ENERGY SYSTEM MODEL INCUBATOR

Hans Ivar Skjelbred 25.09.2020

Incubator - definition

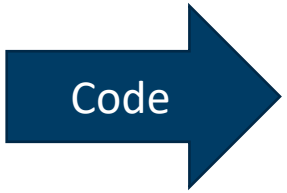
An insulated enclosure in which all environmental conditions can be regulated for optimal growth

(Britannica encyclopedia)

Model incubator - motivation

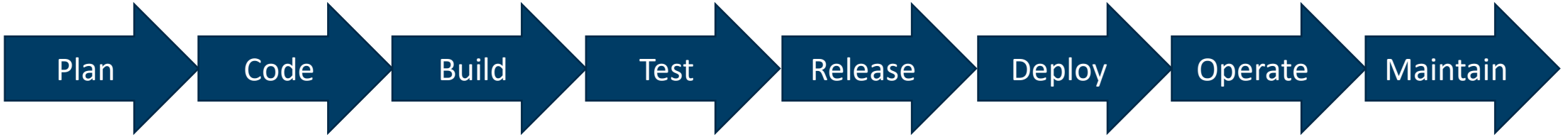
- Internal initiative in SINTEF
  - Responsibility to manage the knowledge that clients have invested in over many years
- Goals
  - Contribute to long-term strategy for model development
  - Increase our ability to use IT professionals in software devlopment
  - Establish quality-assured software development processes
- Budget 1MNOK
- Start in May 2020, end in September 2020

