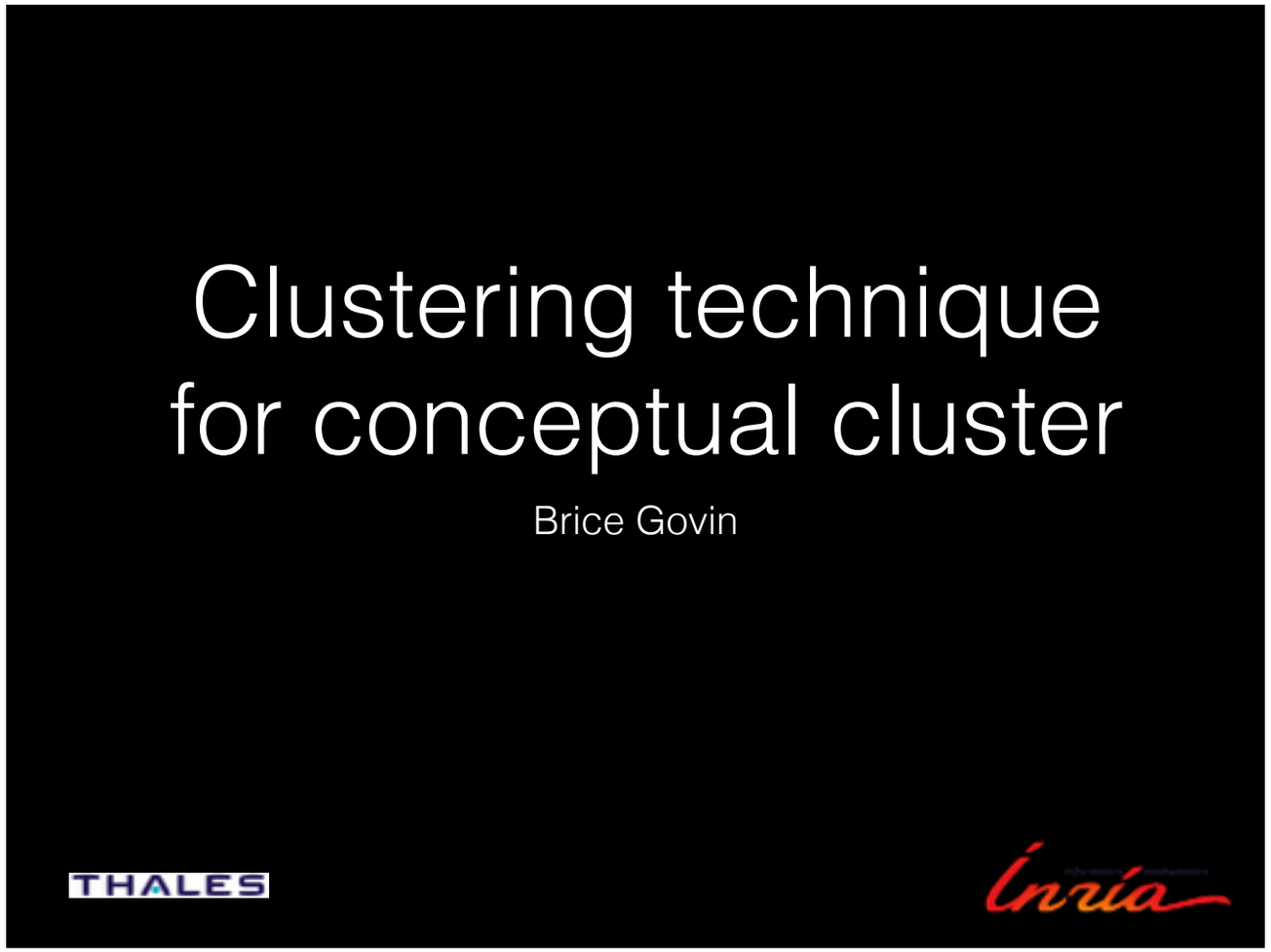
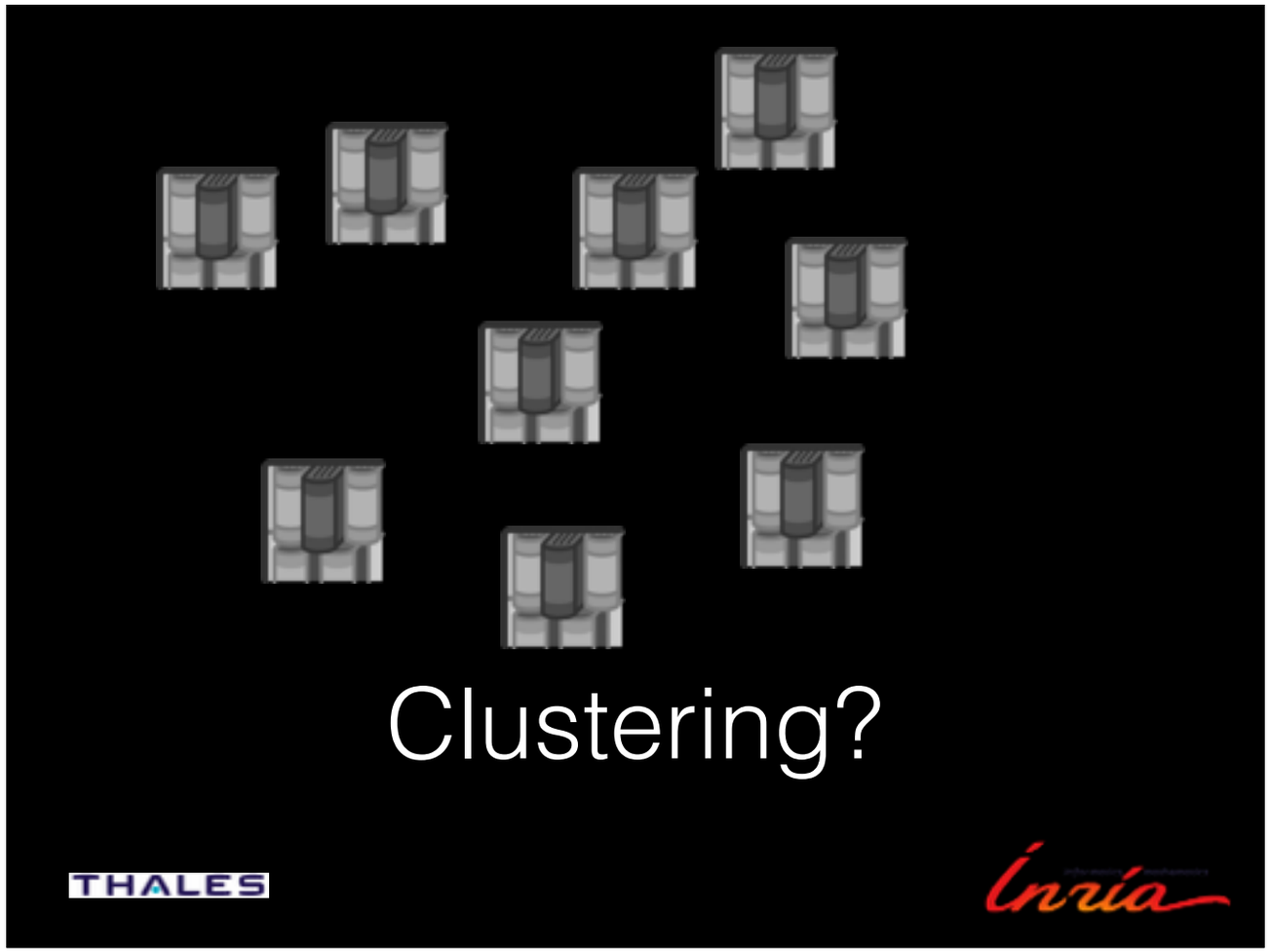


Good morning everybody, for those who don’t know me, I’m Brice Govin and I’m a PhD student in the RMOD team at Inria Lille.
I’m gonna talk to you about a clustering technique we’ve developed in Pharo to recover conceptual cluster.

Clustering technique for conceptual cluster

- Clustering ?
- Concept?
- An iteration to rule them all
- And in an approach bind them
- And come the hobbits and their issues

