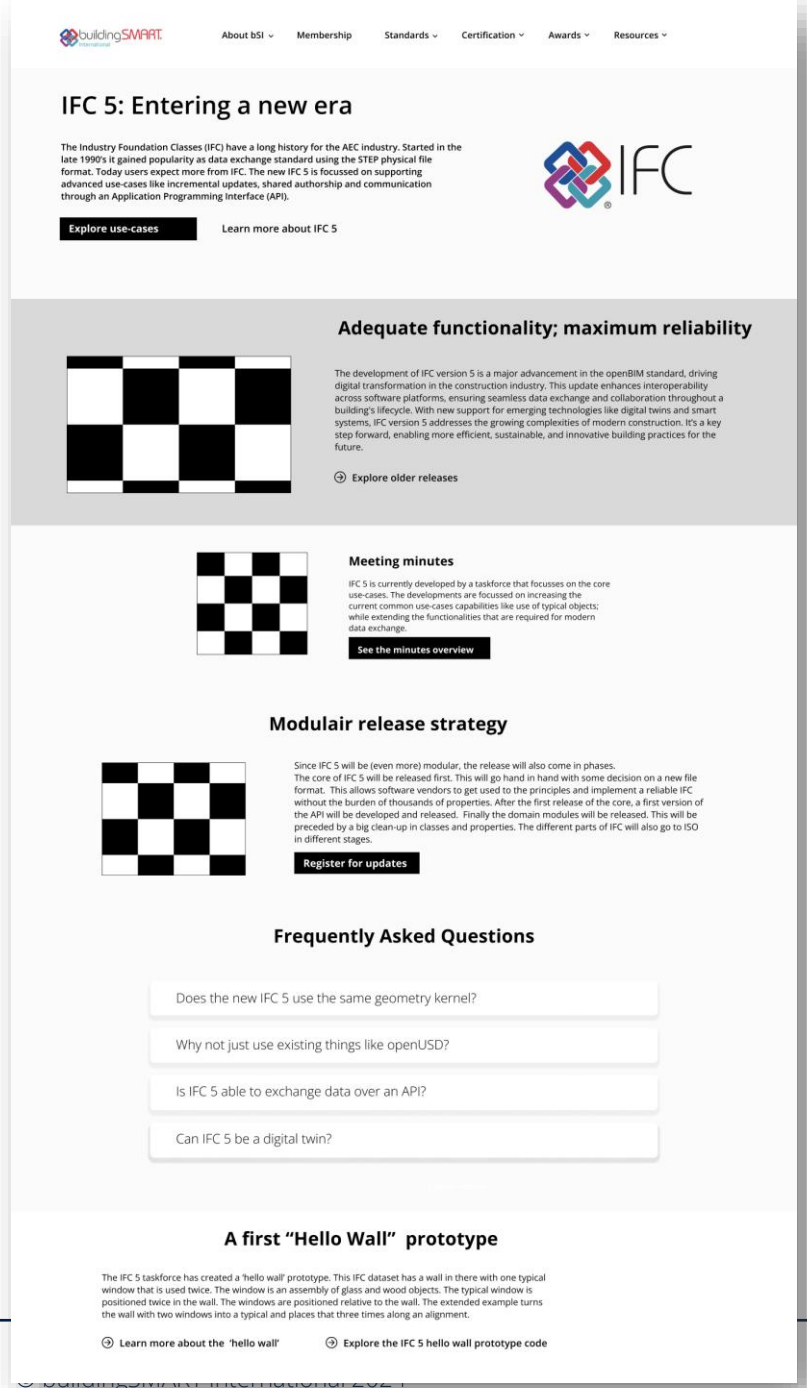
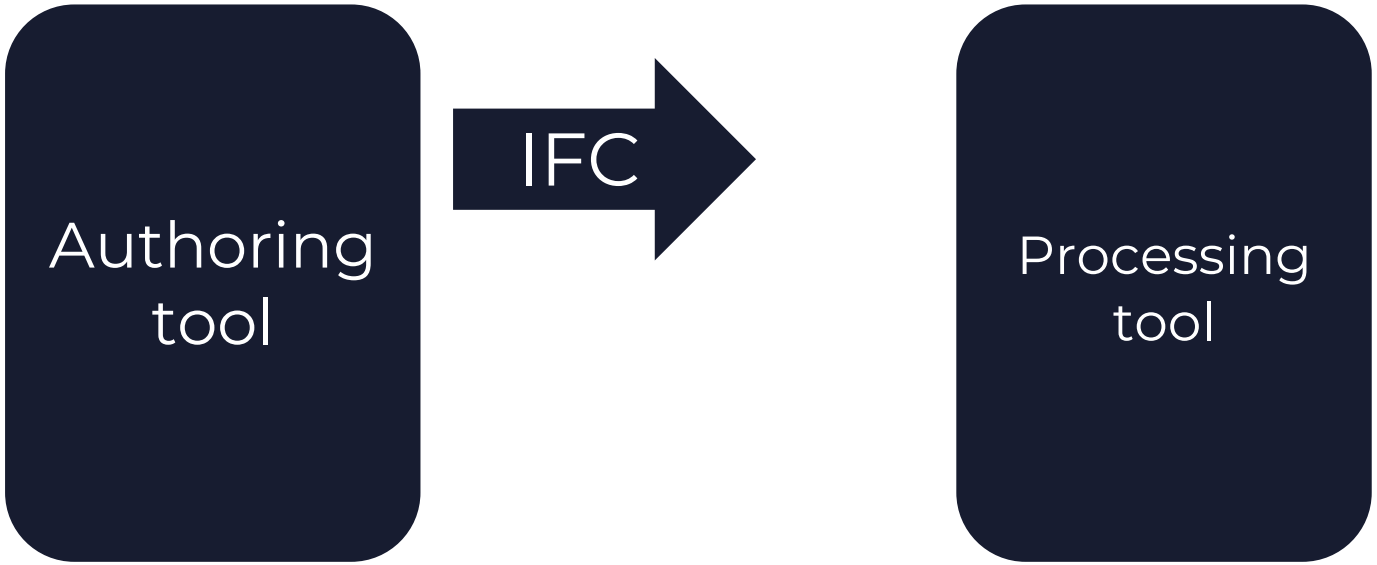


https://buildingsmart.org/standards/bsi-standards/ifc5

Adequate functionality
Maximum reliability

Adequate functionality
Maximum reliability

Yes, this has consequences
We need to reduce functionality (textures, etc)



