Designing a PSS Reuse Strategy

Matthew Balance

Mentor, A Siemens Business

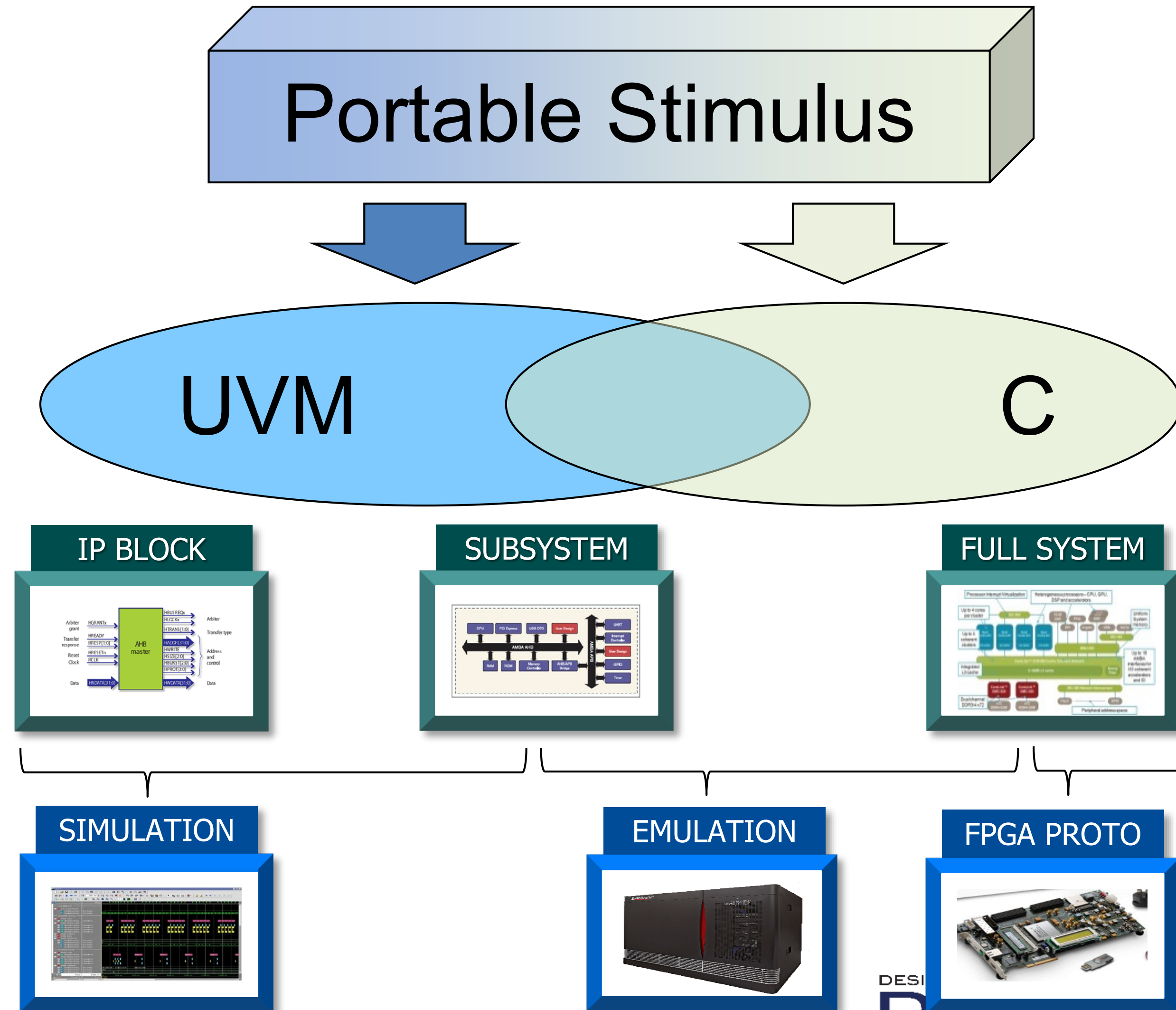
Mentor®
A Siemens Business

accellera
SYSTEMS INITIATIVE

2019
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
EUROPE

Portable Stimulus Vision

- Ambitious Scope
  - Portable across verification levels
  - Portable across verification engines
- Ambitious scope is helpful!
  - Can be applied in many places
  - Can be applied in many ways
- Also complicates adoption
  - Many possible ways to adopt



