output streams
  - Input-output files,

Data Flow Architecture Overview

- The data flow software architecture presents a system as a series of transformations on successive sets of data
- The software system is decomposed into functional process modules (sub-systems) where data directs and controls the order of data computation processing
- Each sub-system (module) component in this architecture transforms its input data to its output data
- The connection between the module components =
  - Input-output streams
  - Input-output files,
  - buffers,

- The data flow software architecture presents a system as a series of transformations on successive sets of data
- The software system is decomposed into functional process modules (sub-systems) where data directs and controls the order of data computation processing
- Each sub-system (module) component in this architecture transforms its input data to its output data
- The connection between the module components =
  - Input-output streams
  - Input-output files,
  - buffers,
  - piped streams

Data Flow Architecture Overview

- The data flow software architecture presents a system as a series of transformations on successive sets of data
- The software system is decomposed into functional process modules (sub-systems) where data directs and controls the order of data computation processing
- Each sub-system (module) component in this architecture transforms its input data to its output data
- The connection between the module components =
  - Input-output streams
  - Input-output files,
  - buffers,
  - piped streams
  - or other type connections

- The focus in the data flow is the data availability

Data Flow Architecture Overview

- The focus in the data flow is the data availability
- The data can flow in a graph topology with cycles (linear, cyclic, tree, etc.)

- The focus in the data flow is the data availability
- The data can flow in a graph topology with cycles (linear, cyclic, tree, etc.)
- There is no interaction between the modules except the data connection between them

Data Flow Architecture Overview

- The focus in the data flow is the data availability
- The data can flow in a graph topology with cycles (linear, cyclic, tree, etc.)
- There is no interaction between the modules except the data connection between them
- The modifiability and reusability are the property attributes of the data flow architecture

- The focus in the data flow is the data availability
- The data can flow in a graph topology with cycles (linear, cyclic, tree, etc.)
- There is no interaction between the modules except the data connection between them
- The modifiability and reusability are the property attributes of the data flow architecture
    - The architectural elements are independent of each other (one element be substituted by another)

- The focus in the data flow is the data availability
- The data can flow in a graph topology with cycles (linear, cyclic, tree, etc.)
- There is no interaction between the modules except the data connection between them
- The modifiability and reusability are the property attributes of the data flow architecture
  - The architectural elements are independent of each other (one element be substituted by another)
  - Each element does not need to know the identity of any other element

There are many different ways to connect the output data of a module to the input of other module

- In terms of execution sequence between modules:

There are many different ways to connect the output data of a module to the input of other module

- In terms of execution sequence between modules:
  - Batch Sequential

Data Flow Architecture Overview

There are many different ways to connect the output data of a module to the input of other module

- In terms of execution sequence between modules:
  - Batch Sequential
  - Pipes are stateless and serve as conduits for moving streams of data between multiple filters

There are many different ways to connect the output data of a module to the input of other module

- In terms of execution sequence between modules:
  - Batch Sequential
  - Pipes are stateless and serve as conduits for moving streams of data between multiple filters
  - Filters are stream modifiers, which process incoming data in some specialized way and send that modified data stream out over a pipe to another filter

Data Flow Architecture Overview

There are many different ways to connect the output data of a module to the input of other module

- In terms of execution sequence between modules:
  - Batch Sequential
  - Pipes are stateless and serve as conduits for moving streams of data between multiple filters
  - Filters are stream modifiers, which process incoming data in some specialized way and send that modified data stream out over a pipe to another filter

- Pipes are special cases of filters; the modification function is the identity

There are many different ways to connect the output data of a module to the input of other module

- In terms of execution sequence between modules:
  - Batch Sequential
  - Pipes are stateless and serve as conduits for moving streams of data between multiple filters
  - Filters are stream modifiers, which process incoming data in some specialized way and send that modified data stream out over a pipe to another filter
- Pipes are special cases of filters; the modification function is the identity
- The close loop process control is another typical data flow architecture style

