

THE DEVELOPERS



Lucian Teodorescu
Software Engineers
vs Developers

14 June 2022, Cluj Innovation Park, www.the-developers.ro

Why I Hate Developers?

alternative title

LUCIAN RADU TEODORESCU
GARMIN



LucTeo

Software
Engineering

Love Story





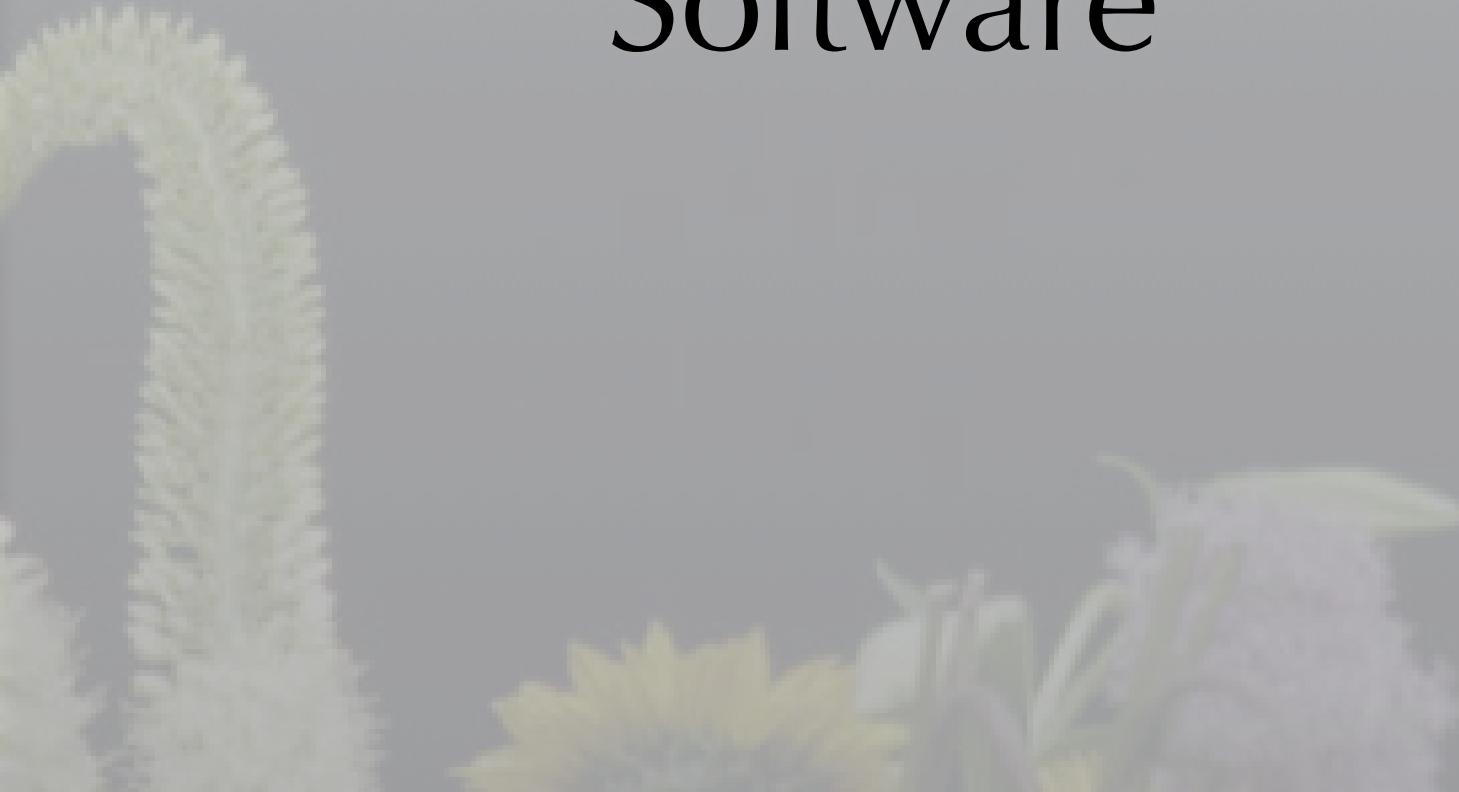
lucteo.ro/pres/2022-thedev

Software Engineering



Mary Shaw

Progress Towards an Engineering Discipline of Software



engineering

creating cost-effective solutions

... to practical problems

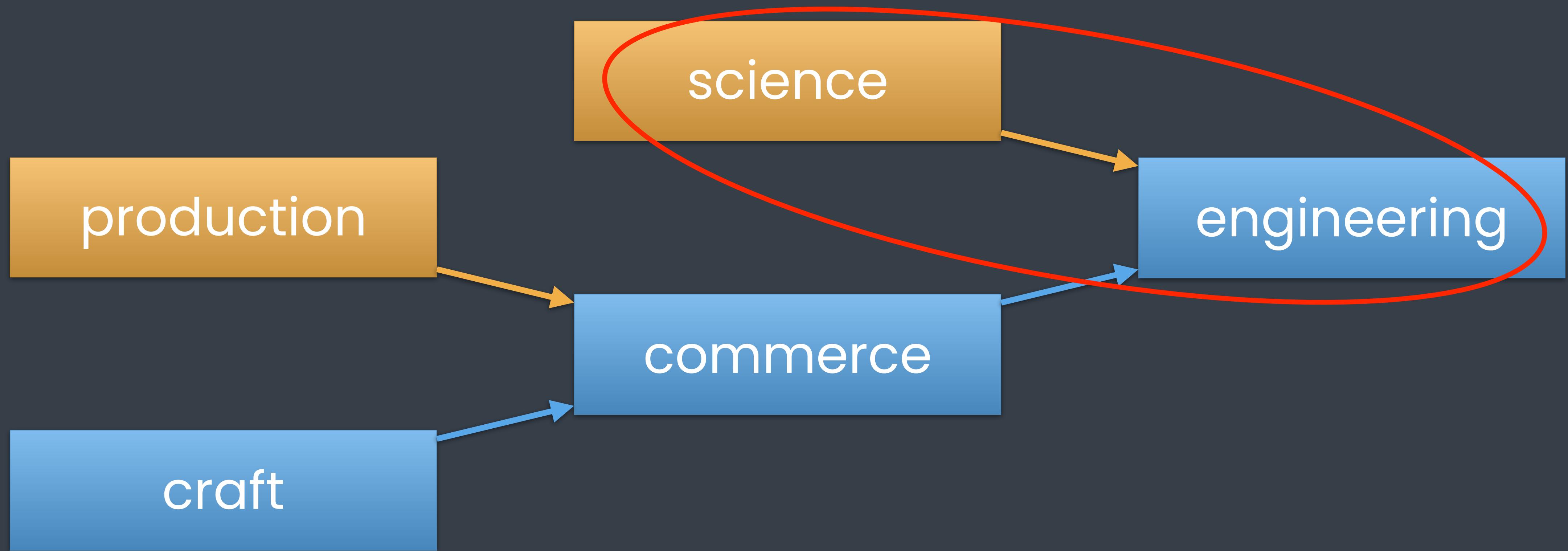
... by applying **scientific knowledge**