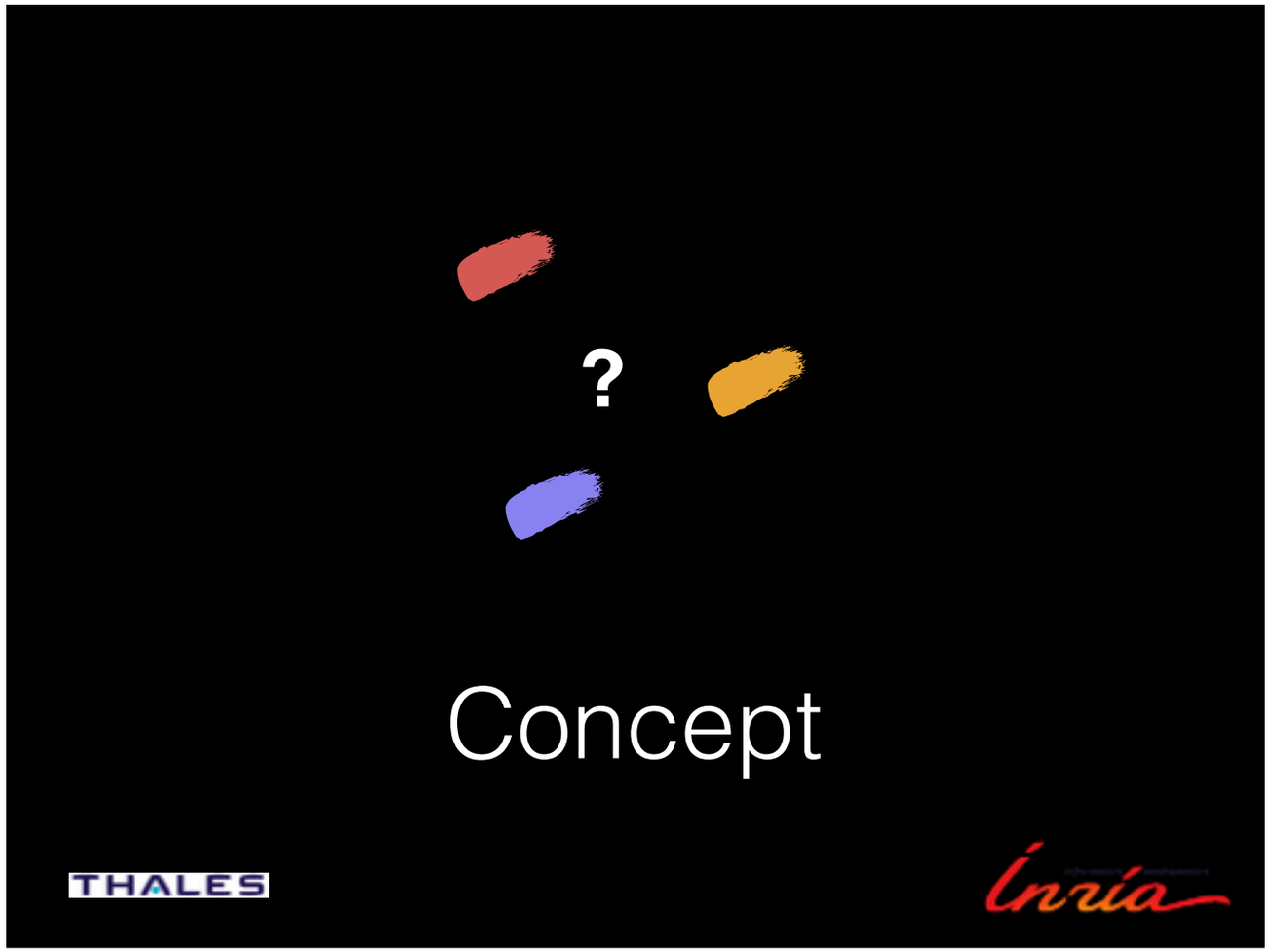


First of all, clustering is a way to gather elements according to parameters.
Basically, when you gather books according to their topic, you are doing clustering where books are elements and topic is the parameter according to which you gather the books.
Of course, clustering is not only use for books but also in computer science from reverse engineering to re-engineering and this is why I'm here today

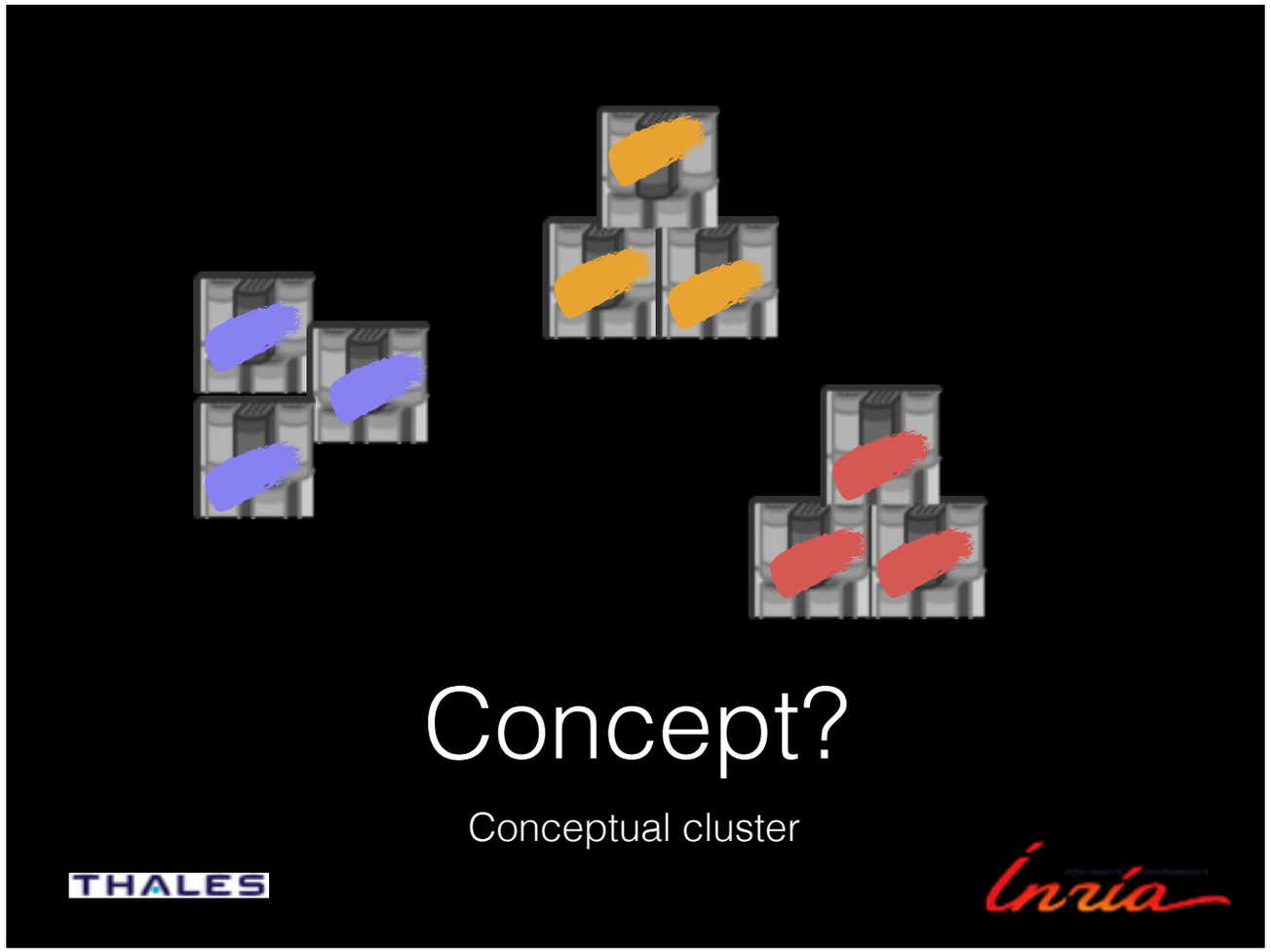
First of all, clustering is a way to gather elements according to parameters.
Basically, when you gather books according to their topic, you are doing clustering where books are elements and topic is the parameter according to which you gather the books.
Of course, clustering is not only use for books but also in computer science from reverse engineering to re-engineering and this is why I'm here today

First of all, clustering is a way to gather elements according to parameters.
Basically, when you gather books according to their topic, you are doing clustering where books are elements and topic is the parameter according to which you gather the books.
Of course, clustering is not only use for books but also in computer science from reverse engineering to re-engineering and this is why I'm here today

First of all, clustering is a way to gather elements according to parameters.
Basically, when you gather books according to their topic, you are doing clustering where books are elements and topic is the parameter according to which you gather the books.
Of course, clustering is not only use for books but also in computer science from reverse engineering to re-engineering and this is why I'm here today



Second, what we call a conceptual cluster is a cluster resulting of a clustering technique for which the parameter is a concept.
A concept is something that has a meaning for a human being but not for a robot/computer (at least until we give them feelings…)
Techniques have been developed to find concepts in a software. Application are multiple like identifying which part of the code has to be tested for a specific feature, or to trace feature in code. You can find example in the Feature Location Field.

