Software assessment

Making effective decisions
about the inside of
your software systems

feenk

The value of your software system today is given by its external functionality. The value of your software system tomorrow is given by how well you'll be able to adapt it. This depends on your ability to understand the system's internals enough to guide its evolution.

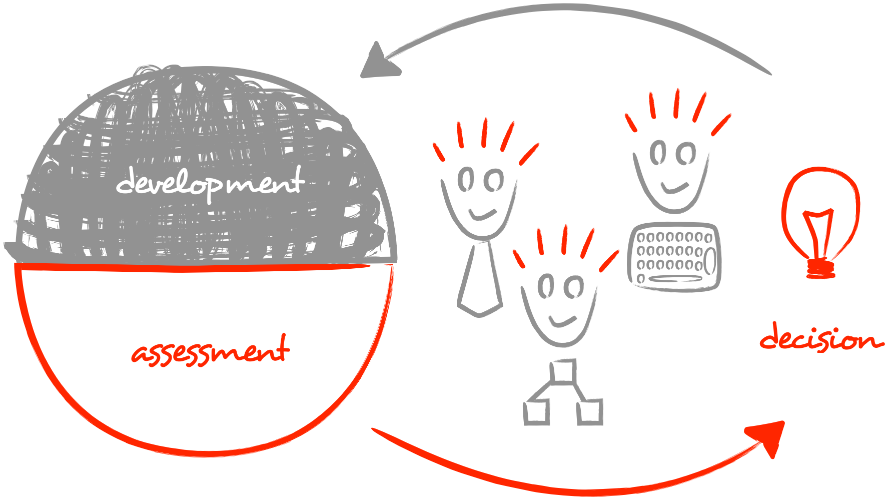
That is software assessment.

Software assessment is pervasive and expensive

When we think of software development, we often think of the active part of creating the system. Yet, the largest cost is spent on assessing the current state of the system. Developers alone spend 50% or more of their time reading code to know what to do next. These are only the direct costs of assessment. The indirect costs can be seen in the consequences of the made decisions.

Software assessment is the single most expensive activity in software development. Yet, currently, it is not addressed explicitly and thus, it never gets optimized. Reading is the most manual possible way to extract information out of data. Given its costs and impact, this has to change.

Assessment must be approached explicitly.



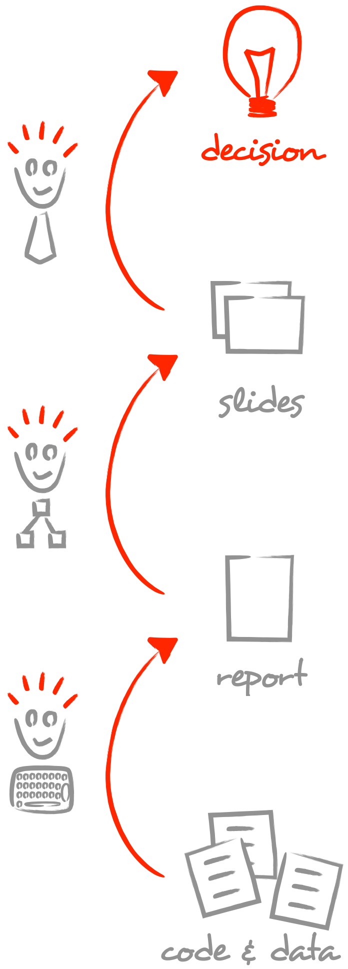
Is software assessment important for a manager?

The internals of systems comprise technical issues. So, shouldn’t assessment be the responsibility of technical people? Why should a manager care?

Two reasons. First, it’s the largest cost. Second, all decisions, both the technical and the business ones, must be based on accurate information.

Put it in perspective: Your system is much larger than humans can read in a reasonable amount of time. A report about your system that is built manually will be at least inaccurate, but most likely wrong.

All decisions about your system must relate to the reality of that system. Everyone must care about how reliable and representative the information is. That’s not a technical issue. It’s a strategic one.

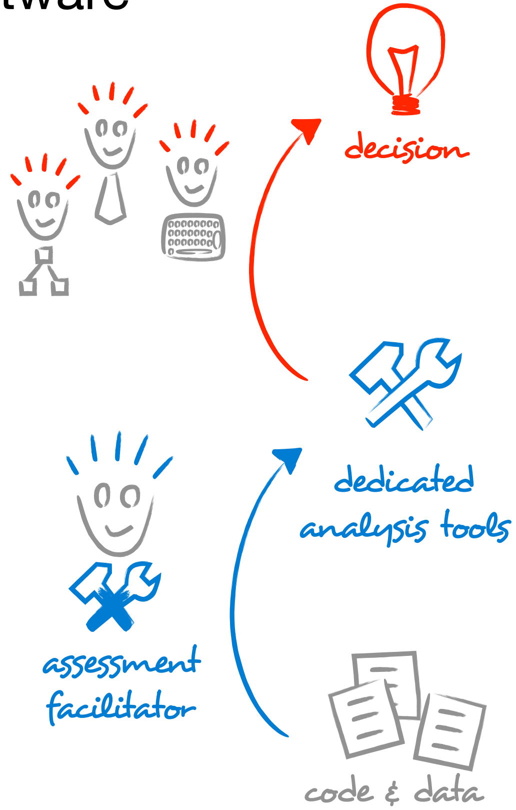


Software assessment is a strategic skill. It's like data science, but for software

For software systems to remain valuable, they have to be adapted to changes in the environment. The evolution challenge is posed by its internal structure. As the dependency to software increases and the need to change becomes ever more critical, it is no longer enough to treat software as a black box: the ability to reason and decide about its internal structure is critical, and software assessment becomes a strategic skill.