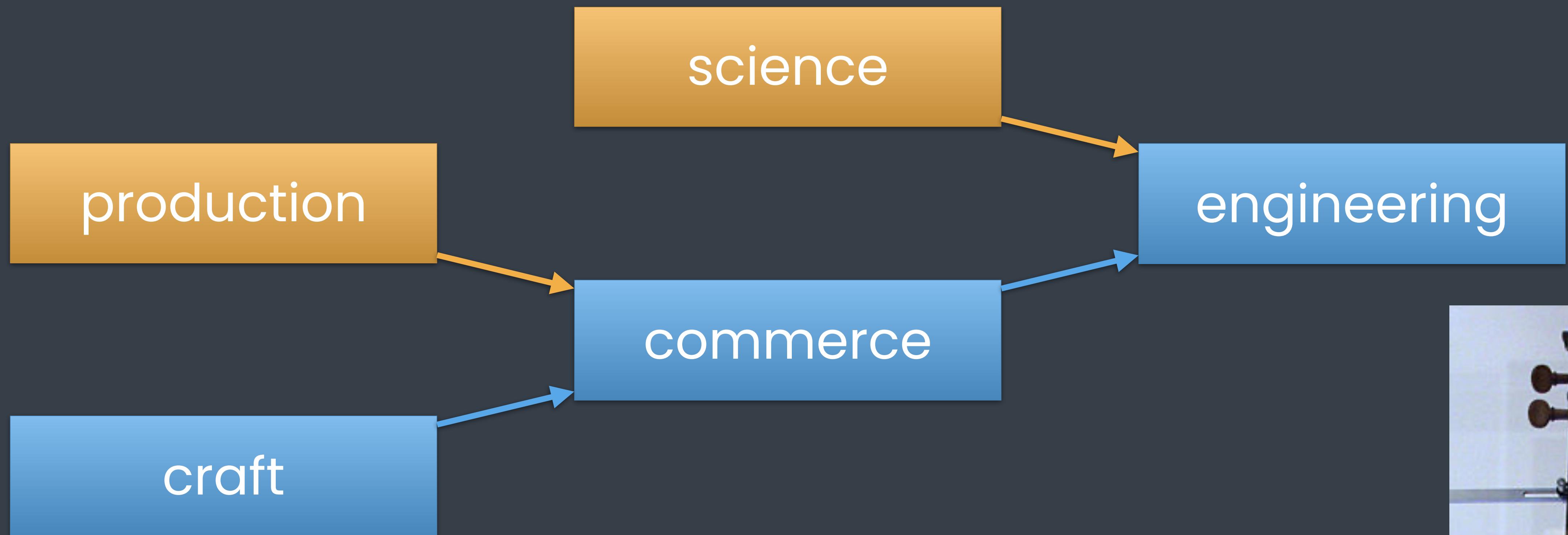
... building things

... in the service of mankind

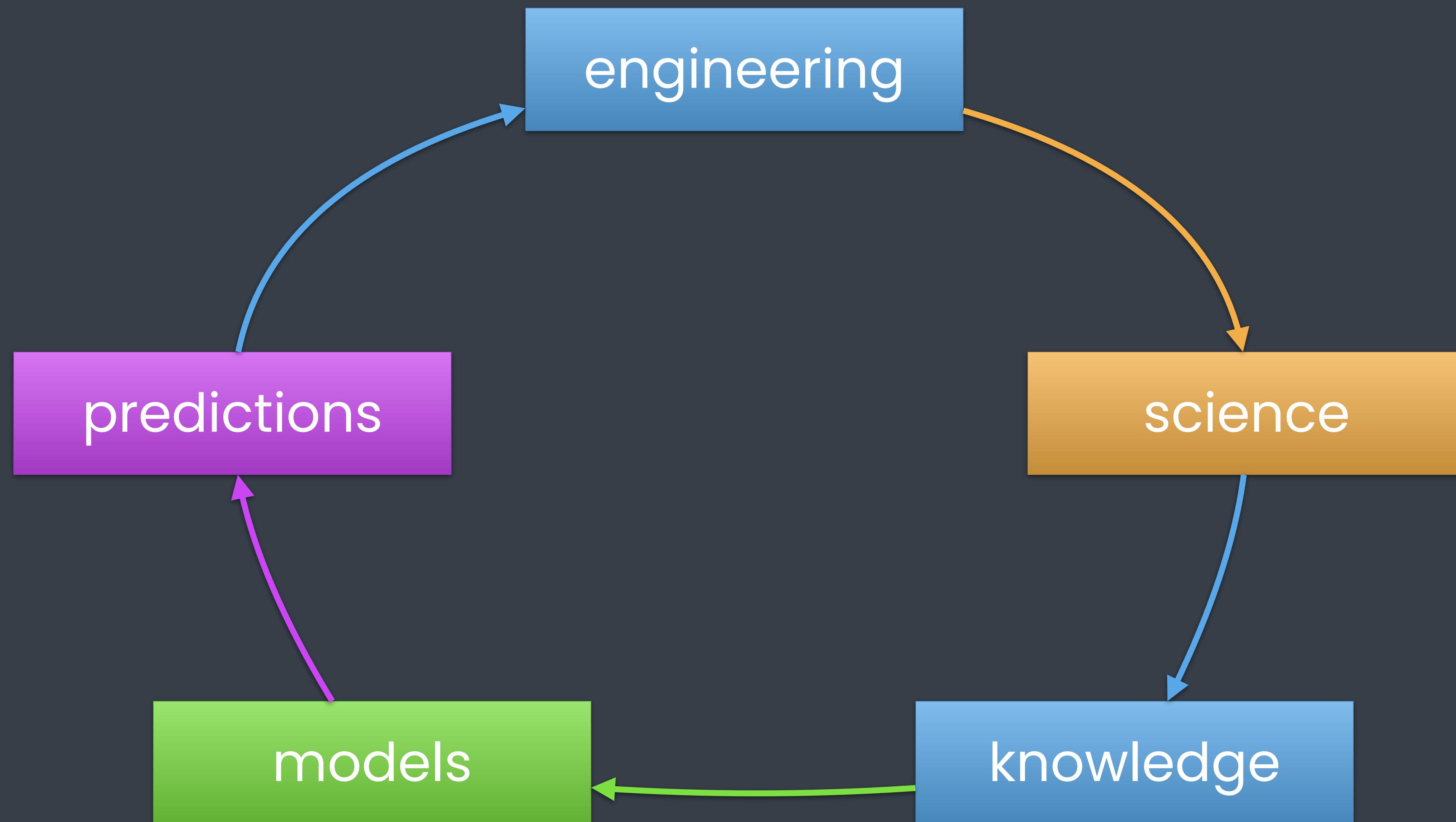
engineering

enabled ordinary people to do things
that formerly required virtuosos









A deck of tarot cards is fanned out across the entire background of the image. The cards are from the "NYVA Tarot" deck, featuring colorful, detailed illustrations of various tarot figures and symbols. Some visible cards include "The FOOL" (0), "The HANGED MAN" (XII), "The SUN" (XIX), and "The HERMIT".

predicting

quality
completion time
scope



knowledge

mostly empirical

scientific principles

assume that you are wrong
iteratively improve – limit the impact of mistakes
always measure
stop when “good enough”



moon landing

orbiting Earth

lunar orbit

lunar impact

lunar landing

human lunar landing

I believe that this nation should commit itself to achieving the goal, before this decade is out, of landing a man on the moon and returning him safely to the earth

John F. Kennedy



however



Mark Seemann, Where's the science?