Data Flow Architecture Overview

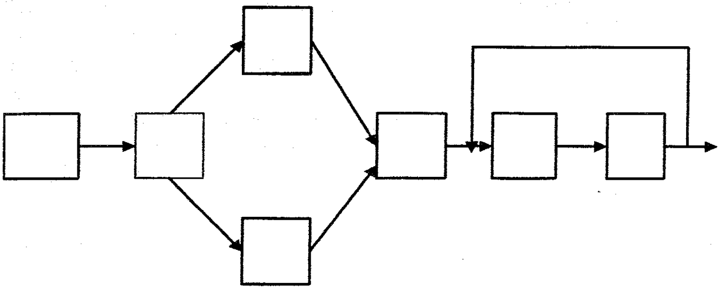


Figure: Block diagram of data flow architecture

- An architectural style that was widely used in 1950' - 1970'

- An architectural style that was widely used in 1950' - 1970'
- RPG (Report Program Generator) and COBOL are two typical programming languages working on this model

- An architectural style that was widely used in 1950' - 1970'
- RPG (Report Program Generator) and COBOL are two typical programming languages working on this model
- Each data transformation element cannot start its process until its previous element completes its computation

- An architectural style that was widely used in 1950' - 1970'
- RPG (Report Program Generator) and COBOL are two typical programming languages working on this model
- Each data transformation element cannot start its process until its previous element completes its computation
- Data flow carries a batch of data as a whole to move from one element to another

Data Flow Architecture Batch Sequential

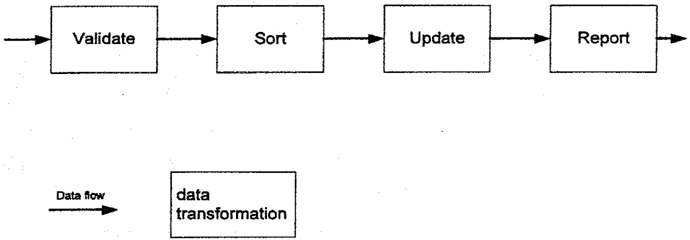


Figure: Batch sequential architecture

Data Flow Architecture Batch Sequential

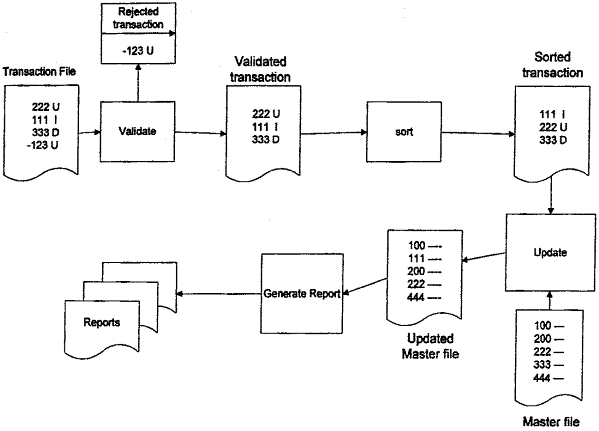


Figure: Batch sequential in business data processing

- Connection links between elements are conducted through temporary files

- Connection links between elements are conducted through temporary files
- Business data processing are typical applications of this architecture

- Connection links between elements are conducted through temporary files
- Business data processing are typical applications of this architecture
- A script is often used to make the batch sequence of the subsystems in the system

Data Flow Architecture Batch Sequential

- Connection links between elements are conducted through temporary files
- Business data processing are typical applications of this architecture
- A script is often used to make the batch sequence of the subsystems in the system

scriptExample.sh

```
ls *.pdf > toto1
grep -ir ".pdf" toto1
 ls *.tex > toto2
cat toto1 toto2 > toto
mail john.do@mcmaster.ca < toto
```

mySHell.sh

```
(exec) searching kwd <inputFile> matchedFile
(exec) counting <matchedFile> countedFile
(exec) sorting <countedFile> myReportfile
```