Software management



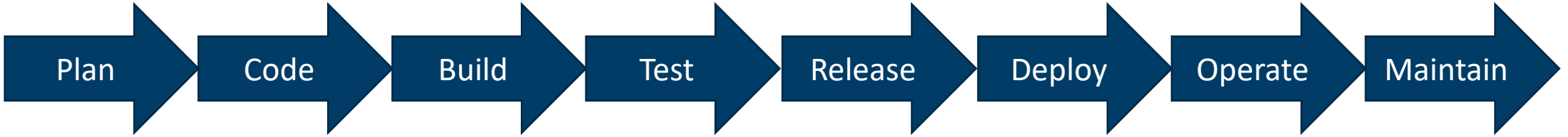
SINTEF

Software management



SINTEF

Software management



Programming language

Interoperability

Code standards

Data structuring

Version control

Test frameworks

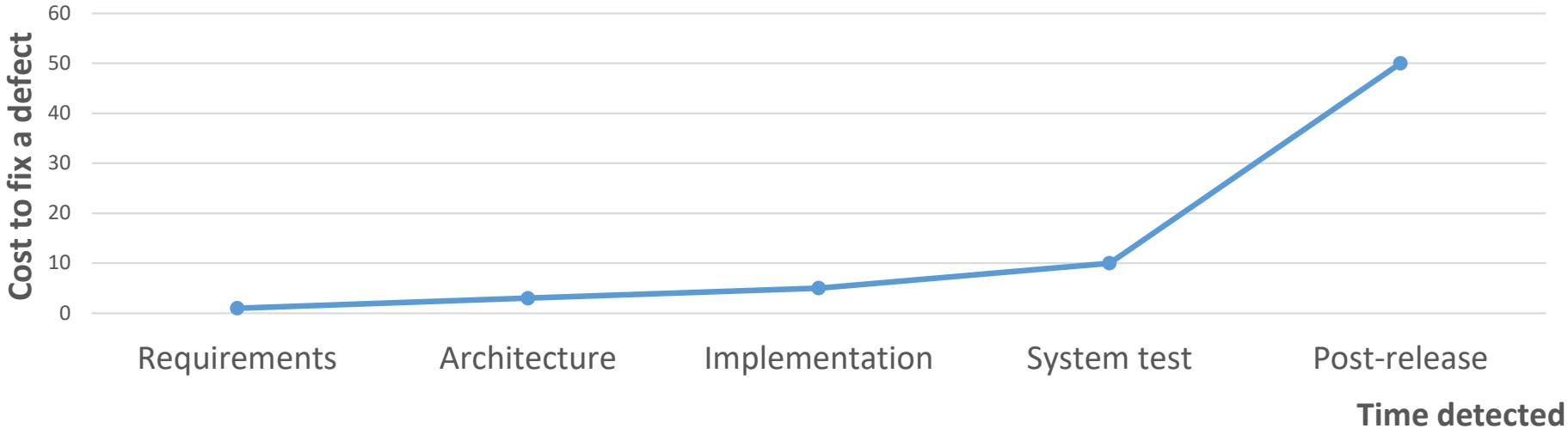
Release pipeline

Documentation

Customer portal

Ticket system

Software management



Plan → Code → Build → Test → Release → Deploy → Operate → Maintain

Programming languages

Interoperability

Code standards

Data structuring

Version control

Test frameworks

Release pipeline

Documentation

Customer portal

Ticket system

Up and coming programming languages

- Systems programming language
- Performance like C
- Designed for safety
  - Prevents memory errors
  - Prevents concurrency issues

- High-level language
- High performance
- Designed for scientific computing
  - Large mathematical function library
  - Easy to call C code

Evaluation of languages

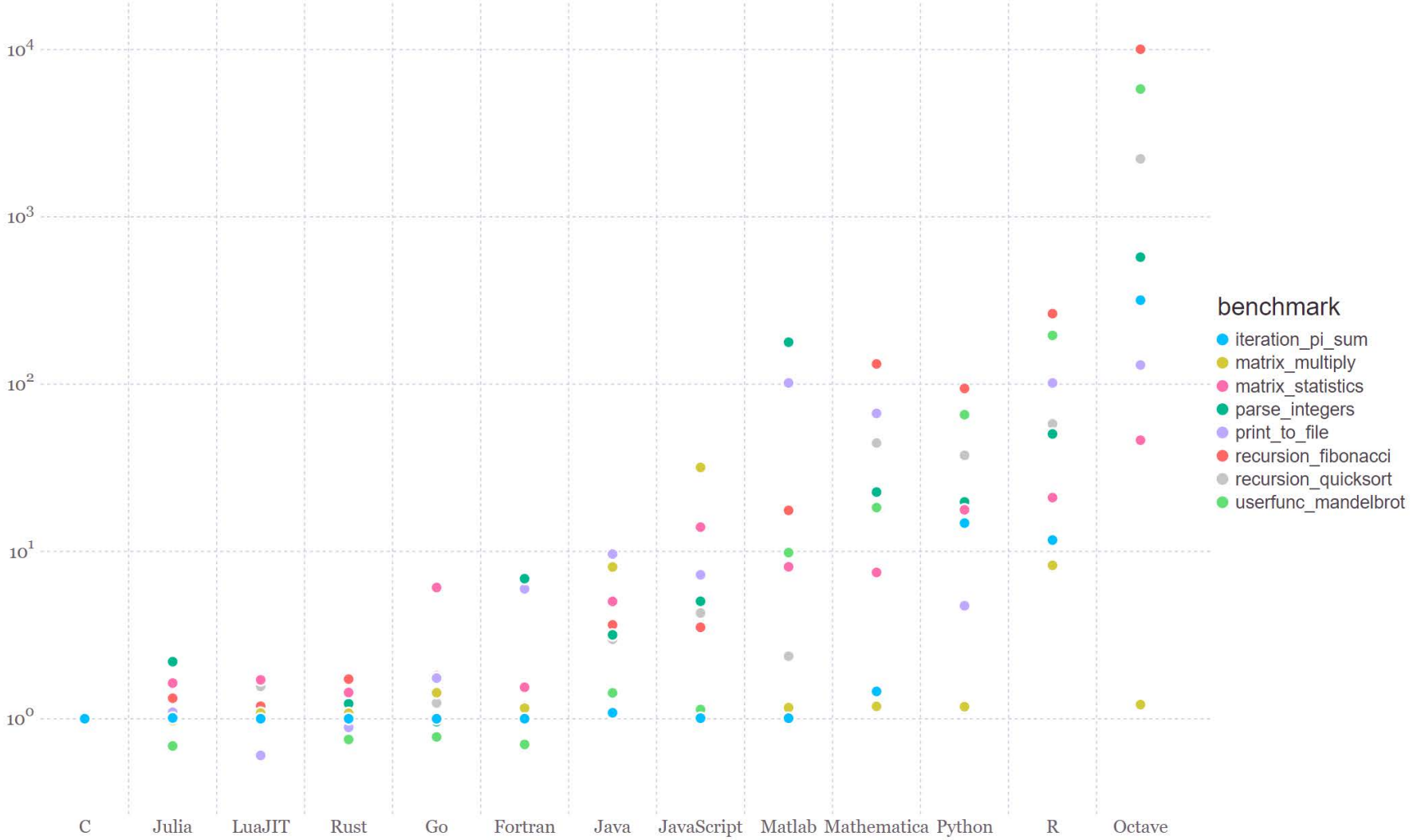
TIOBE Programming Community index
https://www.tiobe.com/tiobe-index/