<https://blog.ploeh.dk/2020/05/25/wheres-the-science/>

Hillel Wayne, Intro to Empirical Software Engineering

<https://www.youtube.com/watch?v=WELBnE33dpY>



Sw Eng vs Developers

2



name

discipline: Software Engineering

Software Engineers → like other engineers

Developers → like Real estate developers?

approach

Software Engineers

use scientific methods
structured
predictable results

Developers

ad-hoc methods
unstructured
unpredictable results

use of knowledge

Software Engineers

contextualized prior knowledge

Developers

magic art?

iterations

Software Engineers

improve knowledge
steps towards vision

Developers

finish “disconnected” features
altering vision

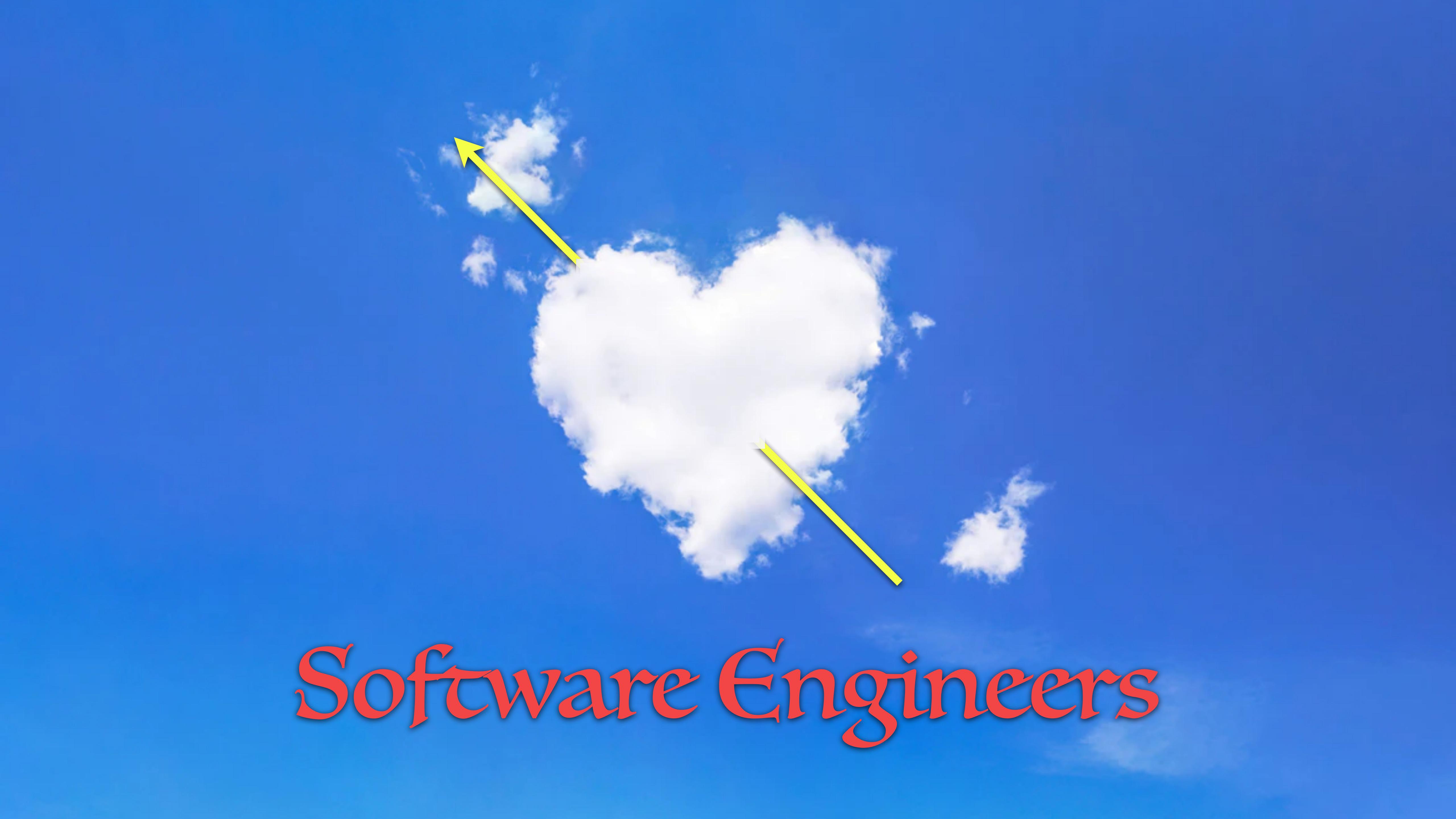
building good software

Software Engineers

ordinary people

Developers

virtuosos



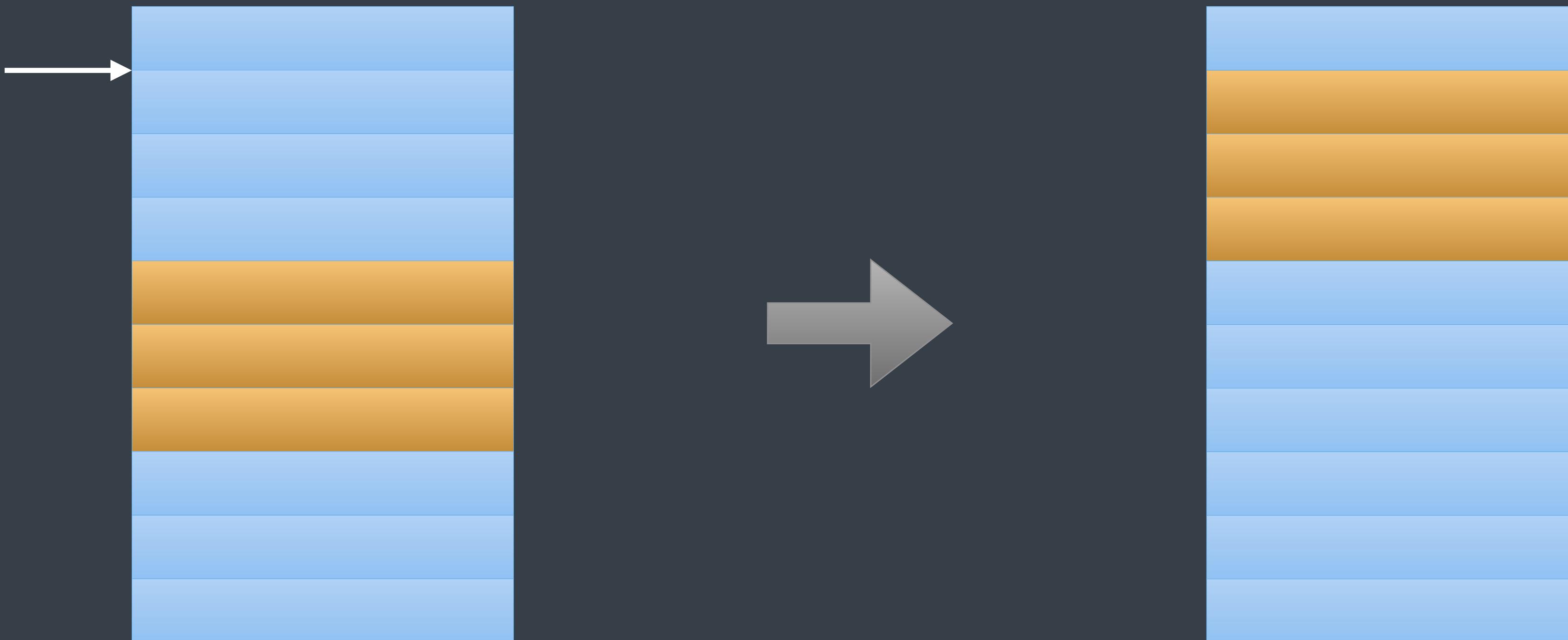
Software Engineers

Engineering the Code

3



1. UI rearrange



dev

```
// Next, check if the panel has moved to the other side of another panel.  
for (size_t i = 0; i < expanded_panels_.size(); ++i) {  
    Panel *panel = expanded_panels_[i].get();  
    if (center_x <= panel->cur_panel_center() ||  
        i == expanded_panels_.size() - 1) {  
        if (panel != fixed_panel) {  
            // If it has, then we reorder the panels.  
            ref_ptr<Panel> ref = expanded_panels_[fixed_index];  
            expanded_panels_.erase(expanded_panels_.begin() + fixed_index);  
            if (i < expanded_panels_.size()) {  
                expanded_panels_.insert(expanded_panels_.begin() + i, ref);  
            } else {  
                expanded_panels_.push_back(ref);  
            }  
        }  
        break;  
    }  
}  
...  
}
```

sw eng

```
// Next, check if the panel has moved to the left side of another panel.  
auto f = begin(expanded_panels_) + fixed_index;  
auto p = lower_bound(begin(expanded_panels_), f, center_x,  
    [](const ref_ptr<Panel> &e, int x) { return e->cur_panel_center() < x; });  
