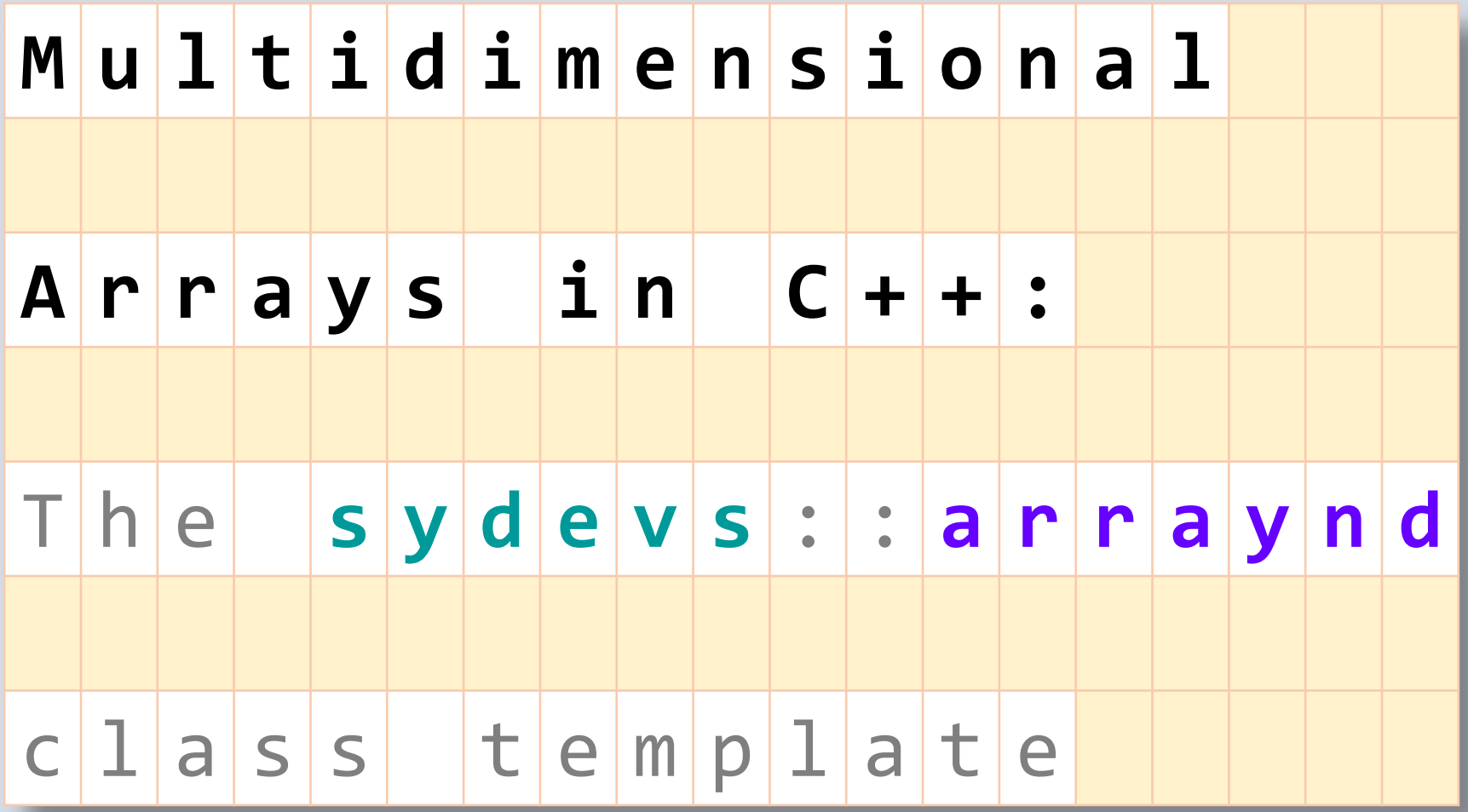