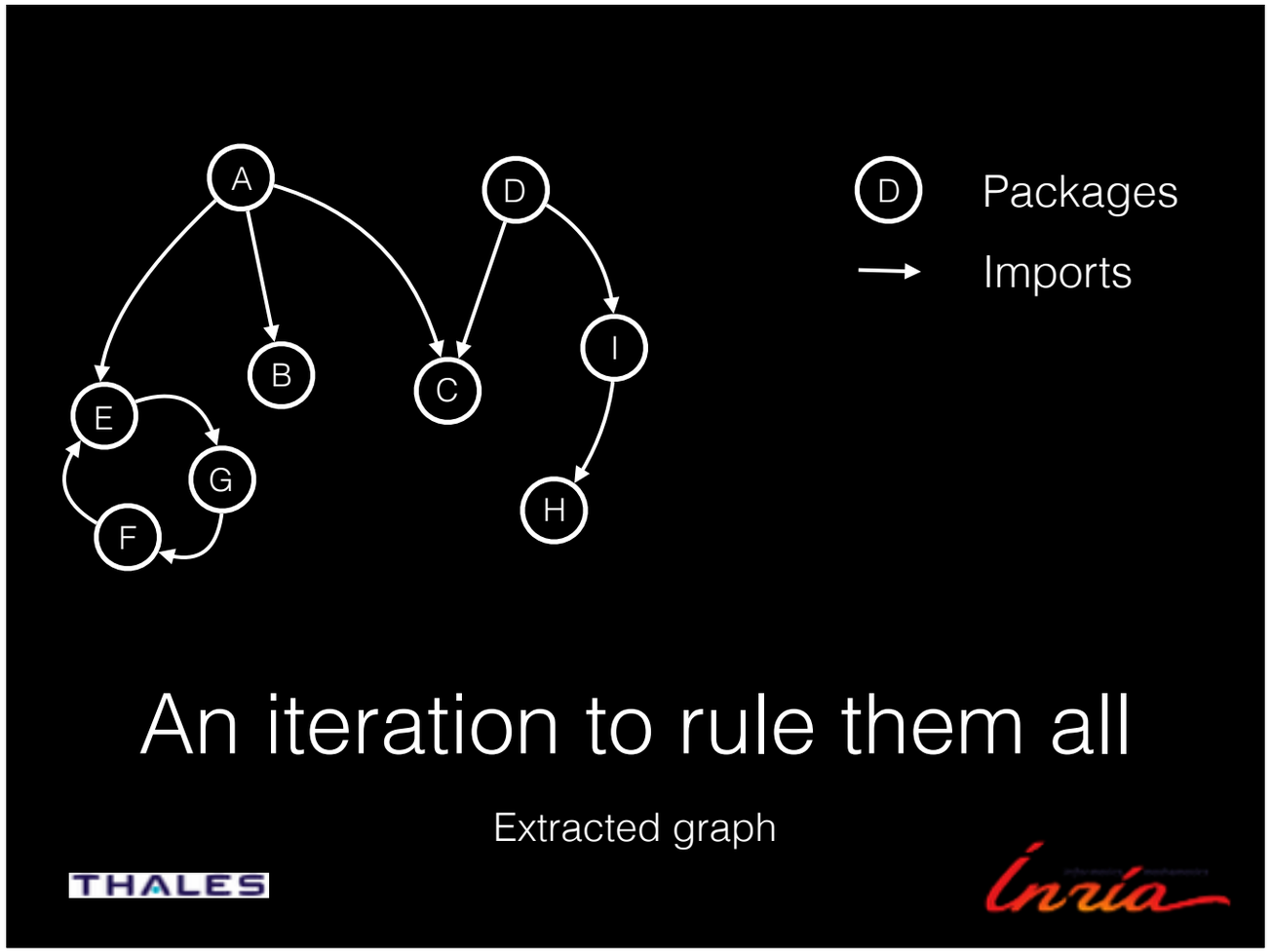


Second, what we call a conceptual cluster is a cluster resulting of a clustering technique for which the parameter is a concept.
A concept is something that has a meaning for a human being but not for a robot/computer (at least until we give them feelings…)
Techniques have been developed to find concepts in a software. Application are multiple like identifying which part of the code has to be tested for a specific feature, or to trace feature in code. You can find example in the Feature Location Field.