// If it has, then we reorder the panels.  
rotate(p, f, f + 1);
```

sw eng

```
template <typename It> // I models RandomAccessIterator
auto slide(It first, It last, It pos) -> pair<It, It>
{
    if (pos < first) return { pos, rotate(pos, first, last) };
    if (last < pos) return { rotate(first, last, pos), pos };
    return { first, last };
}
```

more info

Sean Parent – C++ Seasoning, Going Native 2013

<https://www.youtube.com/watch?v=W2tWOdzgXHA>



2. computing mean, median

mean = average of the data values

median = middle number in the ordered set of data

naïve implementation

```
float mean(float arr[], int n) {
    float sum = 0;
    for (int i = 0; i < n; i++)
        sum += arr[i];

    return sum / n;
}

float median(float arr[], int n) {
    // sort the array
    std::sort(arr, arr + n);
    if (n % 2 == 0)
        return (arr[n / 2 - 1] + arr[n / 2]) / 2;
    return arr[n / 2];
}
```

using STL algorithms

```
float mean(float arr[], int n) {
    return std::reduce(arr, arr + n) / n;
}

float mean_par(float arr[], int n) {
    return std::reduce(std::execution::par, arr, arr + n) / n;
}
```

using STL algorithms

```
float median(float arr[], int n) {
    // partially sort the array
    auto mid = n / 2;
    std::nth_element(arr, arr + mid, arr + n);
    if (n % 2 == 1)
        return arr[mid];
    else {
        auto prev = *std::max_element(arr, arr + mid);
        return std::midpoint(prev, arr[mid])
    }
}
```

percentile calculation

```
float percentile(float arr[], int n, int rank = 90) {
    auto t = static_cast<float>(rank) / 100.0f * (n - 1);
    auto idx_down = static_cast<int>(t);
    std::nth_element(arr, arr + idx_down, arr + n);
    auto lower = arr[idx_down];
    if (idx_down < n - 1) {
        auto upper = *std::min_element(arr + idx_down + 1, arr + n);
        return std::lerp(lower, upper, t - float(idx_down));
    } else
        return lower;
}
```