- We can also implement the batch sequential software architecture at Programming language level

Data Flow Architecture Batch Sequential

```
public class batch_sequential
{
  . . .
  public static void main() {
      searching(kwd, inputFile, matchedFile);
      counting(matchedFile, countedFile);
      sorting (countedFile, reportFile);
  }

  public static void search(String kwd, String
      inFile, String outFile)
  { . . . }

  public static void counting(String inFile,
      String outFile)
  { . . . }

  public static void sorting(String inFile, String
      outFile)
  { . . . }
}
```

Applicable Design Domains:

- Data are batched

Benefits:

Limitation:

Applicable Design Domains:

- Data are batched
- Each sub-system reads related input files and writes output files

Benefits:

Limitation:

Applicable Design Domains:

- Data are batched
- Each sub-system reads related input files and writes output files

Benefits:

- Simple divisions between sub-systems

Limitation:

Applicable Design Domains:

- Data are batched
- Each sub-system reads related input files and writes output files

Benefits:

- Simple divisions between sub-systems
- Each sub-system can be a stand-alone program working on input data and producing output data

Limitation:

Applicable Design Domains:

- Data are batched
- Each sub-system reads related input files and writes output files

Benefits:

- Simple divisions between sub-systems
- Each sub-system can be a stand-alone program working on input data and producing output data

Limitation:

- Requires an external control

Data Flow Architecture Batch Sequential

Applicable Design Domains:

- Data are batched
- Each sub-system reads related input files and writes output files

Benefits:

- Simple divisions between sub-systems
- Each sub-system can be a stand-alone program working on input data and producing output data

Limitation:

- Requires an external control
- Low throughput

Applicable Design Domains:

- Data are batched
- Each sub-system reads related input files and writes output files

Benefits:

- Simple divisions between sub-systems
- Each sub-system can be a stand-alone program working on input data and producing output data

Limitation:

- Requires an external control
- Low throughput
- No interactive interface

- The Pipe and Filter architecture is another type of data flow architecture where the flow is directed by data

- The Pipe and Filter architecture is another type of data flow architecture where the flow is directed by data
- It decomposes the whole system into components of

- The Pipe and Filter architecture is another type of data flow architecture where the flow is directed by data
- It decomposes the whole system into components of
  - data source

- The Pipe and Filter architecture is another type of data flow architecture where the flow is directed by data
- It decomposes the whole system into components of
  - data source
  - filters

Pipe and Filter Architecture

- The Pipe and Filter architecture is another type of data flow architecture where the flow is directed by data
- It decomposes the whole system into components of
  - data source
  - filters
  - pipes

- The Pipe and Filter architecture is another type of data flow architecture where the flow is directed by data
- It decomposes the whole system into components of
  - data source
  - filters
  - pipes
  - data sink

Data Flow Architecture Pipe and Filter Architecture

- The Pipe and Filter architecture is another type of data flow architecture where the flow is directed by data
- It decomposes the whole system into components of
  - data source
  - filters
  - pipes
  - data sink
- The connections between components are data streams

- The Pipe and Filter architecture is another type of data flow architecture where the flow is directed by data
- It decomposes the whole system into components of
  - data source
  - filters
  - pipes
  - data sink
- The connections between components are data streams
- A data stream is a first-in-first-out buffer type data structure

- The Pipe and Filter architecture is another type of data flow architecture where the flow is directed by data
- It decomposes the whole system into components of
  - data source
  - filters
  - pipes
  - data sink
- The connections between components are data streams
- A data stream is a first-in-first-out buffer type data structure
- Almost all operating system and programming languages provide data stream mechanism

- A filter is an independent data stream transformer which

- A filter is an independent data stream transformer which
  - reads data from its input data stream

- A filter is an independent data stream transformer which
  - reads data from its input data stream
  - transforms and processes it