Authoring tool

IFC

Processing tool

Authoring tool

IFC

Processing tool

Authoring tool

IFC

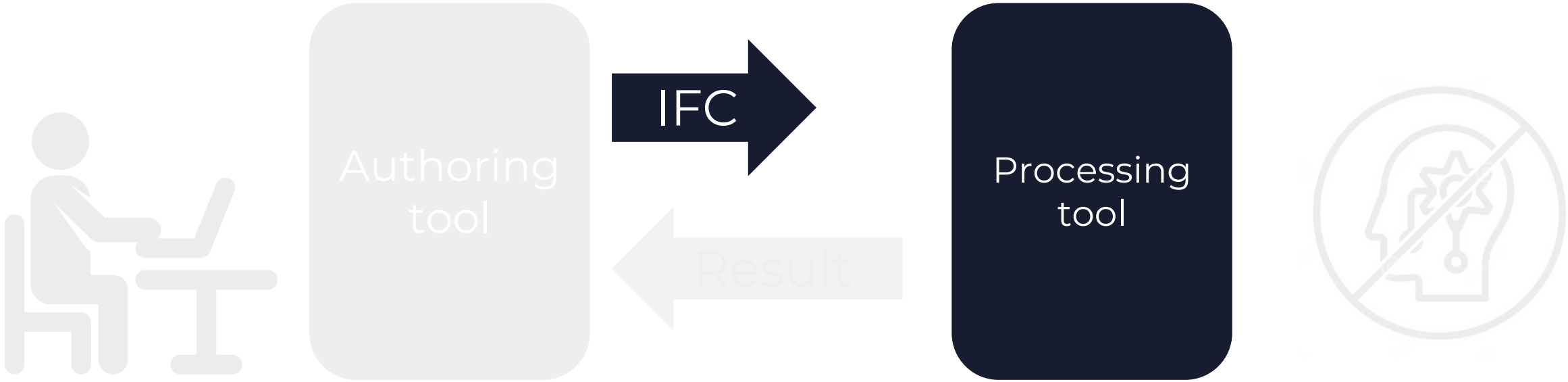
Result

Processing tool

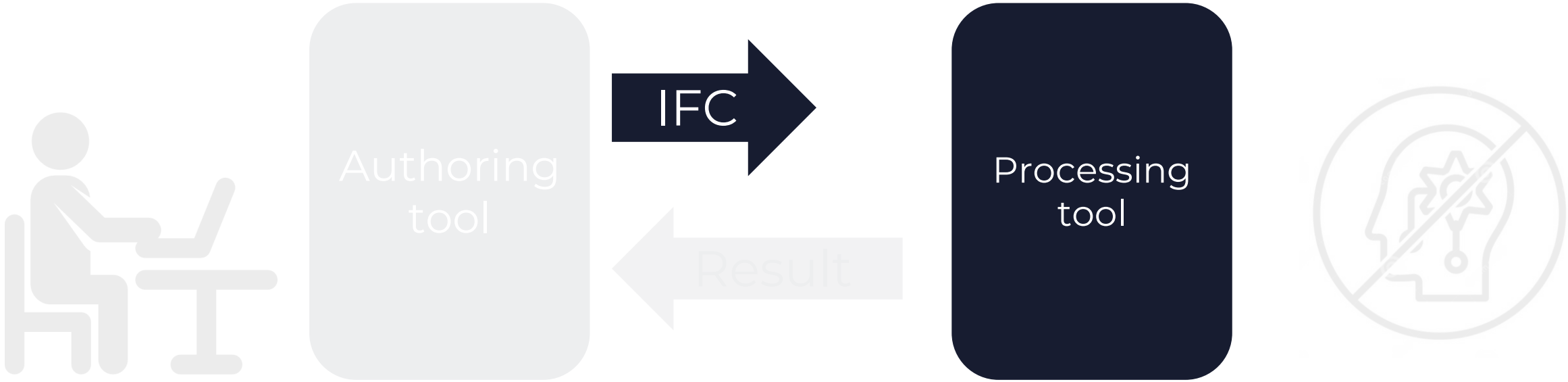
Authoring tool

IFC

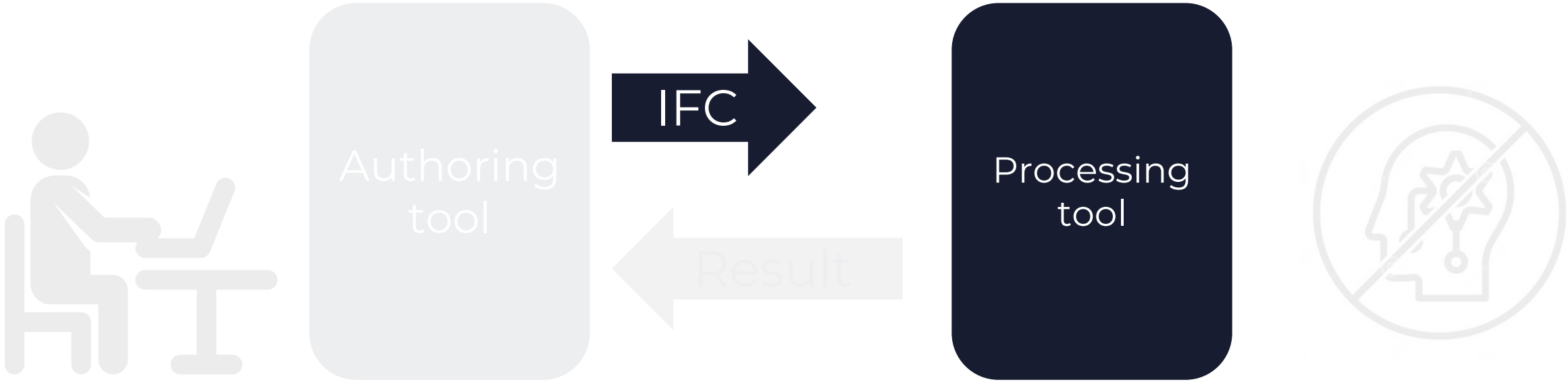
Processing tool



Focus for IFC5



Focus for IFC5
Maximum reliability



Predictable processing

Receiving software needs to understand it.
There needs be clarity and only one obvious way to do things.
No ambiguity and no human interpretation necessary.

IFC 5 objectives

Predictable processing

Receiving software needs to understand it.
There needs be clarity and only one obvious way to do things.
No ambiguity and no human interpretation necessary.

Easy to implement

As in: fast. Within a couple of days
Receiving software needs to understand it: without exceptions.
It needs to be predictable in implementation.
It might need to re-use other standards (like USD)
Yeah yeah... also an API

IFC 5 objectives

Predictable processing

Receiving software needs to understand it.
There needs be clarity and only one obvious way to do things.
No ambiguity and no human interpretation necessary.

Easy to implement

As in: fast. Within a couple of days
Receiving software needs to understand it: without exceptions.
It needs to be predictable in implementation.
It might need to re-use other standards (like USD)
Yeah yeah... also an API

One ~~exchange~~ standard

A data-model. Not a system.
But one for the whole industry (not just coordination)
Maybe even just one file format.

IFC 5 objectives

For buildingSMART:

Do the hard work to make it easy for others.

For buildingSMART:

Do the hard work to make it easy for others.

Focus on the minimal possible solution

For buildingSMART:

Do the hard work to make it easy for others.

Focus on the minimal possible solution

Deliver data model + API + set of rules

NOT for buildingSMART:

- How vendors implement it
- Complex solutions
- 100% coverage of every thinkable use-case
- Solutions that already exists (rendering, pointclouds, etc)

IFC 5 use-cases

New ones:
- Incremental updates (shared authorship)
- Automated processing
- Better types/typicals

IFC 5 use-cases

New ones:
- Incremental updates (shared authorship)
- Automated processing
- Better types/typicals

The ones we might need to drop **(for now)**
- Rendering
- Cost / scheduling
- Style of representation
- Parametric / complex geometry (just simple geometry in first version)

IFC 5 use-cases

New ones:
- Incremental updates (shared authorship)
- Automated processing
- Better types/typicals

The ones we might need to drop **(for now)**
- Rendering
- Cost / scheduling
- Style of representation
- **Parametric / complex geometry** (just simple geometry in first version)

IFC 5 use-cases

New ones:
- Incremental updates (shared authorship)
- Automated processing
- Better types/typicals

The ones we might need to drop (for now)
- Rendering
- Cost / scheduling
- Style of representation
- Parametric / complex geometry (just simple geometry in first version)

The future:
- IDS (requirements)
- BCF (topics/issues)

IFC 5 use-cases

New ones:
- Incremental updates (shared authorship)
- Automated processing
- Better types/typicals

Yeah yeah... also API and binary stuff

The ones we might need to drop (for now)
- Rendering
- Cost / scheduling
- Style of representation
- Parametric / complex geometry (just simple geometry in first version)

The future:
- IDS (requirements)
- BCF (topics/issues)

So base principles IFC 5

- Multiple authors can contribute to a shared data set
- Geometry and data is stored unambiguously
- Ease of implementing readers that can read all geometry and data (not necessarily processing/computing)
- Reference third party data into an IFC data set (the point cloud example)
- Ability to integrate data sets with “IDS 5” and “BCF 5”
- Modular data schemas (that allows validation against multiple data schemas separately)
- Express libraries of objects that can be placed in a data set
- Add properties of library objects on a per-instance, per-project
- Facilitate performant implementations
- Provide basis for partial, atomic transactions

What this means for IFC 5 (v1)

- Only simple geometry (preferably something existing)
- One way of doing placement (alignment)
- Relative placement (tree)
- Control data (placement to origin calculated) to check
- Assembly of components
- Data can be in 1 or n files; doesn’t matter
- Starting with limited number of classes and properties
- Multiple ‘opinions’ possible on the same data
- User decides what data is visualized/shown (layers?)
- Updates are the same as static datasets
- Clearly split authorship