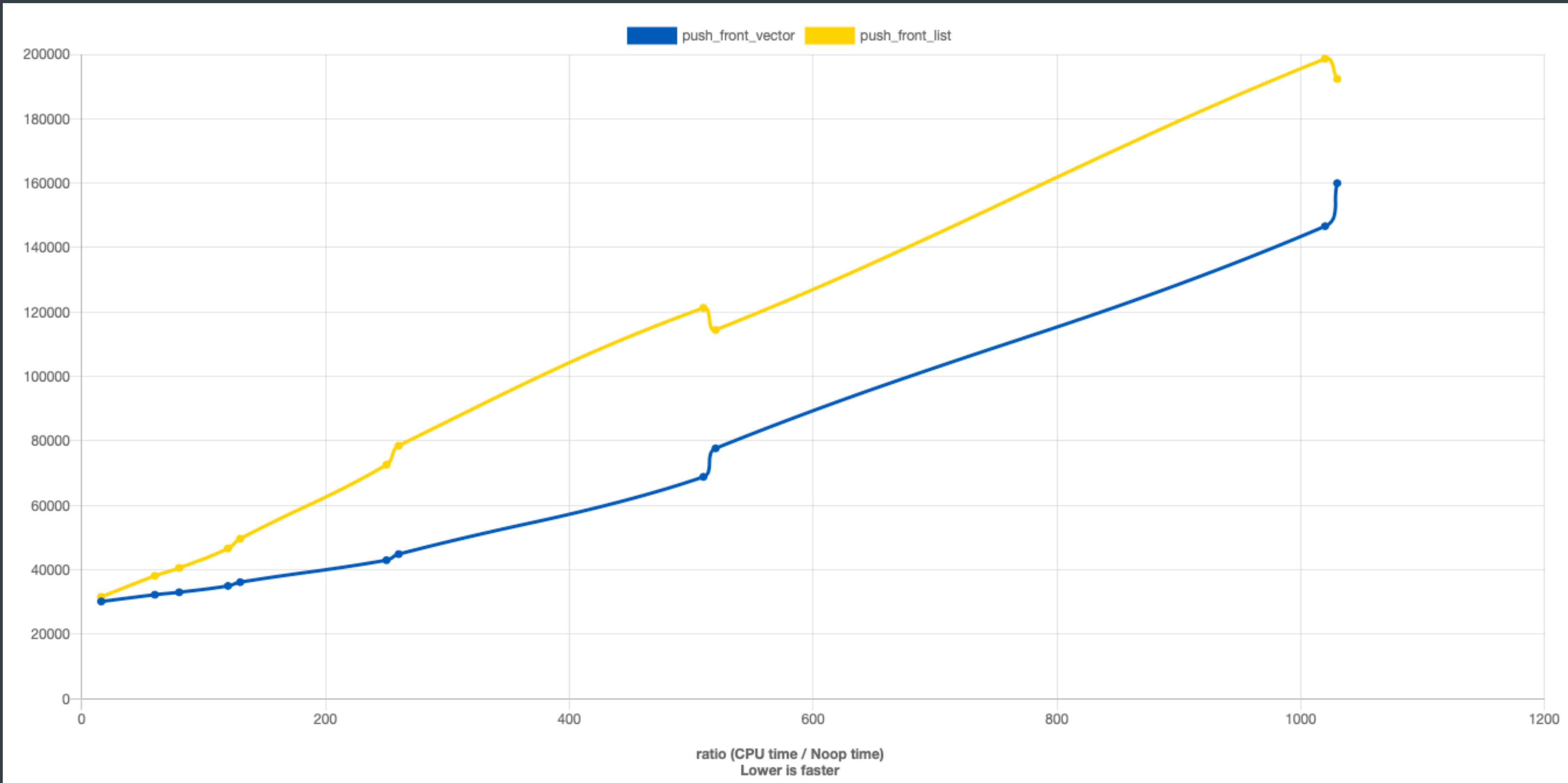
3. choosing containers

insert N elements in the front of a container

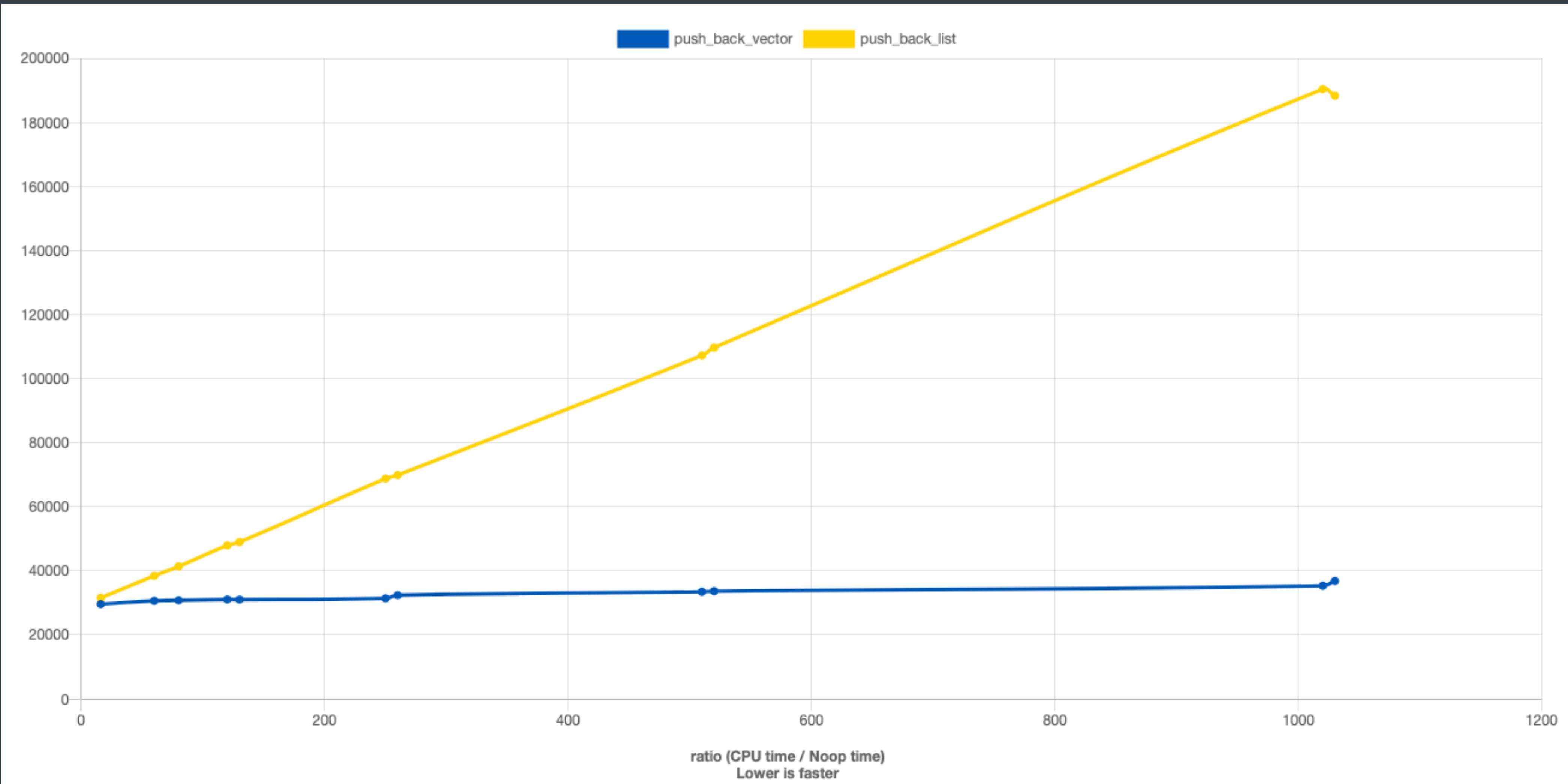
insert front, vector & list

```
// Alternative 1: std::vector, O(N)
std::vector<int> c;
for ( int i=0; i<N; i++ )
    c.insert(c.begin(), i);
```

```
// Alternative 2: std::list, O(1)
std::list<int> c;
for ( int i=0; i<N; i++ )
    c.push_front(i);
```



