C++ Basics
Software Development Process

Intended Learning Outcomes

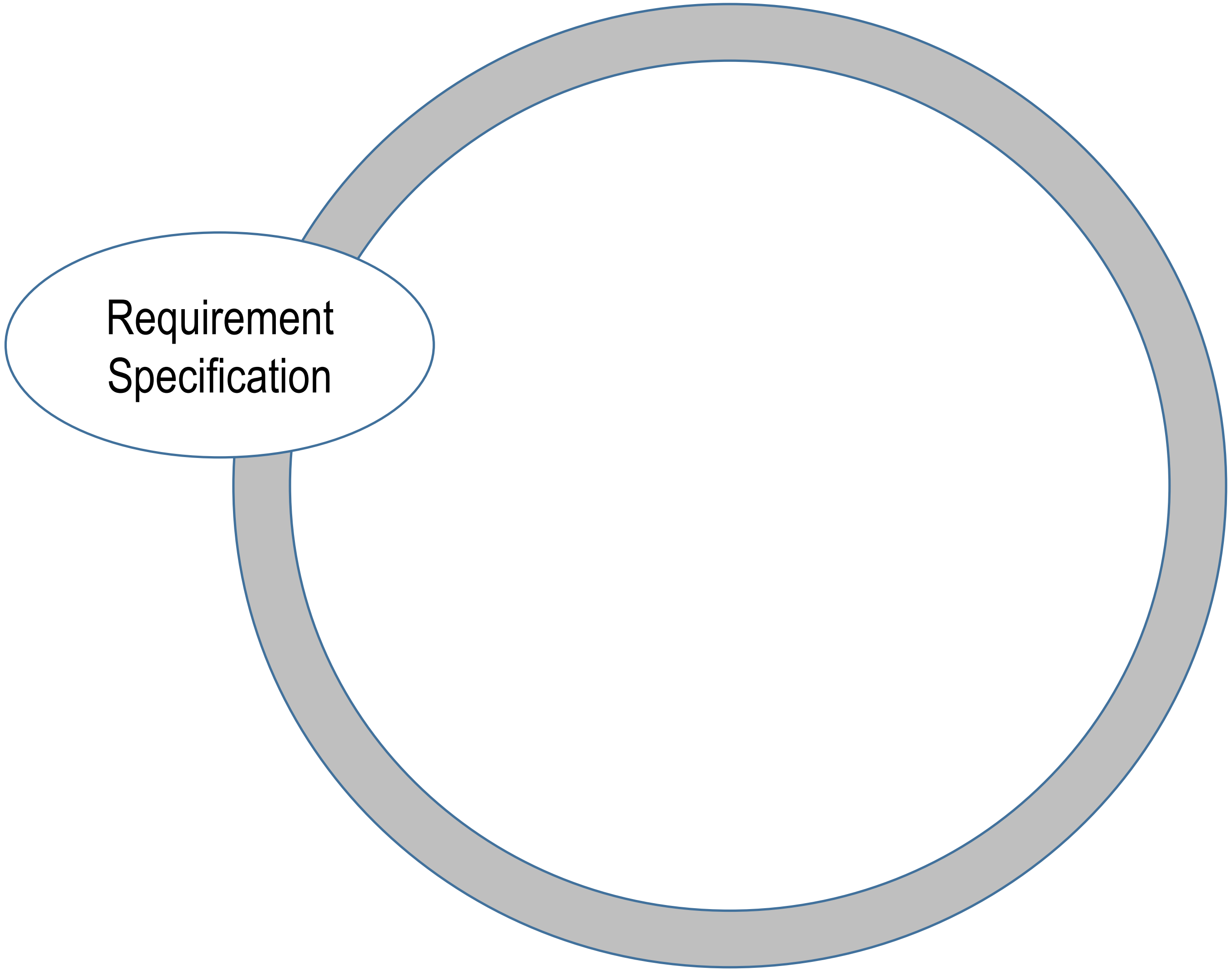
- Describe the process of software development
- Define a base class
- Define a derived class
- Describe the process of Monte Carlo simulation for estimating π