This is relevant both when working with in-house systems, and when working with external providers. The assessment skill offers an infra-red like ability to identify and react to problems before they are escalated.

It's like data science for software. Internalize the skill to augment your ability to make decisions.



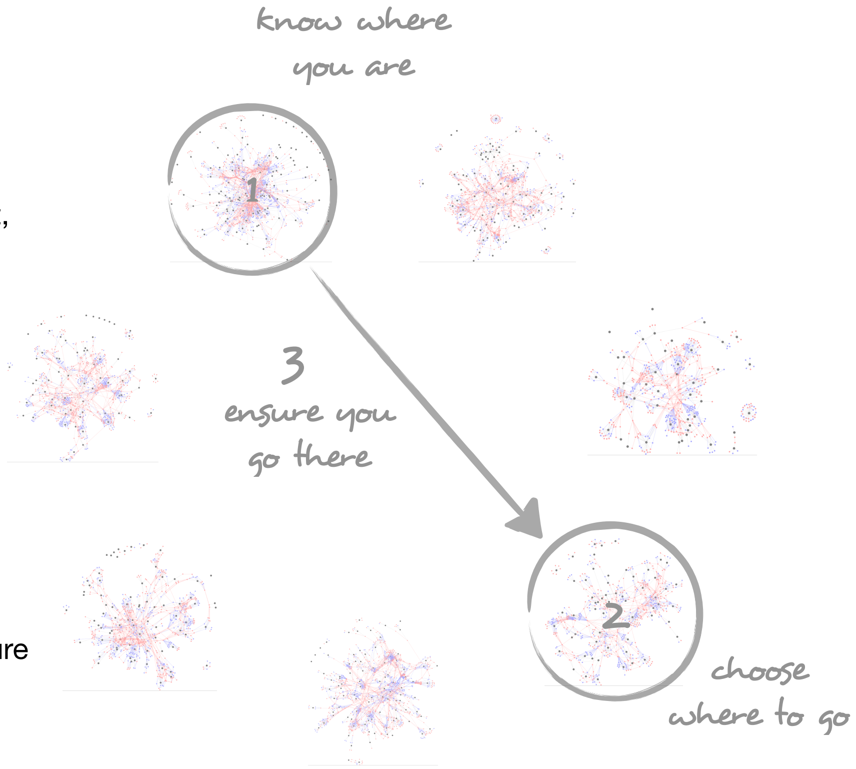
Architecture is a business asset. Steering it relies on assessment

The ability to change the system tomorrow depends on its internal architecture. As the ability to change is of critical importance, it follows that the architecture becomes a business asset. And, like any business asset, you should treat it as an investment, too.

The only architecture that matters is the one in the code. A key challenge is to steer the architecture while the code changes continuously. This implies at least three things:

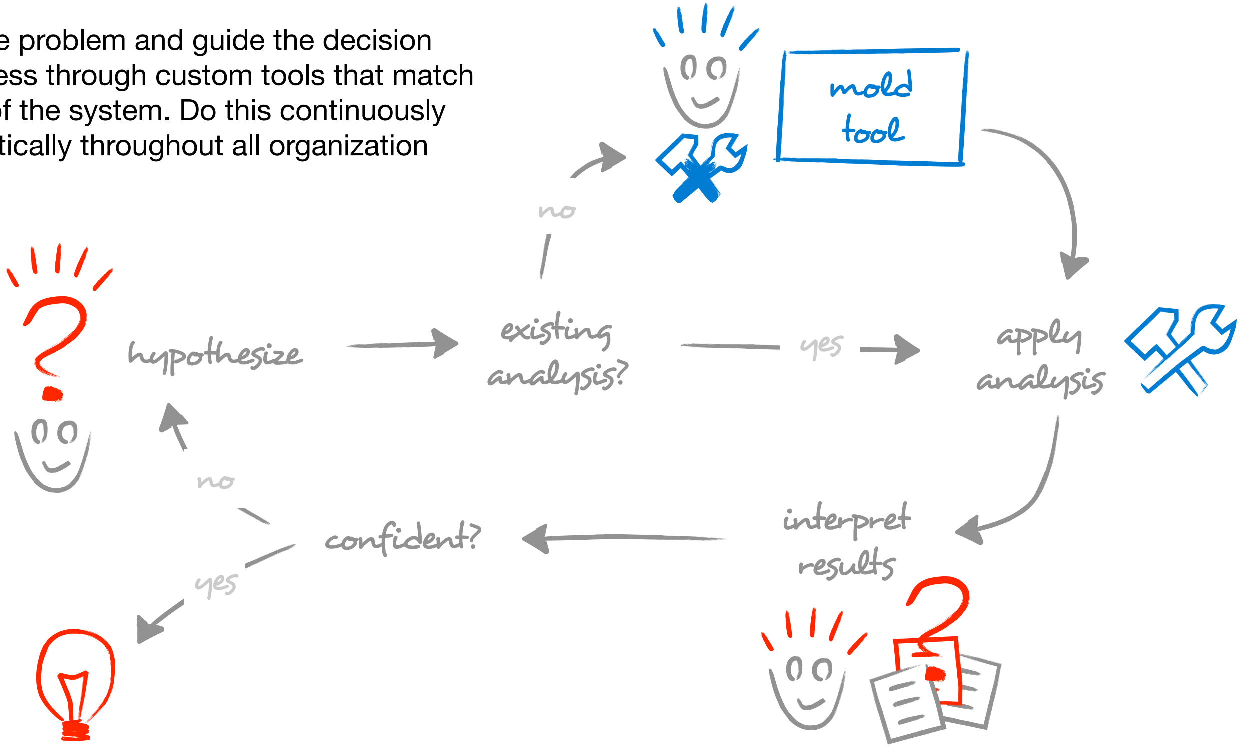
1. know where you are,
2. choose where you want to go,
3. ensure you go there.