Sep 2020	Sep 2019	Change	Programming Language	Ratings	Change
1	2	↑	C	15.95%	+0.74%
2	1	↓	Java	13.48%	-3.18%
3	3		Python	10.47%	+0.59%
4	4		C++	7.11%	+1.48%
5	5		C#	4.58%	+1.18%
6	6		Visual Basic	4.12%	+0.83%
7	7		JavaScript	2.54%	+0.41%
8	9	↑	PHP	2.49%	+0.62%
9	19	⇈	R	2.37%	+1.33%
10	8	↓	SQL	1.76%	-0.19%
11	14	↑	Go	1.46%	+0.24%
12	16	⇈	Swift	1.38%	+0.28%
13	20	⇈	Perl	1.30%	+0.26%
14	12	↓	Assembly language	1.30%	-0.08%
15	15		Ruby	1.24%	+0.03%
16	18	↑	MATLAB	1.10%	+0.04%
17	11	⇊	Groovy	0.99%	-0.52%
18	33	⇈	Rust	0.92%	+0.55%
19	10	⇊	Objective-C	0.85%	-0.99%

Evaulation of languages – architecture

	Python	Julia	Rust	C	C++	Fortran
Compiled?	No	JIT	Yes	Yes	Yes	Yes
Binary distribution	Many 3. Party libs	PackageCompiler Limited usage, Some problems	Yes, inherent	Yes, Inherent	Yes, Inherent	Yes, Inherent
Binary size	…	LARGE!	Small/medium	Small	Small	Small
Binary, protected source code	…	Not well protected	Protected	Protected	Protected	Protected
Run time reqs	Python	Julia	Optional C/C++ libs	Optional C/C++ libs	Optional C/C++ libs	Optional C/C++ libs
Community	Large, Growing	Small/Moderate, Growing	Small/Moderate, Growing	Moderate, Slowly decreasing	Moderate, Stable	Small, Slowly decreasing
Memory model	GC	GC	Unique ownership model	User managed	User managed	User managed

Evaluation of languages - performance



SINTEF

Evaluation of languages - interoperability

Callee\Caller	Python	Julia	Rust	C	C++	Fortran
Python		Yes (built in)	Yes	Yes	Yes	
Julia	Yes (package)		Systemcall	Yes? (hassle)	Yes? (hassle)	Systemcall
Rust		Yes (ccall)		Yes (C api)	Yes (C api)	No? (via C?)
C	Yes (slow!)	Yes (fast, ccall)	Yes (unsafe)			Yes
C++	Yes (slow!)	No? (C api)	No? (C api)			No? (C api)
Fortran	Limited (F2PY)	Yes (fast, ccall)	Yes (iso c)	Yes (iso c)	Yes (iso c)	

Code examples - Initialization

Rust

```rust
#[derive(Clone, Copy)]
struct XY {
    x: Option<u32>,
    y: Option<u32>
}

struct ProdriskCore {
    value_a: Option<XY>
}

impl ProdriskCore {
    pub fn set_xy(&mut self, x: u64, y: u64) {
        let xy = XY{x: Some(x), y: Some(y)};
        self.value_a = Some(xy);
    }

    // Extraction from nested option can be simplified by ?
    pub fn get_a_x(&mut self) -> Option<u64>{
        self.value_a?.x
    }
}

fn main() {
    let mut prodrisk = ProdriskCore {value_a: None};
    println!("Err {:?}", prodrisk.get_a_x());
    prodrisk.set_xy(1, 2);
    let x = prodrisk.get_a_x();
    println!("Err {:?}", x);

    match x {
        Some(i) => prodrisk.set_xy(i*2, 1),
        _ => (),
    }
}
```