Software Development Process

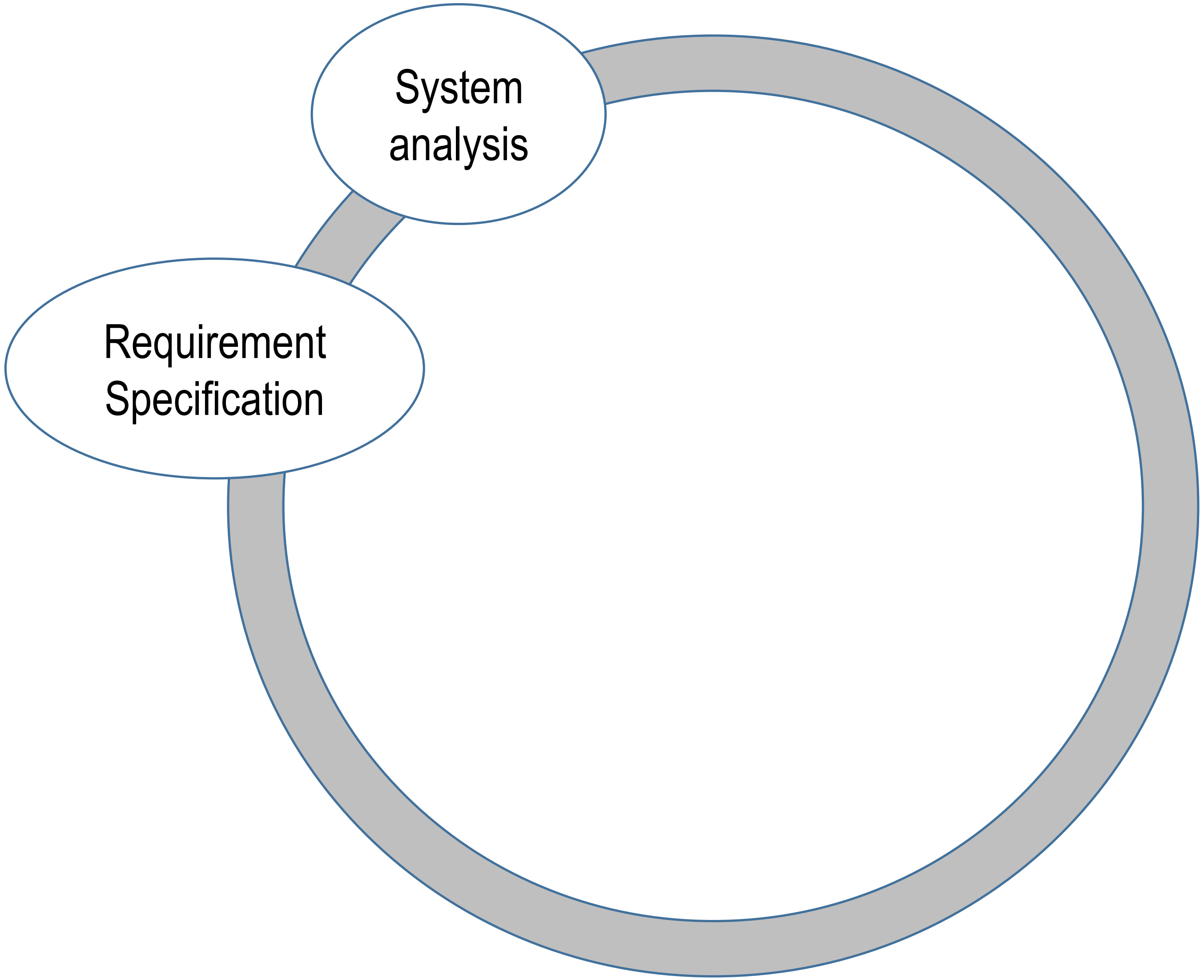
Software Development Process

Requirement Specification

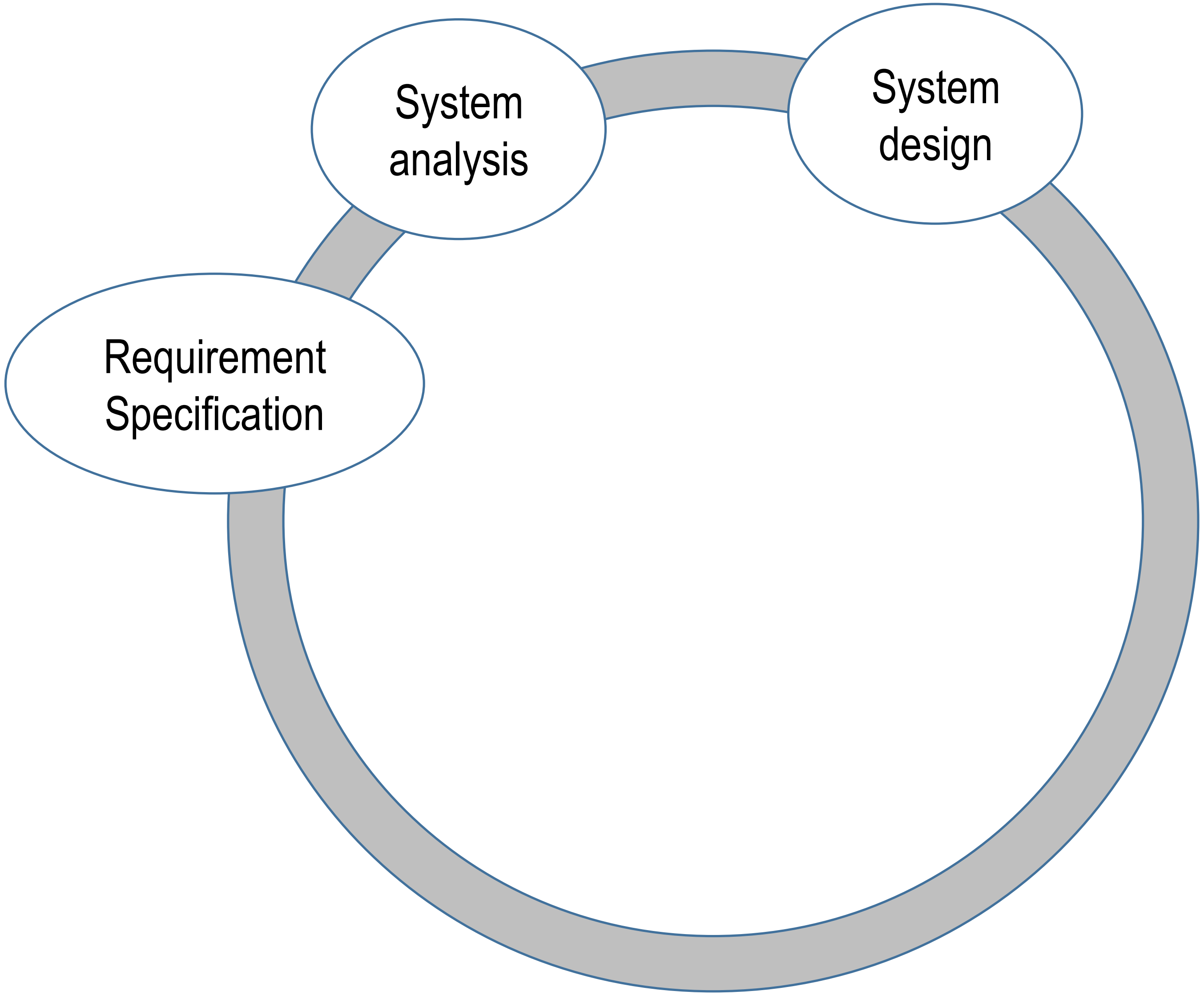
Software Development Process



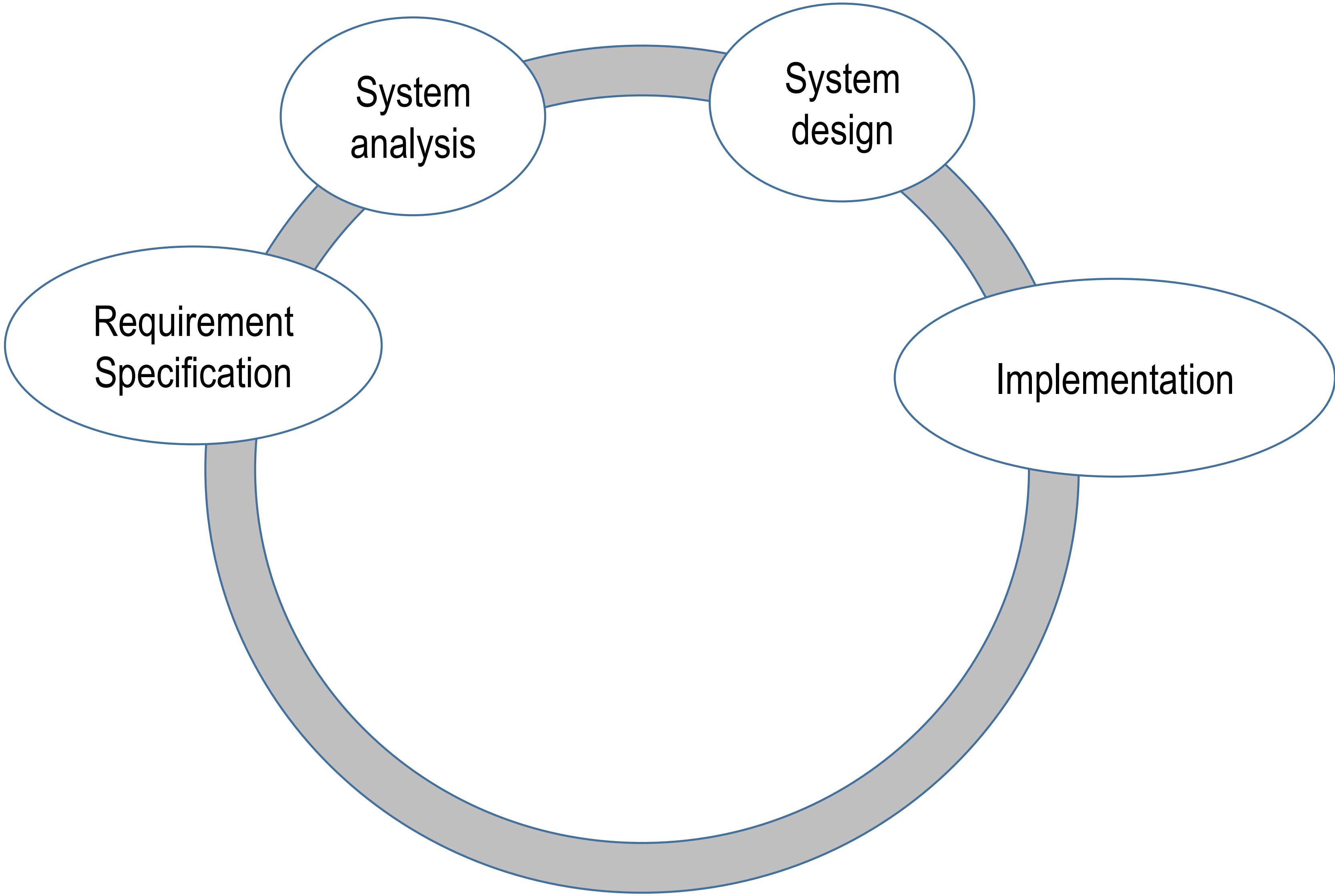
Software Development Process



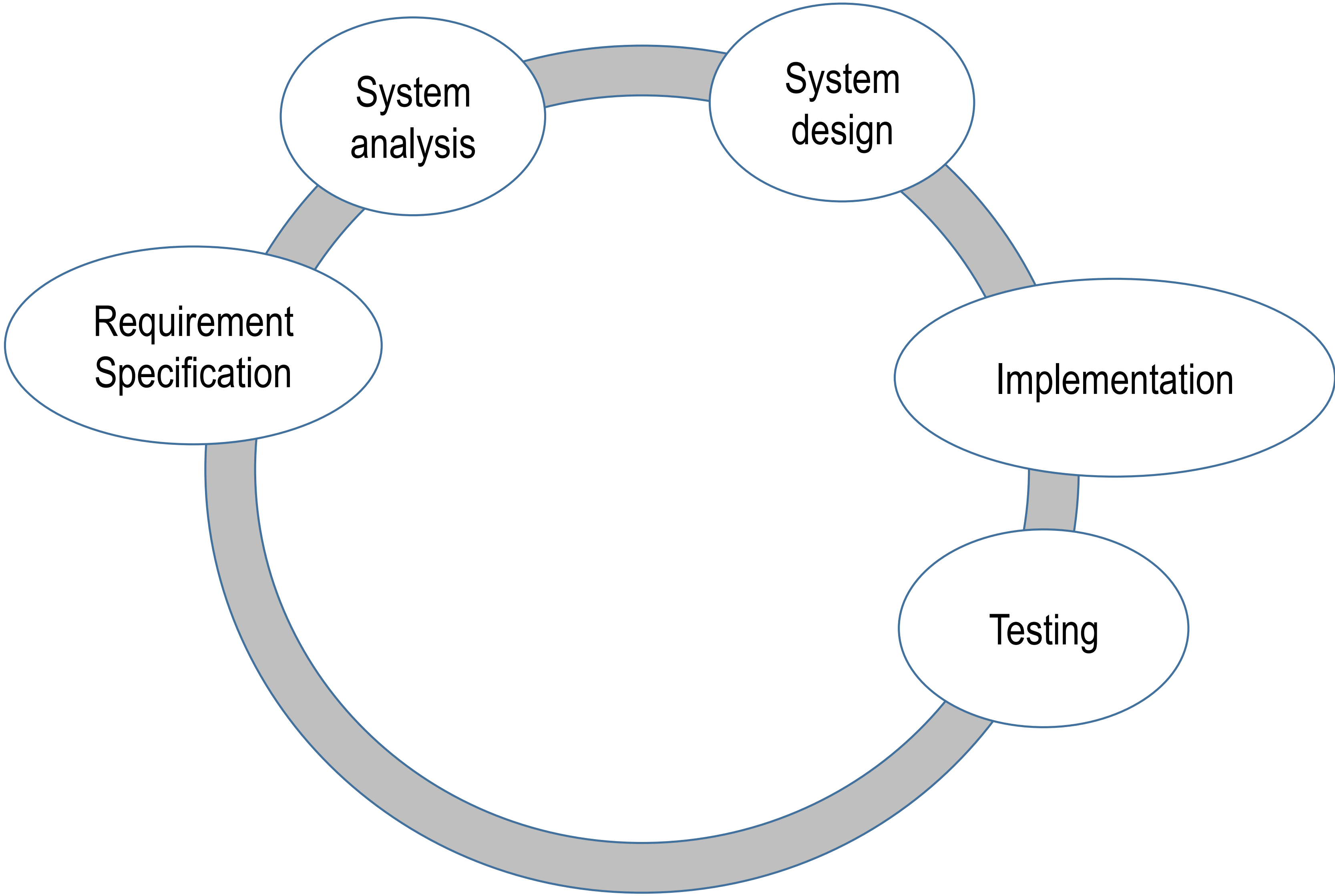
Software Development Process



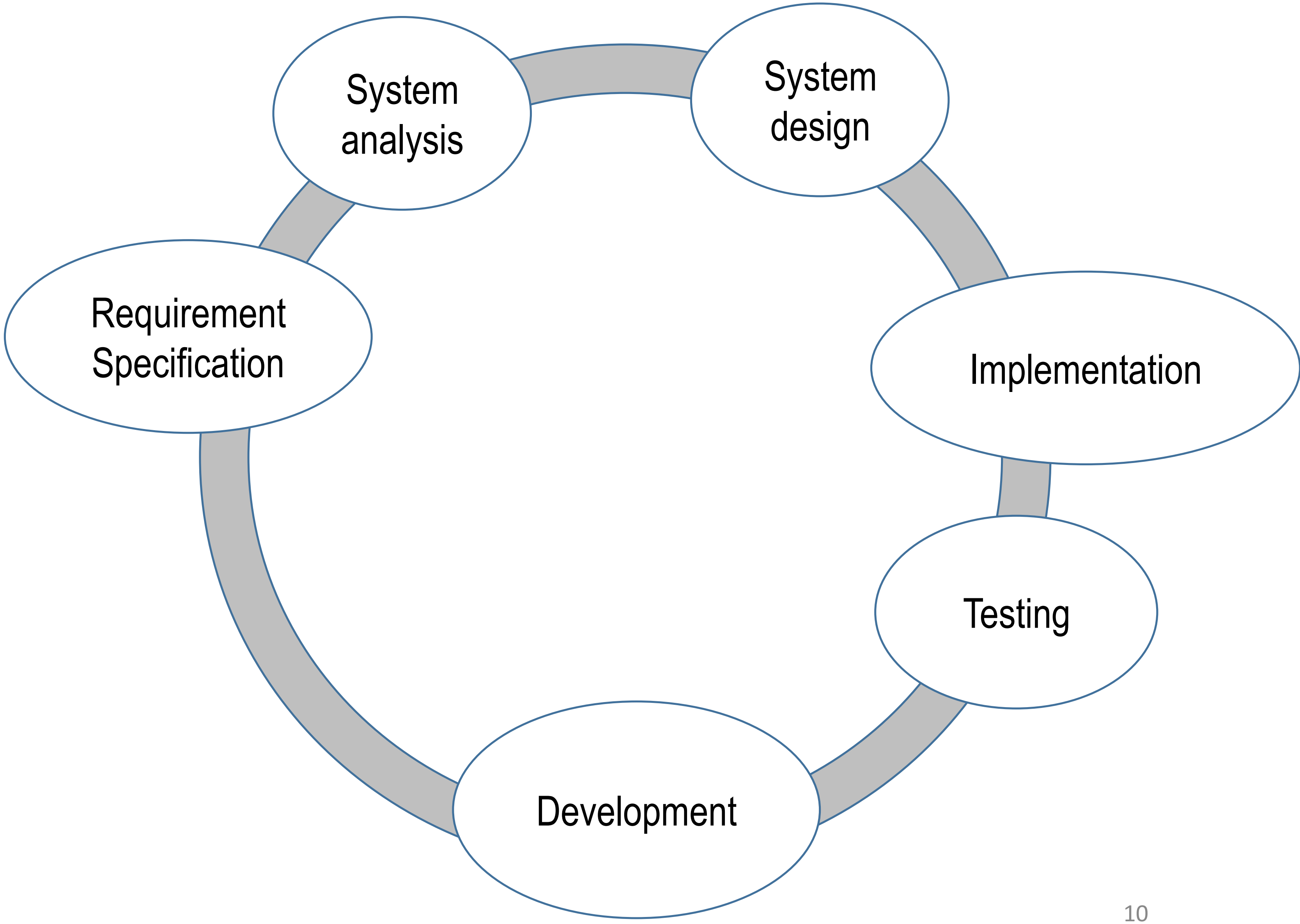
Software Development Process



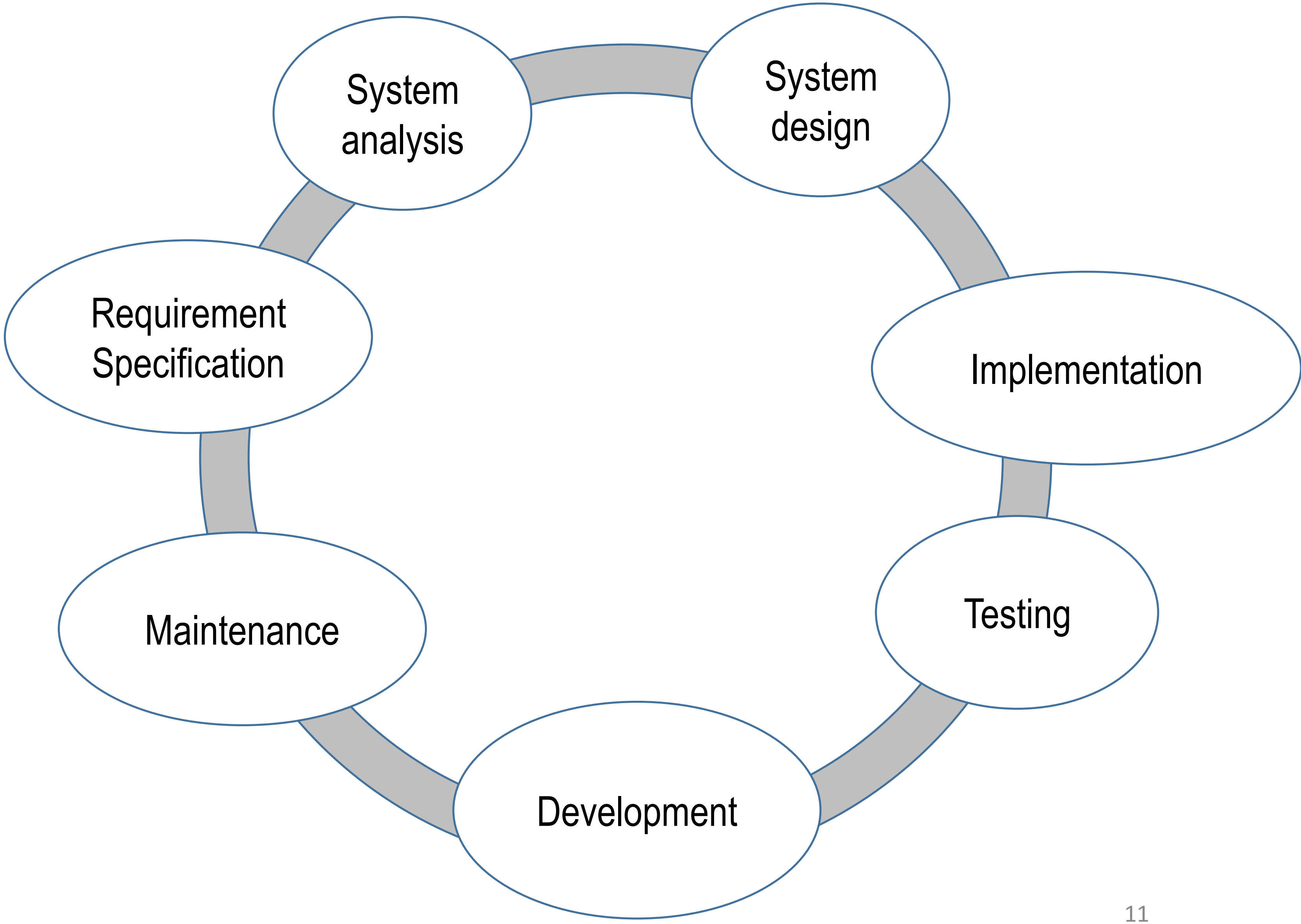
Software Development Process