2 is a design problem that is often covered well. 1 and 3 are assessment activities. Ensure you cover them well, too.



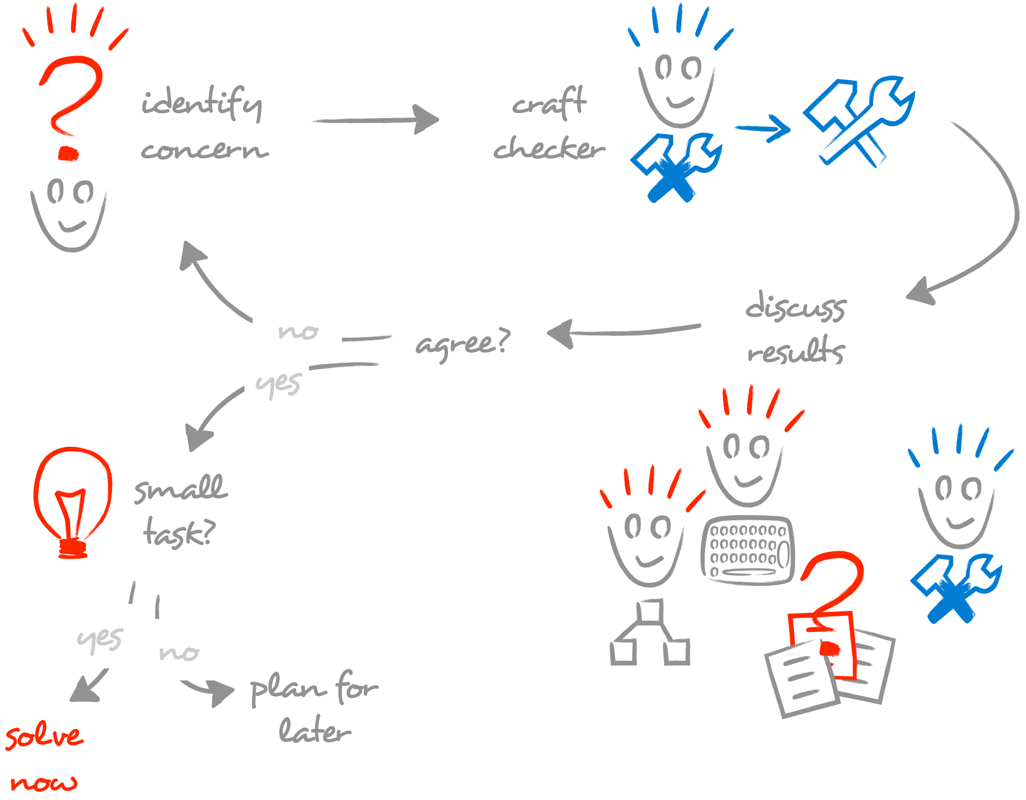
Assessment is not a recipe.
It is a systematic discipline

Start from the problem and guide the decision making process through custom tools that match the context of the system. Do this continuously and systematically throughout all organization layers.



Steering agile architecture is a continuous concern.

The daily assessment process complements the typical development activities. Anyone can raise a concern. A facilitator crafts the tool and the results are discussed in a common space or standup. In this standup only people that have data-based results can speak. And at the end, the goal is to distill concrete actions that are acted upon. This seemingly simple process, enables the team to continuously identify, check and fix relevant technical concerns both about details and about broad architectural issues.



Our assessment services

Strategic assessments

Should you build a new system? Should you invest in the current one? Is the development on track? These are all examples of strategic assessments. We help you evaluate them.