accellera SYSTEMS INITIATIVE

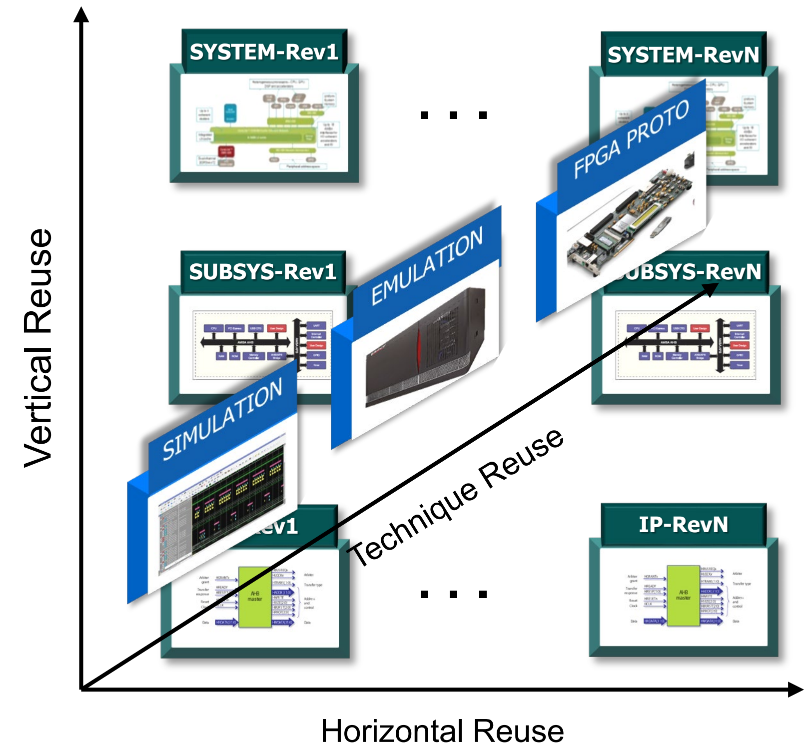
DESIGN AND VERIFICATION DVCON CONFERENCE AND EXHIBITION EUROPE

Reuse Strategy

- Identify a primary target application
  - Keep it simple, initially
  - Can always extend
  - Avoids getting mired in overthinking reuse
- Identify assets to reuse
  - Every organization has some
- Design new assets with reuse in mind
  - Let your primary target application be the guide here

Three Axes of Reuse

- Vertical Reuse
  - Reuse across verification levels
  - PSS test intent created at IP reusable at Subsystem and SoC
  - Reuse accelerates test-creation process at Subsystem and SoC
- Horizontal Reuse
  - Reuse across design revisions
  - PSS description is easily-customizable
- Technique Reuse
  - Reuse same automated-stimulus techniques across platforms
  - Use same modeling techniques in simulation and prototype
  - Automation accelerates test-creation process