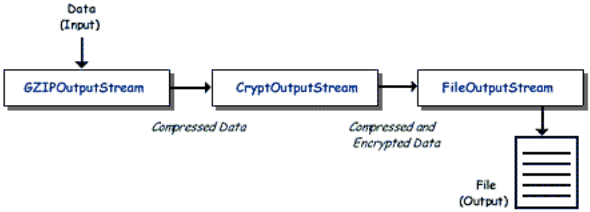
- A filter is an independent data stream transformer which
  - reads data from its input data stream
  - transforms and processes it
  - then writes the transformed data stream over a pipe to next filter

- A filter is an independent data stream transformer which
  - reads data from its input data stream
  - transforms and processes it
  - then writes the transformed data stream over a pipe to next filter
- A filter does not need to wait for batched data as a whole (can start working as soon as the data arrives)

- A filter is an independent data stream transformer which
  - reads data from its input data stream
  - transforms and processes it
  - then writes the transformed data stream over a pipe to next filter
- A filter does not need to wait for batched data as a whole (can start working as soon as the data arrives)
- A filter does not even know the identity of data upstream or data downstream

- A filter is an independent data stream transformer which
  - reads data from its input data stream
  - transforms and processes it
  - then writes the transformed data stream over a pipe to next filter
- A filter does not need to wait for batched data as a whole (can start working as soon as the data arrives)
- A filter does not even know the identity of data upstream or data downstream
- A filter is just working in a local incremental mode

Example of Pipe and Filter Architecture



Data Flow Architecture Pipe and Filter Architecture

- A pipe is a stateless conduit that moves data stream from one filter to another filter

- A pipe is a stateless conduit that moves data stream from one filter to another filter
- A pipe can carry binary or character stream

- A pipe is a stateless conduit that moves data stream from one filter to another filter
- A pipe can carry binary or character stream
- An object type data must be serialized to be able to go over a stream

- A pipe is a stateless conduit that moves data stream from one filter to another filter
- A pipe can carry binary or character stream
- An object type data must be serialized to be able to go over a stream
- Serialization is the process of saving an object onto a storage medium or to transmit it across a network connection link in binary form

- When the resulting series of bytes is reread according to the serialization format, it can be used to create an accurate clone of the original object

- When the resulting series of bytes is reread according to the serialization format, it can be used to create an accurate clone of the original object

- This process of serializing an object is also called deflating or marshalling an object

- When the resulting series of bytes is reread according to the serialization format, it can be used to create an accurate clone of the original object
- This process of serializing an object is also called deflating or marshalling an object
- The opposite operation, extracting a data structure from a series of bytes, is deserialization or unmarshalling

Serializing a structure (C language)

```
 #include "tpl.h"
struct ms_t {
    int i;
    char c[3];
    double f;
};
int main() {
    tpl_node *tn;
    struct ms_t ms = {1, {'a','b','c'}, 3.14};
    tn = tpl_map( "S(ic#f)", &ms, 3);
    tpl_pack( tn, 0 );
    tpl_dump( tn, TPL_FILE, "struct.tpl" );
    tpl_free( tn );
}
```

Deserializing a structure (C language)

```
#include "tpl.h"
struct ms_t {
    int i;
    char c[3];
    double f;
};
int main() {
    tpl_node *tn;
    struct ms_t ms;

    tn = tpl_map( "S(*)", &ms);
    tpl_load( tn, TPL_FILE, "struct.tpl" );
    tpl_unpack( tn, 0 );
    tpl_free( tn );
}
```

There are 3 ways to make the data be flowed:

- Push only (Write only)

There are 3 ways to make the data be flowed:

- Push only (Write only)
  - A data source may push data in a downstream

There are 3 ways to make the data be flowed:

- Push only (Write only)
  - A data source may push data in a downstream
  - A filter may push data in a downstream

There are 3 ways to make the data be flowed:

- Push only (Write only)
  - A data source may push data in a downstream
  - A filter may push data in a downstream
- Pull only (Read only)

There are 3 ways to make the data be flowed:

- Push only (Write only)
  - A data source may push data in a downstream
  - A filter may push data in a downstream
- Pull only (Read only)
  - A data sink may pull data from an upstream

There are 3 ways to make the data be flowed:

- Push only (Write only)
  - A data source may push data in a downstream
  - A filter may push data in a downstream
- Pull only (Read only)
  - A data sink may pull data from an upstream
  - A filter may pull data from an upstream

There are 3 ways to make the data be flowed:

- Push only (Write only)
  - A data source may push data in a downstream
  - A filter may push data in a downstream
- Pull only (Read only)
  - A data sink may pull data from an upstream
  - A filter may pull data from an upstream
- Pull/Push (ReadlWrite)

There are 3 ways to make the data be flowed:

- Push only (Write only)
  - A data source may push data in a downstream
  - A filter may push data in a downstream
- Pull only (Read only)
  - A data sink may pull data from an upstream
  - A filter may pull data from an upstream
- Pull/Push (ReadlWrite)
  - A filter may pull data from an upstream and push transformed data in a downstream

There are 2 type filters:

- Active filter

There are 2 type filters:

- Active filter
  - It pulls in data and push out the transformed data (pull/push)

There are 2 type filters:

- Active filter
    - It pulls in data and push out the transformed data (pull/push)
    - It works with a passive pipe which provides read/write mechanisms for pulling and pushing

There are 2 type filters:

- Active filter
  - It pulls in data and push out the transformed data (pull/push)
  - It works with a passive pipe which provides read/write mechanisms for pulling and pushing
- Passive filter

There are 2 type filters:

- Active filter
  - It pulls in data and push out the transformed data (pull/push)
  - It works with a passive pipe which provides read/write mechanisms for pulling and pushing
- Passive filter
  - It lets connected pipe to push data in and pull data out

There are 2 type filters:

- Active filter
  - It pulls in data and push out the transformed data (pull/push)
  - It works with a passive pipe which provides read/write mechanisms for pulling and pushing
- Passive filter
  - It lets connected pipe to push data in and pull data out
  - It works with active pipes that pull data out from a filter and push data into next filter

Data Flow Architecture Pipe and Filter Architecture

solid lines indicate the class connections; dash lines indicate an alternative configuration for pipes

