

# Architecture, Design and Implementation of Embedded Systems for Real-Time Applications

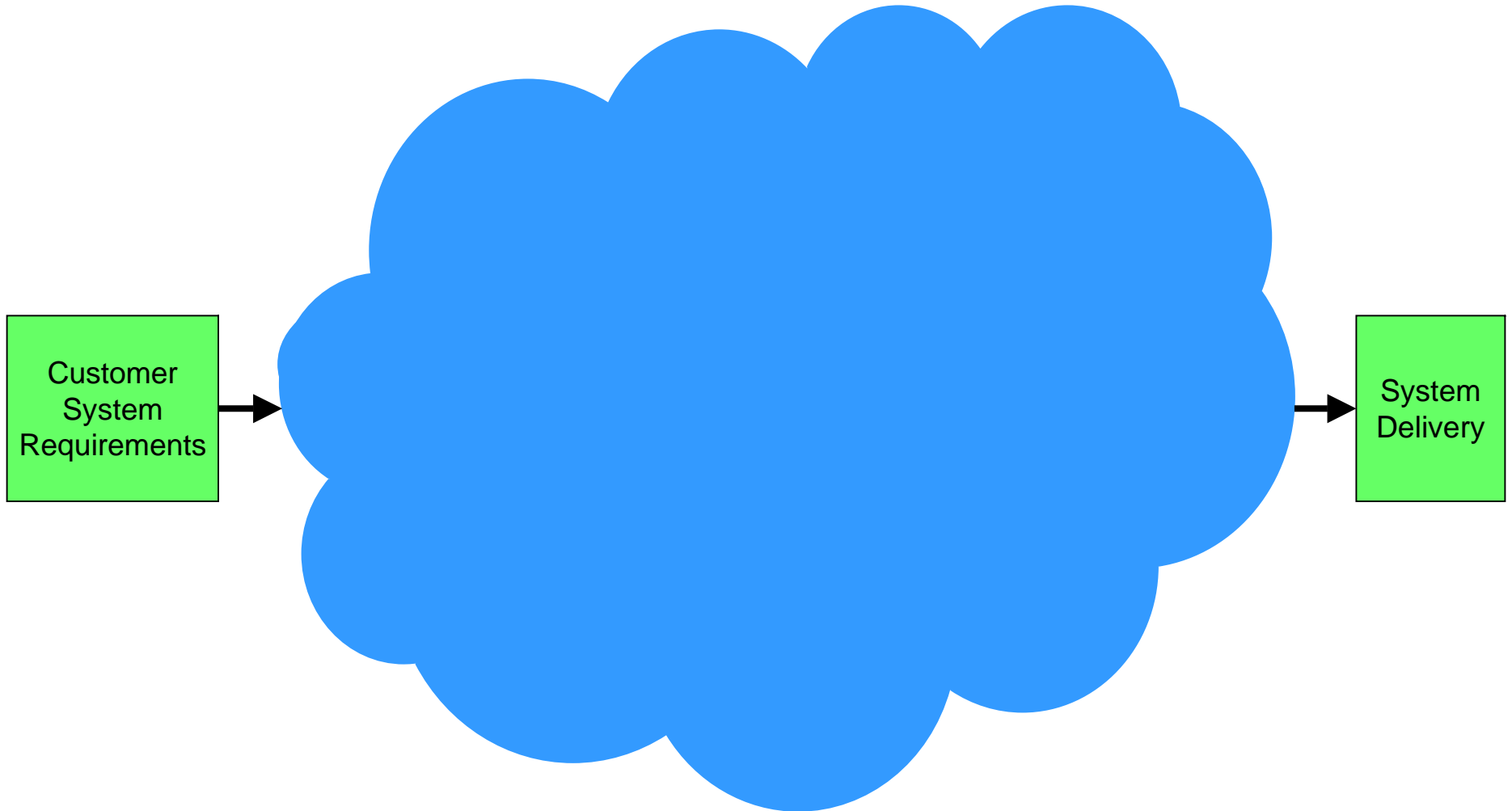
## CpE-450 Spring 05

Class 10

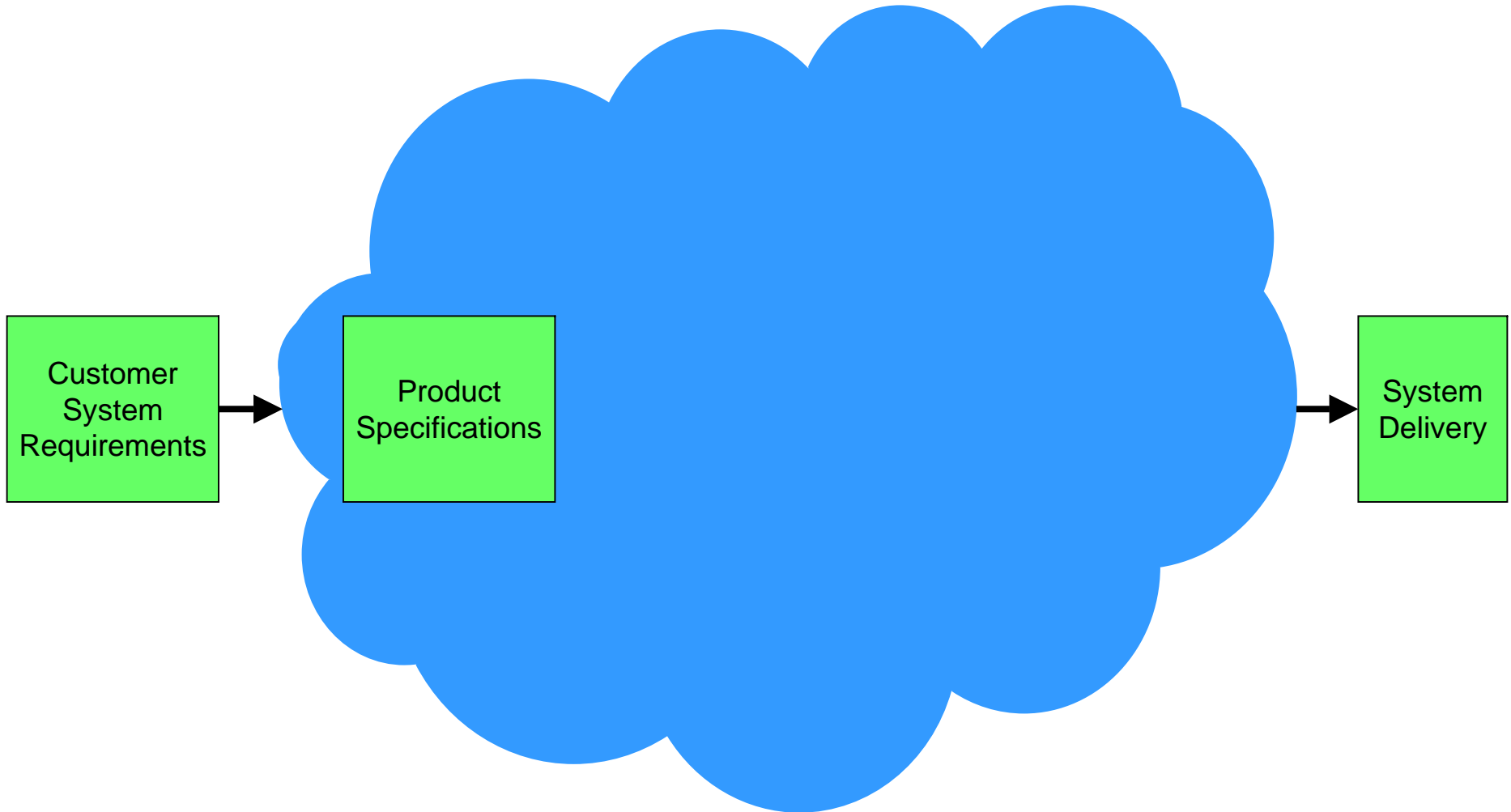
Bruce McNair

[bmcnair@stevens.edu](mailto:bmcnair@stevens.edu)