push_back results



bottom line

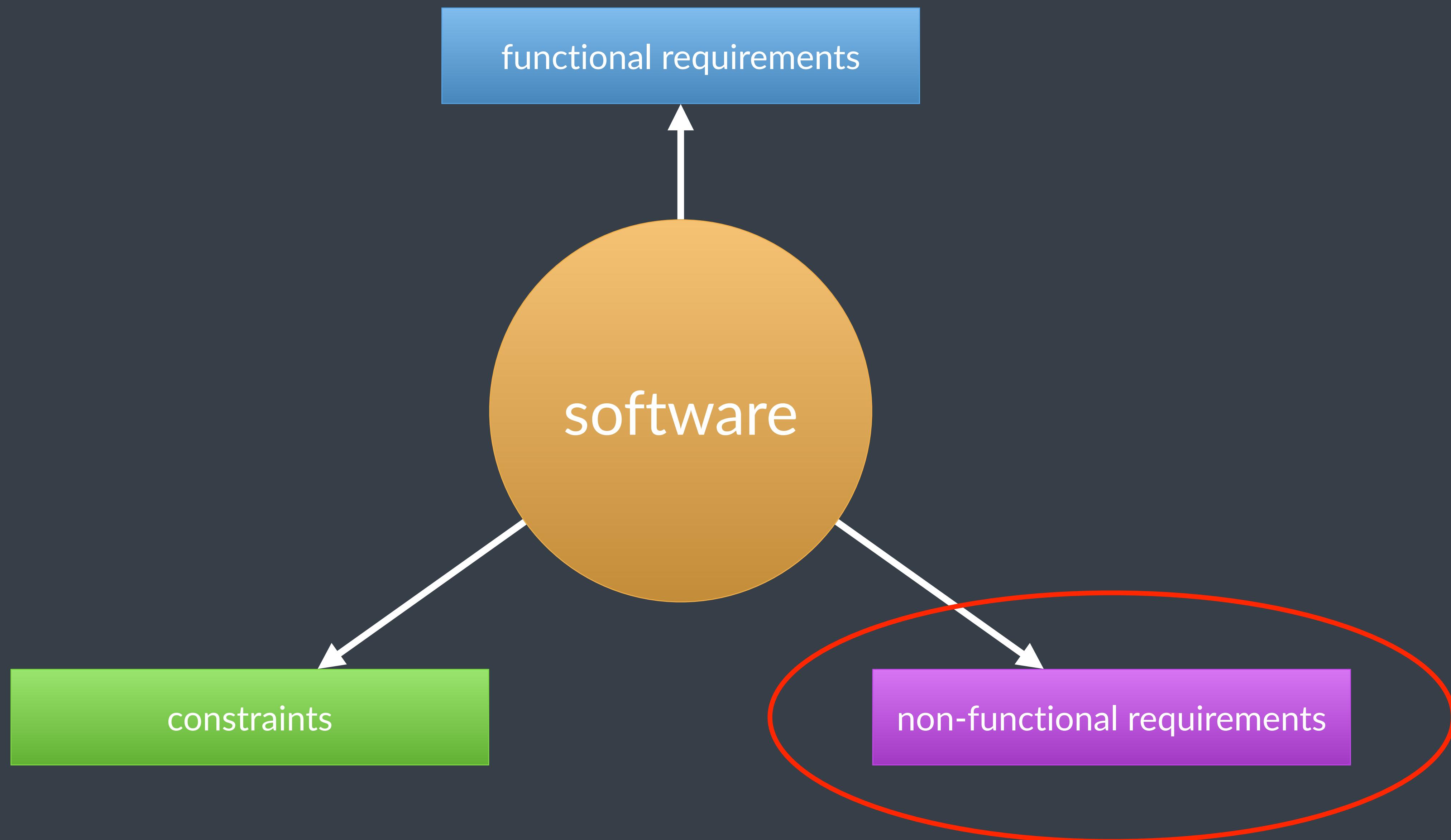
know your algorithms

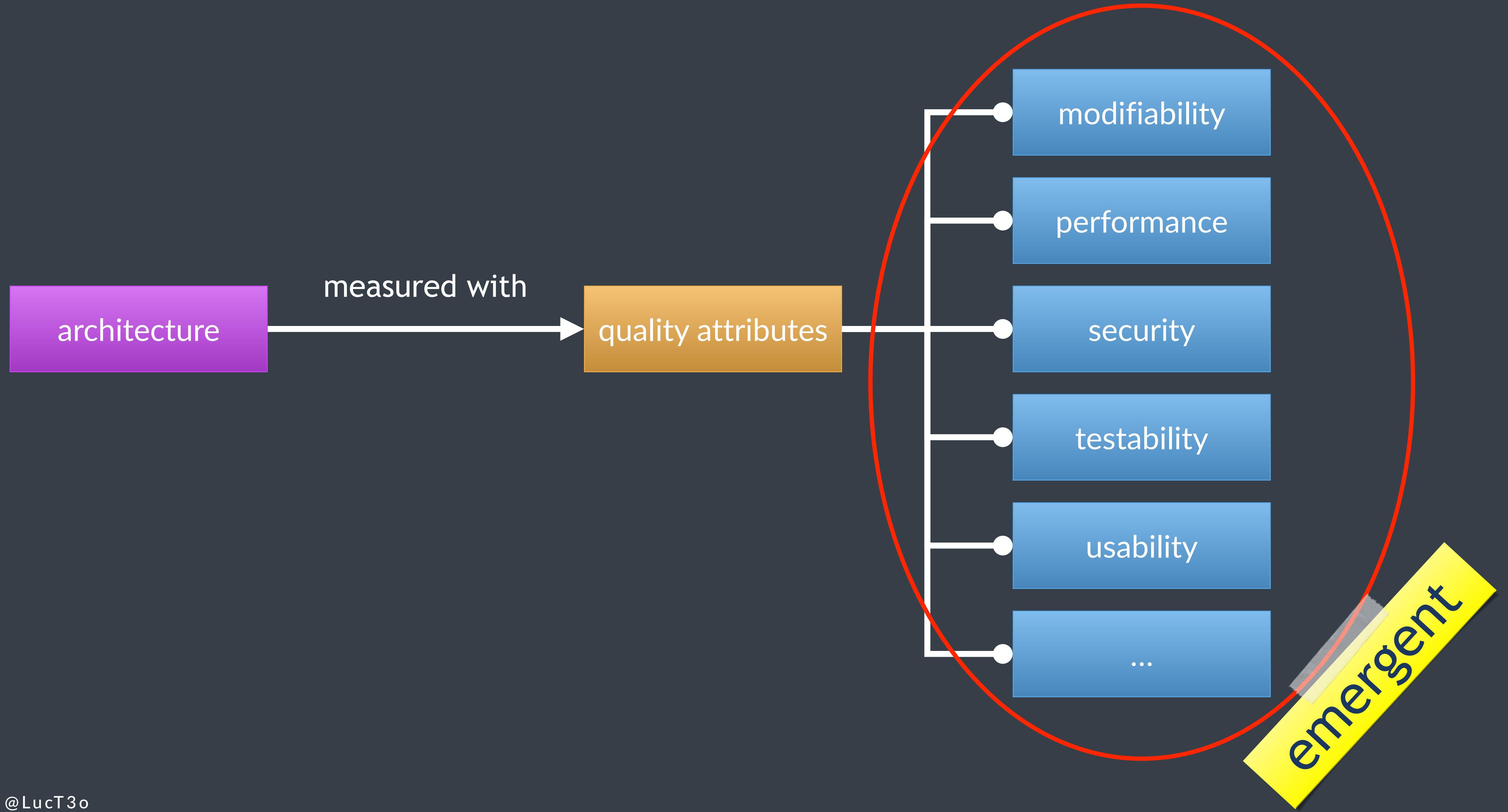
know your data structures

perform experiments

Engineering the Architecture







dynamics

features

incrementally added

quality attributes

always changing

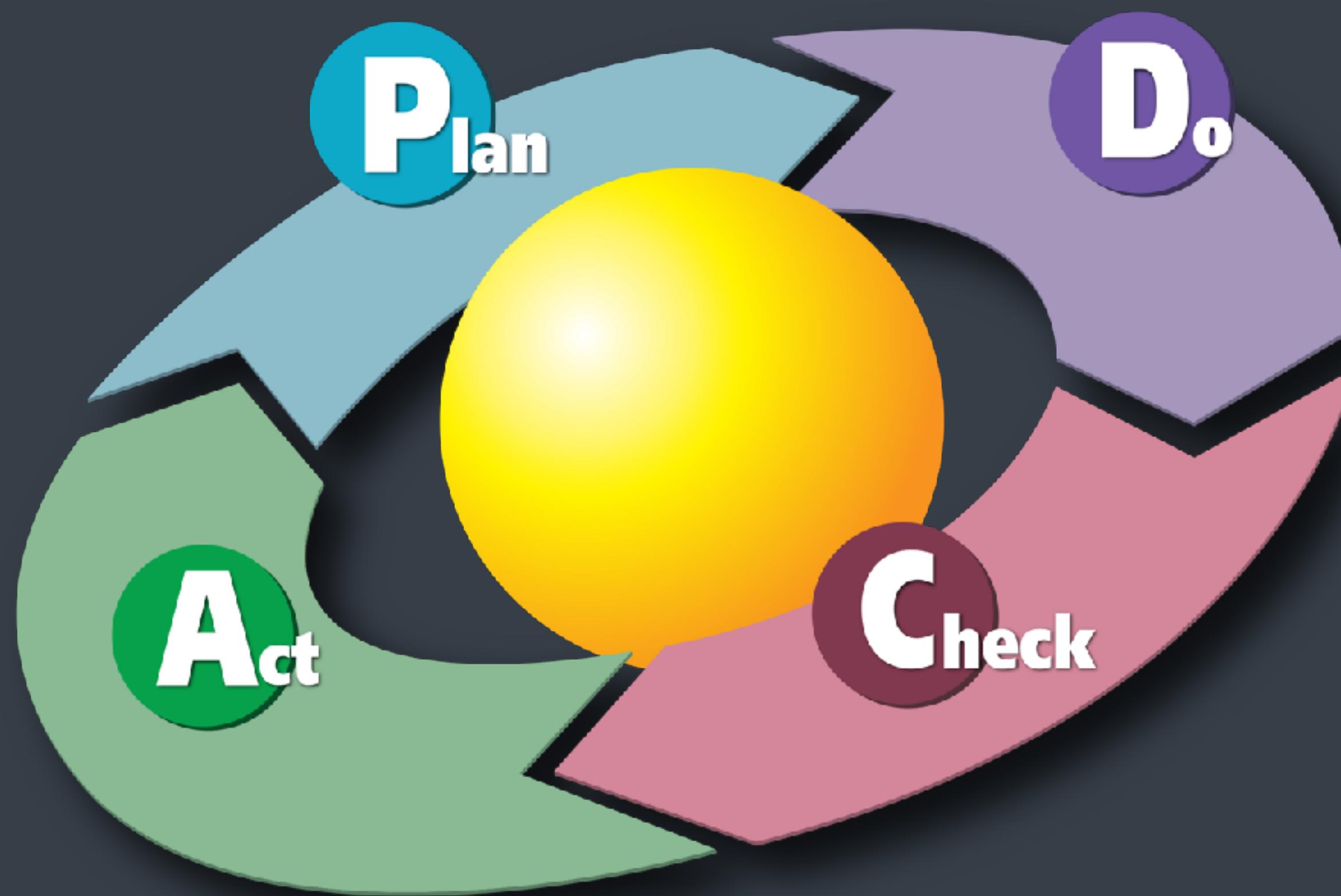
A wide-angle photograph of a waterfall cascading down a rocky cliff into a pool of water. The rocks are heavily covered in vibrant green moss. The surrounding environment is a dense forest with various shades of green foliage. The water is clear and flows rapidly. A large, semi-transparent rectangular overlay covers the middle portion of the image, containing the text.