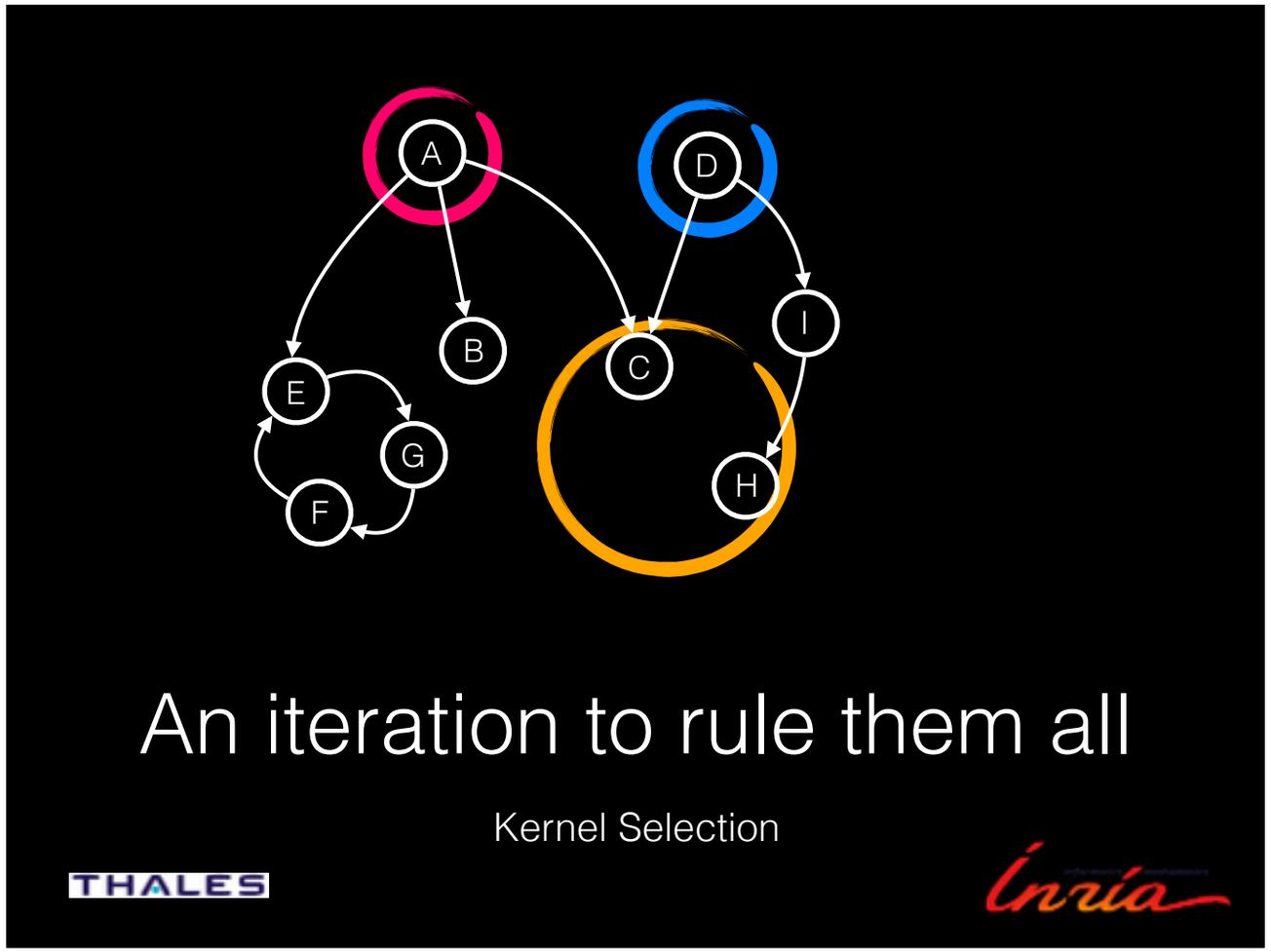
Clustering concepts

THALES

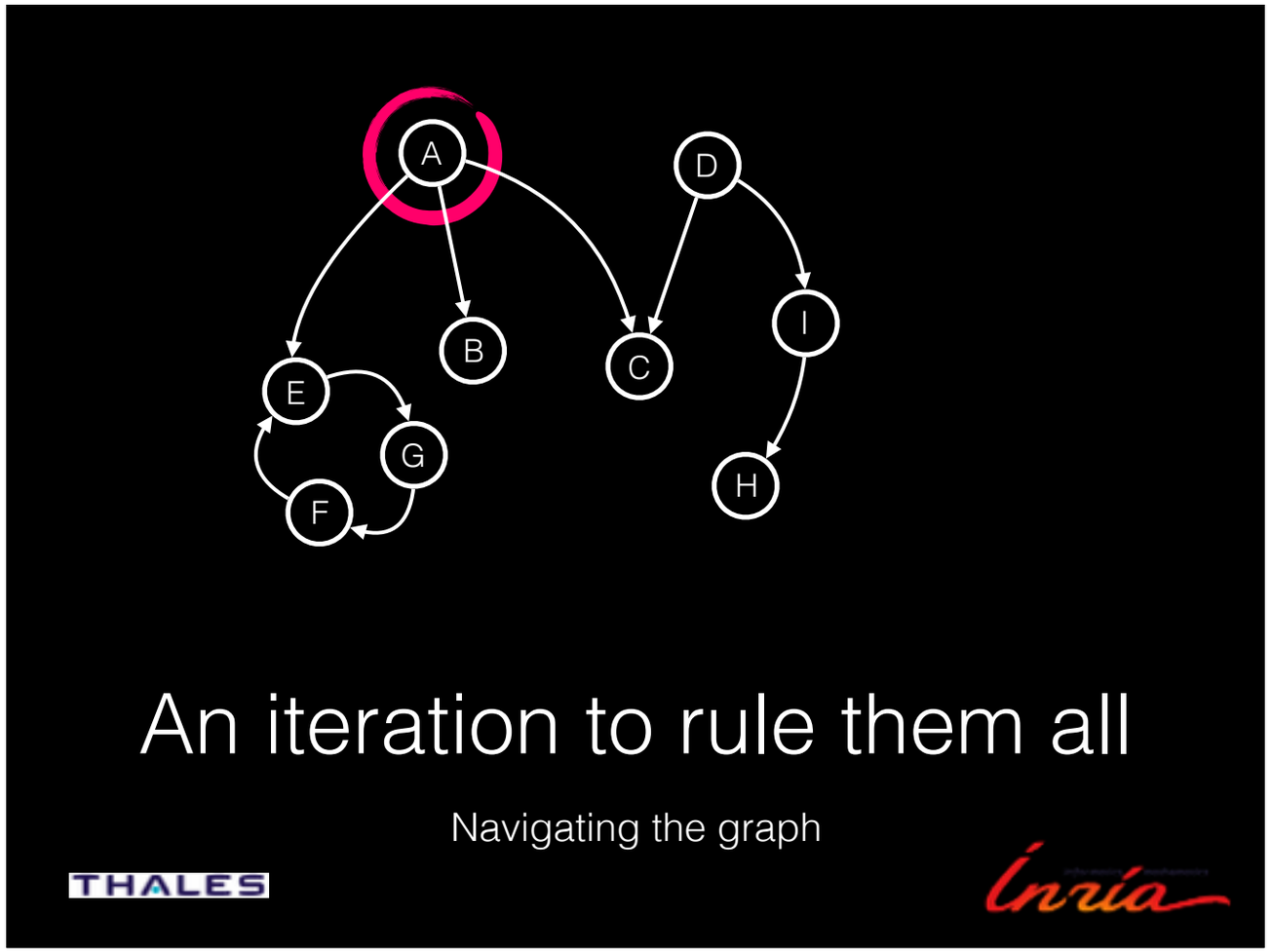
Inria



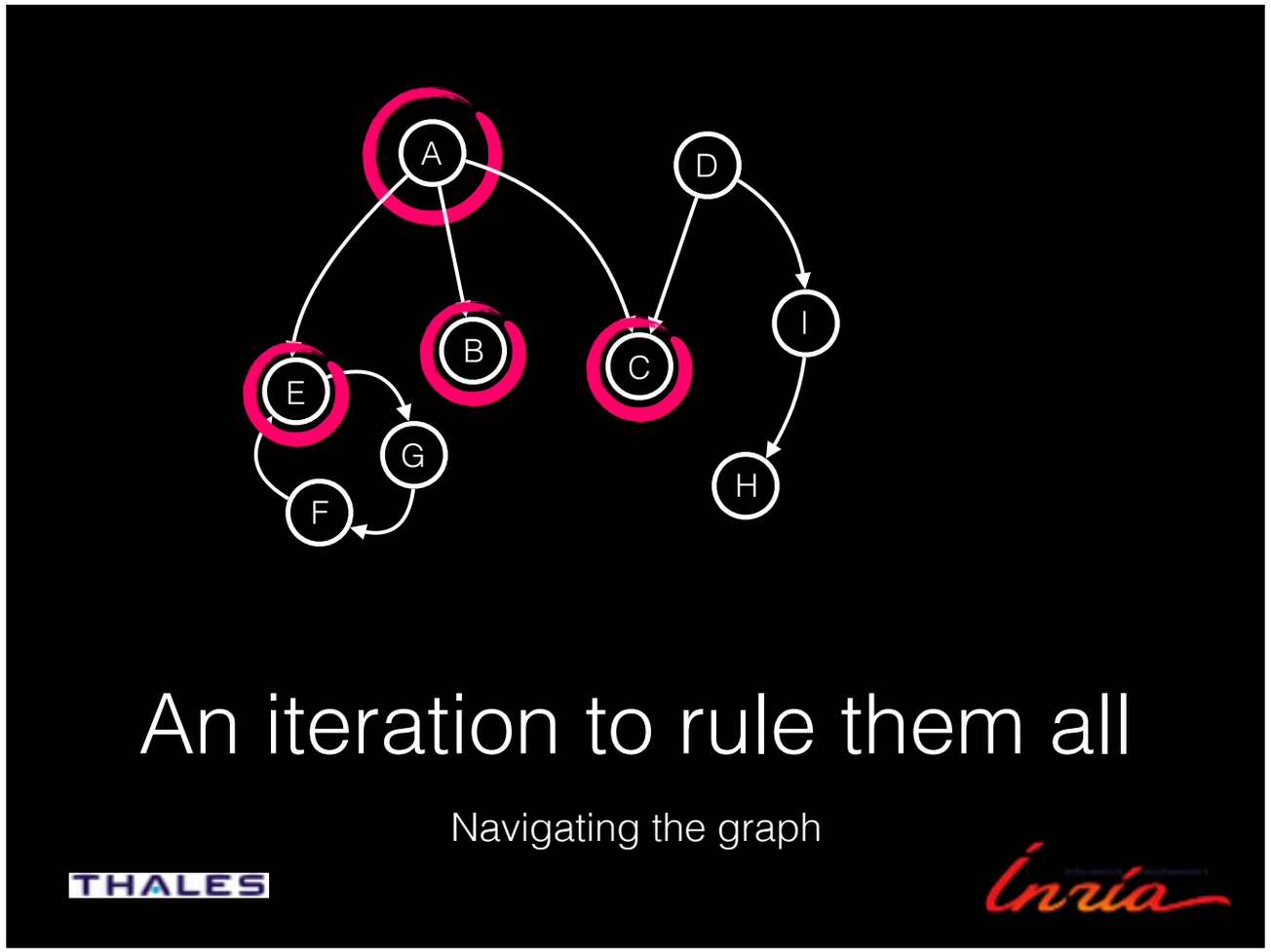
Our method aims to recover such conceptual clusters using an iterative approach.
Each iteration works with an extracted graph. Nodes of the graph, as well as their connections, have to be defined for each iteration. For example, at a given iteration, one can defined nodes as the packages of the software and connections as directed edges that represent the imports between packages.



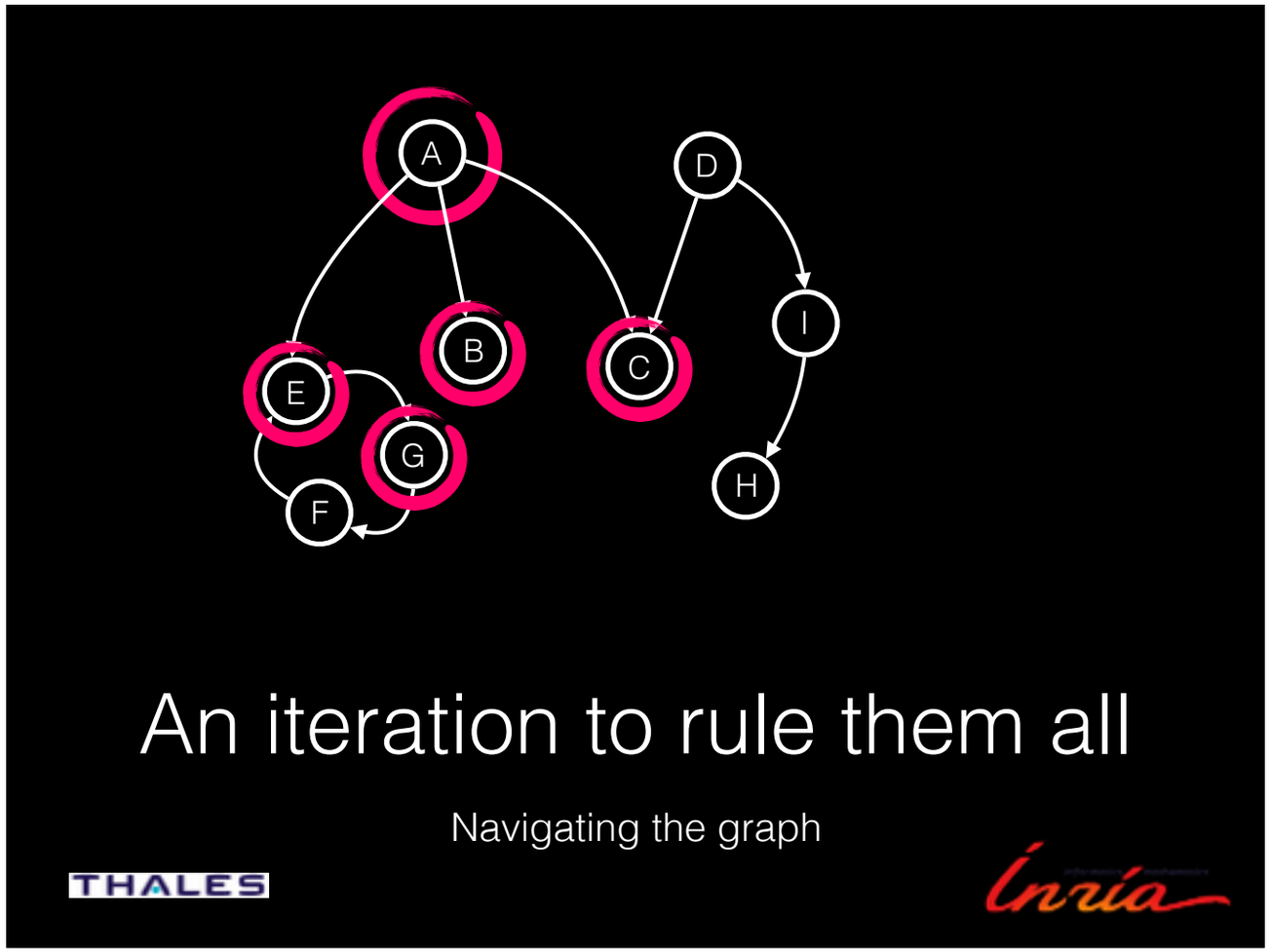
Then, nodes are selected in this graph to be the kernels of the clusters. A cluster kernel can be composed of one or several nodes (it depends on the knowledge of the engineers about the software).
We ask at least one node for a kernel cluster to recover the cluster (sounds logical right?)