Timeline

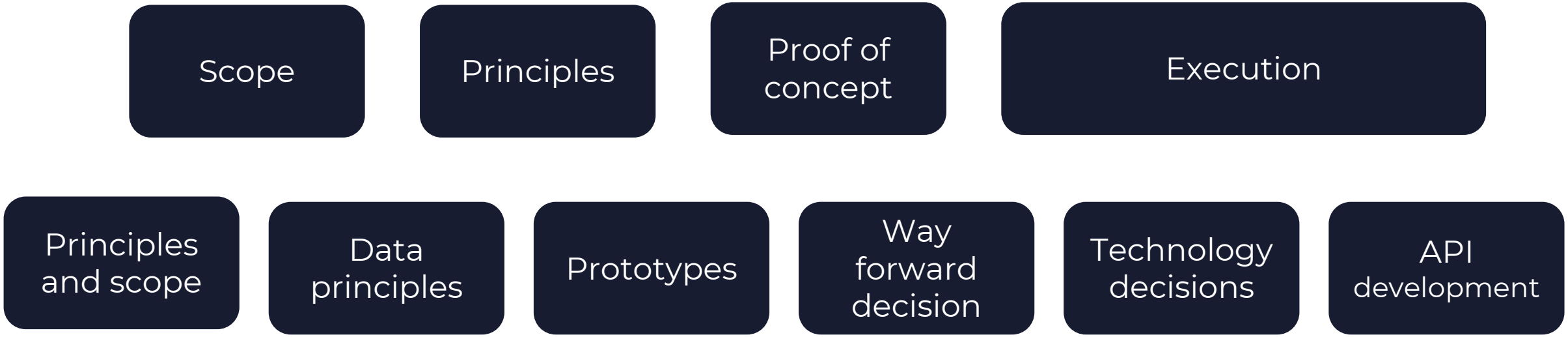
Scope

Principles

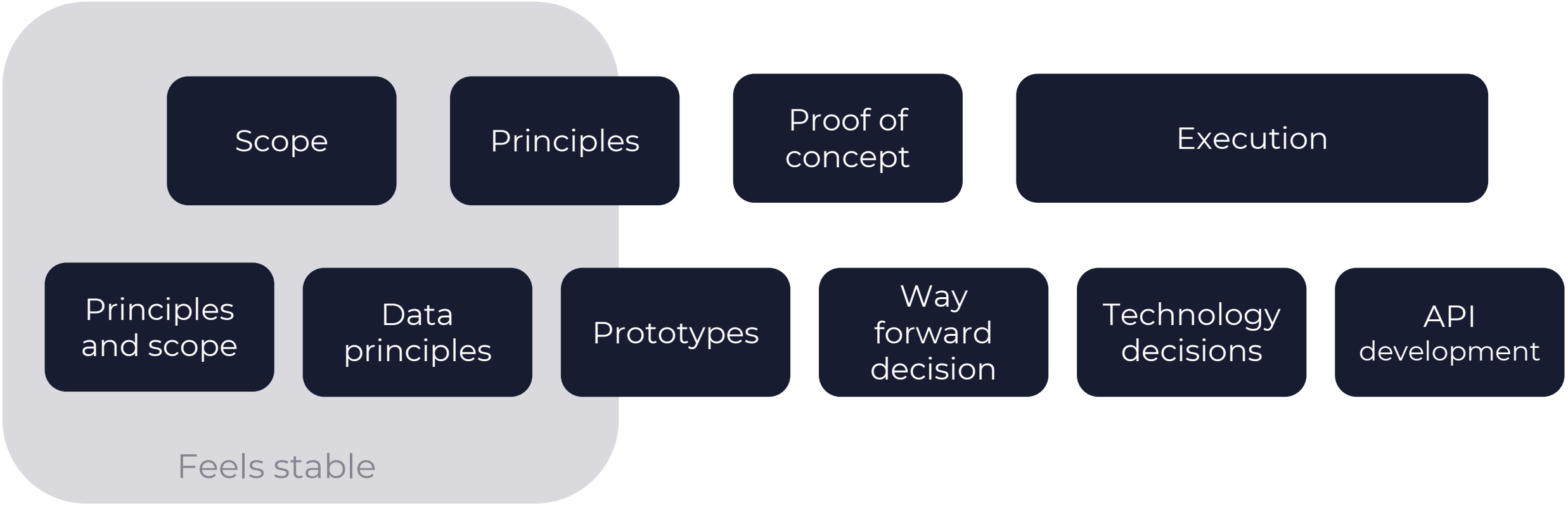
Proof of concept

Execution

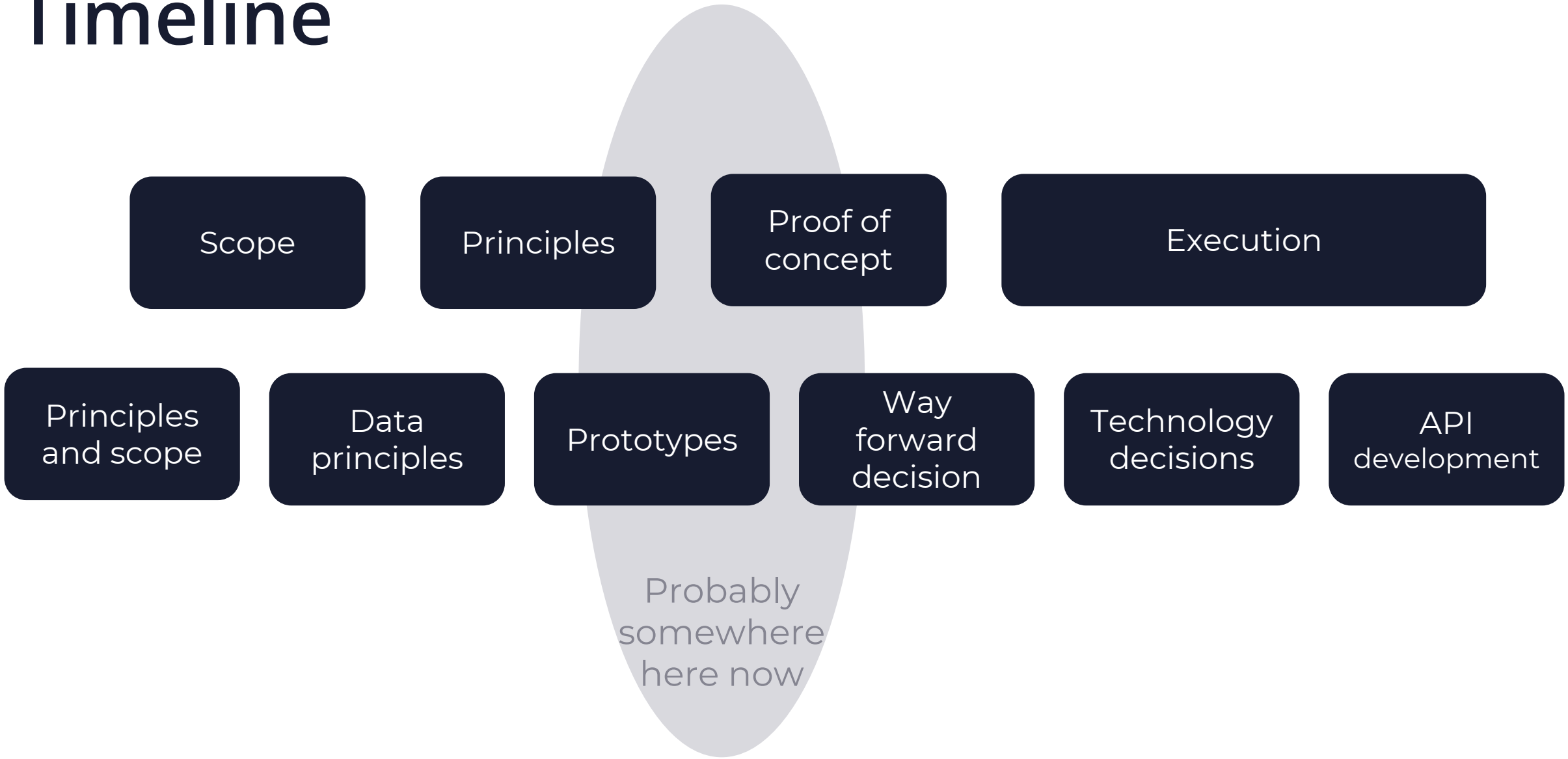
Timeline



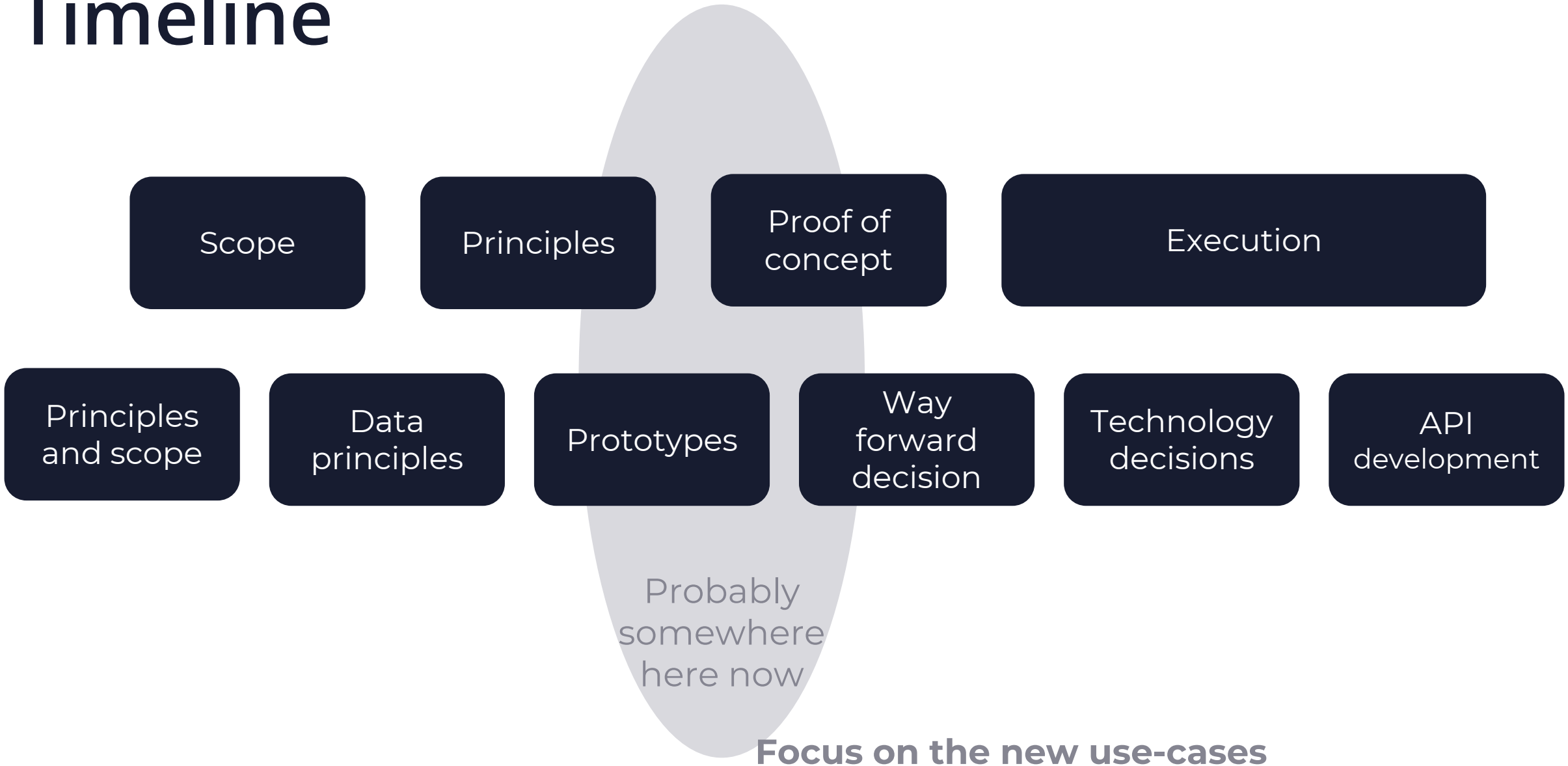
Timeline



Timeline

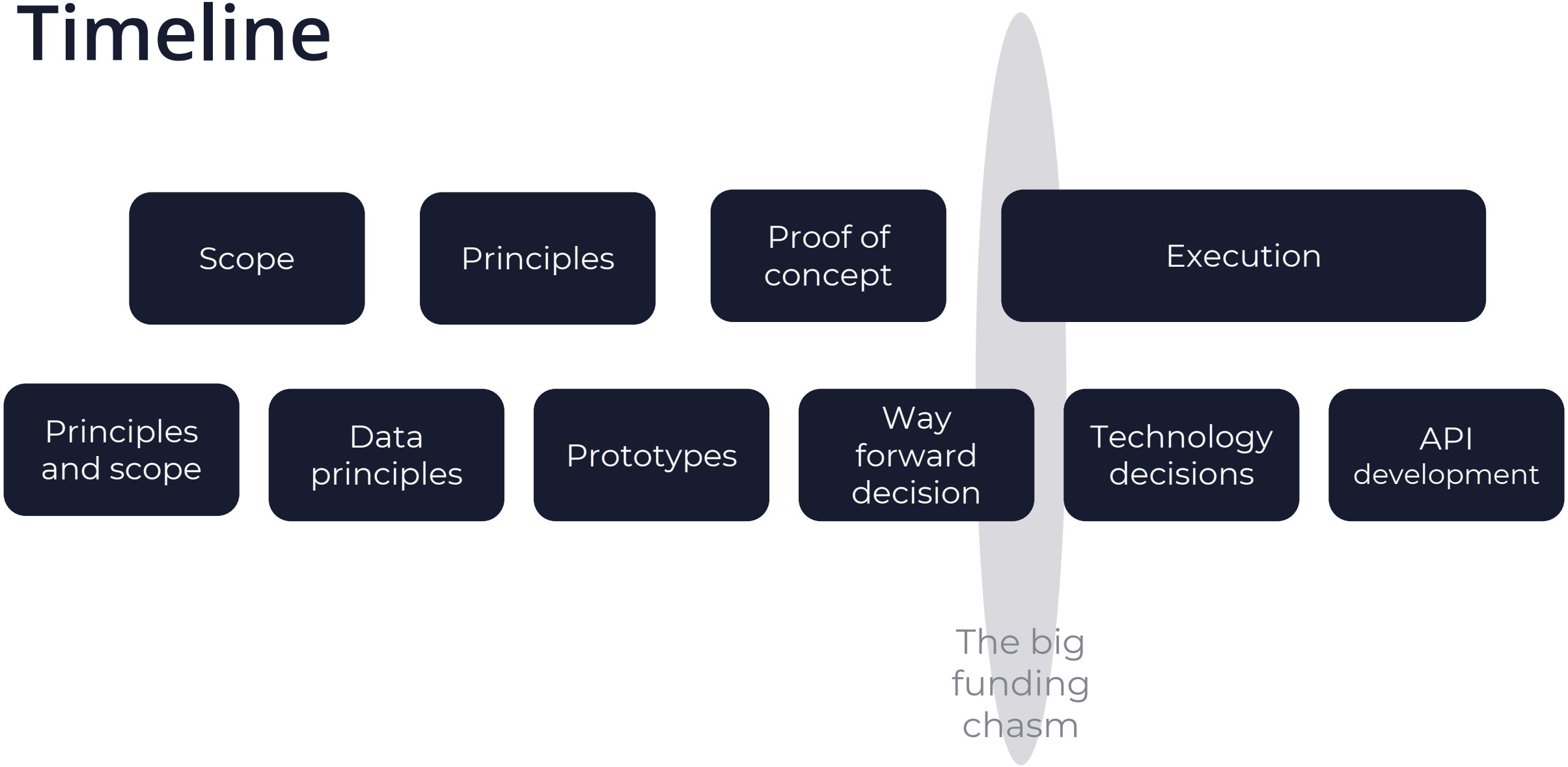


Timeline



Focus on the new use-cases

Timeline



Stop talking and show

Prototypes

What this means for IFC 5 (v1)

- Only simple geometry (preferably something existing)
- One way of doing placement (alignment)
- Relative placement (tree)
- Control data (placement to origin calculated) to check
- Assembly of components
- Data can be in 1 or n files; doesn’t matter