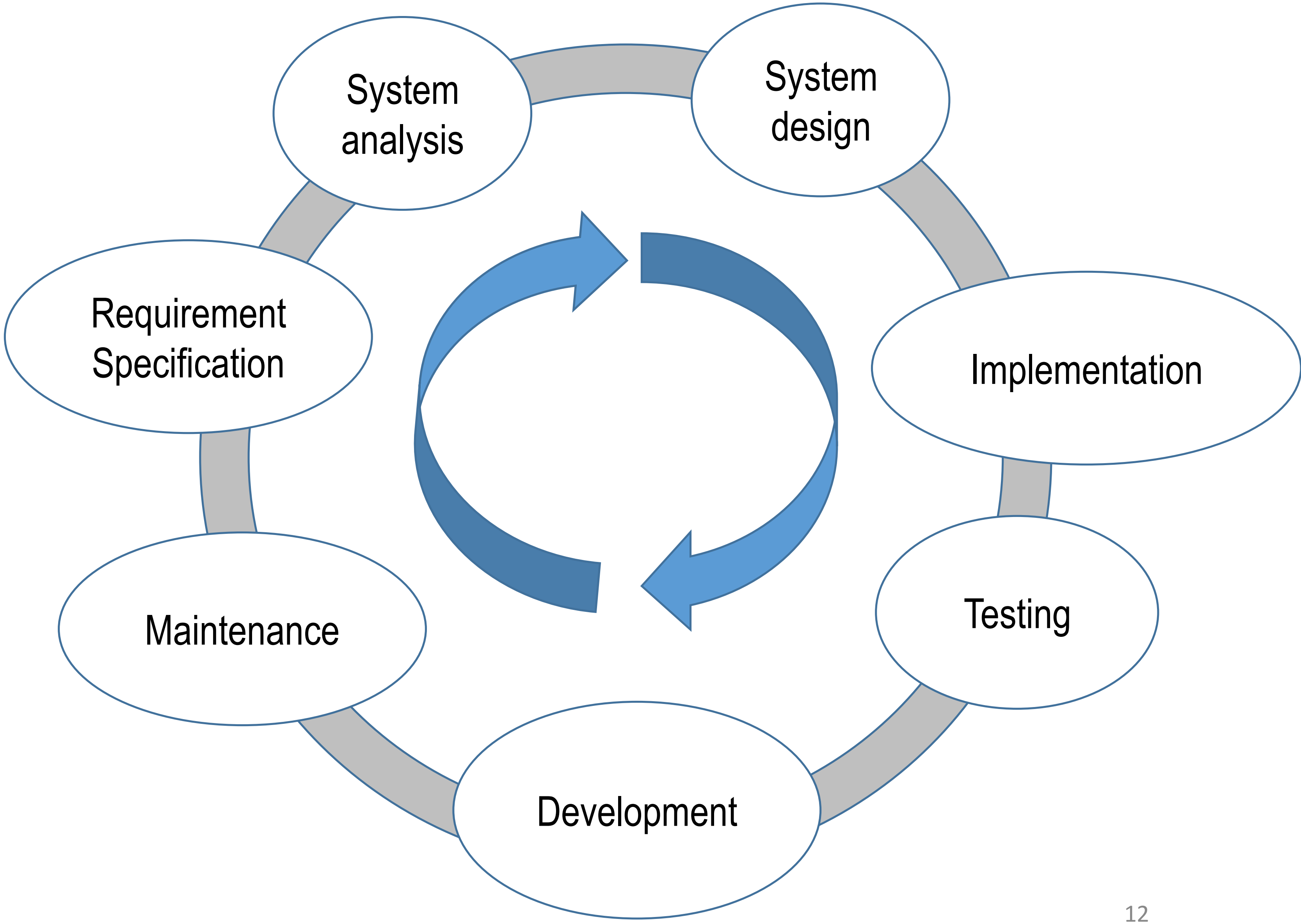
Software Development Process



Software Development Process

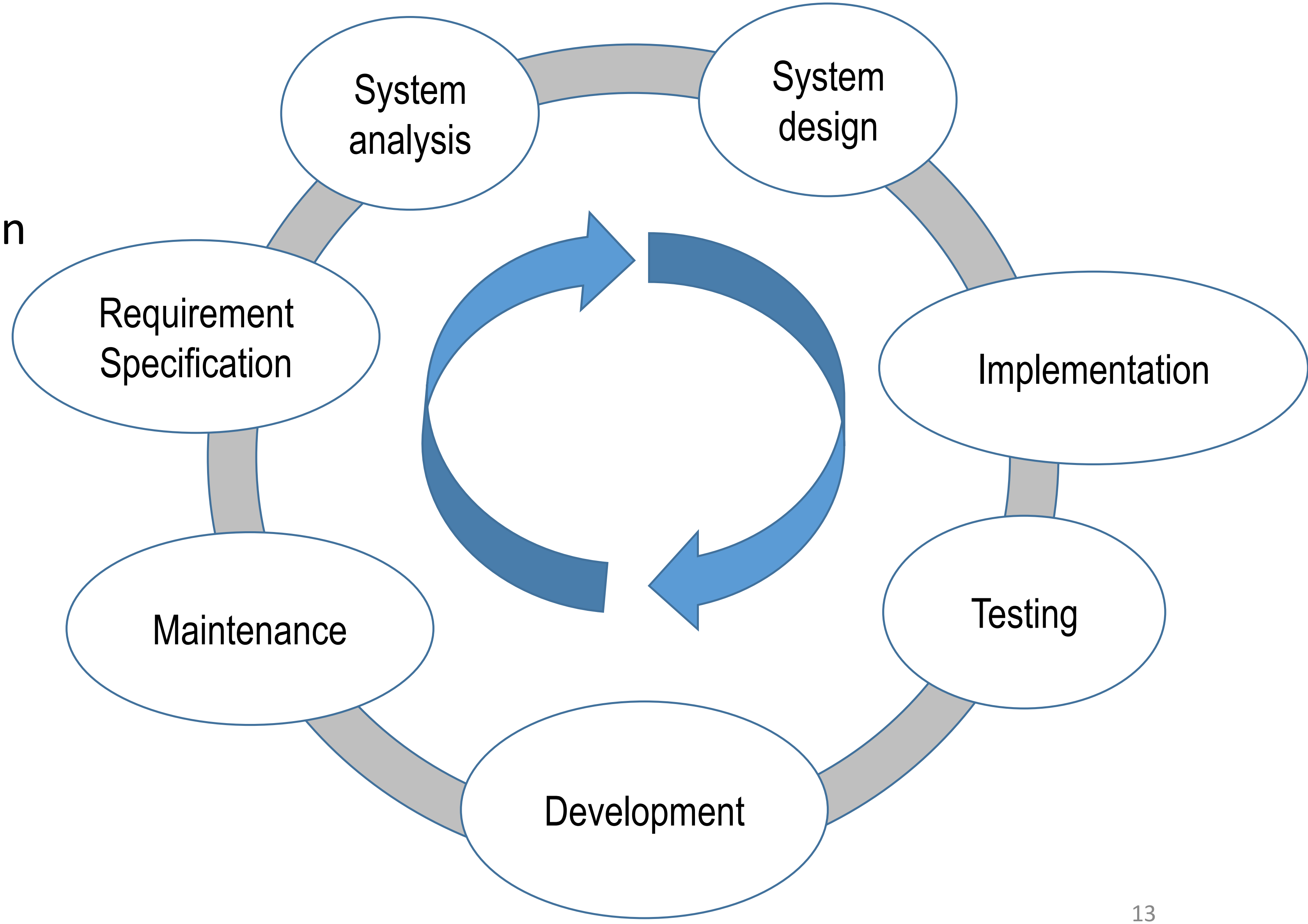


Software Development Process



Software Development Process

1. Requirement specification
2. System analysis
3. System design
4. Implementation
5. Testing
6. Deployment
7. Maintenance



Example: SQUARE

Write a class SQUARE. Implement the following tasks.

1. Ask the user to input the length of the side of a square.
2. Show the area of the square.
3. Show the perimeter of the square

Example: SQUARE

Write a class SQUARE. Implement the following tasks.

1. Ask the user to input the length of the side of a square.
2. Show the area of the square.
3. Show the perimeter of the square

1. **Requirement specification**
2. System analysis
3. System design
4. Implementation
5. Testing
6. Deployment
7. Maintenance

Example: SQUARE

Inputs:

- the length of the side of a square

Output:

- area
- perimeter

Write a class SQUARE. Implement the following tasks.