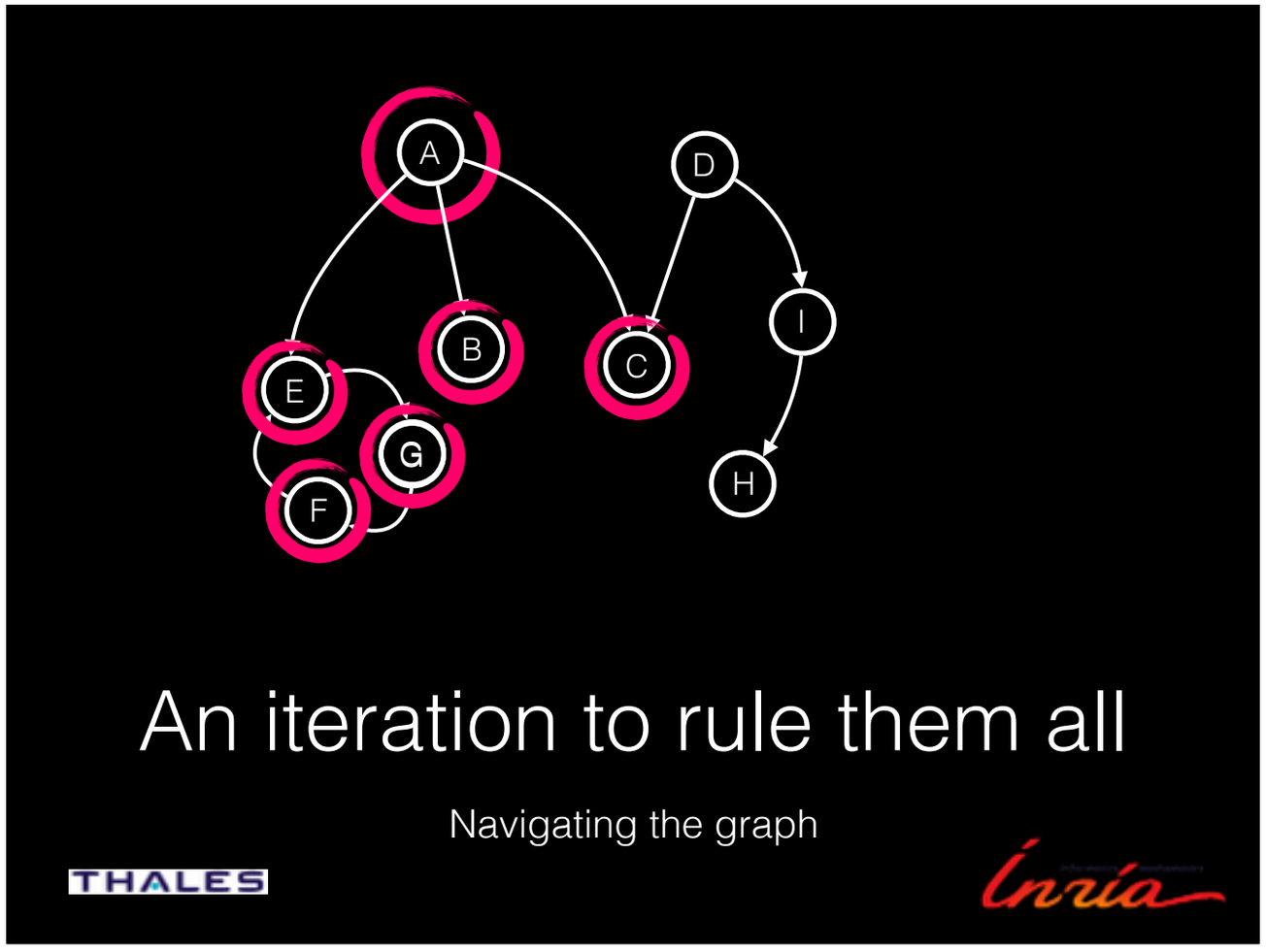
According to a navigation method, that is defined specifically for each iteration, we can navigate the graph (from the kernels).
For each navigated node, a allocation function is applied to check wether the node should be allocate to the clusters of the kernel we started from or not.



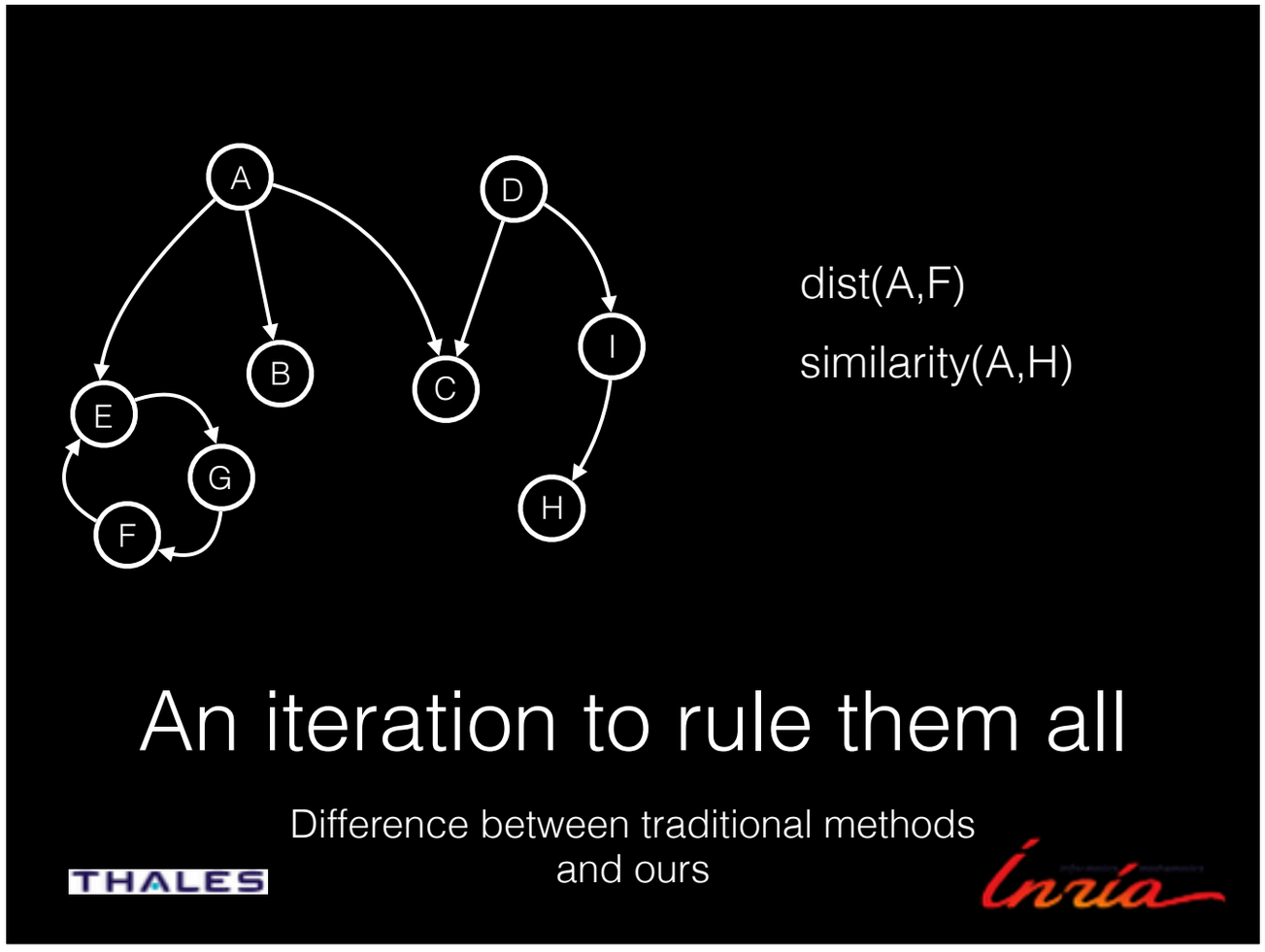
According to a navigation method, that is defined specifically for each iteration, we can navigate the graph (from the kernels).
For each navigated node, a allocation function is applied to check wether the node should be allocate to the clusters of the kernel we started from or not.



According to a navigation method, that is defined specifically for each iteration, we can navigate the graph (from the kernels).
For each navigated node, a allocation function is applied to check wether the node should be allocate to the clusters of the kernel we started from or not.