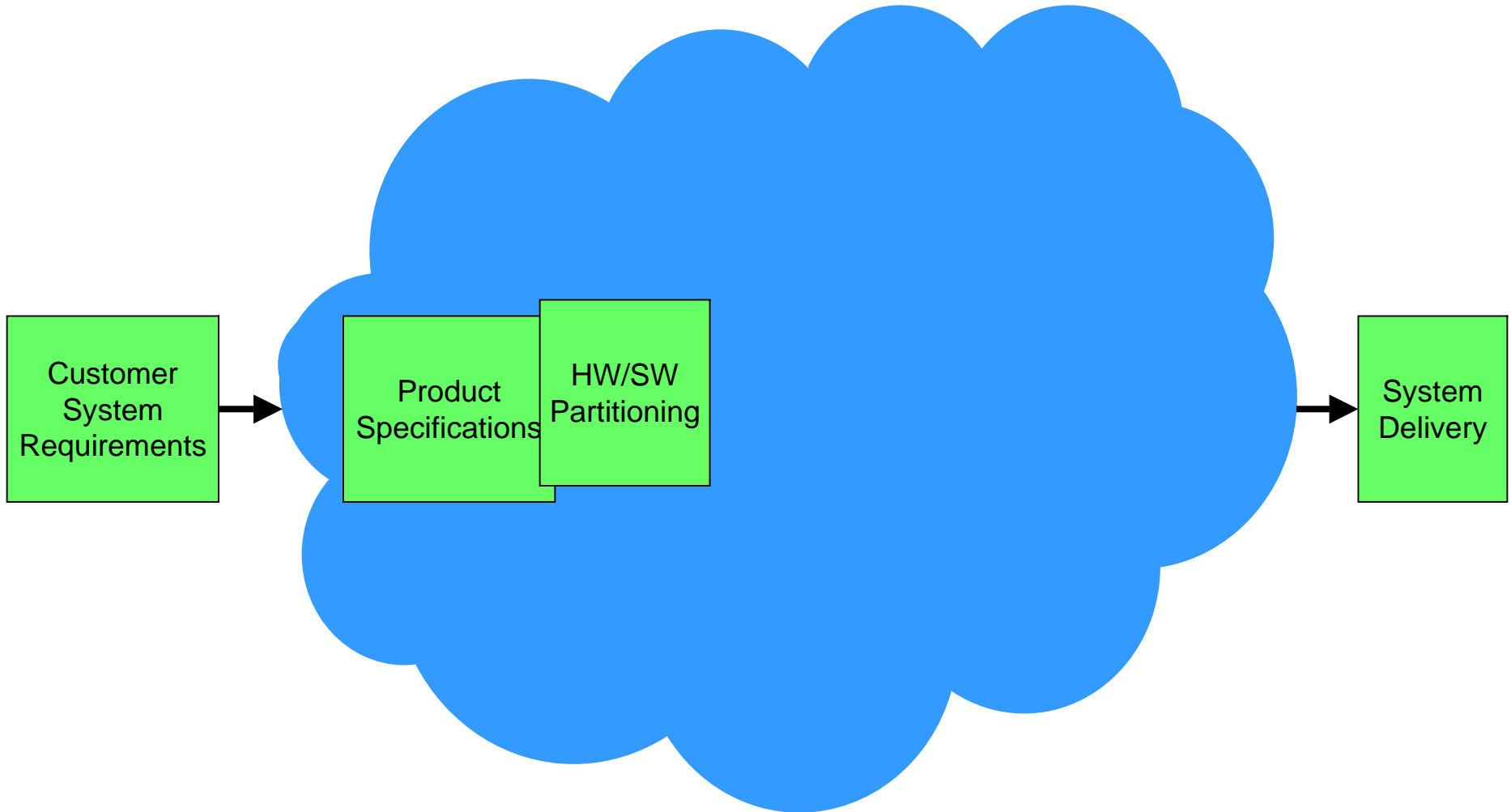
# Embedded System Development



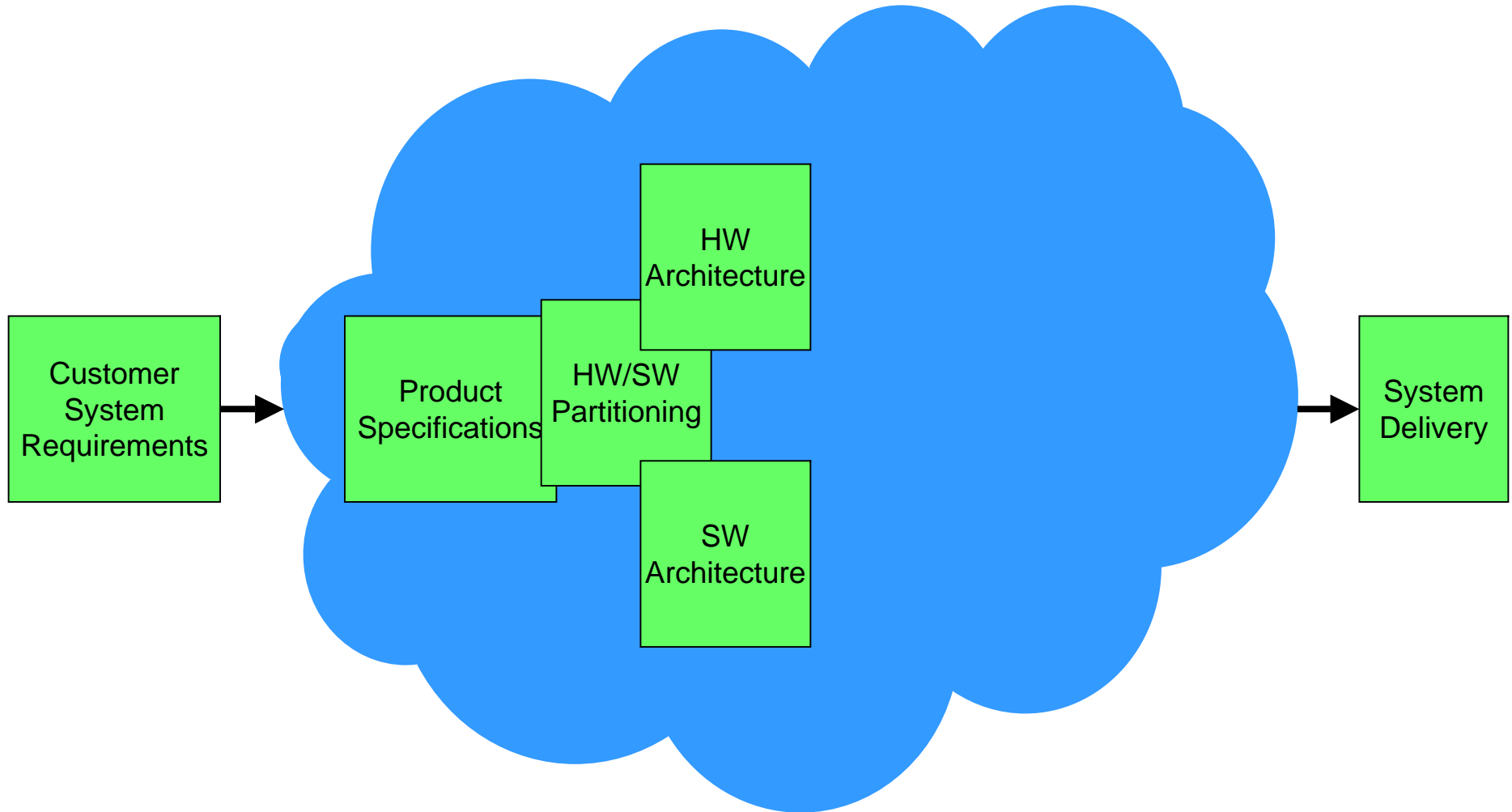
# Embedded System Development



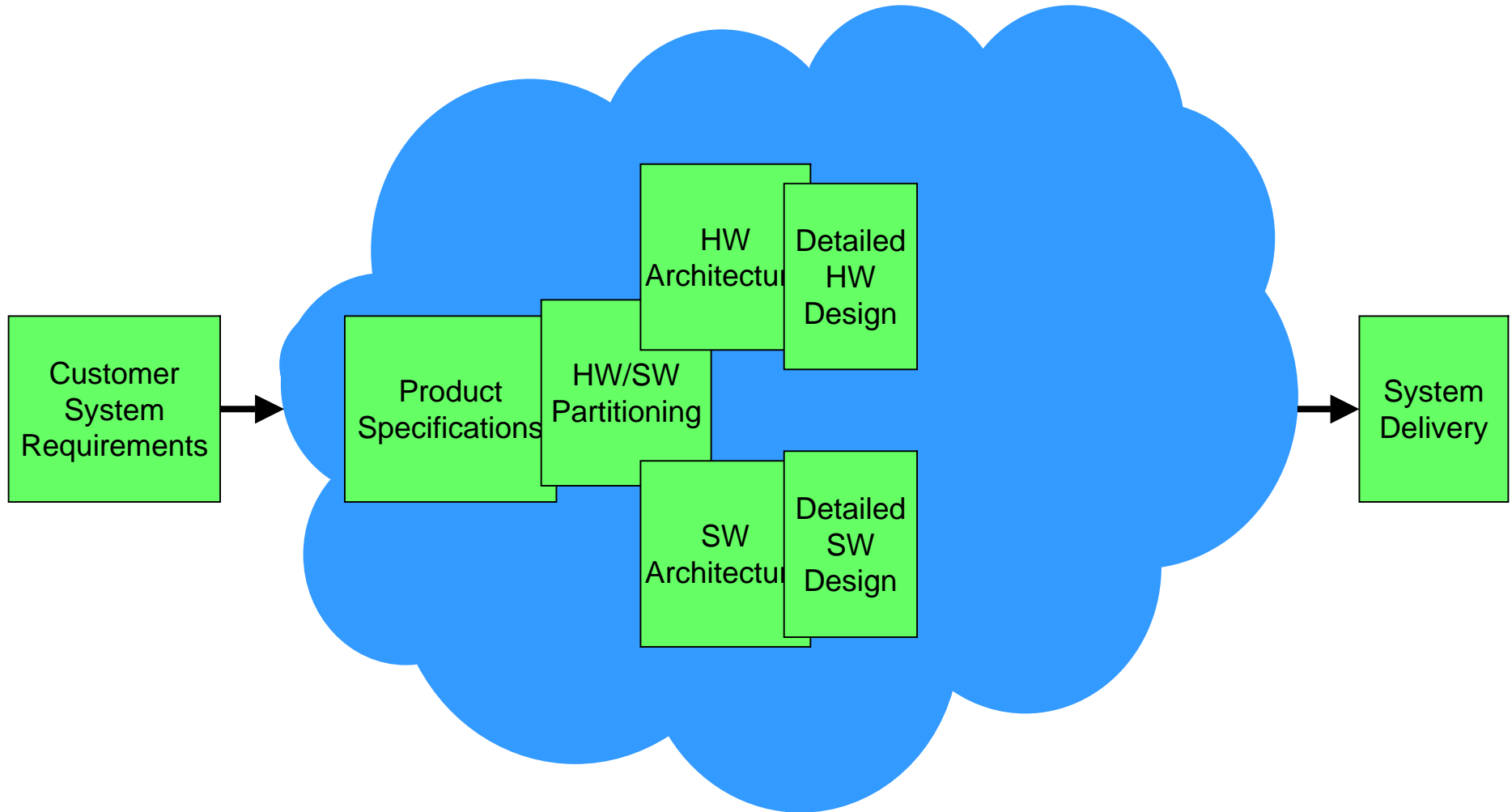
# Embedded System Development



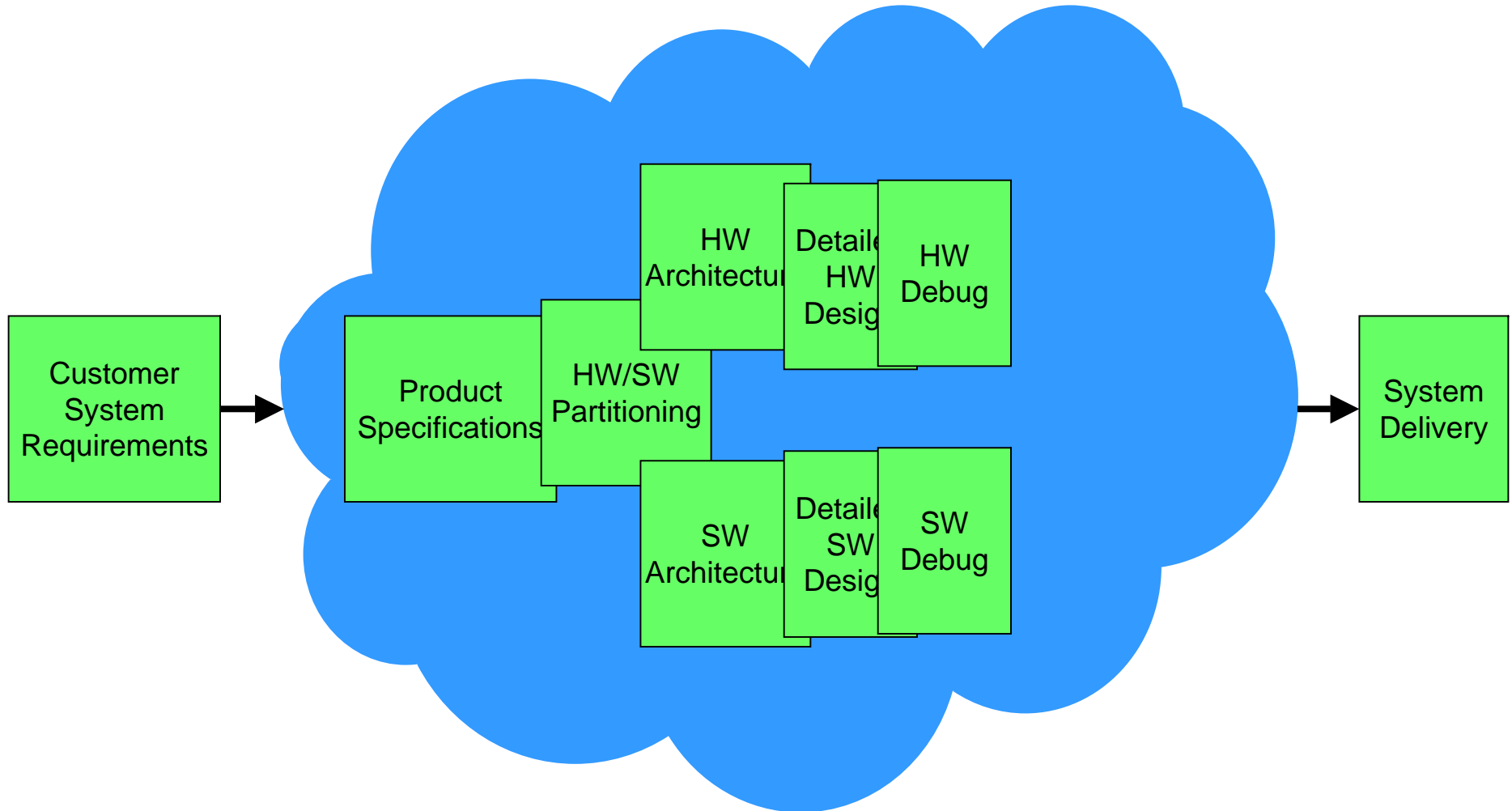
# Embedded System Development



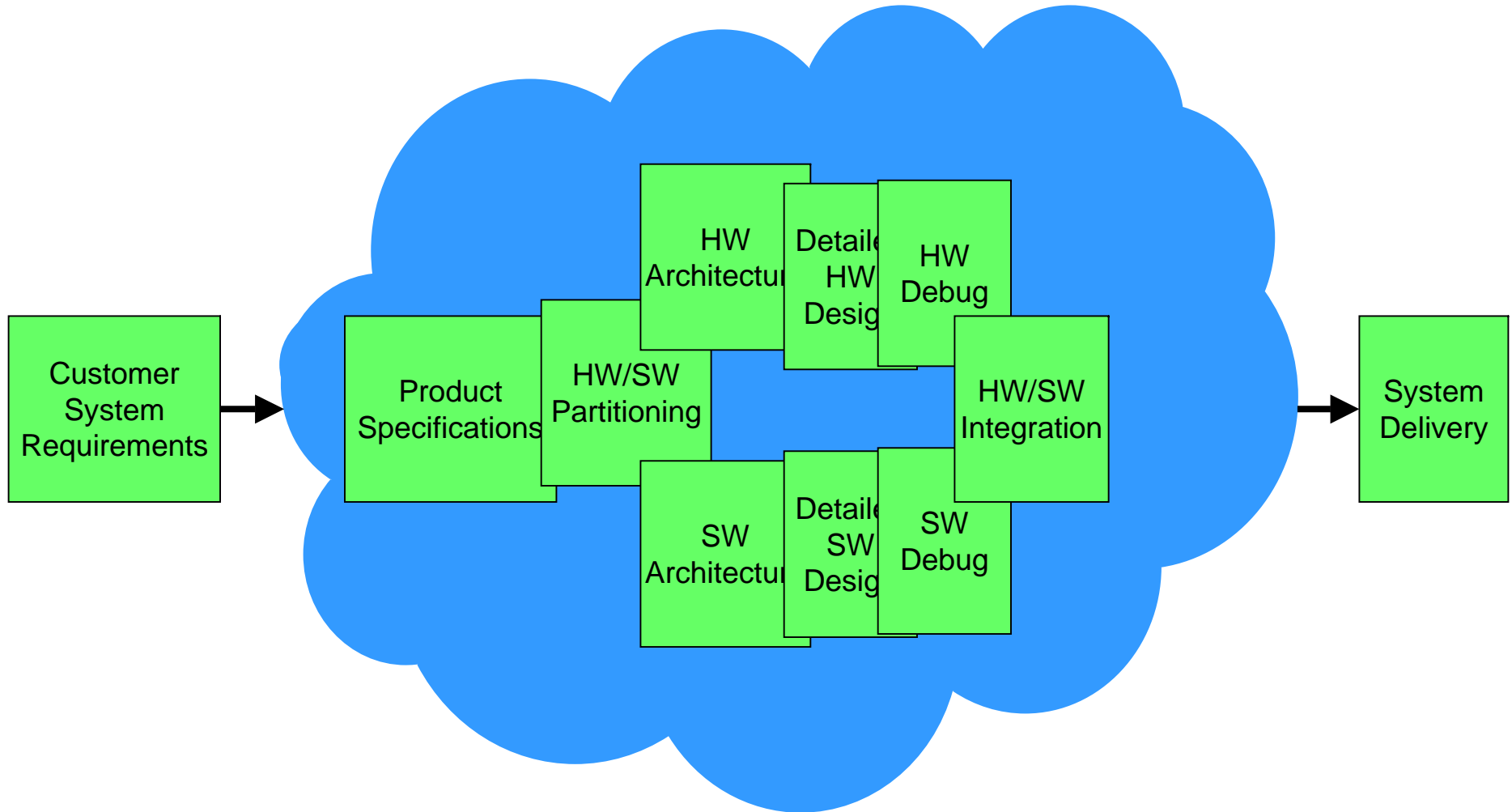
# Embedded System Development



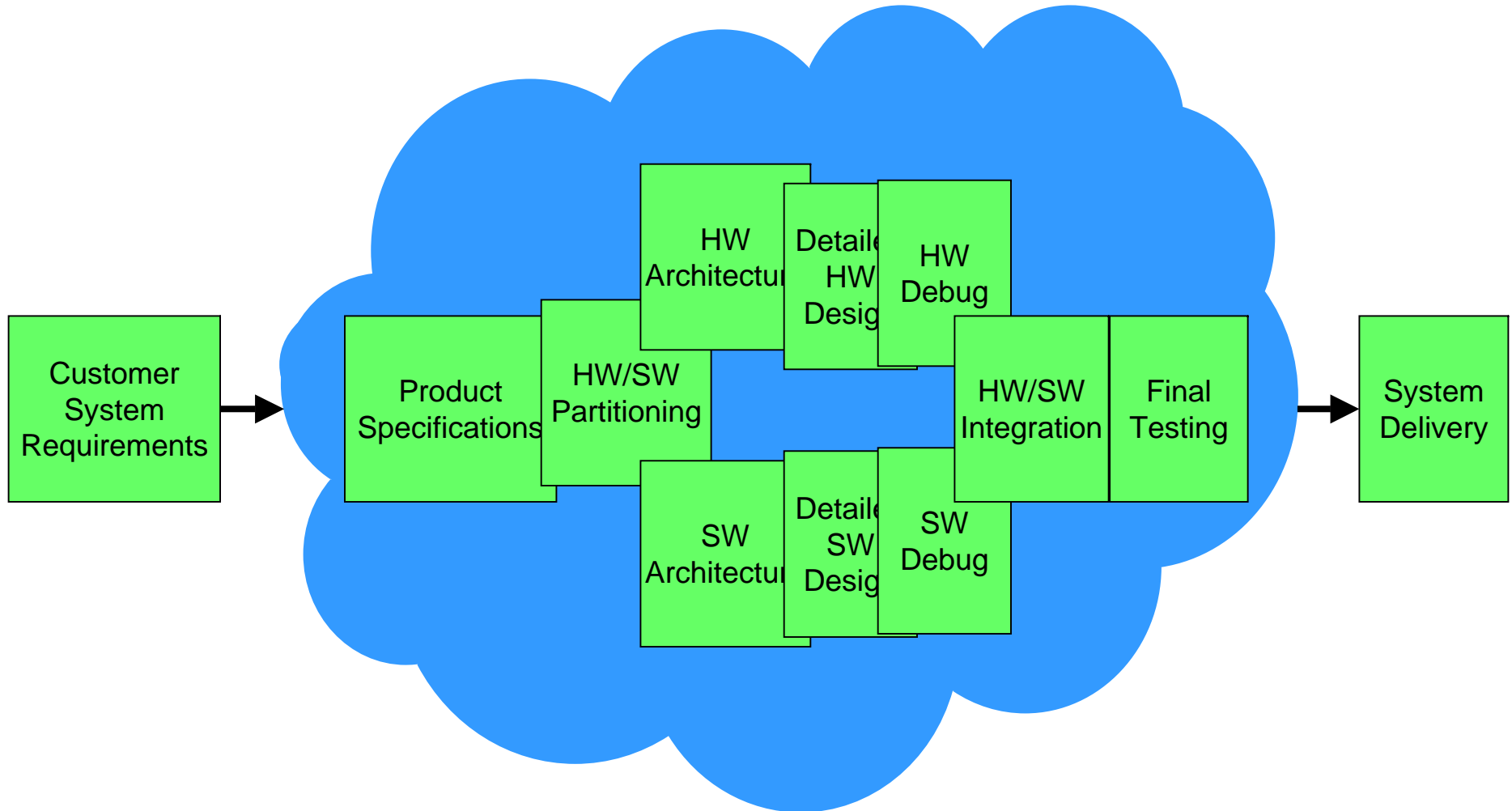
# Embedded System Development



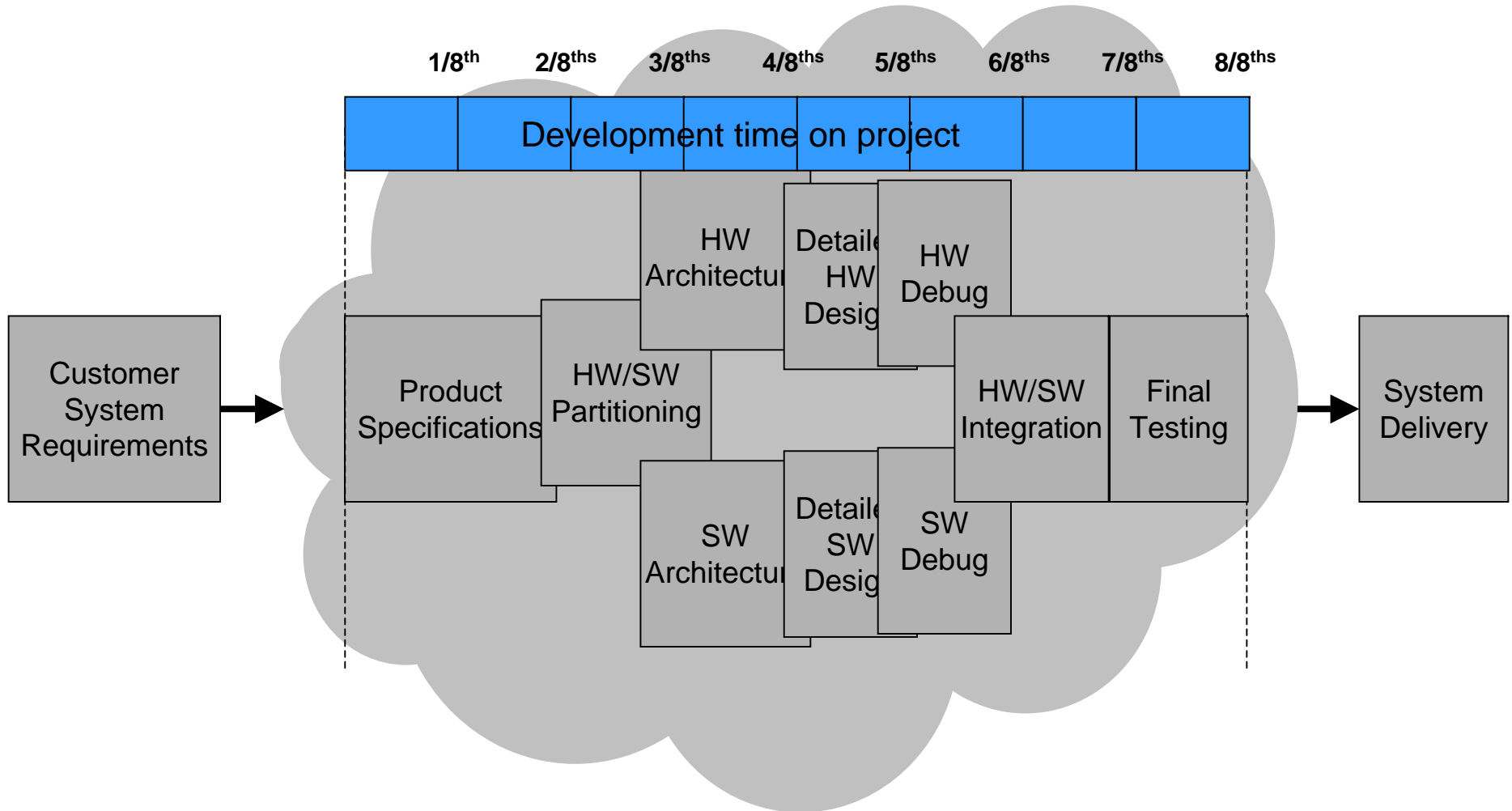
