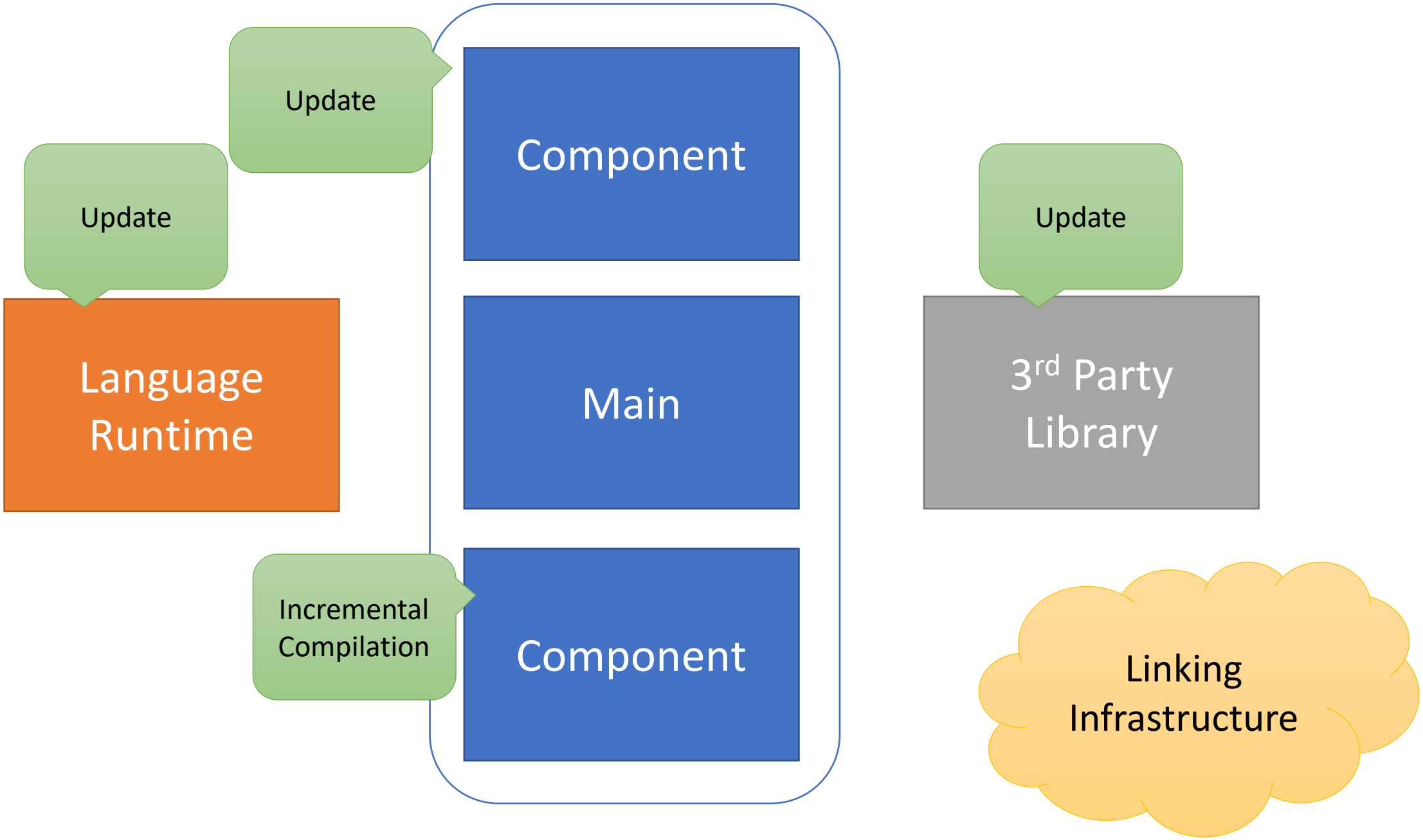
Roadmaps

Ross Tate

Developing a Roadmap

- Identify Goals
- Determine Key Challenges
- Design Strategy and Invariants
- Stage Releases


Update
Update
Component
Update
Language Runtime
Main
3rd Party Library
Incremental Compilation
Component
Linking Infrastructure

Developing a Roadmap

- Identify Goals
- Determine Key Challenges
- Design Strategy and Invariants
- Stage Releases

How to describe memory layouts?

Type Feature

1. Prefix Subtyping
2. Record Subtyping
3. Nominal Typing

Space Abstraction?

1. Cannot hide/reorder/change fields of superclasses
2. Cannot ensure same field is at same offset across references
3. Works (at least so far)

Improved Reusability

Without Space Abstraction

```
Module Crypto {
   void encrypt(i32[] ints,
      i32 from, i32 to);
}
```

Not usable by most languages!