Fortran

```fortran
program main

implicit none

    ! type declarations
    type xy
        integer :: x
        integer :: y
        logical :: x_flag
        logical :: y_flag
    end type xy

    type prodriskcore
        type(xy), pointer :: value_a
        logical :: defined
    end type prodriskcore

    ! local variable declarations
    type(prodriskcore), pointer :: prodrisk
    logical :: prodrisk_defined
    integer :: xval

    ! code
    allocate(prodrisk)
    prodrisk_defined = .true.
    prodrisk%defined = .false.

    if (prodrisk_defined) then
        xval = get_prodrisk_a_x(prodrisk)
    else
        xval = -1
    end if

    call print_positive (xval)

    if (prodrisk_defined) then
        call set_prodrisk_xy(prodrisk, 1, 2)
        xval = get_prodrisk_a_x(prodrisk)
    else
        xval = -1
    end if
    call print_positive(xval)

    if (xval > -1 .and. prodrisk_defined) then
        call set_prodrisk_xy(prodrisk, xval*2, xval)
    end if

    if (prodrisk_defined) then
        if (prodrisk%defined) then
            deallocate(prodrisk%value_a)
        end if
        deallocate(prodrisk)
    end if

contains
    function get_prodrisk_a_x (prodrisk)

    implicit none

        ! dummy arguments
        integer :: get_prodrisk_a_x
        type(prodriskcore), pointer, intent(in) :: prodrisk

        if (prodrisk%defined) then
        ! short circuit logic not guaranteed in fortran
            if (prodrisk%value_a%x_flag) then
                get_prodrisk_a_x = prodrisk%value_a%x
            else
                get_prodrisk_a_x = 0
            end if
        else
            get_prodrisk_a_x = -1
        end if
    end function get_prodrisk_a_x


    subroutine set_prodrisk_xy (prodrisk, x, y)

    implicit none

        ! arguments
        type(prodriskcore), pointer, intent(in) :: prodrisk
        integer, intent(in) :: x
        integer, intent(in) :: y

        if (.not. prodrisk%defined) then
            allocate(prodrisk%value_a)
            prodrisk%defined = .true.
        end if
        prodrisk%value_a%x = x
        prodrisk%value_a%y = y
        prodrisk%value_a%x_flag = .true.
        prodrisk%value_a%y_flag = .true.
    end subroutine set_prodrisk_xy

    subroutine print_positive (x)
    implicit none

        integer :: x

        if (x > -1) then
            print*, x

        else
            print*, 'None'
        end if

    end subroutine print_positive

end program main
```


SINTEF

Toolchains

Git

SVN (Subversion)