1. Ask the user to input the length of the side of a square.
2. Show the area of the square.
3. Show the perimeter of the square

1. Requirement specification
2. **System analysis**
3. System design
4. Implementation
5. Testing
6. Deployment
7. Maintenance

Example: SQUARE

Write a class SQUARE. Implement the following tasks.

1. Ask the user to input the length of the side of a square.
2. Show the area of the square.
3. Show the perimeter of the square

Inputs:

- the length of the side of a square

Output:

- area - method?
- perimeter - method?

1. Requirement specification
2. System analysis
3. **System design**
4. Implementation
5. Testing
6. Deployment
7. Maintenance

Example: SQUARE

Inputs:

- the length of the side of a square

Output:

- area - computeArea( )

- perimeter - computePerimeter( )

Is this design good enough?

Write a class SQUARE. Implement the following tasks.

1. Ask the user to input the length of the side of a square.
2. Show the area of the square.
3. Show the perimeter of the square

1. Requirement specification
2. System analysis
3. **System design**
4. Implementation
5. Testing
6. Deployment
7. Maintenance

Example: SQUARE

Class name? Create a class: SQUARE

Inputs:

- the length of the side of a square

Output:

- area - computeArea( )

- perimeter - computePerimeter( )

Others: (need to add extra functions)

- initialize the data
- ask for input
- display messages to let the user know what they are doing

Write a class SQUARE. Implement the following tasks.

1. Ask the user to input the length of the side of a square.
2. Show the area of the square.
3. Show the perimeter of the square

1. Requirement specification
2. System analysis
3. **System design**
4. Implementation
5. Testing
6. Deployment
7. Maintenance

Example: SQUARE

Class name? Create a class: SQUARE

Inputs:

- the length of the side of a square

Output:

- area	- computeArea( )
- perimeter	- computePerimeter( )

Others: (need to add extra functions)

- initialize the data
- ask for input
- display messages to let the user know what they are doing

Write a class SQUARE. Implement the following tasks.

1. Ask the user to input the length of the side of a square.
2. Show the area of the square.
3. Show the perimeter of the square

1. Requirement specification
2. System analysis
3. **System design**
4. Implementation
5. Testing
6. Deployment
7. Maintenance

Software Development Process

Inputs:

- the length of the side of a square

Output:

- area - computeArea( )
- perimeter - computePerimeter( )

Methods:

- showMessage
- askForInput
- showArea
- showPerimeter
- askForContinue
- showThankYouMessage

1. Requirement specification
2. System analysis
3. System design
4. **Implementation**
5. **Testing**
6. Deployment
7. Maintenance

Software Development Process

Inputs:

- the length of the side of a square

Output:

- area - computeArea( )
- perimeter - computePerimeter( )

Methods:

- showMessage
- askForInput
- showArea
- showPerimeter
- askForContinue
- showThankYouMessage

```
class SQUARE {
protected:
  double sideLength;
  double area;
  double perimeter;
public:
   SQUARE( ) { … }
   process( );
   ……
protected:
   void showMessage( ) const;
   void askForInput( );
   ……
};
```

Software Development Process

Inputs:

- the length of the side of a square

Output:

- area	- computeArea( )
- perimeter	- computePerimeter( )

Methods:

- showMessage
- askForInput
- showArea
- showPerimeter
- askForContinue
- showThankYouMessage

```
void SQUARE::process( ) {
  showMessage( );
  askForInput( );
  computeArea( );
  computePerimeter( );
  showArea( );
  showPerimeter( );
  showThankYouMessage( );
}
```

Software Development Process

Inputs:

- the length of the side of a square

Output:

- area - computeArea( )
- perimeter - computePerimeter( )

Methods:

- showMessage
- askForInput
- showArea
- showPerimeter
- **askForContinue**
- showThankYouMessage

```
void SQUARE::process( ) {
  showMessage( );
  askForInput( );
  computeArea( );
  computePerimeter( );
  showArea( );
  showPerimeter( );
  showThankYouMessage( );
}
```

Software Development Process - askForContinue

Inputs:

- the length of the side of a square

Output:

- area - computeArea( )
- perimeter - computePerimeter( )

Methods:

- showMessage
- askForInput
- showArea
- showPerimeter
- **askForContinue**
- showThankYouMessage

```
void SQUARE::process( ) {
  showMessage( );
  askForInput( );
  computeArea( );
  computePerimeter( );
  showArea( );
  showPerimeter( );
  showThankYouMessage( );
}
```

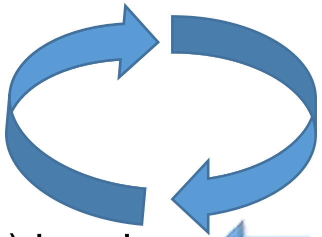
Software Development Process - askForContinue

```
void SQUARE::process( ) {
  askForInput( );
  computeArea( );
  computePerimeter( );
  showArea( );
  showPerimeter( );
}
void SQUARE::handleQuery( ) {
  showMessage( );
  while (true) {
     process( );
  }
  showThankYouMessage( );
}
```

```
void SQUARE::process( ) {
  showMessage( );
  askForInput( );
  computeArea( );
  computePerimeter( );
  showArea( );
  showPerimeter( );
  showThankYouMessage( );
}
```

Software Development Process - askForContinue

```
void SQUARE::process( ) {
  askForInput( );
  computeArea( );
  computePerimeter( );
  showArea( );
  showPerimeter( );
}
void SQUARE::handleQuery( ) {
  showMessage( );
  while (true) {
     process( );
     if ( ! askForContinue( ) ) break;
  }
  showThankYouMessage( );
}
```



```
void SQUARE::process( ) {
  showMessage( );
  askForInput( );
  computeArea( );
  computePerimeter( );
  showArea( );
  showPerimeter( );
  showThankYouMessage( );
}
```

Software Development Process - askForContinue

```
void SQUARE::process( ) {
  askForInput( );
  computeArea( );
  computePerimeter( );
  showArea( );
  showPerimeter( );
}
void SQUARE::handleQuery( ) {
  showMessage( );
  while (true) {
     process( );
     if ( ! askForContinue( ) ) break;
  }
  showThankYouMessage( );
}
```

```
bool SQUARE::askForContinue( ) const {
  bool flg_Y = false;
  while (true) {
      cout << “Do you want to continue (Y/N)?” << endl;
      char c;
      cin >> c;
      flg_Y =  c == ‘Y’ || c == ‘y’ ;
      bool flg_N =  c == ‘N’ || c == ‘n’ ;
      bool  flg = flg_Y || flg_N;
      if ( flg ) break;
  }
  return flg_Y;
} // return true if ‘Y’ or ‘y’ is pressed.
```

Software Development Process - askForContinue

```
void SQUARE::process( ) {
  askForInput( );
  computeArea( );
  computePerimeter( );
  showArea( );
  showPerimeter( );
}
void SQUARE::handleQuery( ) {
  showMessage( );
  while (true) {
     process( );
     if ( ! askForContinue( ) ) break;
  }
  showThankYouMessage( );
}
```

```
bool SQUARE::askForContinue( ) const {
  bool flg_Y = false;
  while (true) {
      cout << “Do you want to continue (Y/N)?” << endl;
      char c;
      cin >> c;
      flg_Y =  ( c == ‘Y’ || c == ‘y’ );
      bool flg_N =  ( c == ‘N’ || c == ‘n’ );
      bool  flg = flg_Y || flg_N;
      if ( flg ) break;
  }
  return flg_Y;
} // return true if ‘Y’ or ‘y’ is pressed.
```

Add parentheses to improve readability

Example: SQUARE new function

Write a class SQUARE. Implement the following tasks.

1. Ask the user to input the length of the side of a square.
2. Show the area of the square.
3. Show the perimeter of the square
4. **If side is positive, go back to step 1. Otherwise, quit the program.**

Example: SQUARE new function

Write a class SQUARE. Implement the following tasks.

1. Ask the user to input the length of the side of a square.
2. Show the area of the square.
3. Show the perimeter of the square
4. **If side is positive, go back to step 1. Otherwise, quit the program.**

We need a loop.

Example: SQUARE new function

Write a class SQUARE. Implement the following tasks.

1. Ask the user to input the length of the side of a square.
2. Show the area of the square.
3. Show the perimeter of the square
4. **If side is positive, go back to step 1. Otherwise, quit the program.**

Write a loop to ask for input until the input value is valid.

The input value is valid if it is equal to or greater than 0.

```
// set sideLength to an invalid value
sideLength = -1.;
while ( sideLength  < 0 ) {
   sideLength = askForInput_SideLength( );
}
```

Be aware of logical bugs.

When sideLength >= 0, exit the while loop.

```
double SQUARE::askForInput_SideLength( ) {
  double len;
   cout << "Input the side length:";
   cin >> len;
   return len;
}
```

Example:
Requirement Specification

Write a class SQUARE_MANAGER. Implement the following tasks.