# Embedded System Development



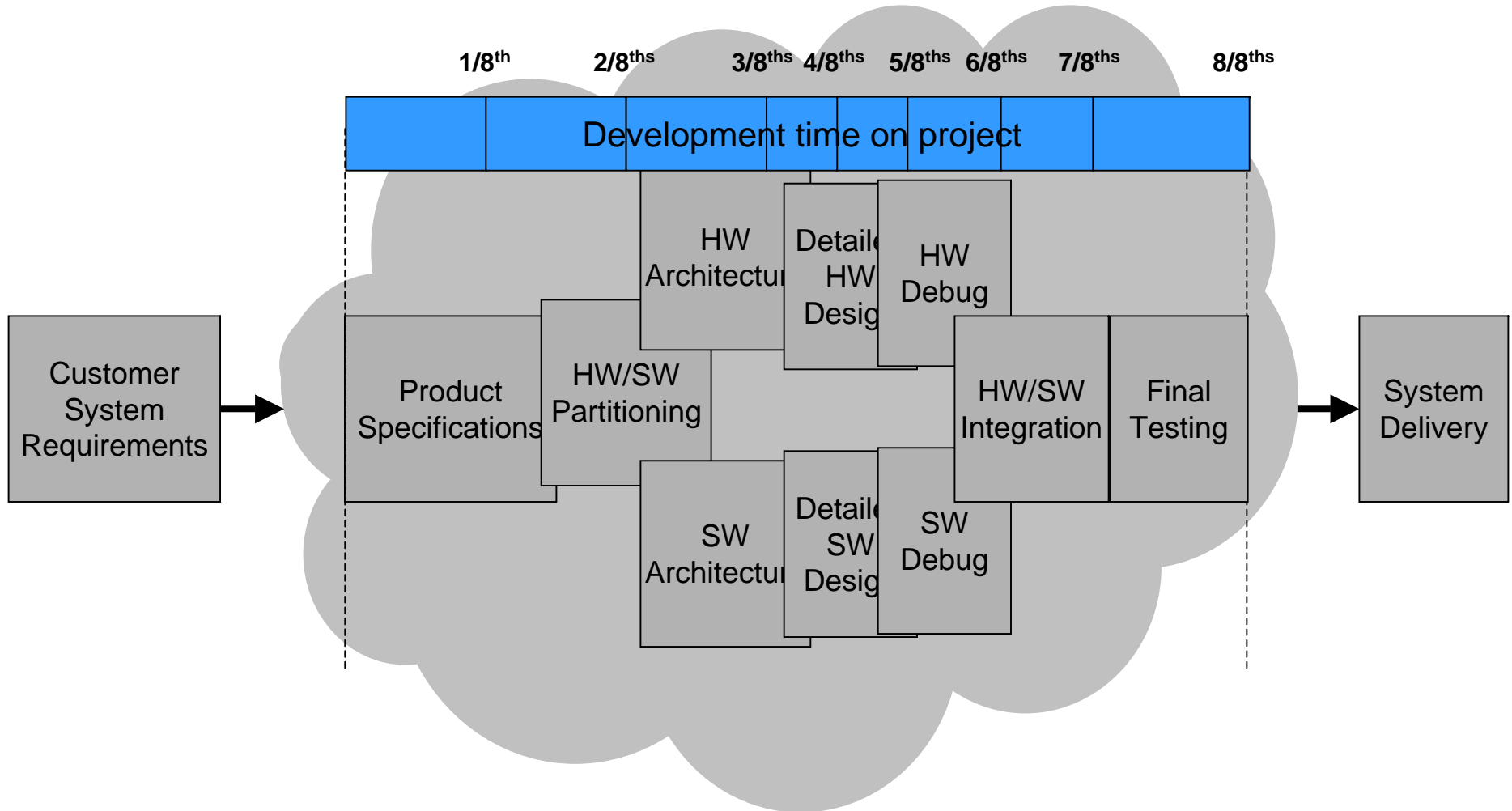
# Embedded System Development



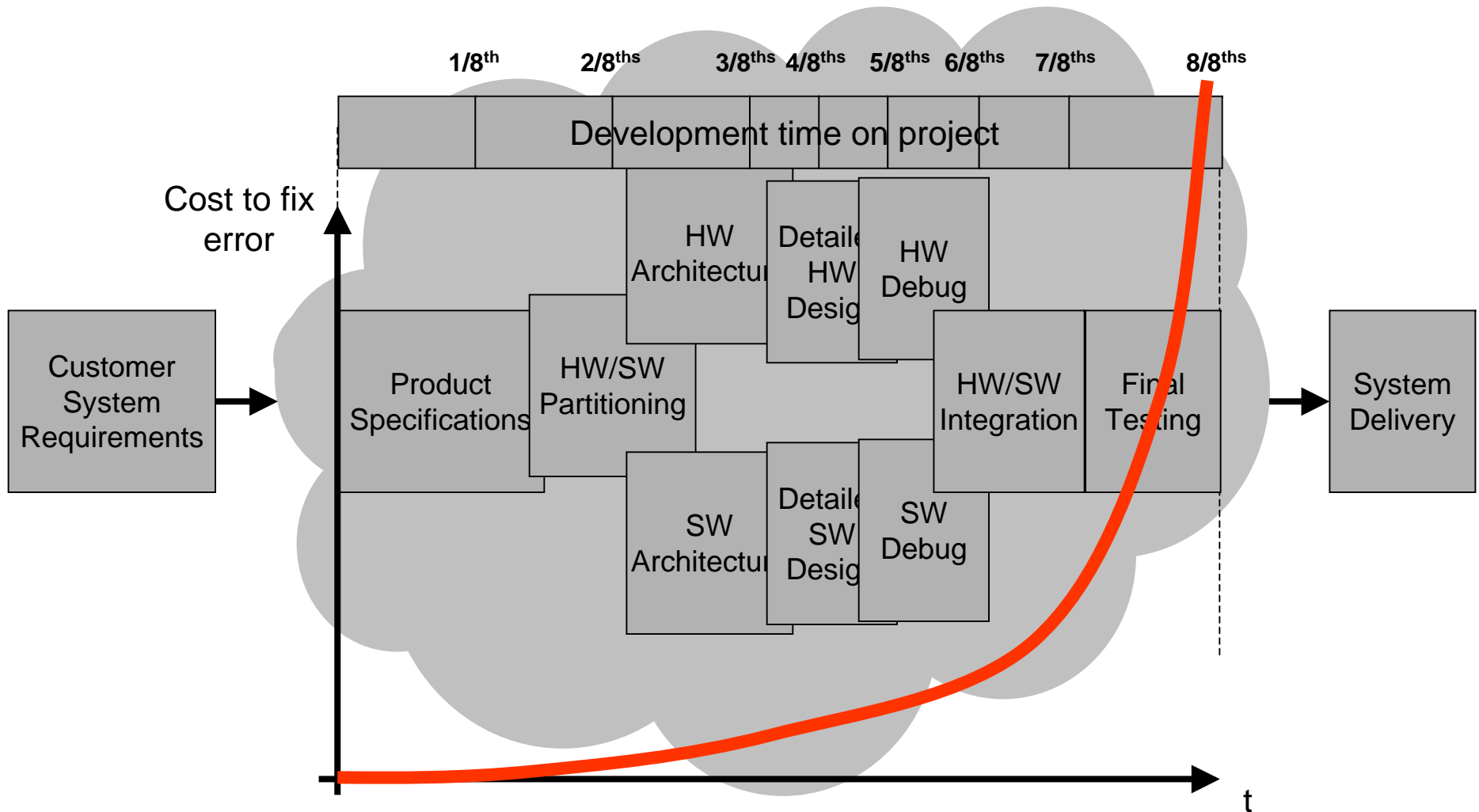
# Embedded System Development



# Embedded System Development



# Embedded System Development



# HW/SW Design Errors

- Consider the following program segment:

```
unsigned long int x;
x = 0x01234567;
y = first_byte(x);

/* what is the value of y? */

byte first_byte(unsigned long int in)
{
    return( (0xFF000000&in)>>24);
}
```