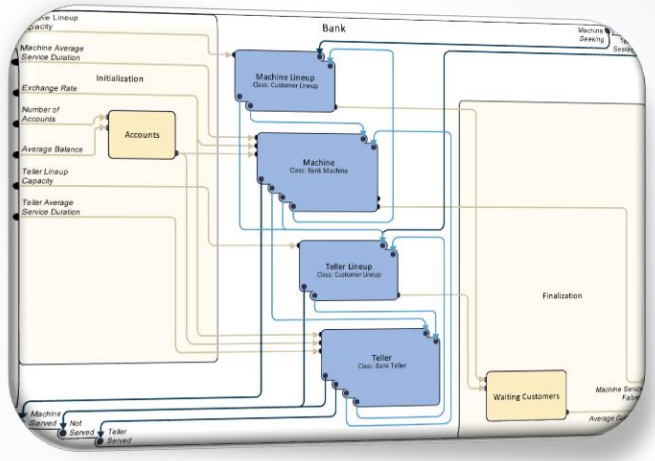
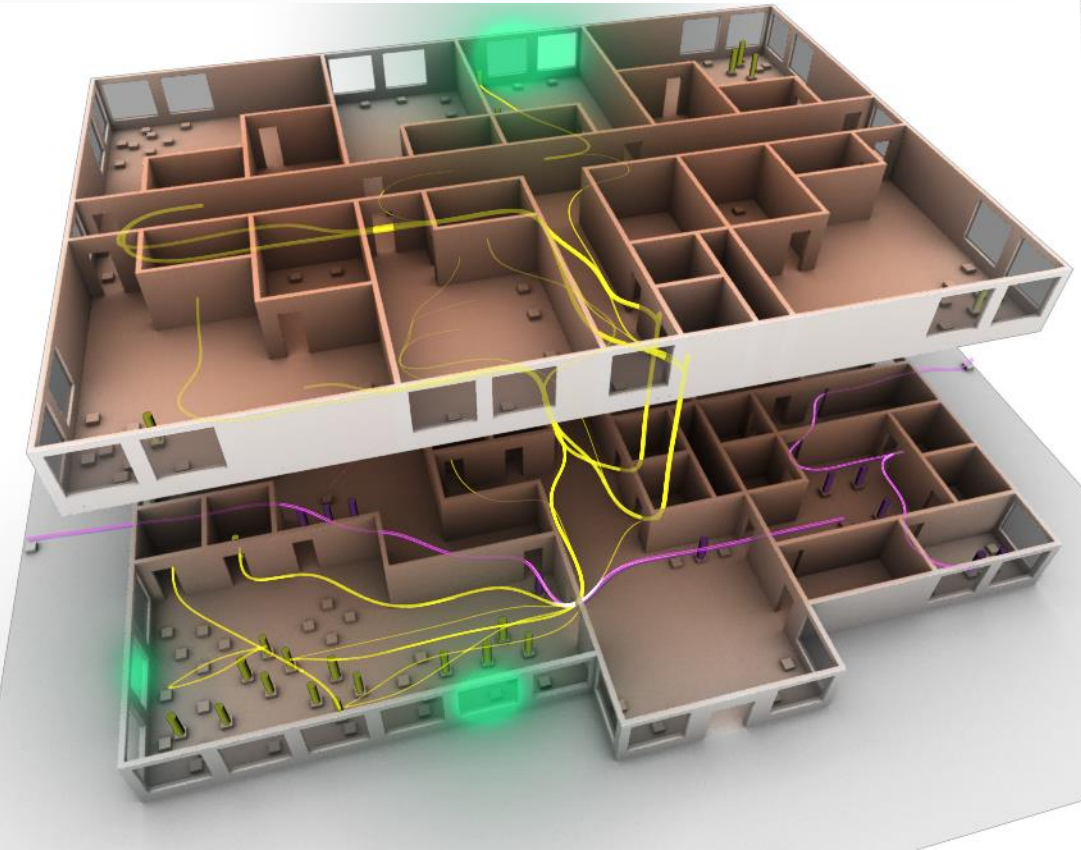