1. Input the number of squares
2. Input the side length of each square
3. Show the average area of the squares
4. Show the standard deviation of the areas of the squares

Example: SQUARE_MANAGER

Tasks:

1. Input the number of squares
2. Input the side length of each square
3. Show the average area of the squares
4. Show the standard deviation of the areas of the squares

Inputs:

- the number of squares
- the side length of each square

Output:

- average area of the squares
- standard deviation of the areas of the squares

1. Requirement specification
2. **System analysis**
3. System design
4. Implementation
5. Testing
6. Deployment
7. Maintenance

Example: SQUARE_MANAGER

Inputs: - the number of squares
- the side length of each square

Output: - average area of the squares
- standard deviation of the areas of the squares

Others: - **initialize the data**
- ask for input
- display messages to let the user know what they are doing
- compute the area of each square

Tasks:

1. Input the number of squares
2. Input the side length of each square
3. Show the average area of the squares
4. Show the standard deviation of the areas of the squares

1. Requirement specification
2. **System analysis**
3. System design
4. Implementation
5. Testing
6. Deployment
7. Maintenance

Example: CIRCLE_MANAGER

Inputs: - the number of **circles**

- the radius of each **circle**

Output: - average area of the **circles**

- standard deviation of the areas of the **circles**

Others: - **initialize the data**

- ask for input

- display messages to let the user know what they are doing

- compute the area of each circle

Tasks:

1. Input the number of **circles**
2. Input the radius of each **circle**
3. Show the average area of the **circles**
4. Show the standard deviation of the areas of the **circles**

1. Requirement specification
2. **System analysis**
3. System design
4. Implementation
5. Testing
6. Deployment
7. Maintenance

Example: CIRCLE_MANAGER

Inputs: - the number of **circles**
- the radius of each **circle**

Output: - average area of the **circles**
- standard deviation of the areas of the **circles**

Others: - **initialize the data**
- **ask for input**
- **display messages to let the user know what they are doing**
- **compute the area of each circle**

Example: SQUARE_MANAGER

Inputs: - the number of squares
- the side length of each square

Output: - average area of the squares
- standard deviation of the areas of the squares

Others: - **initialize the data**
- **ask for input**
- **display messages to let the user know what they are doing**
- **compute the area of each square**

Example: CIRCLE_MANAGER

Inputs: - the number of **circles**

- the radius of each **circle**

Output: - average area of the **circles**

- standard deviation of th

of the **circles**

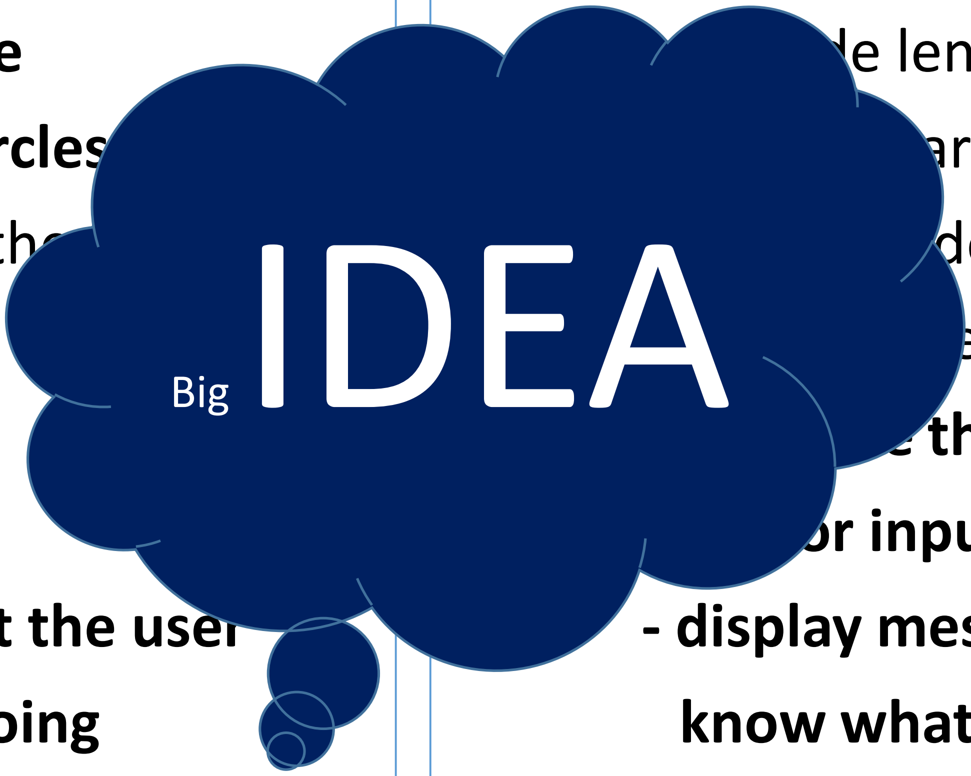
Others: - **initialize the data**

- ask for input

- display messages to let the user

know what they are doing

- compute the area of each circle



Example: SQUARE_MANAGER

Inputs: - the number of squares

le length of each square

area of the squares

deviation of the areas

es

the data

or input

- display messages to let the user

know what they are doing

- compute the area of each square

Example: OBJECT_MANAGER

Inputs: - the number of **objects**
- the attributes of each **object**

Output: - average area of the **objects**
- standard deviation of the areas of the **objects**

Others: - **initialize the data**
- **ask for input**
- **display messages to let the user know what they are doing**
- **compute the area of each object**

Example: SQUARE_MANAGER

Inputs: - the number of squares
- the side length of each square

Output: - average area of the squares
- standard deviation of the areas of the squares

Others: - **initialize the data**
- **ask for input**
- **display messages to let the user know what they are doing**
- **compute the area of each square**

Example: OBJECT_MANAGER

Inputs: - the number of **objects**

- the attributes of each **object**

Output: - average area of the **objects**

- standard deviation of the areas of the **objects**

Others: - **initialize the data**

- ask for input

- display messages to let the user know what they are doing

- compute the area of each object

Tasks:

1. Input the number of **objects**
2. Input the attributes of each **object**
3. Show the average area of the **objects**
4. Show the standard deviation of the areas of the **objects**

1. Requirement specification
2. **System analysis**
3. System design
4. Implementation
5. Testing
6. Deployment
7. Maintenance

Base class and derivation of new classes

```
SQUARE {
  area, perimeter
  computeArea
  computePerimeter
  …
};
```

```
CIRCLE {
  area, perimeter
  computeArea
  computePerimeter
  …
};
```

```
class BASE {
…
  virtual double computeArea( ) = 0;                //Declaration but undefined.
  virtual double computePerimeter( ) = 0;           //Declaration but undefined.
  double getPerimeter() const { return m_Perimeter; }
  double getArea() const { return m_Area; }
protected:
  double m_Perimeter, m_Area;
};

class A : public BASE {
…
  double computeArea( ) { //body….}
};

class B : public BASE {
…
  double computeArea( ) { … }
};
```

Base class and derivation of new classes

```
SQUARE {
   area, perimeter
   computeArea
   computePerimeter
   …
};
```

```
CIRCLE {
   area, perimeter
   computeArea
   computePerimeter
   …
};
```

Objects sharing similar properties

- Common functions
- Common data members (their own data members)
- Common processes
  - To compute the area, we perform similar task(s)
    - Ask for for input
    - Compute some values
    - etc
  - To compute the perimeter, we perform similar task(s)

-> Define a base class and use it to define new classes.

```
class BASE {
…
   virtual double computeArea( ) = 0;


   virtual double computePerimeter( ) = 0;


   double getPerimeter() const { … }
   double getArea() const {… }
protected:
   double m_Perimeter, m_Area;
};
```

```
class CIRCLE : public BASE {
…
   double computeArea( ) { //body….}
};
```

```
class SQUARE: public BASE {
…
   double computeArea( ) { … }
};
```

Objects sharing similar properties

- Common functions
- Common data members (their own data members)
- Common processes
  - To compute the area, we perform similar task(s)
    - Ask for for input
    - Compute some values
    - etc
  - To compute the perimeter, we perform similar task(s)

-> Define a base class and use it to define new classes.

```
class BASE {                                   // A1
…
   virtual double computeArea( ) = 0; // A2

   virtual double computePerimeter( ) = 0; // A3

   double getPerimeter() const { … }
   double getArea() const {… }
protected:
   double m_Perimeter, m_Area;
};
```

```
class CIRCLE : public BASE {
…
   double computeArea( ) { //body….}
};
```

```
class SQUARE: public BASE {
…
   double computeArea( ) { … }
};
```

Objects sharing similar properties

- Common functions
- Common data members (their own data members)
- Common processes
  - To compute the area, we perform similar task(s)
    - Ask for for input
    - Compute some values
    - etc
  - To compute the perimeter, we perform similar task(s)

-> Define a base class and use it to define new classes.

```
class BASE {                          // abstract class
…
   virtual double computeArea( ) = 0; // abstract

   virtual double computePerimeter( ) = 0; // abstract

   double getPerimeter() const { … }
   double getArea() const {… }
protected:
   double m_Perimeter, m_Area;
};
```

```
class CIRCLE : public BASE {
…
   double computeArea( ) { //body….}
};
```

```
class SQUARE: public BASE {
…
   double computeArea( ) { … }
};
```

Objects sharing similar properties

- Common functions
- Common data members   (their own data members)
- Common processes
  - To compute the area, we perform similar task(s)
    - Ask for for input
    - Compute some values
    - etc
  - To compute the perimeter, we perform similar task(s)

-> Define a base class and use it to define new classes.