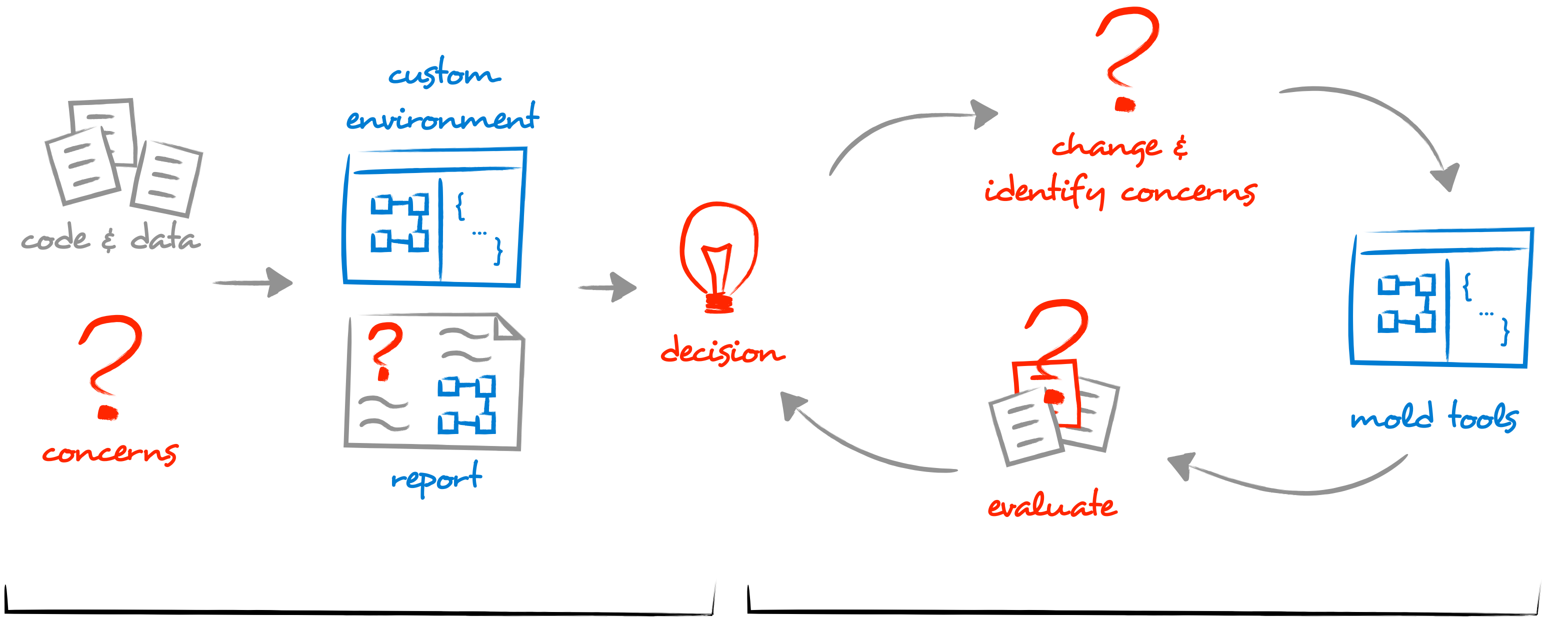
Guiding migrations

Migrations are challenging because of at least two reasons: unclear business needs that are buried in software, and intertwined structural dependencies. In both cases, understanding the technical details is of critical importance. We help the team guide the migration through custom analyses.

Steering agile architecture

Architecture is a business asset. Architecture does not just happen. It requires continuous investment. We coach the team to internalize the skill of software assessment and to steer architecture continuously.

feenk



strategic assessment

1-3 months

guiding migrations & steering agile architecture

3-12 months

feenk

We explain complicated problems covering the complete spectrum of technical decision making:

	strategic assessment	steering agile architecture
We identify technical problems by interpreting the business and technical context.	✓	✓
We assess systems and we architect transformations and migrations.	✓	✓
We construct custom tools that technical and non-technical people use to make their decisions.	✓	✓
We document architecture through automatic checks.		✓
We coach teams to incorporate our techniques and tools to guide the evolution of their systems.		✓
We coach businesses to acknowledge architecture as a business asset.		✓

feenk

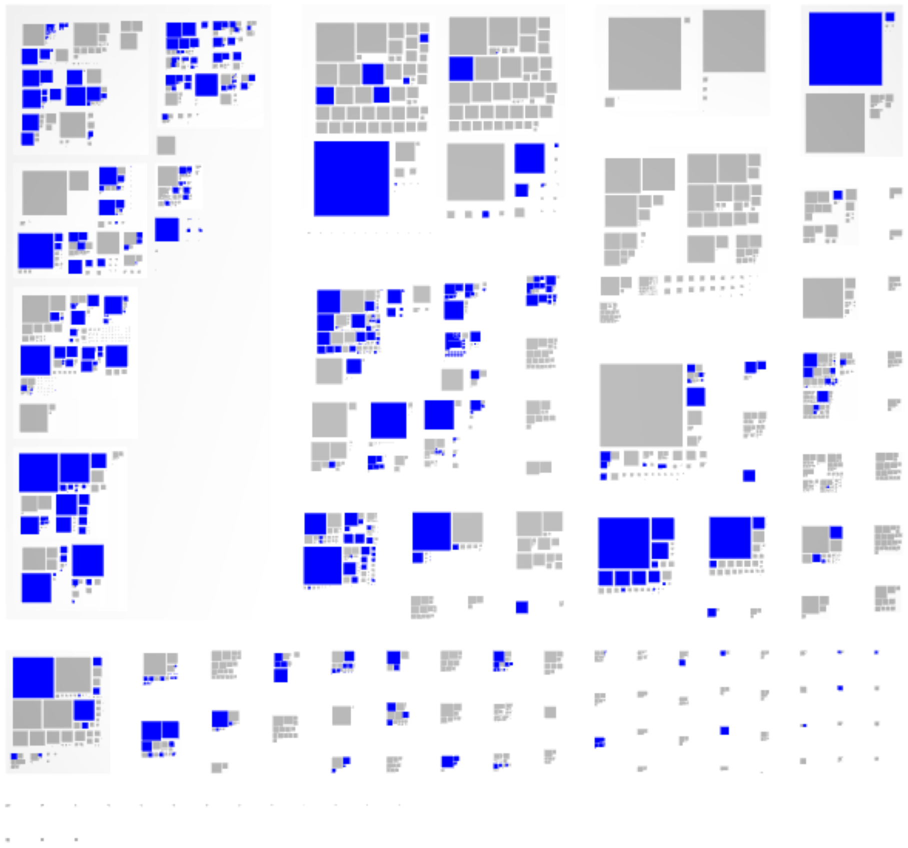
Scenario: Guiding a migration

Migrations are luring. And they are risky.