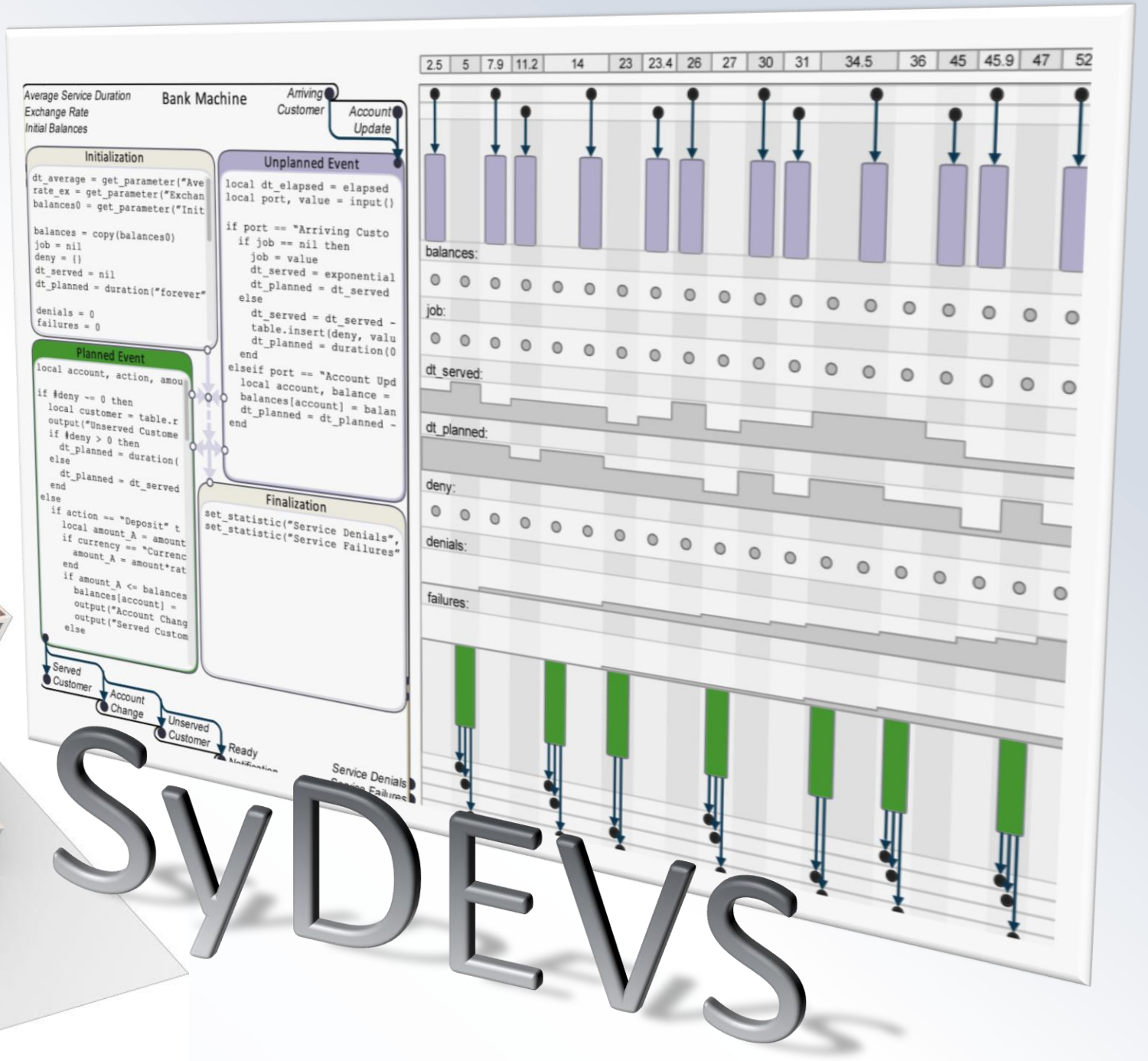