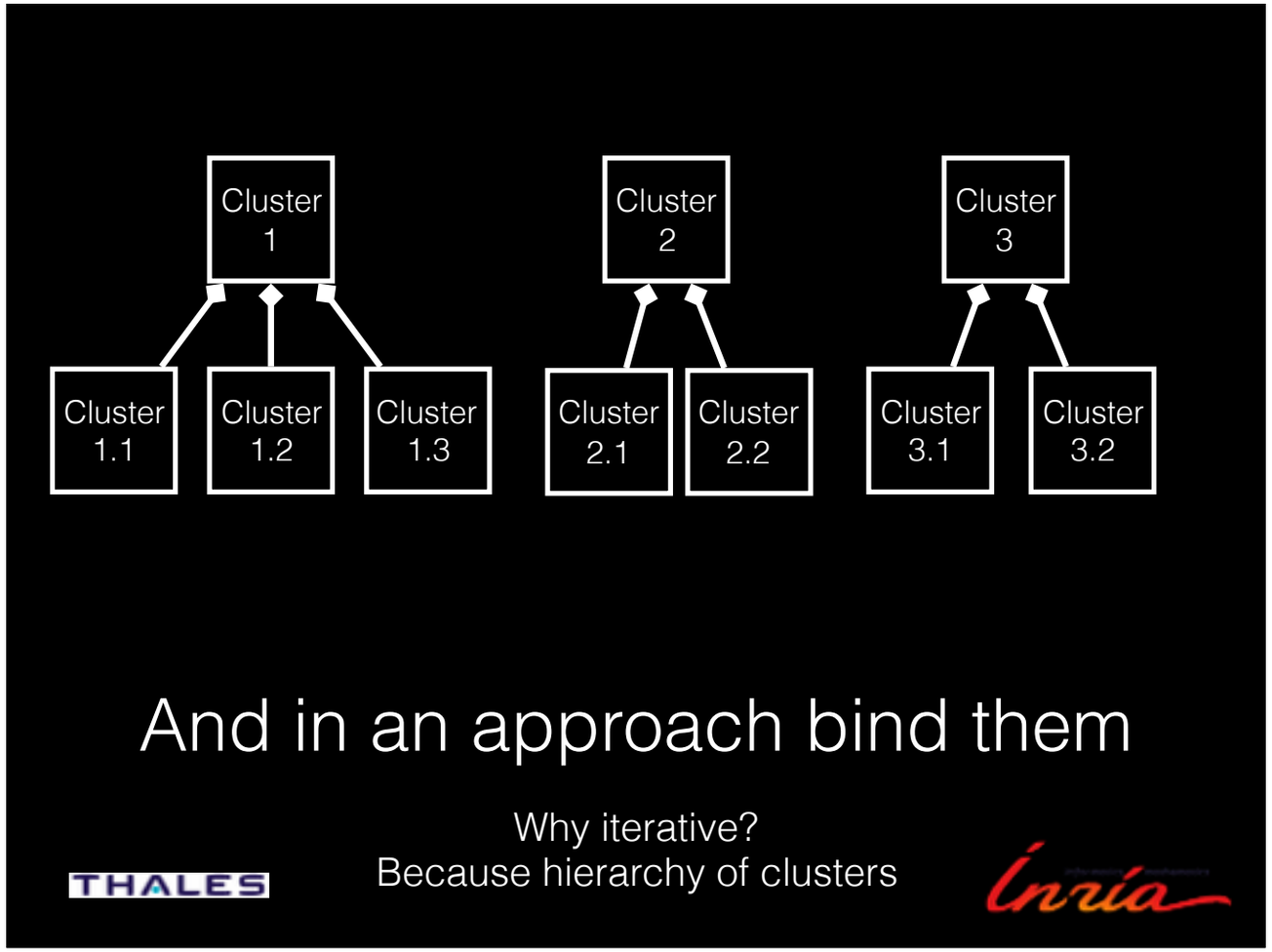


According to a navigation method, that is defined specifically for each iteration, we can navigate the graph (from the kernels).
For each navigated node, a allocation function is applied to check wether the node should be allocate to the clusters of the kernel we started from or not.

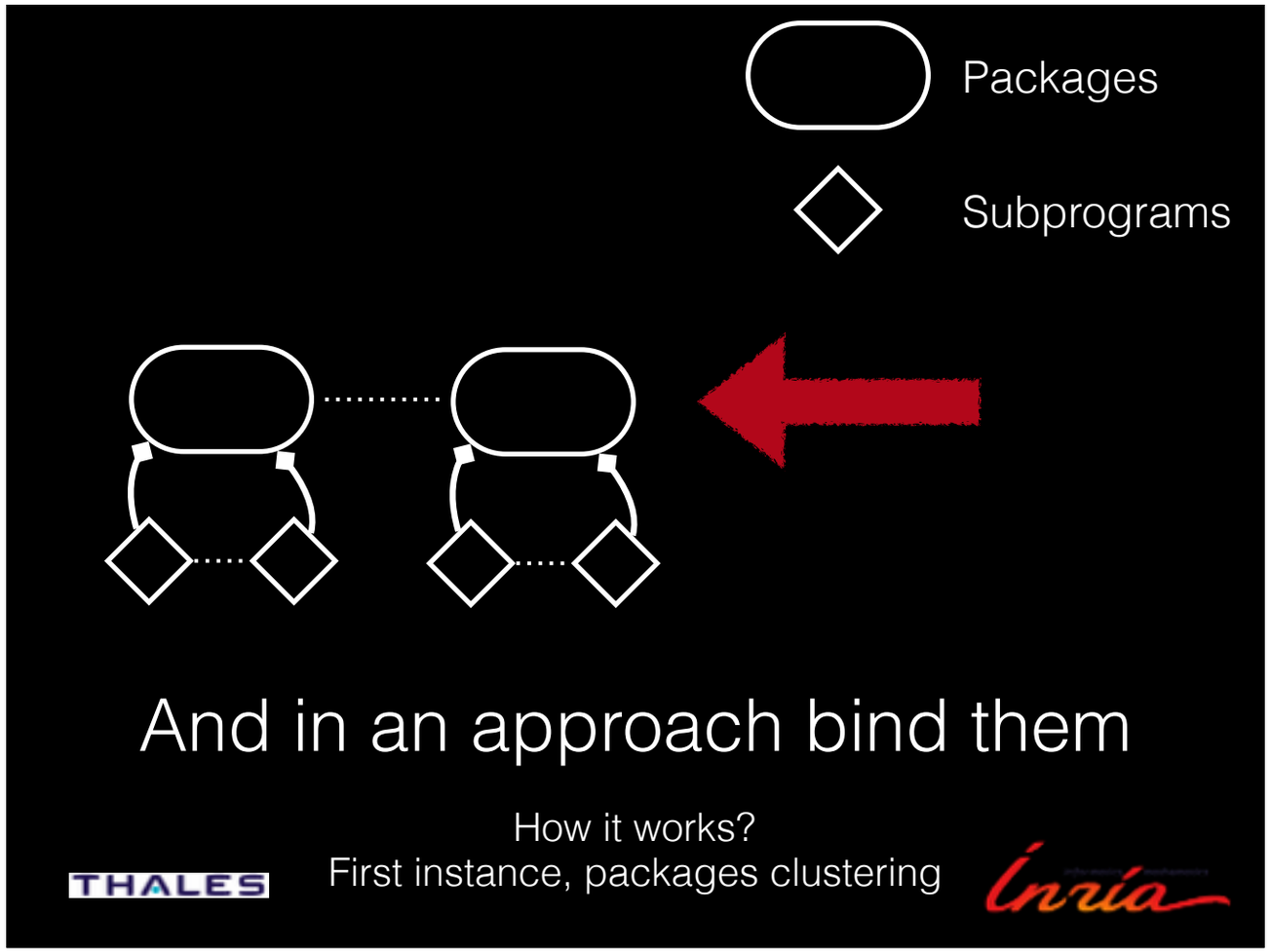


The difference between traditional clustering and our method is that in traditional method, a similarity metric (or distance) is computed from the kernels to all the other nodes of the graph.
Then nodes are allocated to the clusters they are the most similar with (or the cluster they are the closest, for a distance).
What is the similarity metric is generally up to the people who develop the clustering techniques (for example, distance in a call graph from one node to another)

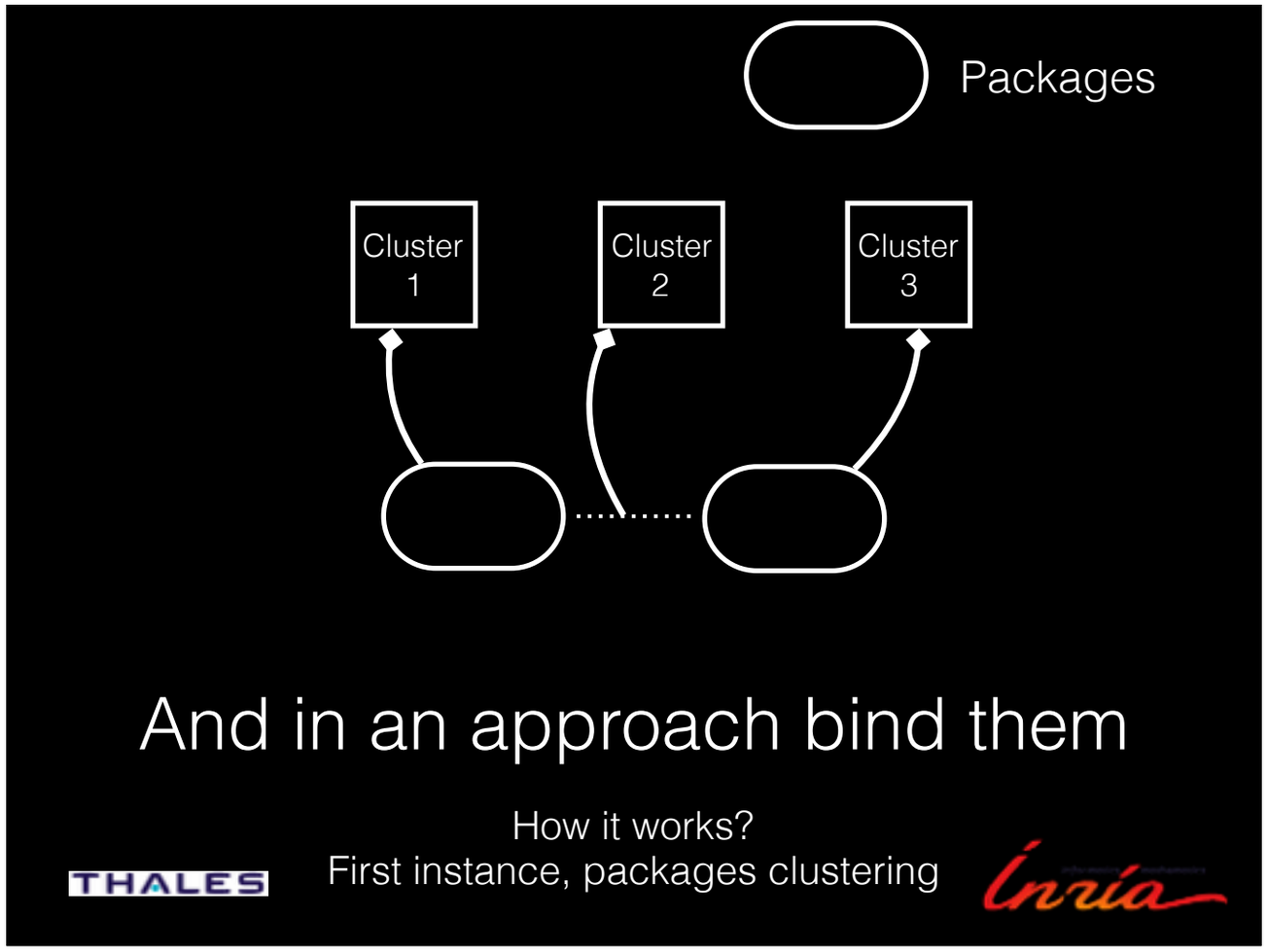
Now that I have explained the generic iteration let’s get deeper on how we use it in our iterative approach.
First, we choose an iterative approach because we consider that we have to cluster the software according to a hierarchy of clusters.
The idea is to start at a high level of software elements (packages), and to apply an instance of our generic iteration, to get a draft of the clusters.
Then we refine each clusters with lower software elements (subprograms) by applying another instance of the generic iteration