The are luring because of the promise of the new technology. They are risky because of the messy reality of the existing system. Yet, the success of a migration depends on the ability to assess that reality.

For example, in a migration case, the team manually estimated that a part that was to be migrated was only used in a few places. To validate this assumption, we crested a custom analysis. The visualization to the right reveals (in blue) that, in fact, the part was used throughout the whole system.

Migrations are indeed risky, but the risks can be mitigated when you know them.

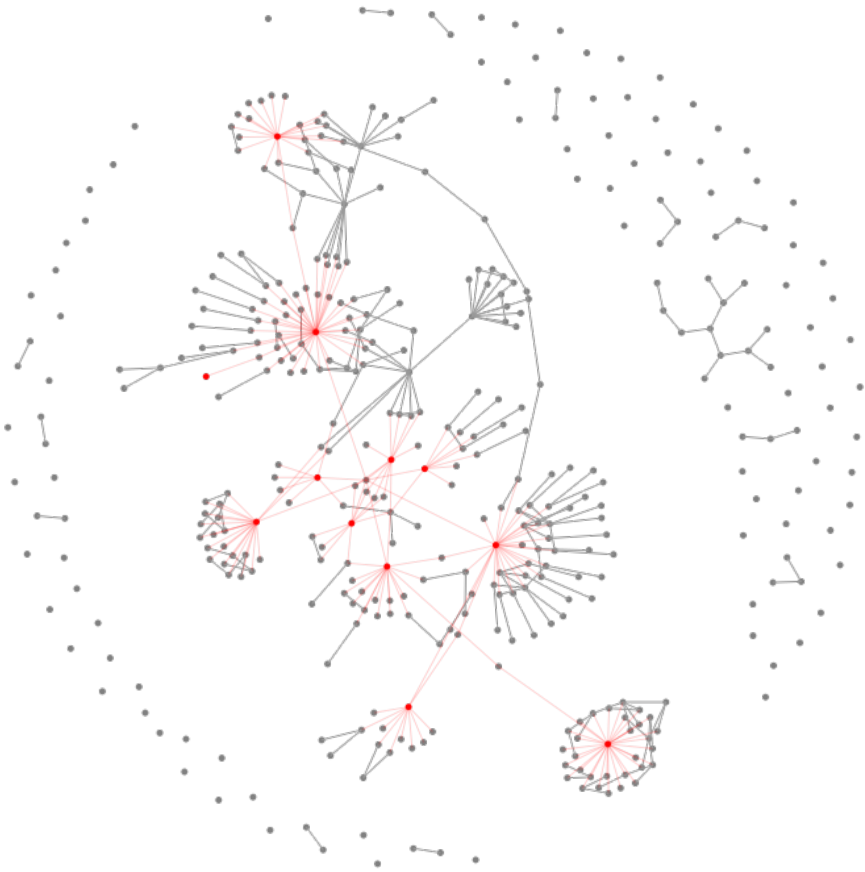


Scenario: Splitting a monolithic application

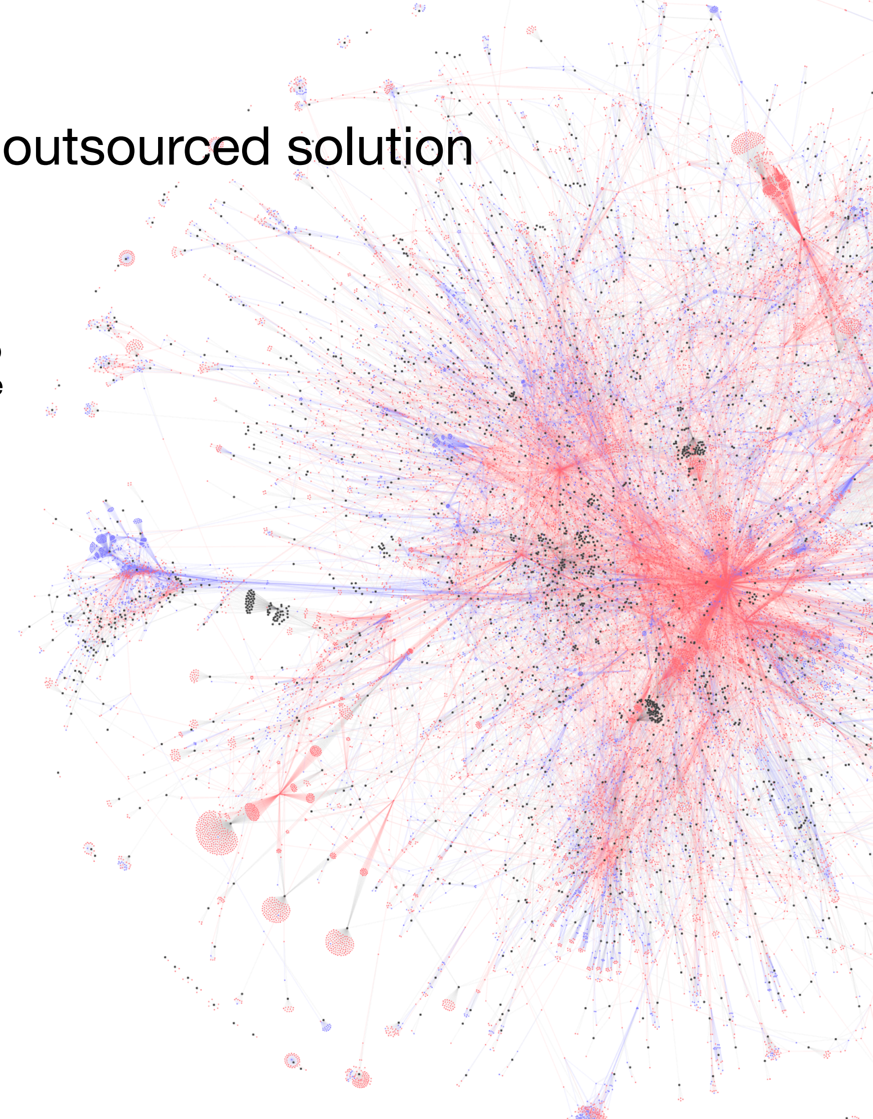
A monolith grows over time. It gets more and more important. It accumulates capabilities until it reaches a point in which, for various reasons, it does not scale anymore. That's when you need to split it. Yet, doing so is hard exactly because it grew organically without clear separation between its inner parts.