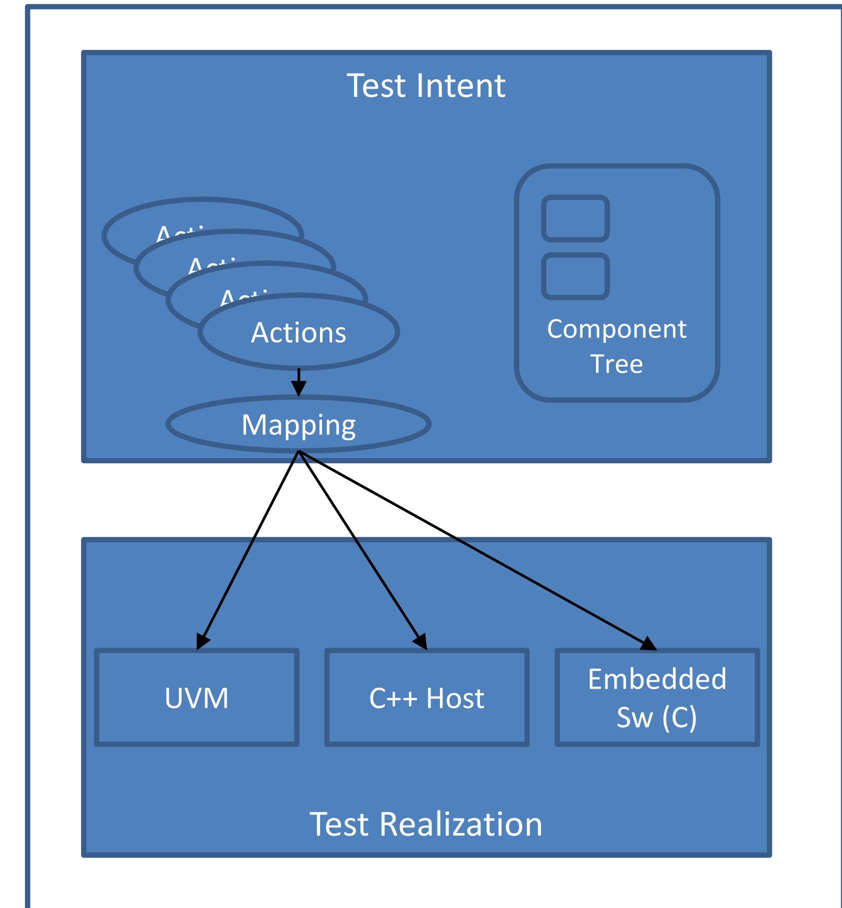


Three Axes of Reuse Tradeoffs

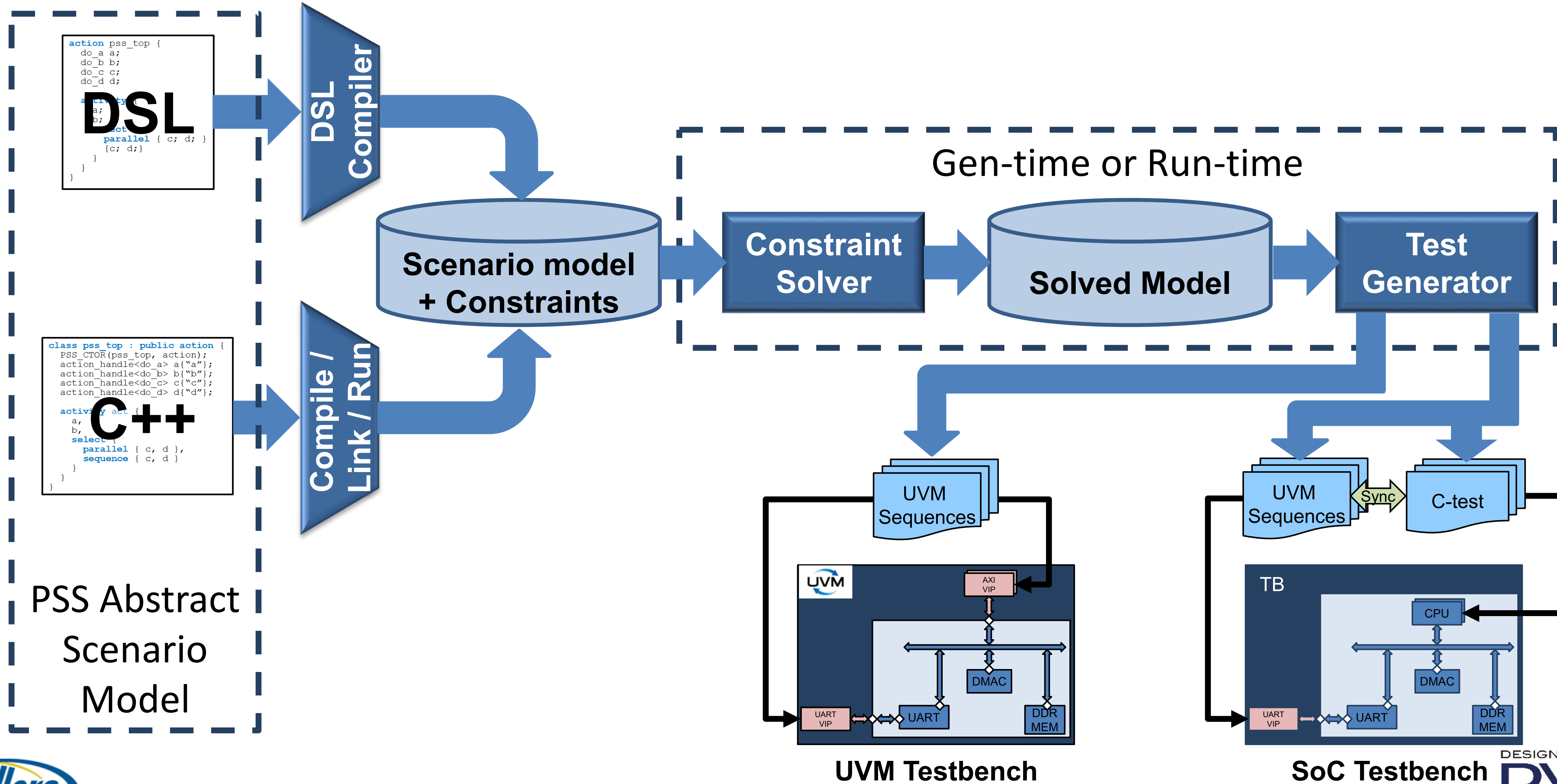
- Vertical Reuse
  - Provides the greatest benefit
  - Has the highest initial cost
- Horizontal Reuse
  - Provides strong benefits
  - Has moderate startup costs
- Technique Reuse
  - Has the lowest initial cost
  - Provides (relatively) the smallest benefit