Rhys Goldstein
Autodesk Research

rhys.goldstein@autodesk.com


Bank
Machine Average Service Duration
Initialization
Exchange Rate
Number of Accounts
Accounts
Average Balance
Teller Lineup Capacity
Teller Average Service Duration
Machine Lineup
Class: Customer Lineup
Machine
Class: Bank Machine
Teller Lineup
Class: Customer Lineup
Teller
Class: Bank Teller
Finalization
Waiting Customers
Machine Served
Not Served
Teller Served




Average Service Duration
Exchange Rate
Initial Balances
Bank Machine
Arriving Customer
Account Update
Initialization
dt_average = get_parameter("Ave
rate_ex = get_parameter("Exchan
balances0 = get_parameter("Init
balances = copy(balances0)
job = nil
deny = {}
dt_served = nil
dt_planned = duration("forever"
denials = 0
failures = 0
Unplanned Event
local dt_elapsed = elapsed
local port, value = input()
if port == "Arriving Custo
if job == nil then
job = value
dt_served = exponential
dt_planned = dt_served
else
dt_served = dt_served -
table.insert(deny, valu
dt_planned = duration(0
end
elseif port == "Account Upd
local account, balance =
balances[account] = balan
dt_planned = dt_planned -
end
Planned Event
local account, action, amou
if #deny ~= 0 then
local customer = table.r
output("Unserved Custome
if #deny > 0 then
dt_planned = duration(
else
dt_planned = dt_served
end
else
if action == "Deposit" t
local amount_A = amount
if currency == "Currenc
amount_A = amount*rat
end
if amount_A <= balances
balances[account] =
output("Account Chang
output("Served Custom
else
Finalization
set_statistic("Service Denials",
set_statistic("Service Failures"
Served Customer
Account Change
Unserved Customer
Ready
Service Denials
2.5
5
7.9
11.2
14
23
23.4
26
27
30
31
34.5
36
45
45.9
47
52
balances:
job:
dt_served:
dt_planned:
deny:
denials:
failures:

SyDEVS

numpy-like operations:

```
a + b
a + 5
5 - a
a/2
(a < b)
(a == b)
(a >= 1 && a <= 4)
all(a >= 0)
any(a == 7)
```

a =

0	1	2
3	4	5

b =

2	1	0
2	4	6

numpy-like operations:

```
a + b
a + 5
5 - a
a/2
(a < b)
(a == b)
(a >= 1 && a <= 4)
all(a >= 0)
any(a == 7)
```

a =

0	1	2
3	4	5

b =

2	1	0
2	4	6

a + b =

2	2	2
5	8	11

numpy-like operations:

```
a + b
a + 5
5 - a
a/2
(a < b)
(a == b)
(a >= 1 && a <= 4)
all(a >= 0)
any(a == 7)
```

a =

0	1	2
3	4	5

b =

2	1	0
2	4	6

a + 5 =

5	6	7
8	9	10

numpy-like operations:

```
a + b
a + 5
5 - a
a/2
(a < b)
(a == b)
(a >= 1 && a <= 4)
all(a >= 0)
any(a == 7)
```

a =

0	1	2
3	4	5

b =

2	1	0
2	4	6

5 - a =

5	4	3
2	1	0

numpy-like operations:

```
a + b
a + 5
5 - a
a/2
(a < b)
(a == b)
(a >= 1 && a <= 4)
all(a >= 0)
any(a == 7)
```

a =

0	1	2
3	4	5

b =

2	1	0
2	4	6

a/2 =

0	0	1
1	2	2

numpy-like operations:

```
a + b
a + 5
5 - a
a/2
(a < b)
(a == b)
(a >= 1 && a <= 4)
all(a >= 0)
any(a == 7)
```

a =

0	1	2
3	4	5

b =

2	1	0
2	4	6

(a < b) =

1	0	0
0	0	1

numpy-like operations:

```
a + b
a + 5
5 - a
a/2
(a < b)
(a == b)
(a >= 1 && a <= 4)
all(a >= 0)
any(a == 7)
```

a =

0	1	2
3	4	5

b =

2	1	0
2	4	6

(a == b) =

0	1	0
0	1	0

numpy-like operations:

```
a + b
a + 5
5 - a
a/2
(a < b)
(a == b)
(a >= 1 && a <= 4)
all(a >= 0)
any(a == 7)
```

a =

0	1	2
3	4	5

b =

2	1	0
2	4	6

(a >= 1 && a <= 4) =

0	1	1
1	1	0

numpy-like operations:

```
a + b
a + 5
5 - a
a/2
(a < b)
(a == b)
(a >= 1 && a <= 4)
all(a >= 0)
any(a == 7)
```