- Starting with limited number of classes and properties
- Multiple ‘opinions’ possible on the same data
- User decides what data is visualized/shown (layers?)
- Updates are the same as static datasets
- Clearly split authorship

Timeline at some point....

Drastically cut down lightweight minimum IFC 5

Super simple geometry;
no bells and whistles

Timeline at some point....

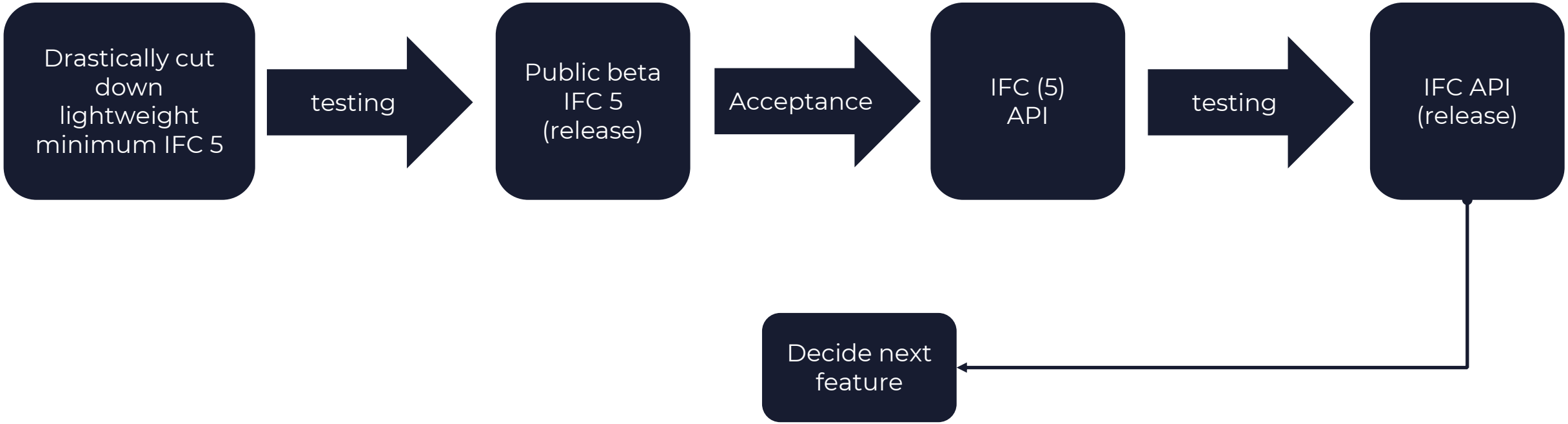
Timeline at some point....

Timeline at some point....

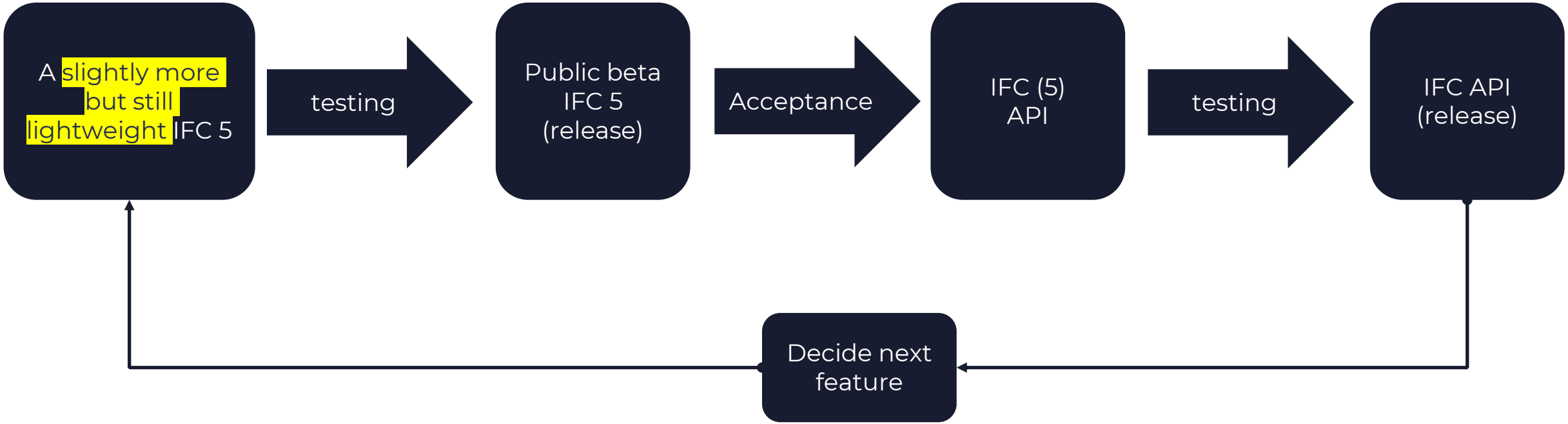
Timeline at some point....

Timeline at some point....

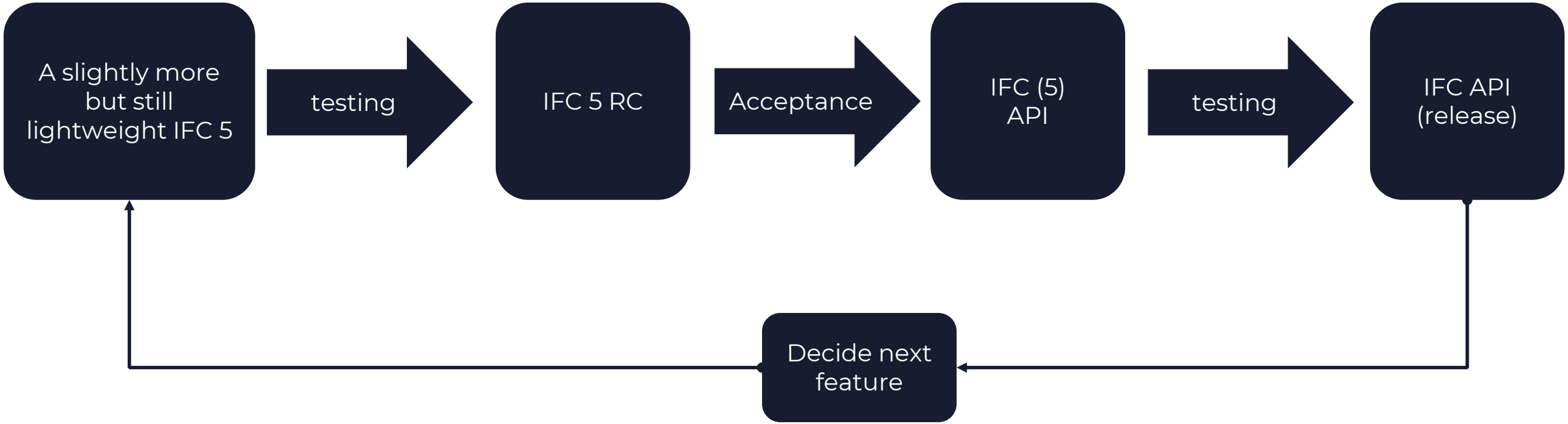
Timeline at some point....