Must A1 the A2 methods in the derived class if you want to create objects of the derived classes.

Case Study:
Requirement Specification

Write a program to perform the following tasks:

1. Input the number of students
2. Input the student ID of each student
3. Input the name of each student
4. Input the score of each student
5. Show the range of the scores,
   i.e., maximum score and minimum score.
6. Show the ID and name of the student(s) with the best score
7. Show the average score
8. Show the standard deviation of the scores

System Design

```
class STUDENT {







};
```

```
class CLASS {







};
```

Tasks:

1. Input the number of students
2. Input the student ID of each student
3. Input the name of each student
4. Input the score of each student
5. Show the range of the scores,
6. Show the ID and name of the student(s)

   with the best score
7. Show the average score
8. Show the standard deviation of the scores

System Design

```
class STUDENT {

};
```

```
class CLASS {

};
```

Tasks:

1. Input the number of students
2. Input the student ID of each student
3. Input the name of each student
4. Input the score of each student
5. Show the range of the scores,
6. Show the ID and name of the student(s)

   with the best score
7. Show the average score
8. Show the standard deviation of the scores

System Design

```
class STUDENT {

};
```

```
class CLASS {

};
```

System Design

Do this exercise in one minute

Tasks:

1. Input the number of students
2. Input the student ID of each student
3. Input the name of each student
4. Input the score of each student
5. Show the range of the scores,
6. Show the ID and name of the student(s)

   with the best score
7. Show the average score
8. Show the standard deviation of the scores

System Design

```
class STUDENT {
public:
……
void showInfo() const;
public:
   int ID;
   double score;
   string name;



};
```

```
class CLASS {














};
```

Tasks:

1. Input the number of students
2. Input the student ID of each student
3. Input the name of each student
4. Input the score of each student
5. Show the range of the scores,
6. Show the ID and name of the student(s)

   with the best score

7. Show the average score
8. Show the standard deviation of the scores

System Design

```
class STUDENT {
public:
……
void showInfo() const;
public:
  int ID;
  double score;
  string name;


};
```

```
class CLASS {
public:
  ......
  void inputStudentInformation();
  void computeScoreRange();
  void showScoreRange() const;
        ……
protected:
  int numStudents;
  std::vector<STUDENT> students;
  double score_range[2];
};
```

Tasks:

1. Input the number of students
2. Input the student ID of each student
3. Input the name of each student
4. Input the score of each student
5. Show the range of the scores,
6. Show the ID and name of the student(s)

   with the best score
7. Show the average score
8. Show the standard deviation of the scores

Case Study: Requirement Specification

Implement a program to use Monte Carlo simulation to estimate π.

1. Input the number of points
2. Input the radius of the circle
3. Randomly generate sample points inside a square enclosing the circle
4. Estimate π
5. Display π

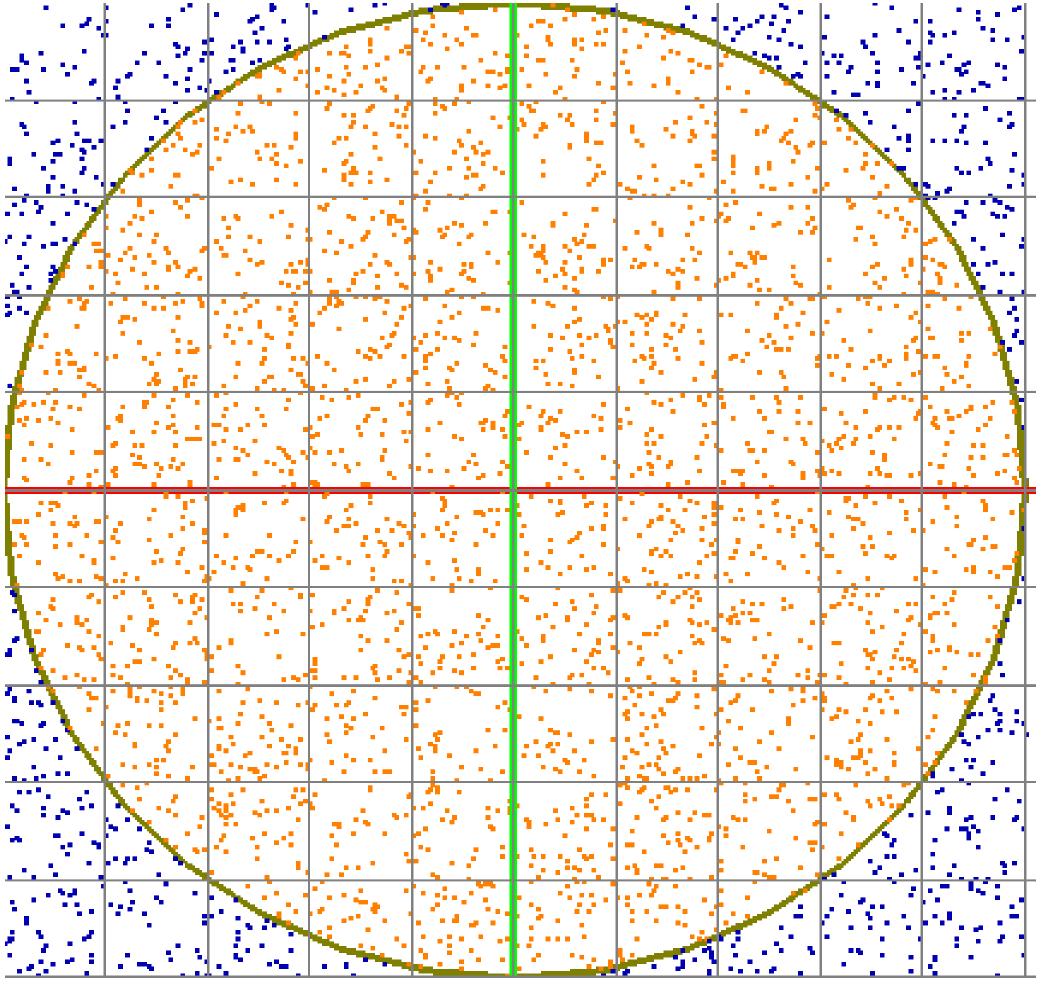
Monte Carlo Simulation

A technique uses random numbers and probability to solve problems.

We use the Monto Carlo simulation to estimate π.

Given a square and an inscribed circle, generate randomly sample points.

Circle radius: r



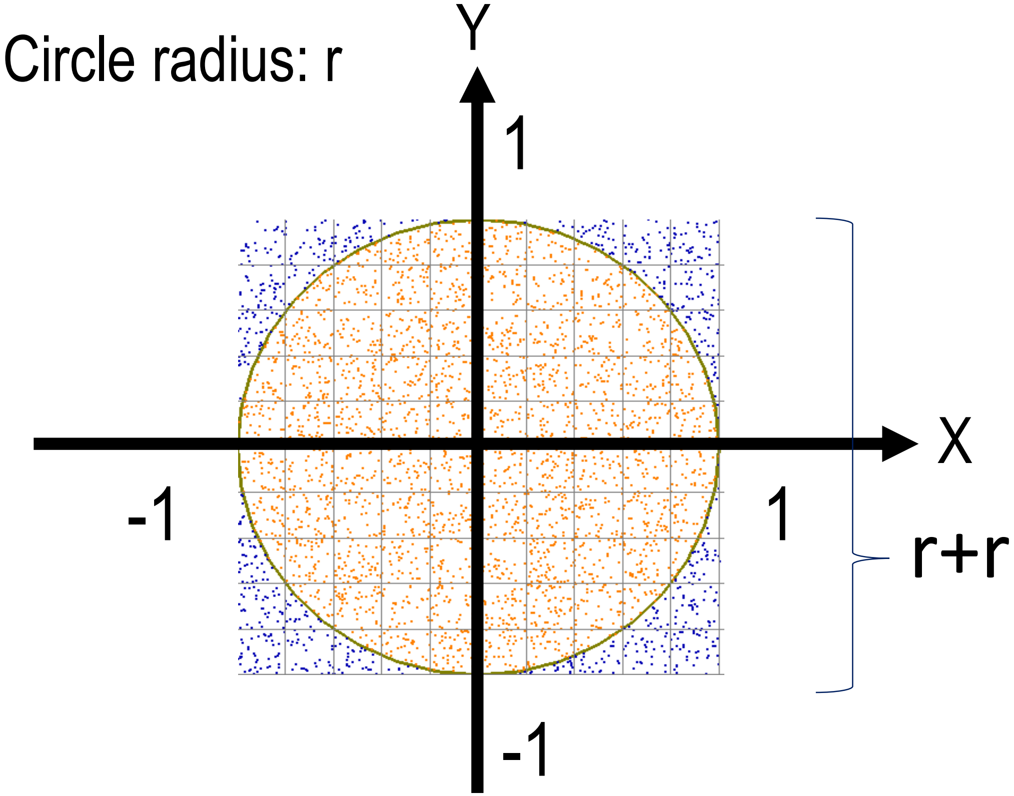
Monte Carlo Simulation

A technique uses random numbers and probability to solve problems.