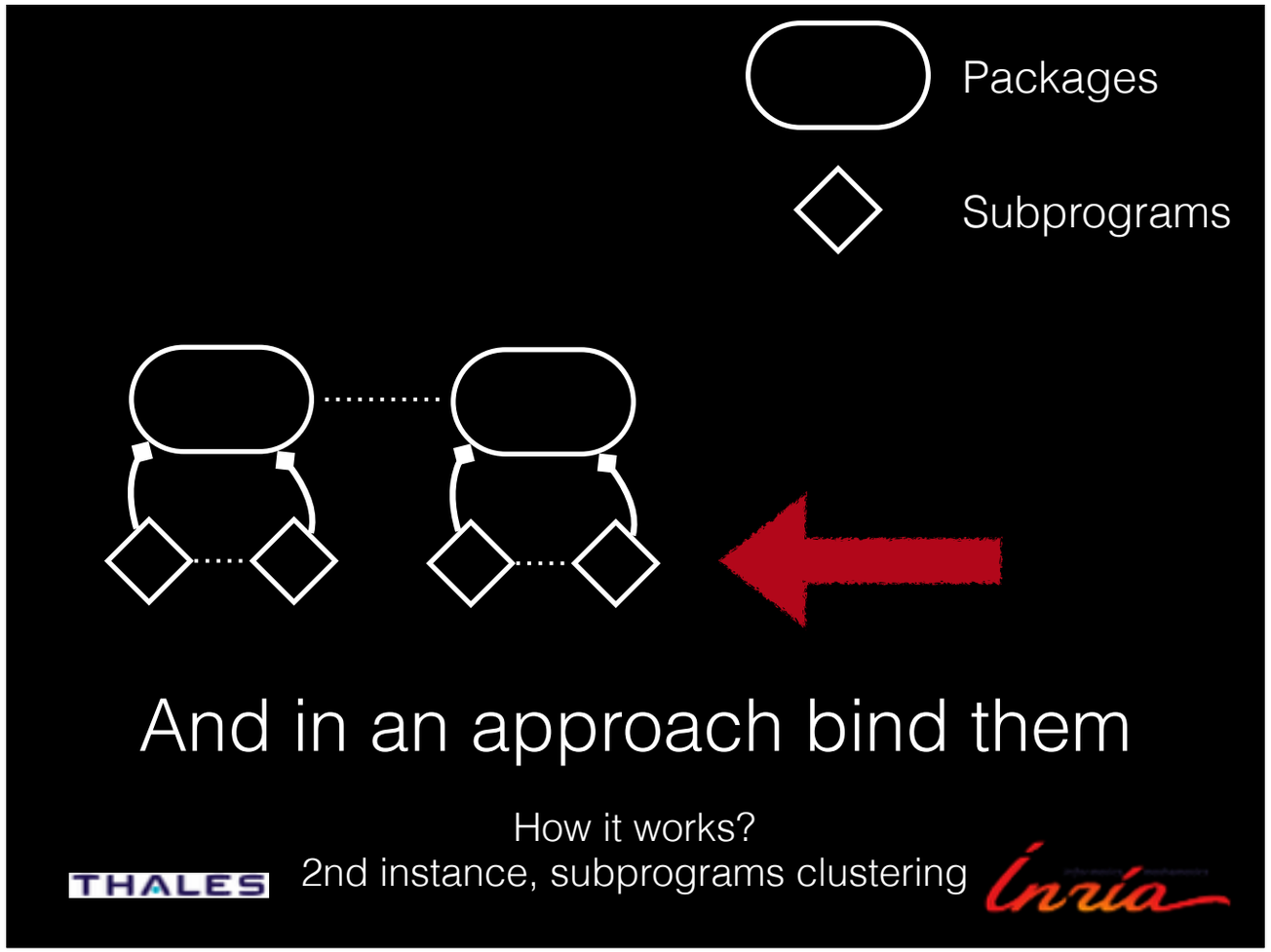
Now that I have explained the generic iteration let’s get deeper on how we use it in our iterative approach.
First, we choose an iterative approach because we consider that we have to cluster the software according to a hierarchy of clusters.
The idea is to start at a high level of software elements (packages), and to apply an instance of our generic iteration, to get a draft of the clusters.
Then we refine each clusters with lower software elements (subprograms) by applying another instance of the generic iteration



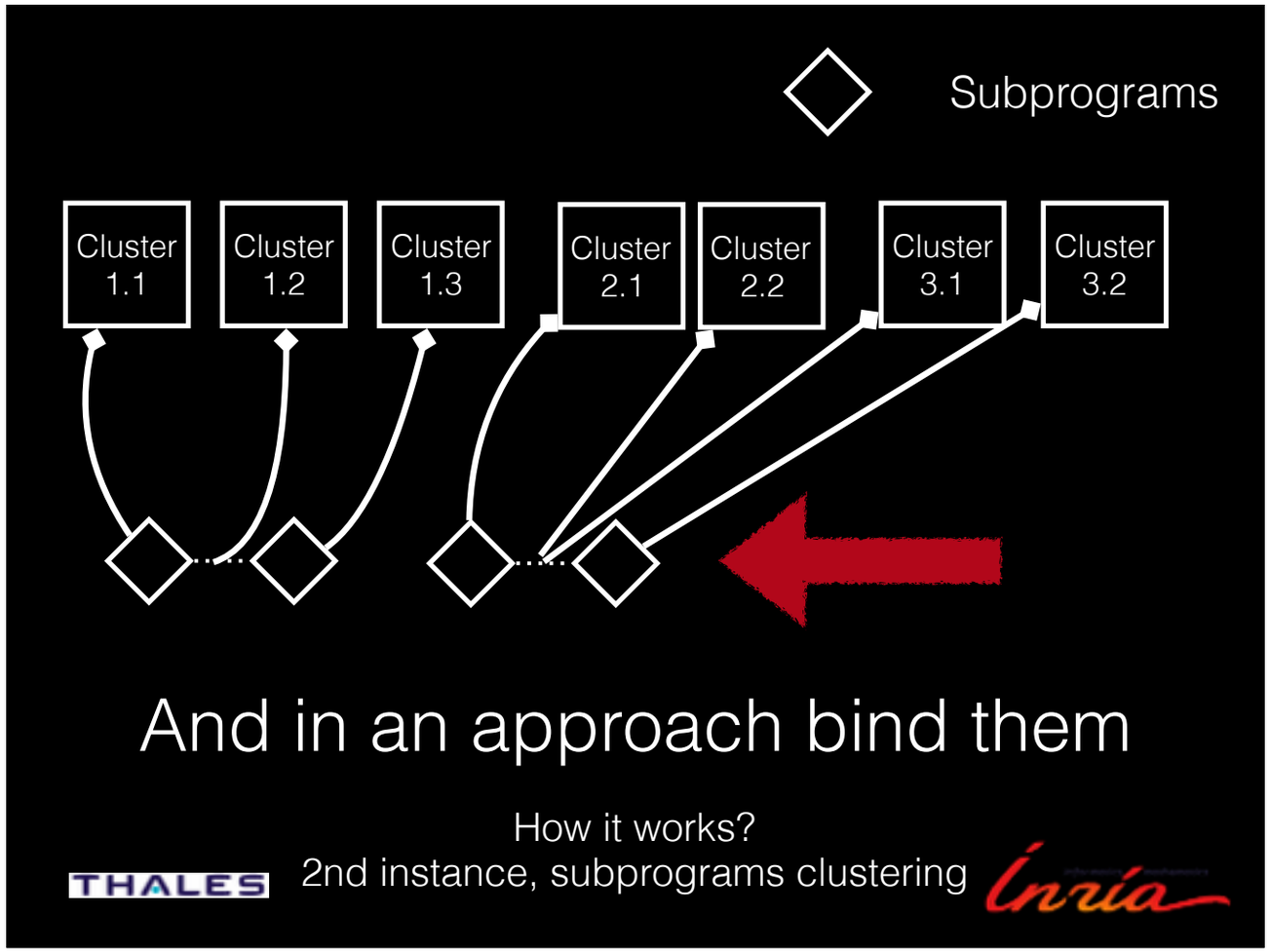
Now that I have explained the generic iteration let’s get deeper on how we use it in our iterative approach.
First, we choose an iterative approach because we consider that we have to cluster the software according to a hierarchy of clusters.
The idea is to start at a high level of software elements (packages), and to apply an instance of our generic iteration, to get a draft of the clusters.
Then we refine each clusters with lower software elements (subprograms) by applying another instance of the generic iteration