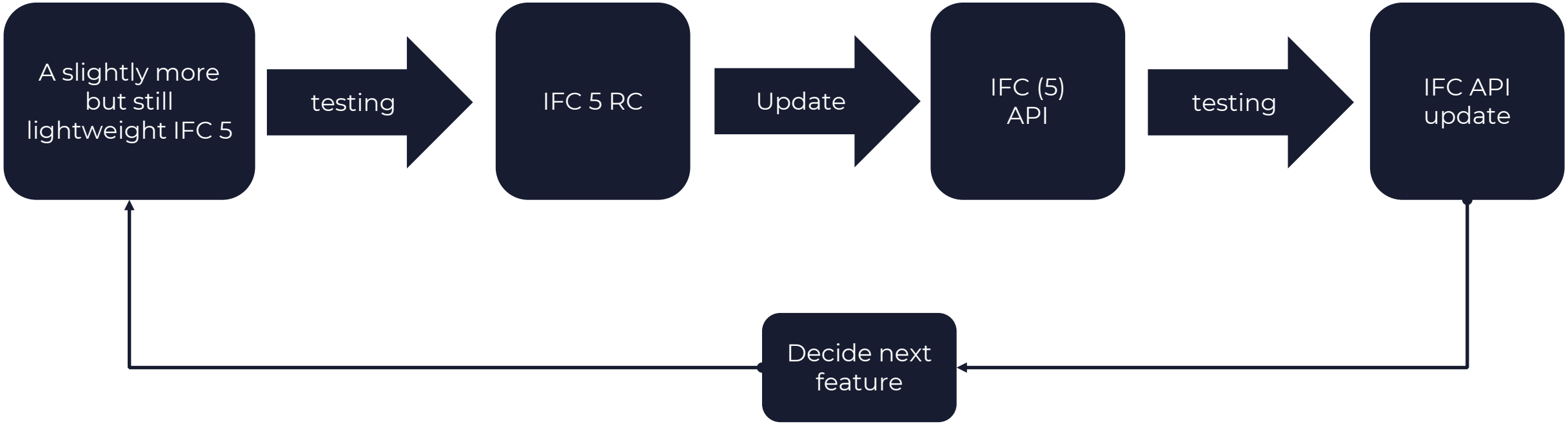
Timeline at some point….



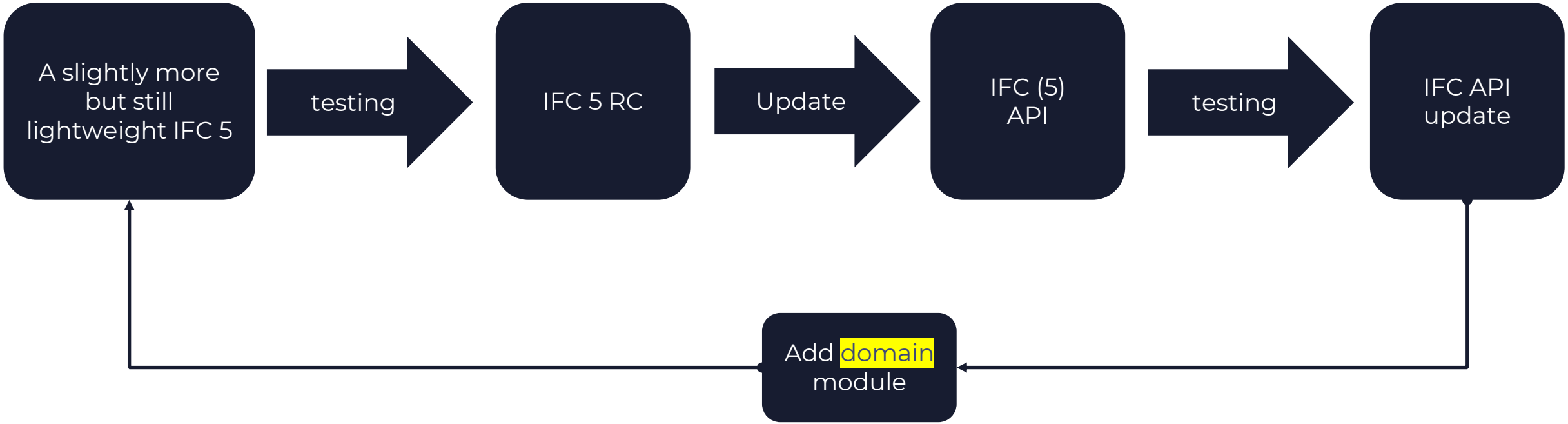
Timeline at some point....



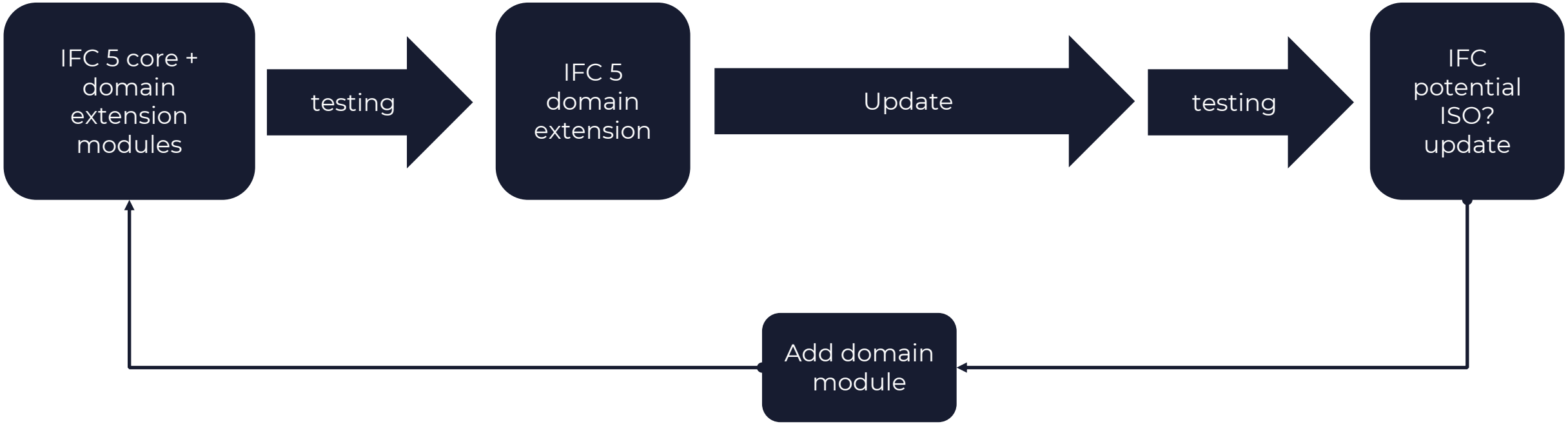
Timeline at some point....



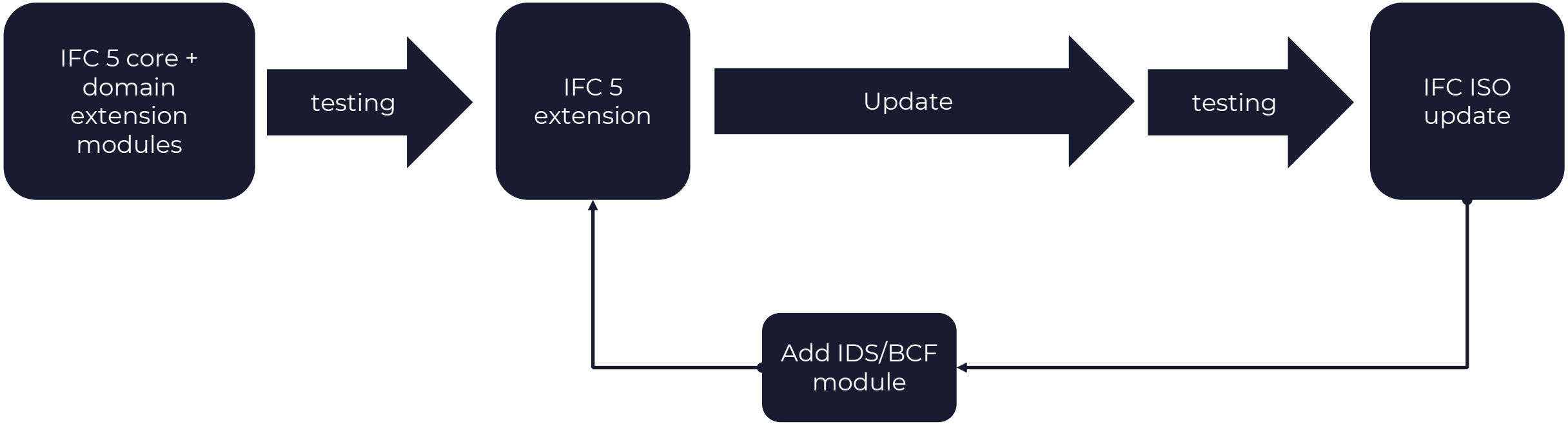
Timeline at some point....