$$a = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}$$

$$b = \begin{bmatrix} 2 & 1 & 0 \\ 2 & 4 & 6 \end{bmatrix}$$

$$\texttt{all(a >= 0)} = 1$$

numpy-like operations:

```
a + b
a + 5
5 - a
a/2
(a < b)
(a == b)
(a >= 1 && a <= 4)
all(a >= 0)
any(a == 7)
```

$$a = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}$$

$$b = \begin{bmatrix} 2 & 1 & 0 \\ 2 & 4 & 6 \end{bmatrix}$$

$$\text{any}(a == 7) = 0$$

numpy-like operations:

```
a + b
a + 5
5 - a
a/2
(a < b)
(a == b)
(a >= 1 && a <= 4)
all(a >= 0)
any(a == 7)
```

```
auto a = array2d<int64>({2, 3}, {0, 1, 2, 3, 4, 5});
auto b = array2d<int64>({2, 3}, {2, 1, 0, 2, 4, 6});

std::cout << a + b;                 // Prints "{{2, 2, 2}, {5, 8, 11}}".
std::cout << a + 5;                 // Prints "{{5, 6, 7}, {8, 9, 10}}".
std::cout << 5 - a;                 // Prints "{{5, 4, 3}, {2, 1, 0}}".
std::cout << a/2;                   // Prints "{{0, 0, 1}, {1, 2, 2}}".
std::cout << (a < b);               // Prints "{{1, 0, 0}, {0, 0, 1}}".
std::cout << (a == b);              // Prints "{{0, 1, 0}, {0, 1, 0}}".
std::cout << (a >= 1 && a <= 4);    // Prints "{{0, 1, 1}, {1, 1, 0}}".
std::cout << all(a >= 0);           // Prints "1".
std::cout << any(a == 7);           // Prints "0".
```

sydevs::arraynd

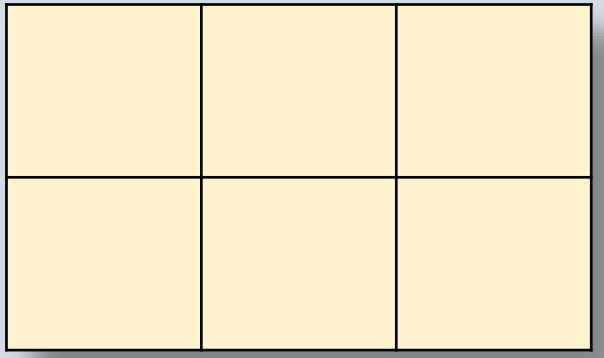
```
auto a = array2d<int64>({2, 3}, {0, 1, 2, 3, 4, 5});
```

```
auto a = array2d<int64>({2, 3}, {0, 1, 2, 3, 4, 5});
```

```
namespace sydevs {

template<typename T, int64 ndims>
class arraynd : ...
{
    ...
};

template<typename T> using array1d = arraynd<T, 1>;
template<typename T> using array2d = arraynd<T, 2>;
template<typename T> using array3d = arraynd<T, 3>;
...
template<typename T> using array9d = arraynd<T, 9>;

}  // namespace
```

```
auto a = array2d<int64>({2, 3}, {0, 1, 2, 3, 4, 5});
```