The splitting strategy must not only take into account the functional side of the system, but its internal structure, too. To make this happen, you need custom tools that reveal the unwanted dependencies.

For example, to the right we see a custom visualization showing the services of a system, in gray, and how they are used by their clients, in red. The clusters reveal probable splitting options. This finding guided the development.



Scenario: Controlling the outsourced solution



Outsourcing the construction of a system can be cost effective as you pay for what you get. And it can be a strategic tool as you can ramp up the development effort fast. Yet, a software system is almost never a one-off investment, and its value is given by how well you can evolve it over time.

You can outsource the work on the system itself, but you should retain the ability to understand its inner workings. Retaining that ability provides two advantages.

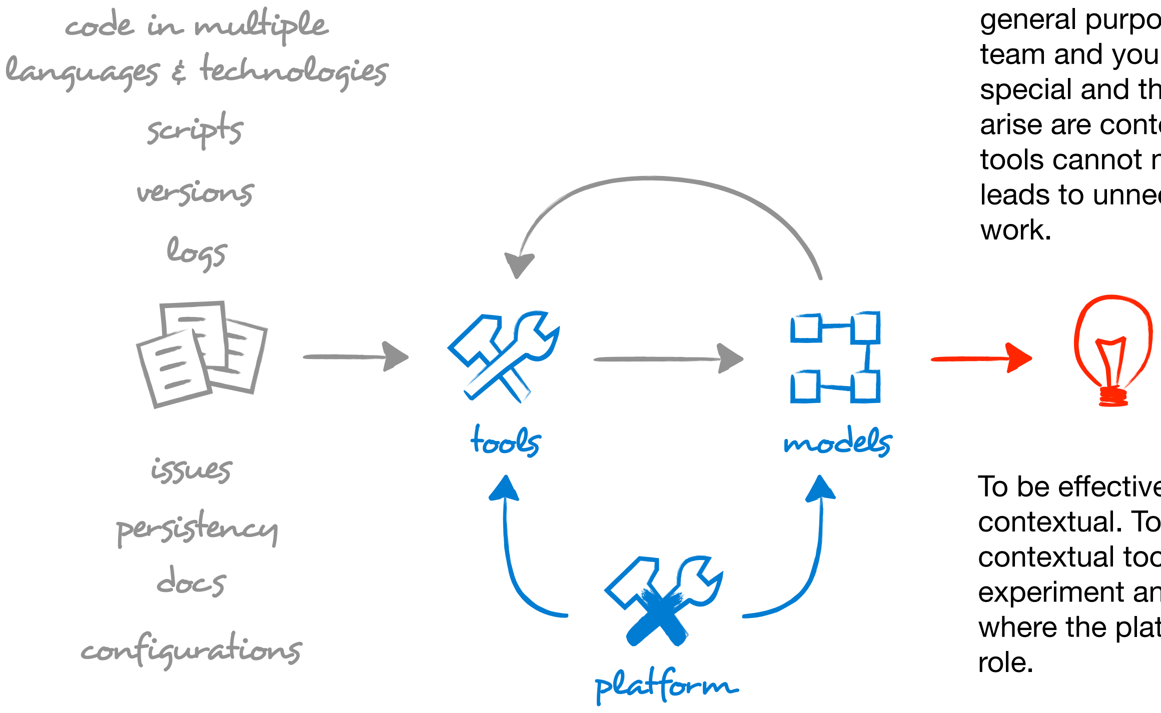
First, outsourcing typically comes with a lock-in effect, and this gives the provider disproportionate negotiation power. Being able to see through the black box levels the playing field.