Timeline at some point....



Timeline at some point....



Sidenote: IFC 5 core and modules

- Technical
- Organizational
- Release + Standardization bodies
  - ISO parts
  - Part 1: current IFC
  - Part x: new IFC core
  - Part xx: domain module
  - Etc…

Parts organization of ISO in next meeting

So a different way to visualize

IFC domain extensions

IFC core extensions ‘other’

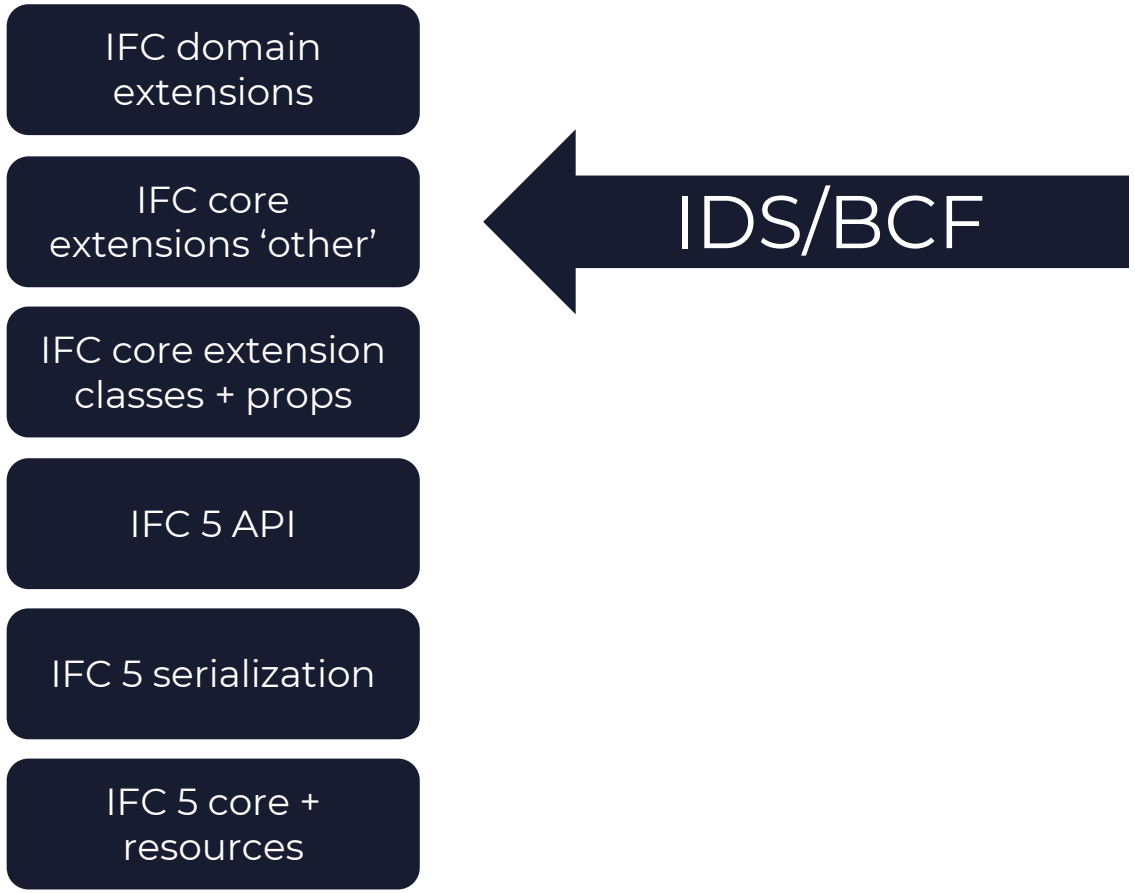
IFC core extension classes + props

IFC 5 API

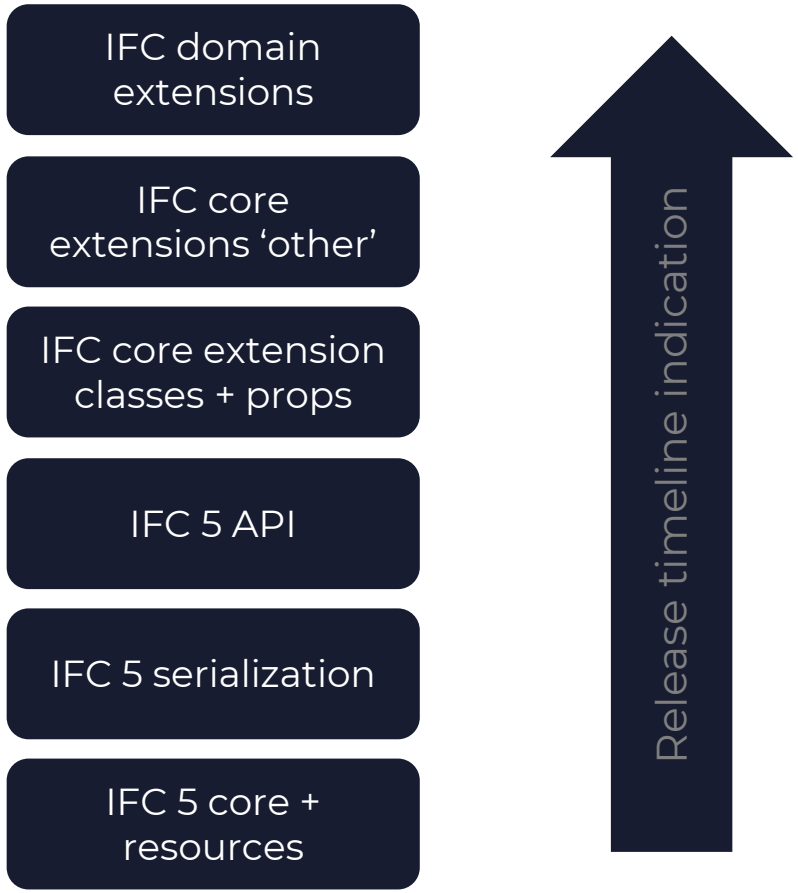
IFC 5 serialization

IFC 5 core + resources

So a different way to visualize



So a different way to visualize



So a different way to visualize

IFC domain extensions

IFC core extensions ‘other’

IFC core extension classes + props

IFC 5 API

IFC 5 serialization

IFC 5 core + resources

Notice a similarity?

So a different way to visualize

- IFC domain extensions
- IFC core extensions ‘other’
- IFC core extension classes + props
- IFC 5 API
- IFC 5 serialization
- IFC 5 core + resources

List of STEP (ISO 10303) parts

Add languages

Article Talk | Read Edit View history Tools

From Wikipedia, the free encyclopedia

An incomplete list of parts making up **STEP** (ISO 10303):

Descriptions methods [edit]

- Part 1 - *Overview and fundamental principles* (1994). Unfortunately outdated, not covering the role of AICs and modules.
- **Part 11 - *EXPRESS language reference manual***
- Part 12 - *EXPRESS -I language reference manual (withdrawn)*
- Part 14 - *EXPRESS -X language reference manual*

Implementation methods [edit]

- Part 15 - *SysML XMI to XSD transformation*[1]
- **Part 21 - STEP-File *Clear text encoding of the exchange structure***
- **Part 22 - SDAI *Standard data access interface specification***
- Part 23 - *C++ language binding of the standard data access interface*
- Part 24 - *C language binding of the standard data access interface*
- Part 25 - *EXPRESS to OMG XMI binding*
- Part 26 - *Binary representation of EXPRESS-driven data* using HDF5
- Part 27 - *Java TM programming language binding to the standard data access interface with Internet/Intranet extensions*
- **Part 28 - STEP-XML *XML representation for EXPRESS-driven data***

Conformance testing methodology and framework [edit]

- Part 31 - *General concepts*
- Part 32 - *Requirements on testing laboratories and clients*
- Part 34 - *Abstract test methods for application protocol implementations*
- Part 35 - *Abstract test methods for SDAI implementations*

Integrated generic resources [edit]

- **Part 41 - *Fundamentals of product description and support***
- **Part 42 - *Geometric and topological representation***
- **Part 43 - *Representation structures***
- **Part 44 - *Product structure configuration***
- Part 45 - *Materials*
- Part 46 - *Visual presentation*: Works in combination with part42 and allows to specify how to display 2D or 3D geometric models together with annotation data. The original design intend was that data according to this part could be displayed by computer systems supporting the Graphical Kernel System or PHIGS. Today other display interfaces such as OpenGL for 3D and Java 2D

So a different way to visualize

- IFC domain extensions
- IFC core extensions ‘other’
- IFC core extension classes + props
- IFC 5 API
- IFC 5 serialization
- IFC 5 core + resources

List of STEP (ISO 10303) parts

Add languages

Article Talk | Read Edit View history Tools

From Wikipedia, the free encyclopedia

An incomplete list of parts making up **STEP** (ISO 10303):

Descriptions methods [edit]

- Part 1 - *Overview and fundamental principles* (1994). Unfortunately outdated, not covering the role of AICs and modules.
- **Part 11 - *EXPRESS language reference manual***
- Part 12 - *EXPRESS -I language reference manual (withdrawn)*
- Part 14 - *EXPRESS -X language reference manual*

Implementation methods [edit]