a =



```
auto a = array2d<int64>({2, 3}, {0, 1, 2, 3, 4, 5});
```

a =

0	1	2
3	4	5

```
auto arr = array3d<int64>({2, 3, 4}, [](const std::array<int64, 3>& indices) {
    return 100*indices[0] + 10*indices[1] + indices[2];
});

std::cout << arr;  // {{{   0,   1,   2,   3 },
                   //   {  10,  11,  12,  13 },
                   //   {  20,  21,  22,  23 }},
                   //  {{ 100, 101, 102, 103 },
                   //   { 110, 111, 112, 113 },
                   //   { 120, 121, 122, 123 }}}
```

```
auto arr = array3d<int64>({2, 3, 4}, [](const std::array<int64, 3>& indices) {
    return 100*indices[0] + 10*indices[1] + indices[2];
});

std::cout << arr;  // {{{   0,   1,   2,   3 },
                   //   {  10,  11,  12,  13 },
                   //   {  20,  21,  22,  23 }},
                   //  {{ 100, 101, 102, 103 },
                   //   { 110, 111, 112, 113 },
                   //   { 120, 121, 122, 123 }}}

std::cout << arr(1, 0, 3);

arr(1, 0, 3) = 7;
```

```
auto arr = array3d<int64>({2, 3, 4}, [](const std::array<int64, 3>& indices) {
    return 100*indices[0] + 10*indices[1] + indices[2];
});

std::cout << arr;  // {{{   0,   1,   2,   3 },
                   //   {  10,  11,  12,  13 },
                   //   {  20,  21,  22,  23 }},
                   //  {{ 100, 101, 102, 103 },
                   //   { 110, 111, 112, 113 },
                   //   { 120, 121, 122, 123 }}}

std::cout << arr[1][0][3];  // Inefficient!

arr[1][0][3] = 7;  // Inefficient!
```

```
auto arr = array3d<int64>({2, 3, 4}, [](const std::array<int64, 3>& indices) {
    return 100*indices[0] + 10*indices[1] + indices[2];
});

std::cout << arr;  // {{{   0,   1,   2,   3 },
                   //   {  10,  11,  12,  13 },
                   //   {  20,  21,  22,  23 }},
                   //  {{ 100, 101, 102, 103 },
                   //   { 110, 111, 112, 113 },
                   //   { 120, 121, 122, 123 }}}

std::cout << arr[range()][1][range().start_at(1).stride_by(2)];

                   //  {{  11,  13 },
                   //   { 111, 113 }}".
```

```
auto arr = array3d<int64>({2, 3, 4}, [](const std::array<int64, 3>& indices) {
    return 100*indices[0] + 10*indices[1] + indices[2];
});

arr[range()][1][range().start_at(1).stride_by(2)].fill(7);

std::cout << arr;  // {{{   0,   1,   2,   3 },
                   //   {  10,   7,  12,   7 },
                   //   {  20,  21,  22,  23 }},
                   //  {{ 100, 101, 102, 103 },
                   //   { 110,   7, 112,   7 },
                   //   { 120, 121, 122, 123 }}}
```

```
auto arr = array3d<int64>({2, 3, 4}, [](const std::array<int64, 3>& indices) {
    return 100*indices[0] + 10*indices[1] + indices[2];
});

auto arr2 = arr[range()][1][range().start_at(1).stride_by(2)];

arr2.fill(7)

std::cout << arr;  // {{{   0,   1,   2,   3 },
                   //   {  10,   7,  12,   7 },
                   //   {  20,  21,  22,  23 }},
                   //  {{ 100, 101, 102, 103 },
                   //   { 110,   7, 112,   7 },
                   //   { 120, 121, 122, 123 }}}
```

```
auto arr = array3d<int64>({2, 3, 4}, [](const std::array<int64, 3>& indices) {
    return 100*indices[0] + 10*indices[1] + indices[2];
});

auto arr2 = arr[range()][1][range().start_at(1).stride_by(2)].copy();

arr2.fill(7)

std::cout << arr;  // {{{   0,   1,   2,   3 },
                   //   {  10,  11,  12,  13 },
                   //   {  20,  21,  22,  23 }},
                   //  {{ 100, 101, 102, 103 },
                   //   { 110, 111, 112, 113 },
                   //   { 120, 121, 122, 123 }}}
```

```
try {
    const auto arr = array2d<std::string>({2, 2}, {"A", "B", "C", "D"});
    auto arr2 = arr[range()];
    arr2(0, 0) = "E";
}
catch (const std::logic_error& e) {
    std::cout << "ERROR: " << e.what() << std::endl;
}


// ERROR: Attempt to obtain a non-const reference to readonly multidimensional
//        array data
```

The `sydevs::arraynd` class template is open source.
Visit https://autodesk.github.io/sydevs/ or Google "SyDEVS".

Also check out `xtensor` by QuantStack.
Visit https://xtensor.readthedocs.io/ or Google "xtensor".

Rhys Goldstein
Autodesk Research

rhys.goldstein@autodesk.com