We use the Monto Carlo simulation to estimate π.

Given a square and an inscribed circle, generate randomly sample points.

Circle radius: r



$$\text{circleArea} = r^2\ \pi$$
$$\text{squareArea} = (r+r)*(r+r) = 4\ r^2$$

$$\text{circleArea / squareArea} = \ \pi / 4$$

$$\pi = 4*(\text{circleArea / squareArea })$$

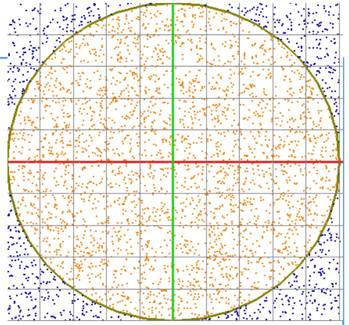
Generate randomly 100000 points inside the square in a uniform manner.

$$\pi \ \approx 4 * \underbrace{\text{numberOfInteriorPoints / 100000}}_{\text{circleArea / squareArea}}$$

Monte Carlo Simulation

A technique uses random numbers and probability to solve problems.

We use the Monto Carlo simulation to estimate π.



The process

- Identify the relation between the samples and the domain of interest
- Generate the samples in domain
- Determine the samples that match the criteria
- Compute the result
- Report the result

$$\text{circleArea} = r^2\,\pi$$

$$\text{squareArea} = (r+r)*(r+r) = 4\,r^2$$

$$\text{circleArea} / \text{squareArea} = \pi / 4$$

$$\pi = 4*(\text{circleArea} / \text{squareArea}\ )$$

Generate randomly 100000 points inside the square in a uniform manner.

$$\pi \approx 4 * \underbrace{\text{numberOfInteriorPoints} / 100000}_{\text{circleArea / squareArea}}$$

Case Study: Requirement Specification

Implement a program to use Monte Carlo simulation to estimate π.

1. Input the number of points
2. Input the radius of the circle
3. Randomly generate sample points inside a square enclosing the circle
4. Estimate π
5. Display π

Design and implement the system now

Intended Learning Outcomes

- Describe the process of software development
- Define a base class
- Define a derived class
- Describe the process of Monte Carlo simulation for estimating π

Supplemental Material

Declaration and definition

```
void foo( );            // forward declaration

void g( ) {
      foo( );
}

// definition
void foo( ) {
                        //Implementation of the function body
}
```

Converting Decimals to Hexadecimals

To convert a decimal number d to a hexadecimal number:

We need to find the hexadecimal digits

$h_n, h_{n-1}, h_{n-2}, ... , h_2, h_1,$ and h_0 such that

$$d = h_n\ 16^n + h_{n-1}\ 16^{n-1} + \ldots + h_1\ 16^1 + h_0 16^0$$

For example, d = 35 = 2*16 + 3*1

$h_1 = 2, h_0 = 3$

Hexadecimal Digits

0
1
2
3
4
5
6
7
8
9

A = 10 (Dec)
B = 11 (Dec)
C = 12 (Dec)
D = 13 (Dec)
E = 14 (Dec)
F = 15 (Dec)

Converting Decimals to Hexadecimals

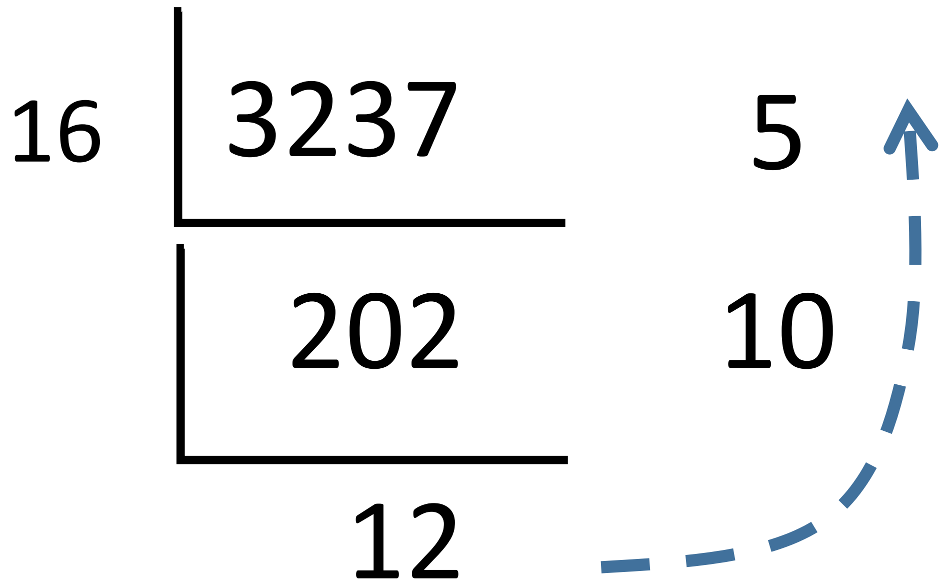
$d = h_n\ 16^n + h_{n-1}\ 16^{n-1} + \ldots + h_1\ 16^1 + h_0 16^0$

- 37
- 37/16 = 2, remainder 5
- Ans: 25h, or 0x25
- 25h = 2*16 + 5

- 258
- 258/16 = 16, remainder 2
- 16/16 = 1, remainder 0
- Ans: 102h, or 0x102

Converting Decimals to Hexadecimals

$d = h_n\ 16^n + h_{n-1}\ 16^{n-1} + \ldots + h_1\ 16^1 + h_0 16^0$



16

3237

5

202

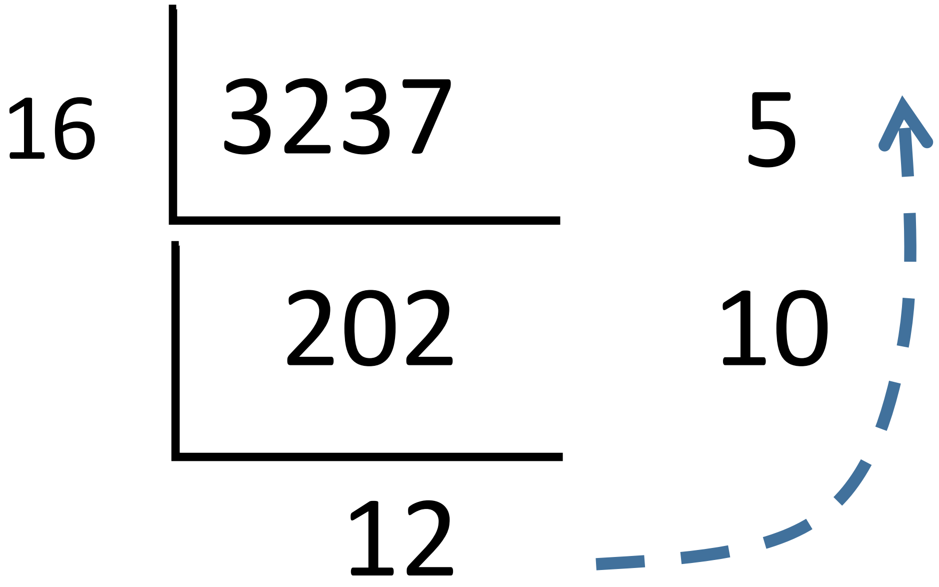
10

12

Answer: CA5

Converting Decimals to Hexadecimals

$d = h_n\ 16^n + h_{n-1}\ 16^{n-1} + \ldots + h_1\ 16^1 + h_0 16^0$



16 | 3237 5

202 10

12

(3237/(16*16))%16

(3237/(16))%16

(3237)%16

Answer: CA5

🟧 🟩 🟦

Converting Decimals to Hexadecimals System Analysis

- Input
- Output

Converting Decimals to Hexadecimals System Design

- Data fields
- Methods (functions)

Display prime numbers Exercise

- What is a prime number?
- A positive integer greater than 1 and its only positive divisor is 1 or itself.

- Write a program to show prime numbers smaller than or equal to a number n.

e.g., n = 10

2, 3, 5, 7