- Part 15 - *SysML XMI to XSD transformation*[1]
- **Part 21 - STEP-File *Clear text encoding of the exchange structure***
- **Part 22 - SDAI *Standard data access interface specification***
- Part 23 - *C++ language binding of the standard data access interface*
- Part 24 - *C language binding of the standard data access interface*
- Part 25 - *EXPRESS to OMG XMI binding*
- Part 26 - *Binary representation of EXPRESS-driven data* using HDF5
- Part 27 - *Java TM programming language binding to the standard data access interface with Internet/Intranet extensions*
- **Part 28 - STEP-XML *XML representation for EXPRESS-driven data***

Conformance testing methodology and framework [edit]

- Part 31 - *General concepts*
- Part 32 - *Requirements on testing laboratories and clients*
- Part 34 - *Abstract test methods for application protocol implementations*
- Part 35 - *Abstract test methods for SDAI implementations*

Integrated generic resources [edit]

- **Part 41 - *Fundamentals of product description and support***
- **Part 42 - *Geometric and topological representation***
- **Part 43 - *Representation structures***
- **Part 44 - *Product structure configuration***
- Part 45 - *Materials*
- Part 46 - *Visual presentation*: Works in combination with part42 and allows to specify how to display 2D or 3D geometric models together with annotation data. The original design intend was that data according to this part could be displayed by computer systems supporting the Graphical Kernel System or PHIGS. Today other display interfaces such as OpenGL for 3D and Java 2D

Yes, we are turning ISO 16739 into parts as well

(current IFC 4.3 = part 1)

So a different way to visualize

IFC domain extensions

IFC core extensions ‘other’

IFC core extension classes + props

IFC 5 API

IFC 5 serialization

IFC 5 core + resources

IFC 5 timeline

- Gradual releases
- Finetuning as early as possible

- Critical success factor: keeping things out of scope

IFC 5 timeline

- Gradual releases
- Finetuning as early as possible

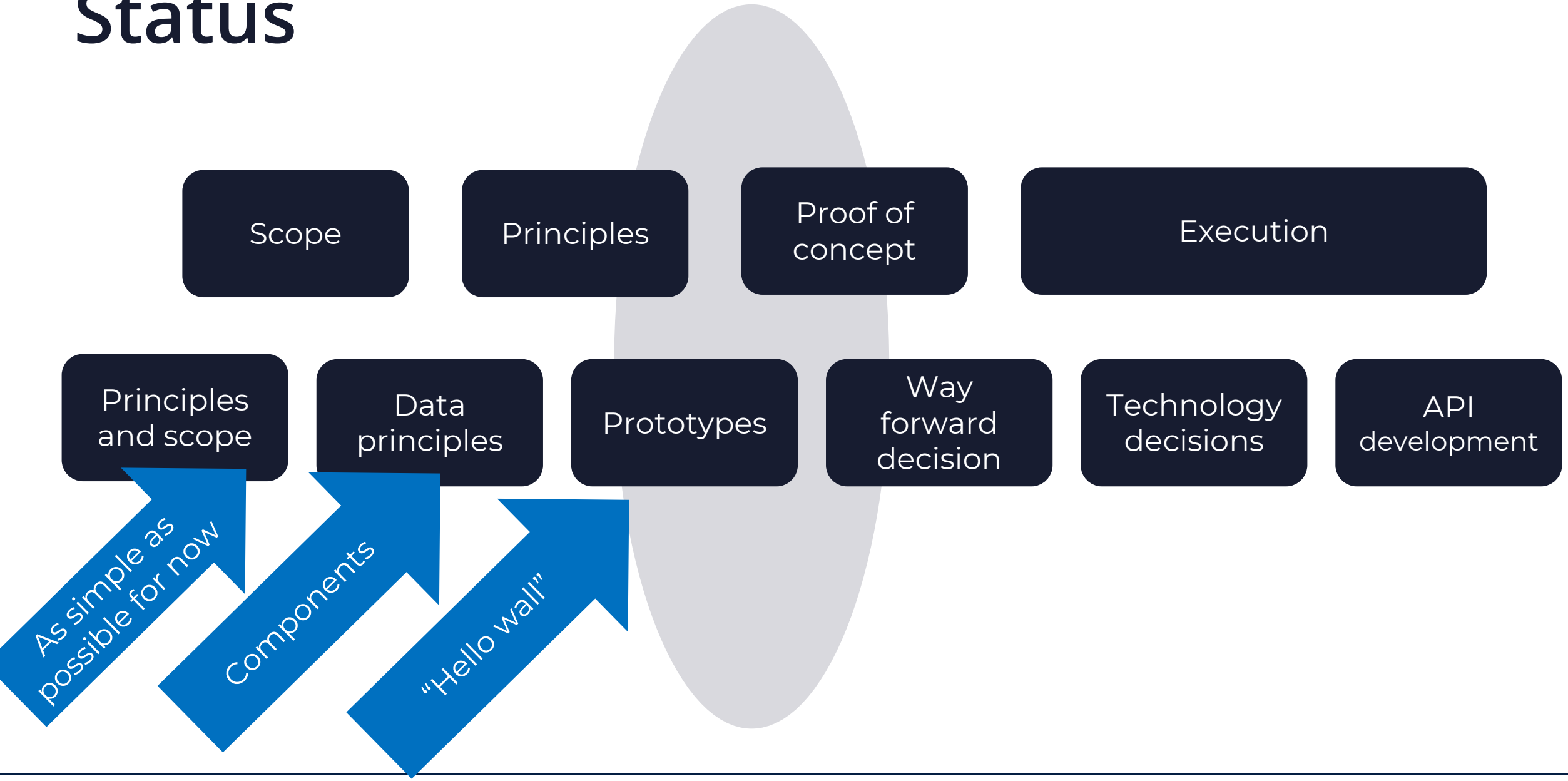
- Critical success factor: keeping things out of scope