Second, the same ability is key when the system is handed over either internally or to another provider.

Your system is special. Match it with custom tools

In many industries, it is common to have dedicated people that build special purpose tools to speed up production. They are toolsmiths and they are essential.

In contrast, in software development it is common to use general purpose tools. Yet, your team and your system are special and the problems that arise are contextual. Generic tools cannot match that and this leads to unnecessary manual work.



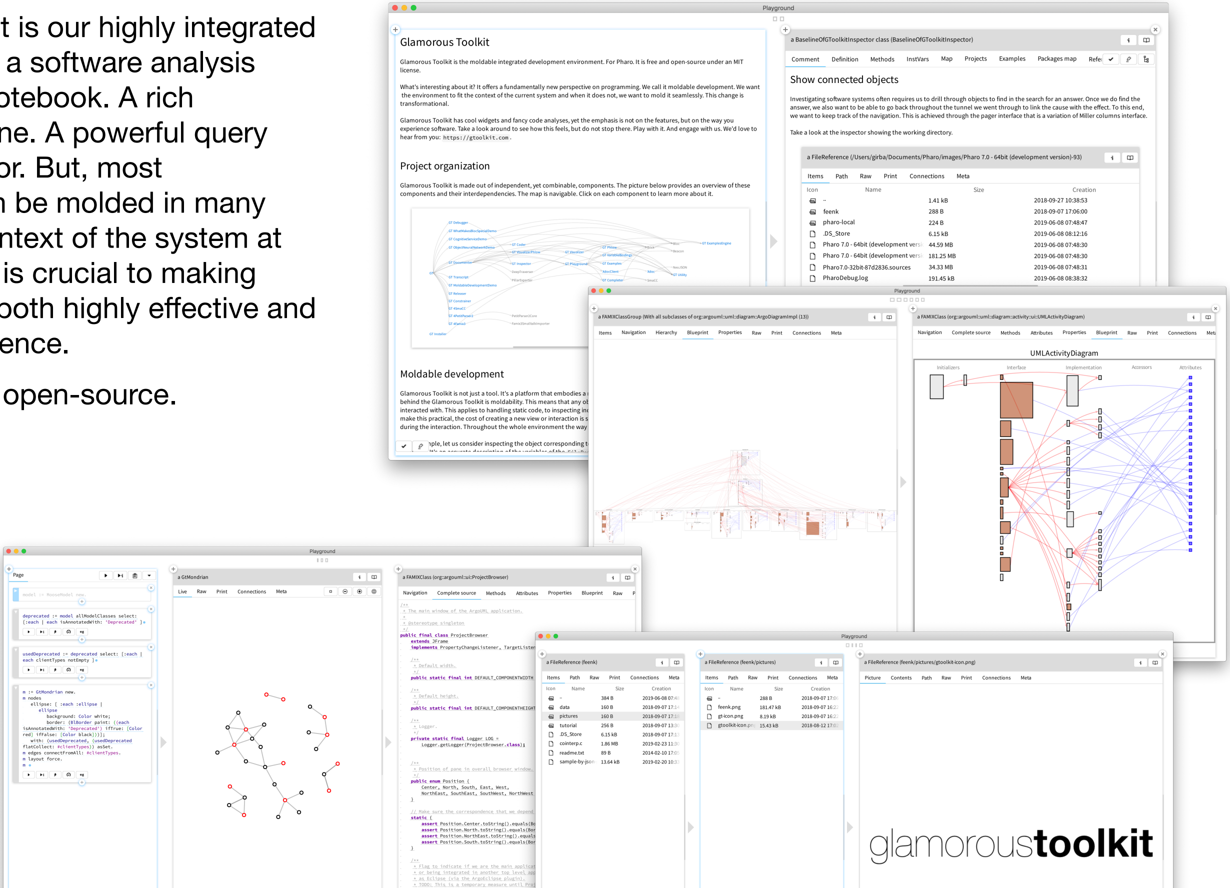
To be effective, tools must be contextual. To find the right contextual tool, you need to experiment and explore. Here is where the platform plays a key role.

Glamorous Toolkit: the moldable environment

Glamorous Toolkit is our highly integrated environment. It is a software analysis platform. A live notebook. A rich visualization engine. A powerful query tool. A fancy editor. But, most importantly, it can be molded in many ways to fit the context of the system at hand. This ability is crucial to making decision making both highly effective and a beautiful experience.

And it is free and open-source.

gtoolkit.com



Glamorous Toolkit is the moldable integrated development environment. For Pharo. It is free and open-source under an MIT license.

What's interesting about it? It offers a fundamentally new perspective on programming. We call it moldable development. We want the environment to fit the context of the current system and when it does not, we want to mold it seamlessly. This change is transformational.

Glamorous Toolkit has cool widgets and fancy code analyses, yet the emphasis is not on the features, but on the way you experience software. Take a look around to see how this feels, but do not stop there. Play with it. And engage with us. We'd love to hear from you: https://gtoolkit.com.

Glamorous Toolkit is made out of independent, yet combinable, components. The picture below provides an overview of these components and their interdependencies. The map is navigable. Click on each component to learn more about it.

Investigating software systems often requires us to drill through objects to find in the search for an answer. Once we do find the answer, we also want to be able to go back throughout the tunnel we went through to link the cause with the effect. To this end, we want to keep track of the navigation. This is achieved through the pager interface that is a variation of Miller columns interface.