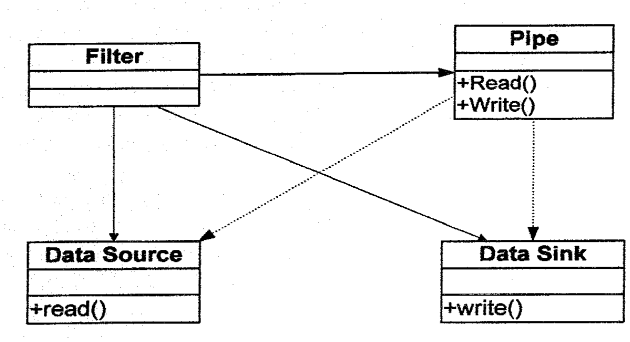


Figure: Pipe and active filter class diagram

Data Flow Architecture Pipe and Filter Architecture

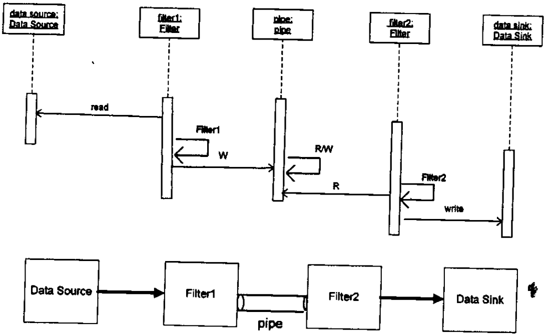


Figure: Pipe and filter block diagram and sequence diagram

Data Flow Architecture Pipe and Filter Architecture

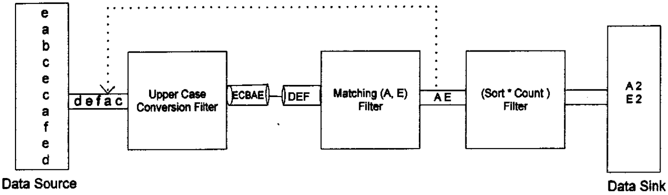


Figure: Pipelined pipe and filter

Data Flow Architecture Pipe and Filter Architecture

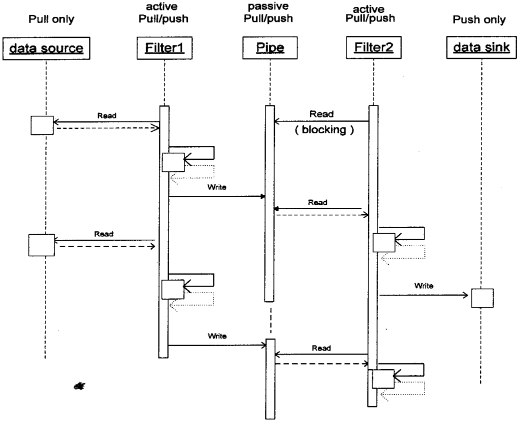


Figure: Pipe and filter sequence diagram

Data Flow Architecture Pipe and Filter Architecture

- Applicable Design Domain of Pipe and Filter Architecture

Data Flow Architecture Pipe and Filter Architecture

- Applicable Design Domain of Pipe and Filter Architecture
  - Wherever the system can be broken into a series of processing steps over data stream

- Applicable Design Domain of Pipe and Filter Architecture
  - Wherever the system can be broken into a series of processing steps over data stream
  - Data format on the data stream is simple and stable, and easy to be adapted if it is necessary

- Applicable Design Domain of Pipe and Filter Architecture
  - Wherever the system can be broken into a series of processing steps over data stream
  - Data format on the data stream is simple and stable, and easy to be adapted if it is necessary
  - There are significant work which can be pipelined to gain the performance

- Applicable Design Domain of Pipe and Filter Architecture
  - Wherever the system can be broken into a series of processing steps over data stream
  - Data format on the data stream is simple and stable, and easy to be adapted if it is necessary
  - There are significant work which can be pipelined to gain the performance
  - Suitable for producer/consumer model

Data Flow Architecture Pipe and Filter Architecture

- Applicable Design Domain of Pipe and Filter Architecture
  - Wherever the system can be broken into a series of processing steps over data stream
  - Data format on the data stream is simple and stable, and easy to be adapted if it is necessary
  - There are significant work which can be pipelined to gain the performance
  - Suitable for producer/consumer model

- Benefits of Pipe and Filter:

- Applicable Design Domain of Pipe and Filter Architecture
  - Wherever the system can be broken into a series of processing steps over data stream
  - Data format on the data stream is simple and stable, and easy to be adapted if it is necessary
  - There are significant work which can be pipelined to gain the performance
  - Suitable for producer/consumer model
- Benefits of Pipe and Filter:
  - Concurrency

- Applicable Design Domain of Pipe and Filter Architecture
  - Wherever the system can be broken into a series of processing steps over data stream
  - Data format on the data stream is simple and stable, and easy to be adapted if it is necessary
  - There are significant work which can be pipelined to gain the performance
  - Suitable for producer/consumer model
- Benefits of Pipe and Filter:
  - Concurrency
  - Reusability: Encapsulation of filters makes it easy to plug and play and to substitute

Data Flow Architecture Pipe and Filter Architecture

- Applicable Design Domain of Pipe and Filter Architecture
  - Wherever the system can be broken into a series of processing steps over data stream
  - Data format on the data stream is simple and stable, and easy to be adapted if it is necessary
  - There are significant work which can be pipelined to gain the performance
  - Suitable for producer/consumer model
- Benefits of Pipe and Filter:
  - Concurrency
  - Reusability: Encapsulation of filters makes it easy to plug and play and to substitute
  - Modifiability: Low coupling between filters