Diff languages have diff arrays

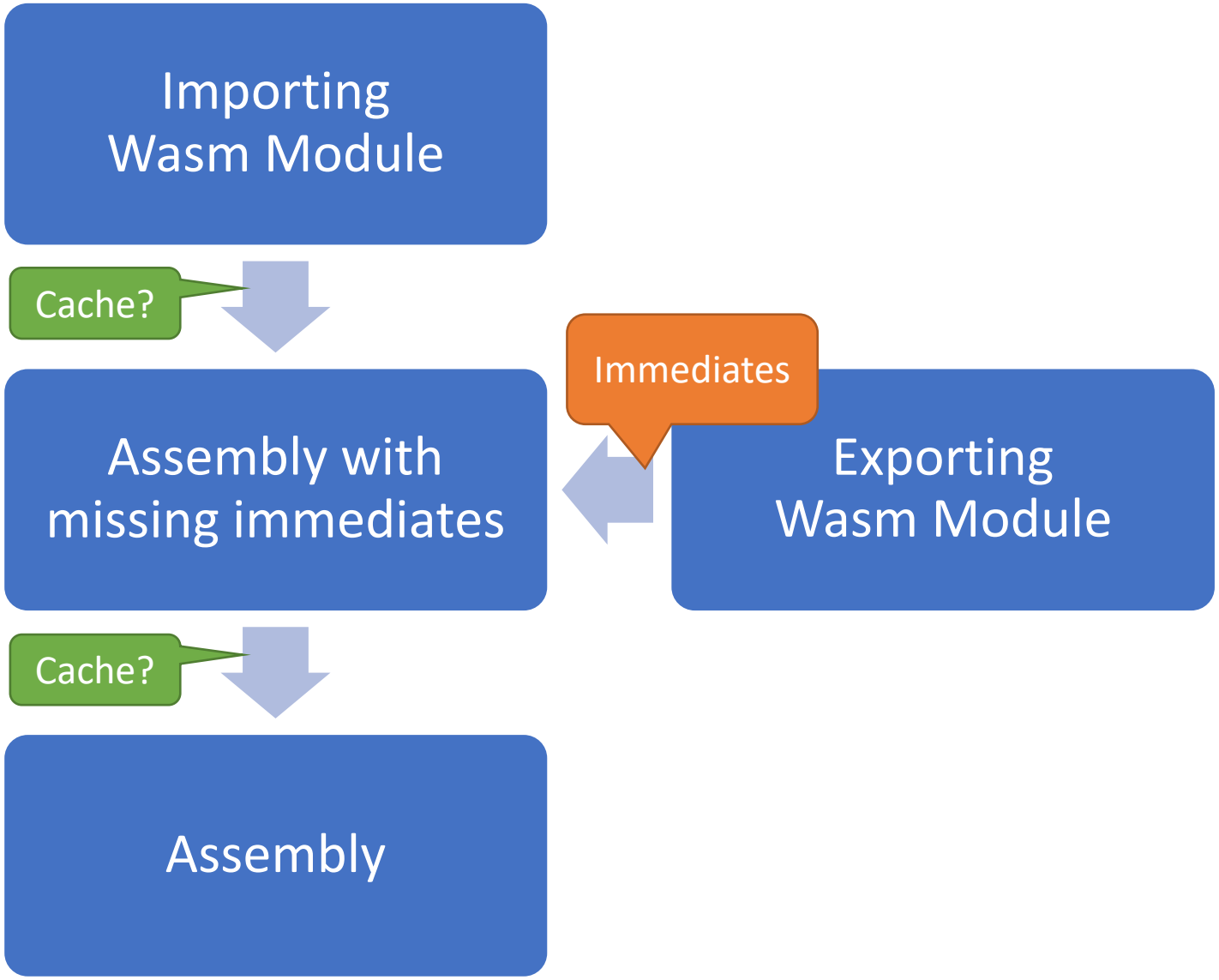
With Space Abstraction

```
Module Crypto {
   import scheme $i32s
      with indexed field int : i32;
   void encrypt((gcref $i32s) ints,
      i32 from, i32 to);
}
```

Usable by most languages!

Including linear memory!

How?



Module links against
same runtime module
⇓
Cache hits

Developing a Roadmap

Stage 1:

Identify Goals

- Export satisfies import only if fields in same order
- Compatible with Single-Stage Compilation

Stage 2?

Determine Key Challenges

- Export/import field order can differ
- Requires Two-Stage Compilation *if used*

Design Strategy and Invariants

Stage Releases

Developing a Roadmap

- Identify Goals
- Determine Key Challenges
- Design Strategy and Invariants
- Test Validity
- Stage Releases