# HW/SW Design Errors

- Consider the following program segment:

```
unsigned long int x;  
x = 0x01234567;  
y = first_byte(x);  
  
/* what is the value of y? */  
  
byte first_byte(unsigned long int in)  
{  
    return( (0xFF000000&in)>>24);  
}
```



y = 0x67

# HW/SW Co-design Errors

- Consider the following program segment:
- Sun workstations and TI DSPs address memory differently than SGI workstations and Intel '86 family PCs

```

unsigned long int x;
x = 0x01234567;
y = first_byte(x);

/* what is the value of y? */

byte first_byte(unsigned long int in)
{
    return( (0xFF000000&in)>>24);
}
    
```

“Big Endian vs. Little Endian” arithmetic

Depends on processor arithmetic architecture



y = 0x67



y = 0x01

# More on HW/SW Co-design Errors

- Previous case was easy to debug. What about this one:

```
struct port
{
    byte status;
    byte data;
}
.
.
.

port.status = DATA_RATE | BITS_7 | NO_PARITY;
.
.
.
port.data = *x++;
```

# Still More on HW/SW Co-design Errors

- First case was easy to debug. Second case was a little harder.

What about this one:

```
boolean PN_generator (long unsigned int *state)
{
    next_bit = parity(state & MASK);
    *state = *state>>1 | next_bit<<31;
}
```

```
boolean parity(long unsigned int x)
{
    boolean p=0;
    for(i=0;i<32;i++)
    {
        p ^= x&0x01;
        x = x>>1;
    }
    return(p);
}
```

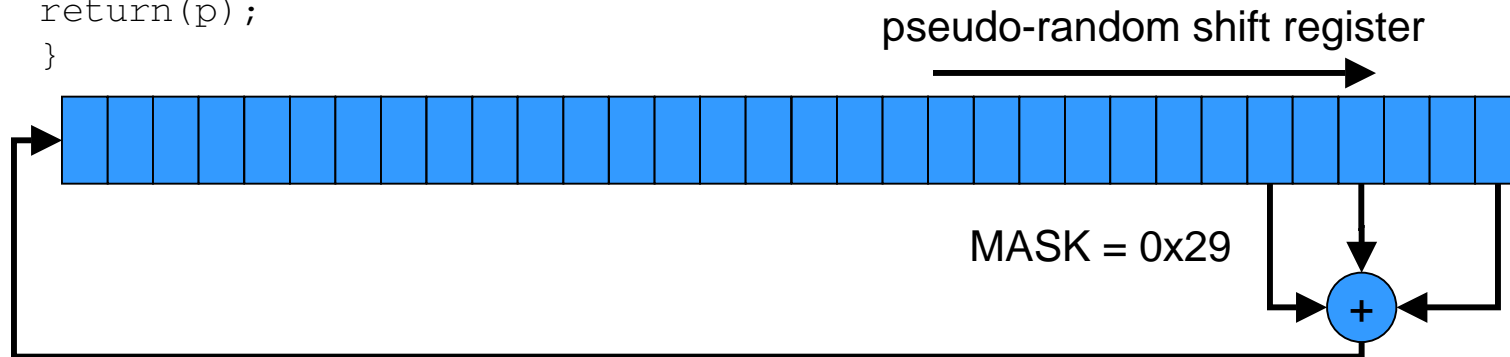
# Still More on HW/SW Co-design Errors

- First case was easy to debug. Second case was a little harder.

What about this one:

```
boolean PN_generator (long unsigned int state)
{
    next_bit = parity(state & MASK);
    state = state>>1 | next_bit<<31;
}
```

```
boolean parity(long unsigned int x)
{
    boolean p=0;
    for(i=0;i<32;i++)
    {
        p ^= x&0x01;
        x = x>>1;
    }
    return(p);
}
```



# Still More on HW/SW Co-design Errors

- First case was easy to debug. Second case was a little harder.

What about this one:

```
boolean PN_generator (long unsigned int state)
{
    next_bit = parity(state & MASK);
    state = state>>1 | next_bit<<31;
}
```

```
boolean parity(long unsigned int x)
{
    boolean p=0;
    for(i=0;i<32;i++)
    {
        p ^= x&0x01;
        x = x>>1;
    }
    return(p);
}
```

Or:

```
unsigned int rand(unsigned int x)
{
    return(mod(A*x+B,C));
}
```

A, B, and C are constants chosen to generate maximal length random sequence (or do they?)