Anatomy of a PSS Description

- Declarative high-level specification
  - Actions
  - Constraints
  - Rules
- Link to environment-specific test realization
  - SystemVerilog
  - C/C++
  - Assembly



Creating Tests with Portable Stimulus



accellera SYSTEMS INITIATIVE

2019 DESIGN AND VERIFICATION DVCON CONFERENCE AND EXHIBITION EUROPE

Reusable Assets
Constraints

- PSS descriptions are heavily constraint-based
  - PSS is a *declarative* specification

- SystemVerilog constraints are also a *declarative* specification
  - Very similar (nearly identical) format to PSS

- Reusing SV constraints can jump-start PSS creation
  - Reuse already-developed and debugged logic

Reusable Assets
Test Realization

- Existing functions can often be reused as test realization
- May require some modification
  - Simplify arguments
  - Align with other functions' calling convention
- Reuse saves significant time over developing

```
task init_single_transfer(
  int unsigned     channel,
  int unsigned     src,
  int unsigned     inc_src,
  int unsigned     dst,
  int unsigned     inc_dst,
  int unsigned     sz
  );
  wb_dma_ch ch = m_regs.ch[channel];
  uvm_status_e status;
  uvm_reg_data_t value;


  // Disable the channel
  ch.CSR.read(status, value);
  value[0] = 0;
  ch.CSR.write(status, value);
```


accellera SYSTEMS INITIATIVE
2019 DESIGN AND VERIFICATION DVCON CONFERENCE AND EXHIBITION EUROPE

Designing for Reuse Libraries

- Designing PSS for reuse a key consideration
- Consider defining common data structures
  - Address/size data buffer
- Consider defining common base types
  - Actions with common fields
  - IP-specific common types

```
struct data_mem_t {
  rand bit[31:0]     addr;
  rand bit[31:0]     sz;
}
```

```
abstract action dma_dev_a : pvm_dev_a {
  // All transfers involve a channel
  rand bit[7:0] in [0..7]   channel;
  // Size of each transfer
  rand bit[4] in [1,2,4] trn_sz;
}

/**
 * Transfer memory-to-memory
 */
action mem2mem_a : dma_dev_a {
  input data_ref_mem_b      dat_i;
  output data_ref_mem_b     dat_o;
  . . .
}
```

accellera SYSTEMS INITIATIVE

CONFERENCE AND EXHIBITION EUROPE

Designing for Reuse Checking

- Designing reusable checking is challenging
  - Visibility is different in different environments
  - Requirements are different
  - Performance is different
- Focus on making functional tests portable
  - Is the end result correct?
- Add in environment-specific checks as needed
  - Detailed scoreboards
  - Environment-specific checking actions

Designing for Reuse
Test Realization

- Design test-realization for reuse
- Specify common APIs
  - SystemVerilog
  - Embedded C
  - Host C
- Doesn't cost much
  - But avoids complexity

```
task mem2mem(
  int unsigned                      channel,
  int unsigned                      src,
  int unsigned                      dst,
  int unsigned                      sz);
  init_single_transfer(channel, src, 1, dst, 1, sz);
  wait_complete_irq(channel);
endtask
```