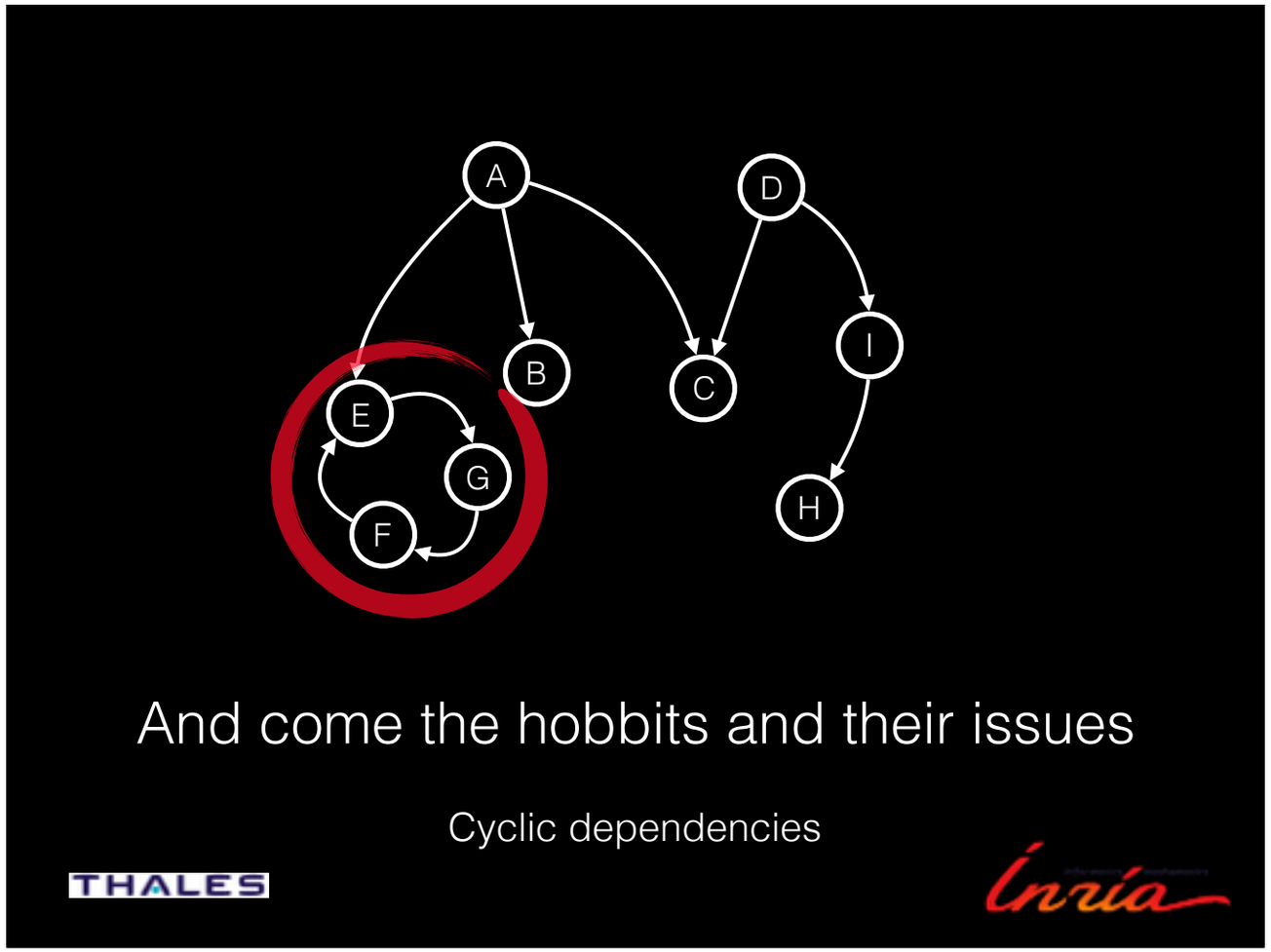
Now that I have explained the generic iteration let’s get deeper on how we use it in our iterative approach.
First, we choose an iterative approach because we consider that we have to cluster the software according to a hierarchy of clusters.
The idea is to start at a high level of software elements (packages), and to apply an instance of our generic iteration, to get a draft of the clusters.
Then we refine each clusters with lower software elements (subprograms) by applying another instance of the generic iteration



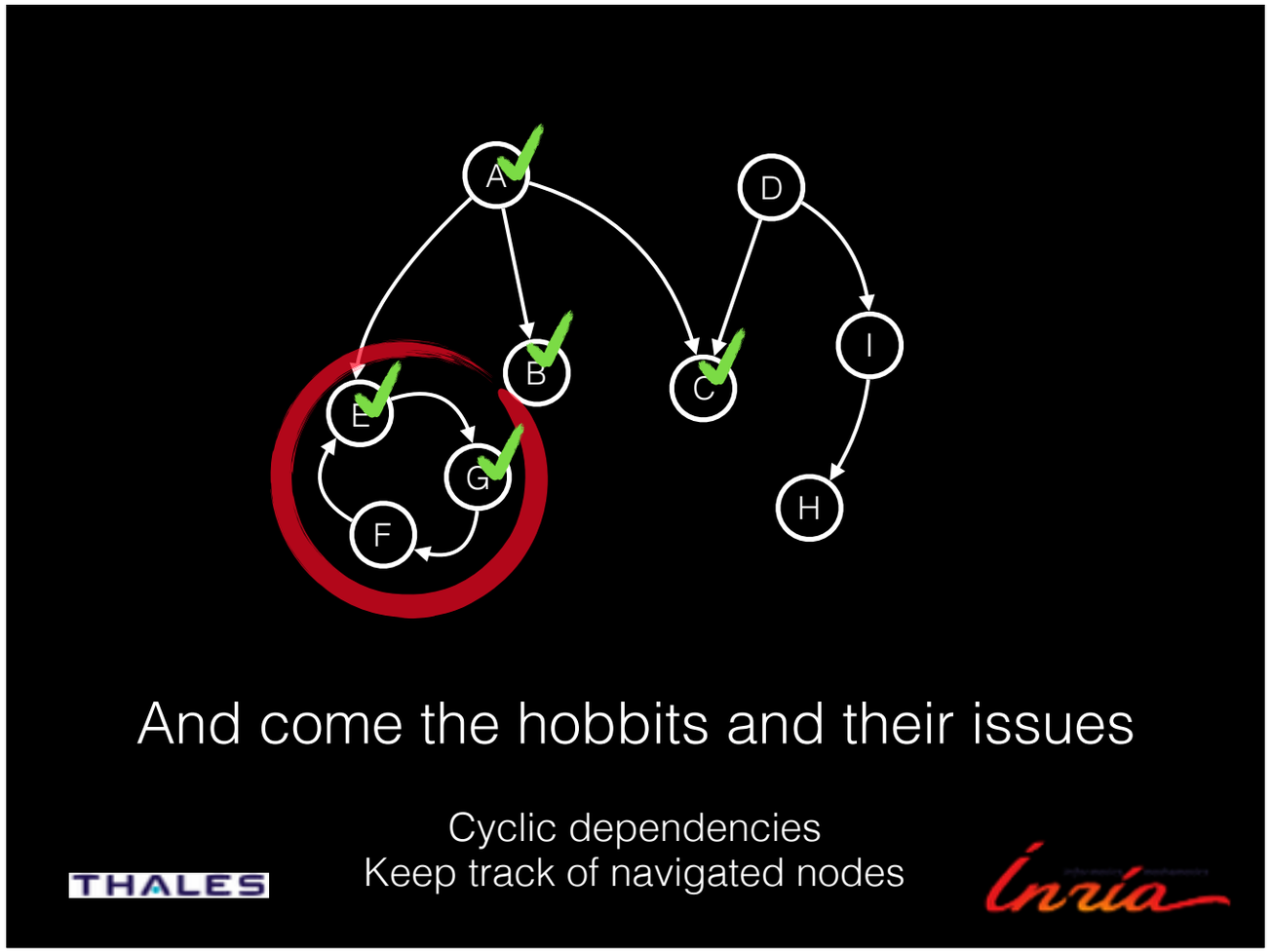
Now that I have explained the generic iteration let’s get deeper on how we use it in our iterative approach.
First, we choose an iterative approach because we consider that we have to cluster the software according to a hierarchy of clusters.
The idea is to start at a high level of software elements (packages), and to apply an instance of our generic iteration, to get a draft of the clusters.
Then we refine each clusters with lower software elements (subprograms) by applying another instance of the generic iteration



Now that I have explained the generic iteration let’s get deeper on how we use it in our iterative approach.
First, we choose an iterative approach because we consider that we have to cluster the software according to a hierarchy of clusters.
The idea is to start at a high level of software elements (packages), and to apply an instance of our generic iteration, to get a draft of the clusters.
Then we refine each clusters with lower software elements (subprograms) by applying another instance of the generic iteration



However, navigating a graph brings a specific issue.
You might think (and you will be right) that cyclic dependencies can occur in the extracted graph and since we navigate the graph such cycle can make the process enters an infinite loop.
This aspect is considered also in our approach by navigating nodes only once.
Indeed, if a node has already been navigated that means all its outgoing nodes have also been navigated.



However, navigating a graph brings a specific issue.
You might think (and you will be right) that cyclic dependencies can occur in the extracted graph and since we navigate the graph such cycle can make the process enters an infinite loop.
This aspect is considered also in our approach by navigating nodes only once.
Indeed, if a node has already been navigated that means all its outgoing nodes have also been navigated.

A second issue that occurs is due to the fact that in our solution the allocation function allows to allocate an element in several clusters.
For example A imports C / D imports C .
A and D are respectively kernels of 2 different clusters.
According to our allocation function, C will be in two clusters, the one of A and the one of D
We have decided to treat such cases (that we can call CONFLICT) differently for each iteration.
It appears therefore that another definition is required, on a function to resolve the conflicts.
For the first instance of our generic iteration (where packages are clustered), the function to resolve conflicts is to put every packages in conflicts in a Library components.
(One third of the packages are then allocated to this Library)
In the second instance (where subprograms are clustered), we decided to not solve conflicts.
Subprograms in conflicts mean that they have to be split (for example using Extract Method or program slicing to identify sets of consistent instructions)

1st Instance: packages clustering

Precision: 98%

Recall: 88%

2nd Instance: subprograms clustering

Precision: ~60%

Recall: ~50%%

Talk about numbers

THALES

Inria

What next?

THALES

Inria