Take a look at the inspector showing the working directory.

a FileReference (/Users/girba/Documents/Pharo/images/Pharo 7.0 - 64bit (development version)-93)

Icon	Name	Size	Creation
	..	1.41 kB	2018-09-27 10:38:53
	feenk	288 B	2018-09-07 17:06:00
	pharo-local	224 B	2019-06-08 07:48:47
	.DS_Store	6.15 kB	2019-06-08 08:12:16
	Pharo 7.0 - 64bit (development vers	44.59 MB	2019-06-08 07:48:30
	Pharo 7.0 - 64bit (development vers	181.25 MB	2019-06-08 07:48:30
	Pharo7.0-32bit-87d2836.sources	34.33 MB	2019-06-08 07:48:31
	PharoDebug.log	191.45 kB	2019-06-08 08:38:32

Glamorous Toolkit is not just a tool. It's a platform that embodies a ... behind the Glamorous Toolkit is moldability. This means that any ob... interacted with. This applies to handling static code, to inspecting in... make this practical, the cost of creating a new view or interaction is s... during the interaction. Throughout the whole environment the way

a FAMIXClassGroup (With all subclasses of org::argouml::uml::diagram::ArgoDiagramImpl (13))

a FAMIXClass (org::argouml::uml::diagram::activity::ui::UMLActivityDiagram)

```
model := MooseModel new.

deprecated := model allModelClasses select:
[:each | each isAnnotatedWith: 'Deprecated' ]

usedDeprecated := deprecated select: [:each |
each clientTypes notEmpty ]

m := GtMondrian new.
m nodes
    ellipse: [ :each :ellipse |
      ellipse
        background: Color white;
        border: (BlBorder paint: ((each
isAnnotatedWith: 'Deprecated') ifTrue: [Color
red] ifFalse: [Color black]))];
    with: (usedDeprecated, (usedDeprecated
flatCollect: #clientTypes)) asSet.
m edges connectFromAll: #clientTypes.
m layout force.
m
```

a GtMondrian

a FAMIXClass (org::argouml::ui::ProjectBrowser)

```
/**
 * The main window of the ArgoUML application.
 *
 * @stereotype singleton
 */
public final class ProjectBrowser
    extends JFrame
    implements PropertyChangeListener, TargetListen

    /**
     * Default width.
     */
    public static final int DEFAULT_COMPONENTWIDTH

    /**
     * Default height.
     */
    public static final int DEFAULT_COMPONENTHEIGHT

    /**
     * Logger.
     */
    private static final Logger LOG =
        Logger.getLogger(ProjectBrowser.class);

    /**
     * Position of pane in overall browser window.
     */
    public enum Position {
        Center, North, South, East, West,
        NorthEast, SouthEast, SouthWest, NorthWest
    }

    // Make sure the correspondence that we depend
    static {
        assert Position.Center.toString().equals(
        assert Position.North.toString().equals(
        assert Position.NorthEast.toString().equals
        assert Position.South.toString().equals(
    }

    /**
     * Flag to indicate if we are the main applicat
     * or being integrated in another top level app
     * as Eclipse (via the ArgoEclipse plugin).
     * TODO: This is a temporary measure until Proj
```

a FileReference (feenk)

Icon	Name	Size	Creation
	..	384 B	2019-06-08 07:4
	data	160 B	2018-09-07 17:1
	pictures	160 B	2018-09-07 17:1
	tutorial	256 B	2018-09-07 13:3
	.DS_Store	6.15 kB	2018-09-07 17:1
	cointerp.c	1.86 MB	2019-02-23 11:3
	readme.txt	89 B	2014-02-10 17:0
	sample-by-json	13.64 kB	2019-02-20 10:3

a FileReference (feenk/pictures)

Icon	Name	Size	Creation
	..	288 B	2018-09-07 17:0
	feenk.png	181.47 kB	2018-09-07 16:2
	gt-icon.png	8.19 kB	2018-09-07 16:2
	gtoolkit-icon.pn	15.43 kB	2018-08-12 17:0

a FileReference (feenk/pictures/gtoolkit-icon.png)

feenk

We make your systems explainable.

feenk.com

We work as a consulting company.
Yet, we are not just consultants.
We are authors and researchers.

We invented Moldable Development
and we develop Glamorous Toolkit.

feenk

2009 After 7 years in research, Tudor Gîrba authors the humane assessment method which pioneers the software assessment domain: it shows why software development should be regarded primarily as a decision making business, and how systems can be made explainable through systematic construction of custom tools.

2014 After intensive applications of software assessment in industry, Tudor's work receives international recognition through the Dahl-Nygaard Junior Prize. He is the only recipient of the prestigious prize acting from industry.

2015 feenk is founded with the goal of reshaping the development experience to make systems explainable. feenk takes on ambitious client projects.

2020 feenk launched the first productive version of Glamorous Toolkit, the moldable development environment that changes the economics of software development by integrating software assessment deep in the development process.

feenk

Before we part …

Software is not a cost.
It’s an investment.

Black boxes are (too) risky.
Software assessment is a strategic skill.

Profitable systems are recyclable.
Architecture is a business asset.

Hand-drawn pictures represent either wishes or beliefs.
Decisions should be based on facts.

Tools matter.
Pick them carefully.

Legacy is a positive thing.