Existing Systems

Industry

- JVM
- .NET

Research

- LILC+SST
- iTalX

Nominal

- LISP/Scheme/Racket

anyref

- jFlint

Structural

Evaluating a Roadmap

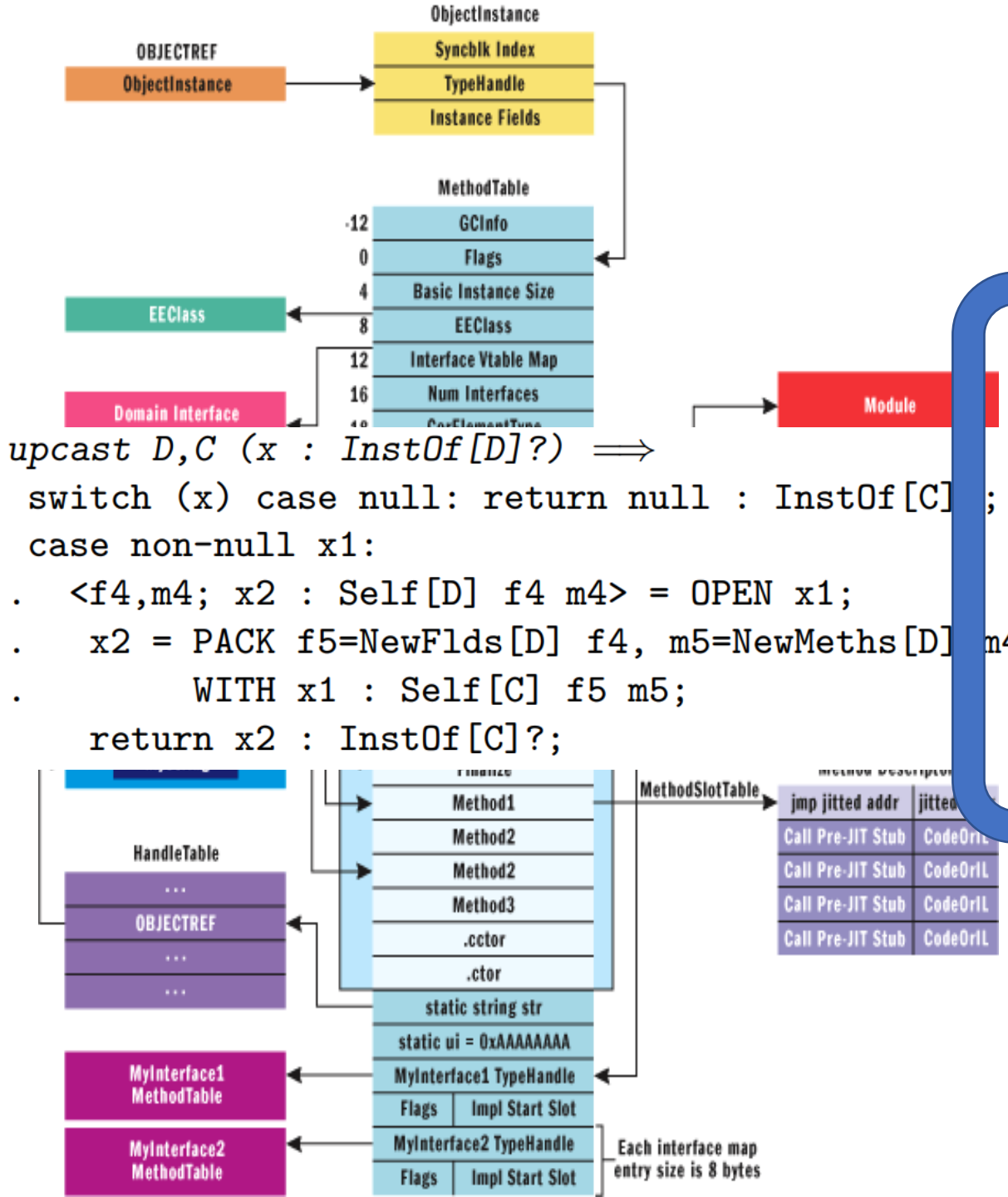
- Identify Goals
- Determine Key Challenges
- Design Strategy and Invariants
- Test Validity
- Stage Releases

Uniform Representation

- Claim
  - functional languages will use funcref for closures
- Reality
  - Many functional languages need examinable closures
  - E.g. Scheme uses “eq?” to compare function pointers
    - But funcref is not a subtype of eqref
  - E.g. Scheme can (optionally) serialize closures
    - But funcref provides no way to look at closed-over values

Evaluating a Roadmap

- Identify Goals
- Determine Key Challenges
- Design Strategy and Invariants
- Stage Releases




```
upcast D,C (x : InstOf[D]?) ⟹
 switch (x) case null: return null : InstOf[C]?;
 case non-null x1:
.   <f4,m4; x2 : Self[D] f4 m4> = OPEN x1;
.    x2 = PACK f5=NewFlds[D] f4, m5=NewMeths[D] m4
.         WITH x1 : Self[C] f5 m5;
    return x2 : InstOf[C]?;
```

- Structural casts
  - Heavy machinery & unpredictable
- Nominally-tagged casts
  - Might as well use nominal types
- Existential types
  - Burdensome and inexpressive without nominal types

Dynamic Languages

Scheme's "equal?"

- Deep equality
- >10 cases

Structural/ Uniform

- Cast anyref
- >10 times

Nominal/ Specialized

- Enumerate cases
- Switch table

Evaluating a Roadmap

Identify Goals

Design Strategy and Invariants

Stage Releases

Evaluating a Roadmap

Everyone else went the other way after hitting a dead end

Identify Goals

Unsolved key challenges

Design Strategy and Invariants

No language encodings

Shipping features with no known uses