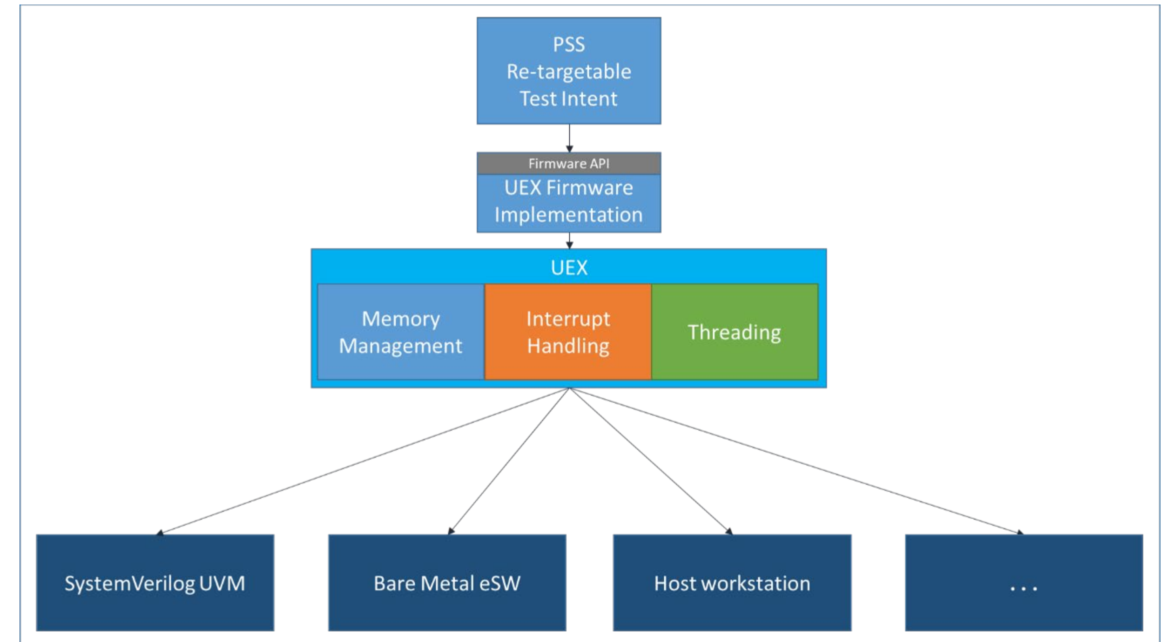
```
void wb_dma_dev_mem2mem(
    uint32_t                             devid,
    uint32_t                             channel,
    uint32_t                             src,
    uint32_t                             dst,
    uint32_t                             sz,
    uint32_t                             trn_sz) {
  wb_dma_dev_t        *drv = (wb_dma_dev_t *)uex_get_device(devid);
  uint32_t csr, sz_v;

  // Disable the channel
  csr = uex_ioread32(&drv->regs->channels[channel].csr);
  csr &= ~(1);
  uex_iowrite32(csr, &drv->regs->channels[channel].csr);
```


accellera SYSTEMS INITIATIVE
CONFERENCE AND EXHIBITION EUROPE

Designing for Reuse
Consider a Hardware Abstraction Layer

- A Hardware Abstraction Layer (HAL) makes test realization portable
- Provides common API
- Provides different implementations
- Simplifies test-realization code



accellera SYSTEMS INITIATIVE

2019 DESIGN AND VERIFICATION DVCON CONFERENCE AND EXHIBITION EUROPE

Designing for Reuse
Consider a Hardware Abstraction Layer

- Micro-Executor (UEX) is one example of a HAL
- Example code shows an ISR
  - uex_ioread32 access memory
  - uex_event_signal notifies waiting

```
static void wb_dma_dev_irq(struct uex_dev_s *devh) {
  wb_dma_dev_t *dev = (wb_dma_dev_t *)devh;
  uint32_t i;
  uint32_t src_a;

  src_a = uex_ioread32(&dev->regs->int_src_a);

  // Need to spin through the channels to determine
  // which channel to activate
  for (i=0; i<8; i++) {
    if (src_a & (1 << i)) {
      // Read the CSR to clear the interrupt
      uint32_t csr = uex_ioread32(&dev->regs->chan[i].csr);
      dev->status[i] = 0;
      uex_event_signal(&dev->xfer_ev[i]);
    }
  }
}
```

accellera SYSTEMS INITIATIVE

DESIGN AND VERIFICATION DVCON CONFERENCE AND EXHIBITION EUROPE

Summary

- Creating a Reuse Strategy helps to get the biggest benefit from PSS
  - Keeps focus on a primary application
  - Reduces complexity
  - Can always expand scope on subsequent projects
- Identify a primary target application
- Identify assets to reuse
- Design new assets with reuse in mind

Questions

Finalize slide set with questions slide