Adequate functionality
Maximum reliability

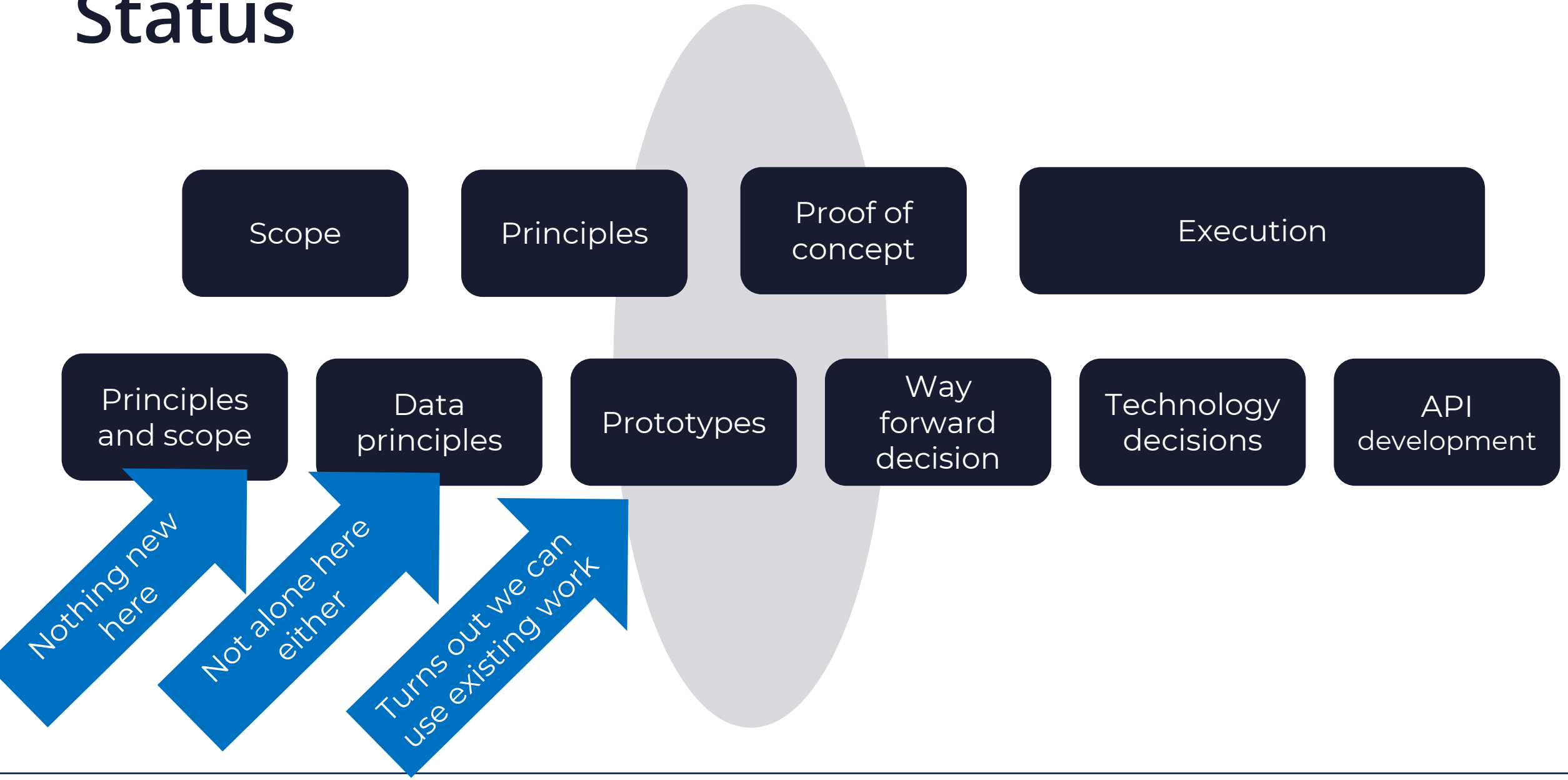
Stop talking and show

Prototypes

Status



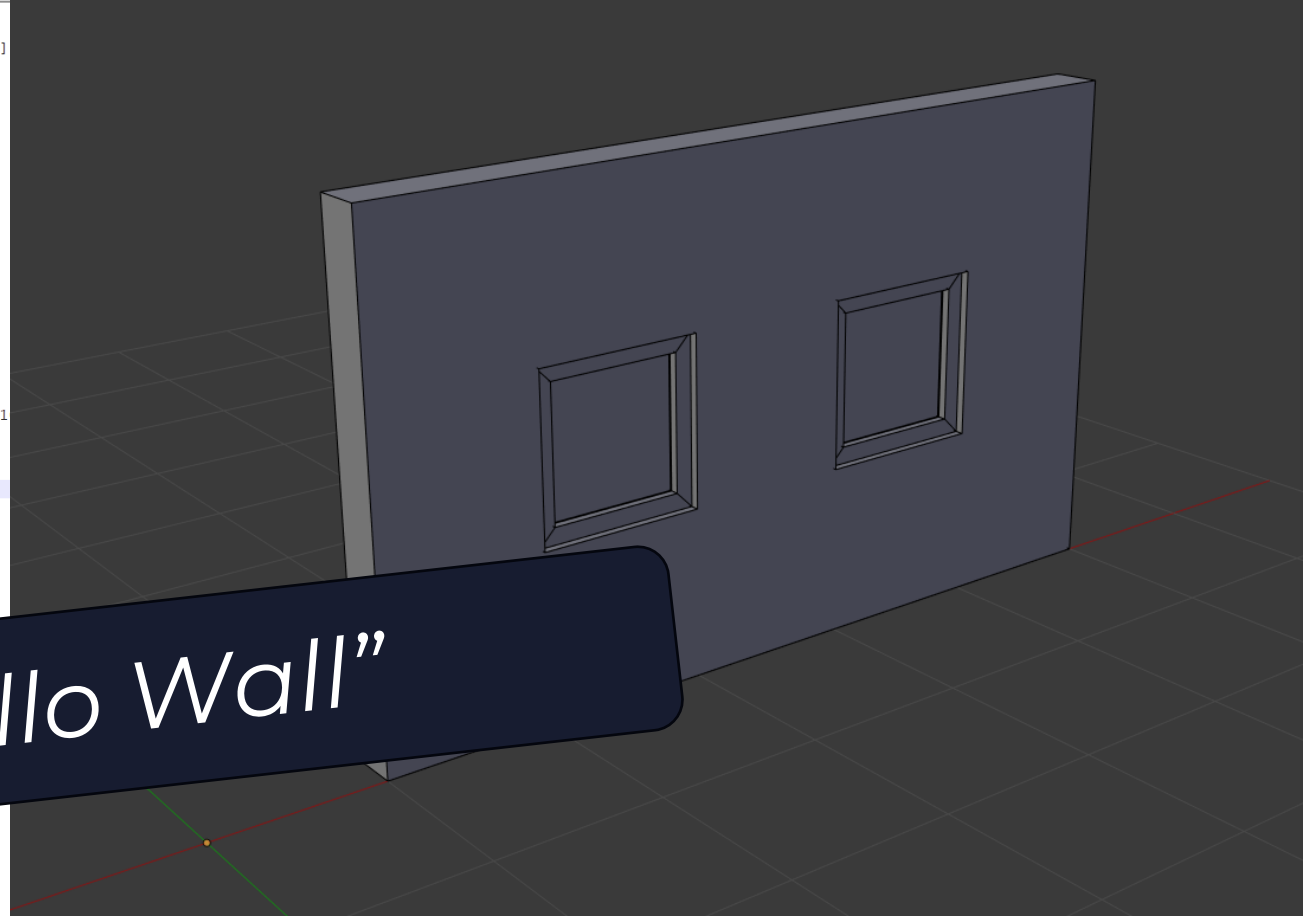
Status



```
# wall
{"type": "urn:bsi:ifc5:polyline", "entity": "wallpoly", "points": [[0.0, 0.0], [5.0, 0.0], [5.0, 0.3], [0.0, 0.3], [0.0, 0.0]]}
{"type": "urn:bsi:ifc5:extrusion", "entity": "wallext", "profile": {"points": [[0.0, 0.0], [5.0, 0.0], [5.0, 0.3], [0.0, 0.3], [0.0, 0.0]
{"type": "urn:bsi:ifc5:geometry", "tag": "axis", "entity": "wall", "ref": "wallpoly"}
{"type": "urn:bsi:ifc5:geometry", "tag": "body", "entity": "wall", "ref": "wallext"}
{"type": "urn:bsi:ifc5:relative-placement", "entity": "wall", "location": [1.0, 0.0, 0.0]}

# window type
{"type": "urn:bsi:ifc5:assembly", "entity": "windowtype", "ref": "frame"}
{"type": "urn:bsi:ifc5:assembly", "entity": "frame", "ref": "bar1"}
{"type": "urn:bsi:ifc5:assembly", "entity": "frame", "ref": "bar2"}
{"type": "urn:bsi:ifc5:assembly", "entity": "frame", "ref": "bar3"}
{"type": "urn:bsi:ifc5:assembly", "entity": "frame", "ref": "bar4"}
{"type": "urn:bsi:ifc5:assembly", "entity": "windowtype", "ref": "glazing"}
{"type": "urn:bsi:ifc5:type", "entity": "bar1", "ref": "bartype"}
{"type": "urn:bsi:ifc5:type", "entity": "bar2", "ref": "bartype"}
{"type": "urn:bsi:ifc5:type", "entity": "bar3", "ref": "bartype"}
{"type": "urn:bsi:ifc5:type", "entity": "bar4", "ref": "bartype"}
{"type": "urn:bsi:ifc5:geometry", "tag": "body", "entity": "bartype", "ref": "bargeometry"}
{
    "type": "urn:bsi:ifc5:triangle-mesh",
    "entity": "bargeometry",
    "positions":[[0.0, -0.05, 0.0], [0.0, -0.05, 1.0], [0.0, 0.05, 0.0], [0.0, 0.05, 1.0], [0.10, -0.05, 0.10], [0.10, -0.05, 0.90], [0.1
    "indices": [[1, 2, 0], [3, 6, 2], [7, 4, 6], [5, 0, 4], [6, 0, 2], [3, 5, 7], [1, 3, 2], [3, 7, 6], [7, 5, 4], [5, 1, 0], [6, 4, 0],
}
{
    "type": "urn:bsi:ifc5:relative-placement",
    "entity": "bar1",
    "axes":   [[1.0, 0.0, 0.0],
               [0.0, 1.0, 0.0],
               [0.0, 0.0, 1.0]],
    "location":[0.0, 0.0, 0.0]
}
{
    "type": "urn:bsi:ifc5:relative-placement",
    "entity": "bar2",
    "axes":   [[0.0, 0.0, -1.0],
               [0.0, 1.0,  0.0],
               [1.0, 0.0,  0.0]],
    "location":[1.0, 0.0,  0.0]
}
{
    "type": "urn:bsi:ifc5:relative-placement",
    "entity": "bar3",
    "axes":   [[ 0.0, 0.0, 1.0],
               [ 0.0, 1.0, 0.0],
               [-1.0, 0.0, 0.0]],
    "location":[ 0.0, 0.0, 1.0]
}
{
    "type": "urn:bsi:ifc5:relative-placement",
    "entity": "bar4",
    "axes":   [[-1.0, 0.0, -0.0],
               [ 0.0, 1.0,  0.0],
               [ 0.0, 0.0, -1.0]],
    "location":[ 1.0, 0.0,  1.0]
}
{"type": "urn:bsi:ifc5:geometry", "tag": "body", "entity": "glazing", "ref": "glazingext"}
{"type": "urn:bsi:ifc5:extrusion", "entity": "glazingext", "profile": {"points": [[0.0, 0.0], [0.8, 0.0], [0.8, 0.01], [0.0, 0.01], [0.0, 0.0]]}, "depth": 0.8}
{
    "type": "urn:bsi:ifc5:relative-placement",
    "entity": "glazing",
    "location":[0.1, 0.0, 0.1]
}
{"type": "urn:bsi:ifc5:geometry", "tag": "void", "entity": "windowtype", "ref": "windowvoid"}
{"type": "urn:bsi:ifc5:extrusion", "entity": "windowvoid", "profile": {"points": [[0.0, -0.5], [1.0, -0.5], [1.0, 0.5], [0.0, 0.5], [0.0, -0.5]]}, "depth": 1.0}


# window1 and 2
{"type": "urn:bsi:ifc5:type", "entity": "window1", "ref": "windowtype"}
```



“Hello Alignment”

Decisions:

- Automation first; predictable and stable as a result requirement
- Versioning is external of the data
- Layers per user; layer per change request (putting in layer from user when accepted)
- At some point discard history and ‘merge’ layers
- Communicate changes per layer (change request == change set)
- No groups anymore; or groups are components with one-to-one relations from the (instance)entity to the group entity
- Classification reference as a ‘property’
- Extensions cannot have information in the header
- Extensions also need to have the non-extension info (alignment needs to have resolved station points as well)
- Don’t oversell it
- Space boundaries need to be object
- Header in the layer/file to trace back where the component information came from
- GUID version: ask Autodesk and trimble what version they use
- UTF 8
- All SI units;
- Display units are different from storage units

Todo:

- Header info (version? Date? Hash?) → what needs to be in the header besides what is already in USD?
- Example file with GUIDs
- Example file from Revit
- Web rendering / binary file streaming
- Experiment with IFC 5 schema on top of USD
- How fast is composition on a flat list

- Explain why we don't want extrusions anymore; but 'original'info about how a mesh is created
- API?
- Change things
- How are we going to call it?
- Guidelines for 'diff' (which is just data)