- Applicable Design Domain of Pipe and Filter Architecture
  - Wherever the system can be broken into a series of processing steps over data stream
  - Data format on the data stream is simple and stable, and easy to be adapted if it is necessary
  - There are significant work which can be pipelined to gain the performance
  - Suitable for producer/consumer model
- Benefits of Pipe and Filter:
  - Concurrency
  - Reusability: Encapsulation of filters makes it easy to plug and play and to substitute
  - Modifiability: Low coupling between filters
  - Simplicity: Clear division between piped filters

- Applicable Design Domain of Pipe and Filter Architecture
  - Wherever the system can be broken into a series of processing steps over data stream
  - Data format on the data stream is simple and stable, and easy to be adapted if it is necessary
  - There are significant work which can be pipelined to gain the performance
  - Suitable for producer/consumer model
- Benefits of Pipe and Filter:
  - Concurrency
  - Reusability: Encapsulation of filters makes it easy to plug and play and to substitute
  - Modifiability: Low coupling between filters
  - Simplicity: Clear division between piped filters
  - Flexibility: It supports sequential + parallel execution

Limitations of Pipe and Filter:

- Not suitable for dynamic interactions

Limitations of Pipe and Filter:

- Not suitable for dynamic interactions
- Low Common Denominator is required for data transmission in the ASCII formats

Limitations of Pipe and Filter:

- Not suitable for dynamic interactions
- Low Common Denominator is required for data transmission in the ASCII formats
- Overhead of data transformation among filters such as parsing overhead in two consecutive filters

Limitations of Pipe and Filter:

- Not suitable for dynamic interactions
- Low Common Denominator is required for data transmission in the ASCII formats
- Overhead of data transformation among filters such as parsing overhead in two consecutive filters
- Difficult to configure a P&F system dynamically

Limitations of Pipe and Filter:

- Not suitable for dynamic interactions
- Low Common Denominator is required for data transmission in the ASCII formats
- Overhead of data transformation among filters such as parsing overhead in two consecutive filters
- Difficult to configure a P&F system dynamically
- Error handling issue

- The process-control software architecture is suitable for the embedded system software design

- The process-control software architecture is suitable for the embedded system software design
- The process-control architecture decomposes the whole system into two type sub-systems:

Process-Control Architecture

- The process-control software architecture is suitable for the embedded system software design
- The process-control architecture decomposes the whole system into two type sub-systems:
    - executor processor unit for changing process control variables

Process-Control Architecture

- The process-control software architecture is suitable for the embedded system software design

- The process-control architecture decomposes the whole system into two type sub-systems:
  - executor processor unit for changing process control variables

  - controller unit for calculating the amounts of the changes

Data Flow Architecture Process-Control Architecture

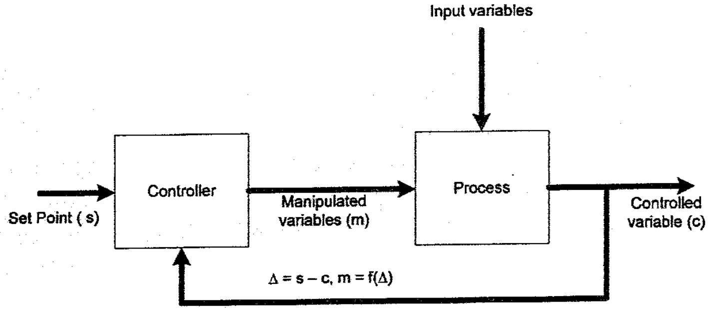


Figure: Data flow in the process control architecture

A process control system must have the following process control data:

- Controlled variable:

Process-Control Architecture is subject of SFWR ENG 3DX3 Dynamic models and control of physical systems

A process control system must have the following process control data:

- Controlled variable:
  - Example: speed in a cruise control system

Process-Control Architecture is subject of SFWR ENG 3DX3 Dynamic models and control of physical systems

A process control system must have the following process control data:

- Controlled variable:
  - Example: speed in a cruise control system
  - It has a set point which is the goal to reach

Process-Control Architecture is subject of SFWR ENG 3DX3 Dynamic models and control of physical systems

A process control system must have the following process control data:

- Controlled variable:
  - Example: speed in a cruise control system
  - It has a set point which is the goal to reach
  - The controlled variable data should be measured by sensors

Process-Control Architecture is subject of SFWR ENG 3DX3 Dynamic models and control of physical systems

A process control system must have the following process control data:

- Controlled variable:
  - Example: speed in a cruise control system
  - It has a set point which is the goal to reach
  - The controlled variable data should be measured by sensors
- Input variable:

Process-Control Architecture is subject of SFWR ENG 3DX3 Dynamic models and control of physical systems

A process control system must have the following process control data:

- Controlled variable:
  - Example: speed in a cruise control system
  - It has a set point which is the goal to reach
  - The controlled variable data should be measured by sensors
- Input variable:
  - Example: measured input data such as the temperature of return air in a temperature control system

Process-Control Architecture is subject of SFWR ENG 3DX3 Dynamic models and control of physical systems

A process control system must have the following process control data:

- Controlled variable:
  - Example: speed in a cruise control system
  - It has a set point which is the goal to reach
  - The controlled variable data should be measured by sensors
- Input variable:
  - Example: measured input data such as the temperature of return air in a temperature control system
- Manipulated variable: can be adjusted by the controller

Process-Control Architecture is subject of SFWR ENG 3DX3 Dynamic models and control of physical systems

- Applicable domains

- Applicable domains
  - Embedded software system involving continuing actions

- Applicable domains
  - Embedded software system involving continuing actions
  - The system needs to maintain an output data at a stable level

- Applicable domains
  - Embedded software system involving continuing actions
  - The system needs to maintain an output data at a stable level
  - The system can have a target point

- Applicable domains
  - Embedded software system involving continuing actions
  - The system needs to maintain an output data at a stable level
  - The system can have a target point
- Benefits

- Applicable domains
  - Embedded software system involving continuing actions
  - The system needs to maintain an output data at a stable level
  - The system can have a target point
- Benefits
  - Better solution to the control system where no precise formula can be used to decide the manipulated variable

Data Flow Architecture Process-Control Architecture

- Applicable domains
  - Embedded software system involving continuing actions
  - The system needs to maintain an output data at a stable level
  - The system can have a target point
- Benefits
  - Better solution to the control system where no precise formula can be used to decide the manipulated variable
  - The software can be completely embedded in the devices

Process-Control Architecture

- Applicable domains
  - Embedded software system involving continuing actions
  - The system needs to maintain an output data at a stable level
  - The system can have a target point
- Benefits
  - Better solution to the control system where no precise formula can be used to decide the manipulated variable
  - The software can be completely embedded in the devices
- Limitations: Can be unstable and requires a thorough mathematical analysis

Data Flow Architecture

What class model would be appropriate to implement a given data flow?

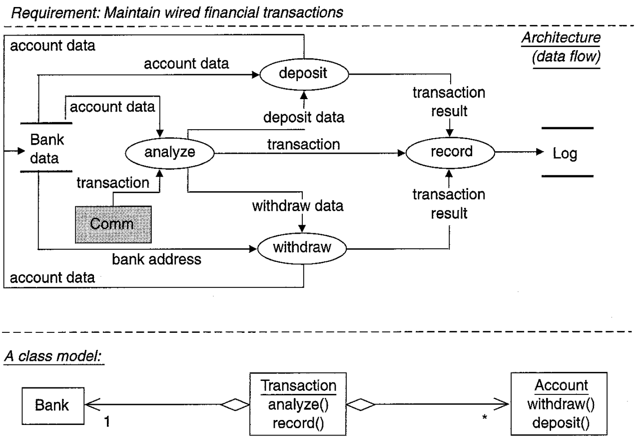


Figure: Example of Data Flow Architecture and Corresponding Class Model