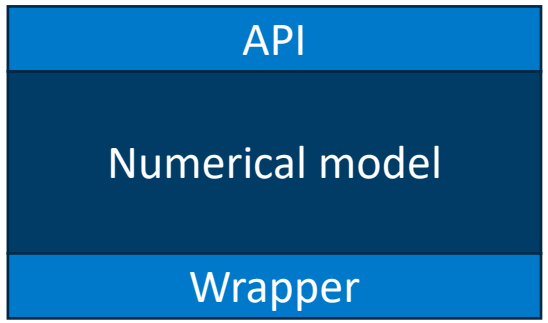
TeamCity build server

Markdown documentation

Model architecture

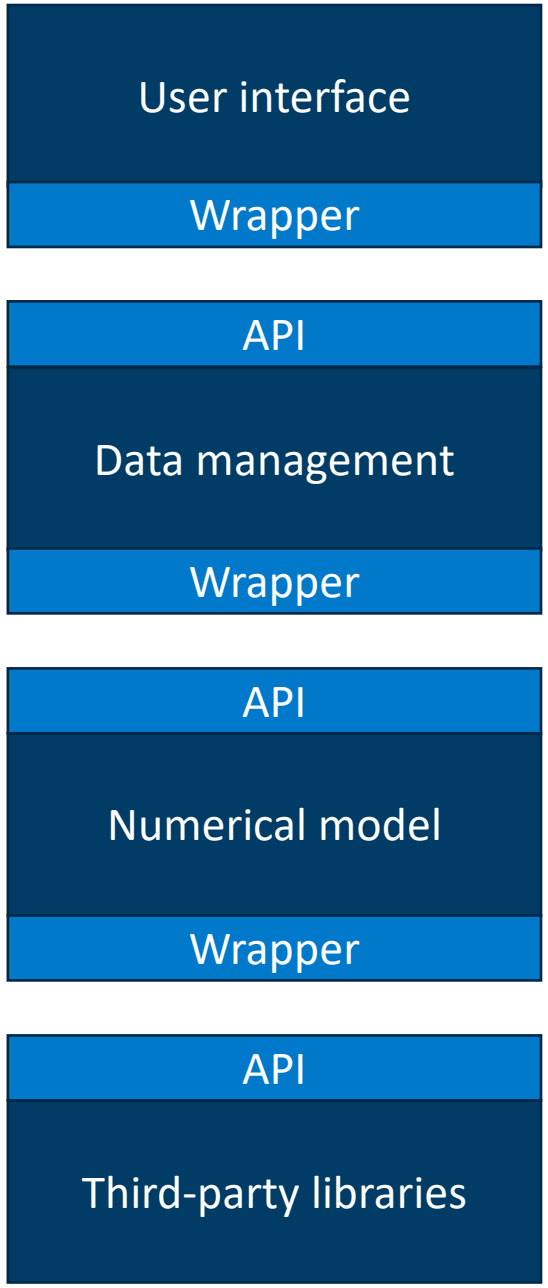
Numerical model

Model architecture



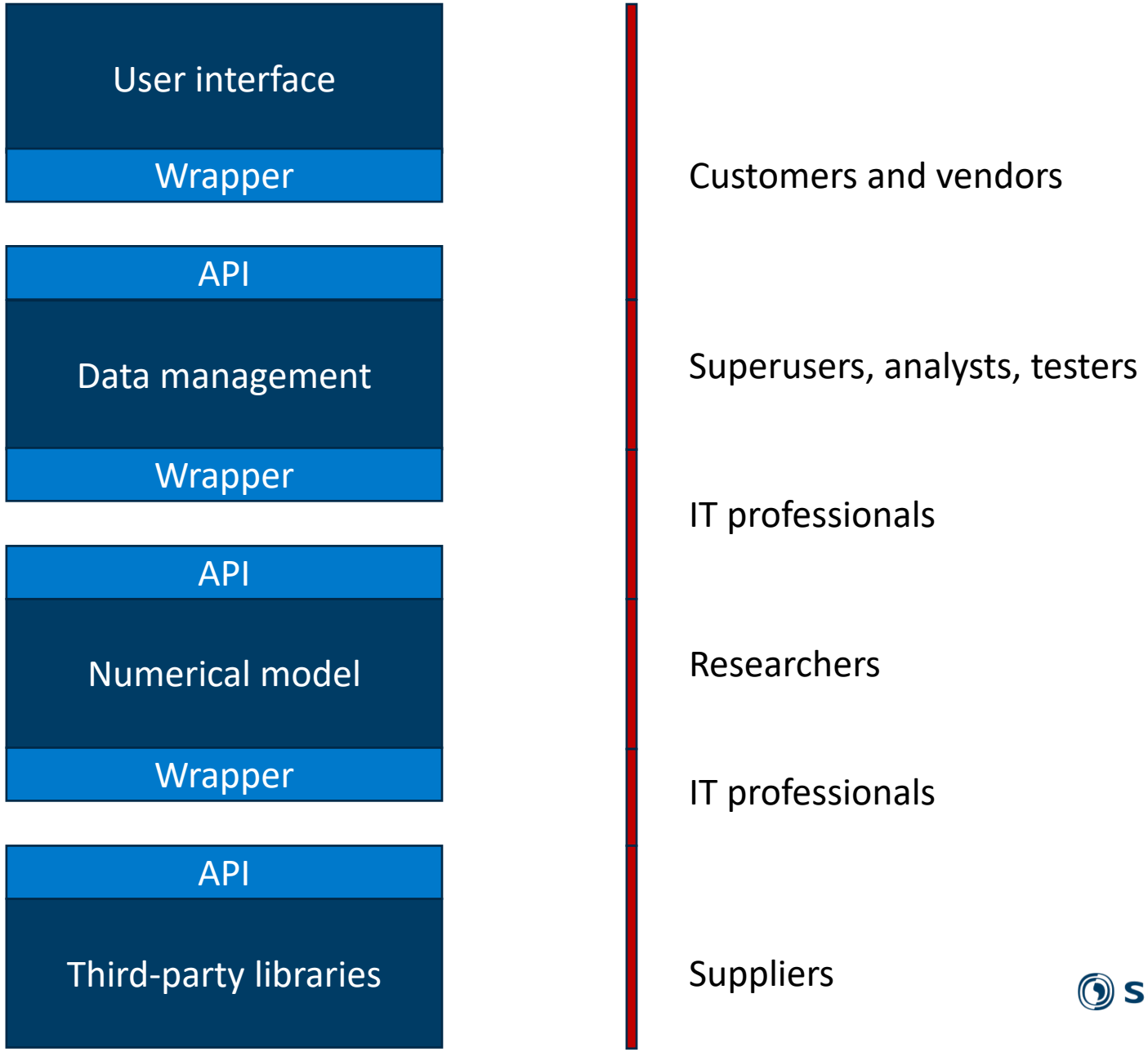
SINTEF

Model architecture



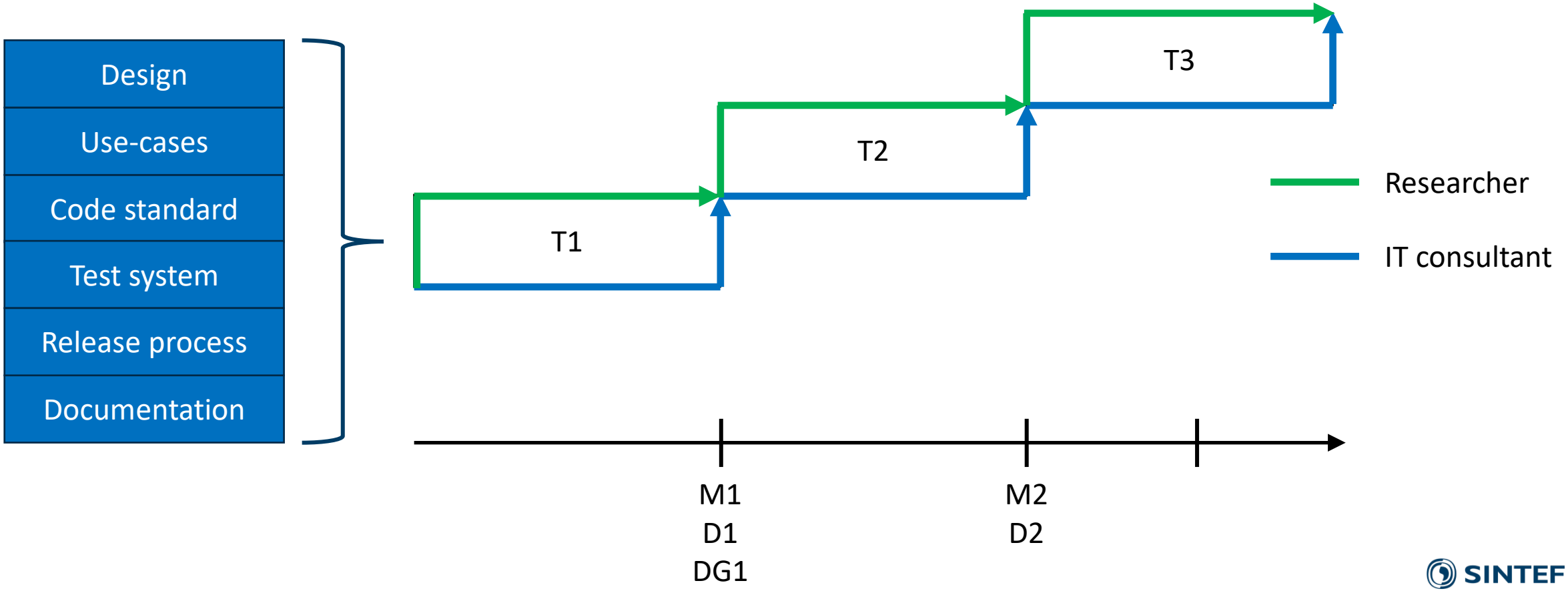
SINTEF

Model architecture



SINTEF

Realization in practice



SINTEF

Results from the Model Incubator project

- Recommended usage areas for different programming languages
- Roadmap for management and restructuring of legacy code
- Evaluation of toolchains for version control, testing and release
- Strategy for integration of IT professionals in the development process
- Assessments on actual code to demonstrate feasibility and impact

SINTEF

70 år

1950-2020

Teknologi for et bedre samfunn