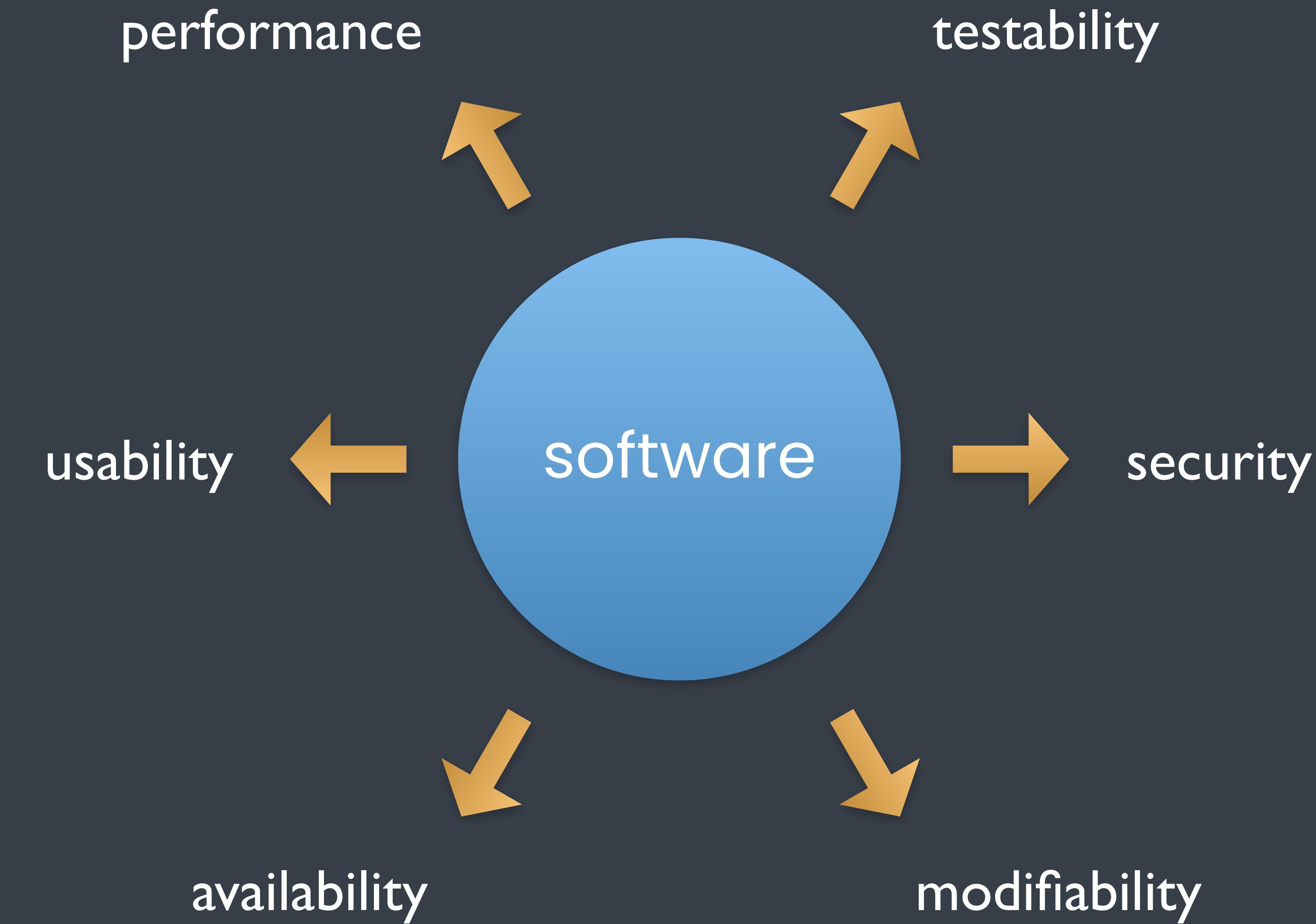
Architecture is always in flux

engineering architecture

set goals for quality attributes
constantly measure important QAs
dedicate work for improving QAs when needed

engineering architecture





Engineering the Processes

5



main problems

software design is unpredictable
QAs can be fragile

waterfall

big design upfront

no iterations

results often misaligned with the goals

common agile

small sprints

deliver functionality in each iteration

upfront design is nonexistent

both are wrong



controlled iterative

learn through iterations
measure, measure, measure
constantly improve design
reduce risks

engineering

initial guesses are wrong
improved with each iteration
empirical approach

another take: **risks**

initial state

high risk

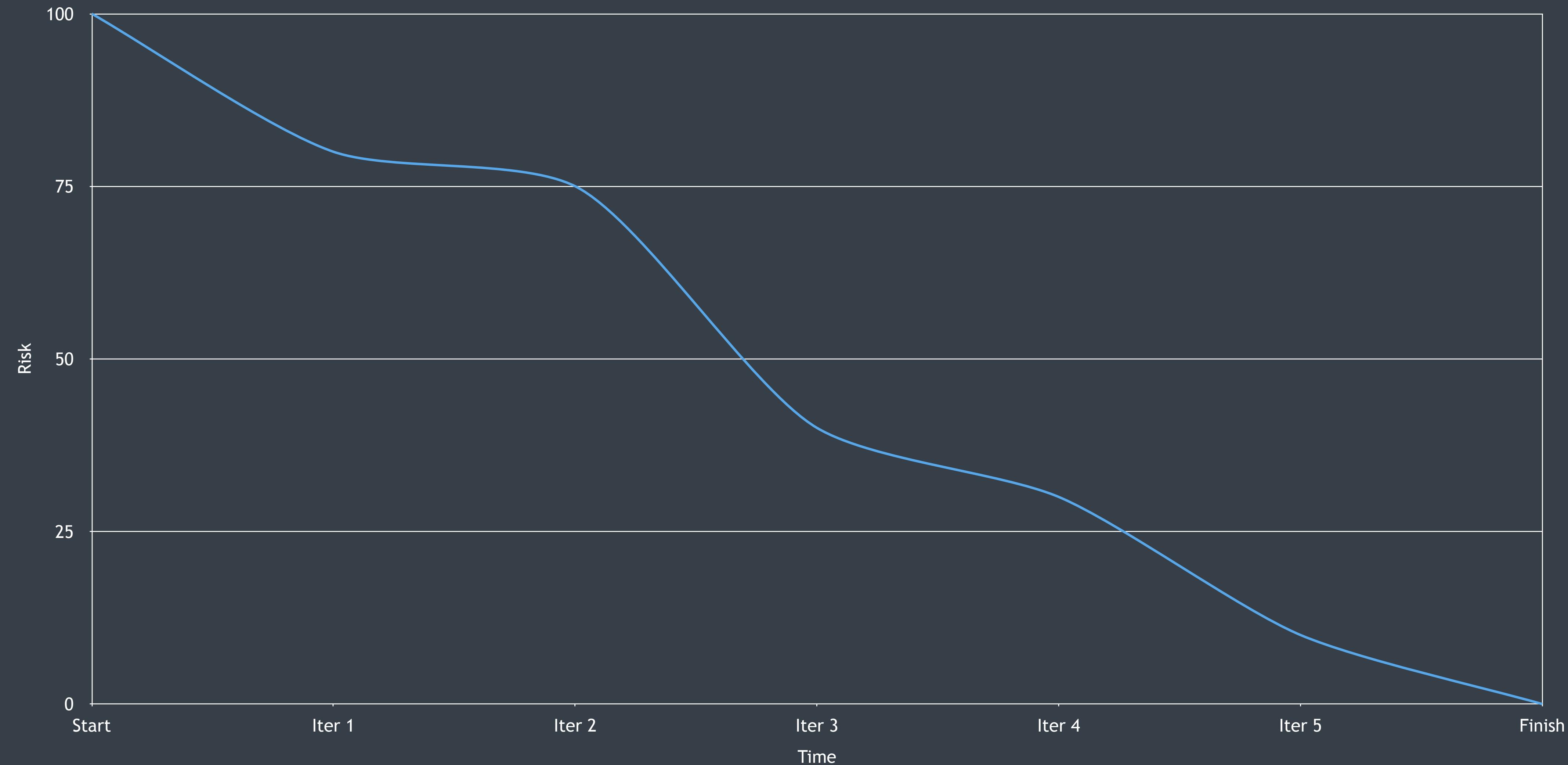
project completed

zero risk

process goal

iteratively decrease risk

ideal project



tips

constantly identify risks

mitigate risks asap

spend design time around risks

prototype around risks



no risk == success

Conclusions

6



engineering

know what engineering is
knowledge → software

use engineering

in code
for architecture
with the processes

A photograph of a coastal path. A paved walkway leads up a grassy hill, flanked by wooden railings. In the middle of the path is a set of wide concrete steps. The sky is blue with some clouds.

software engineer

developer



Love It

Thank You



@LucT3o



lucteo.ro

