Upgrades

Henri Piron, Technical consultant Upgrade Team

Donat van Steenbergh, Team Leader Channel

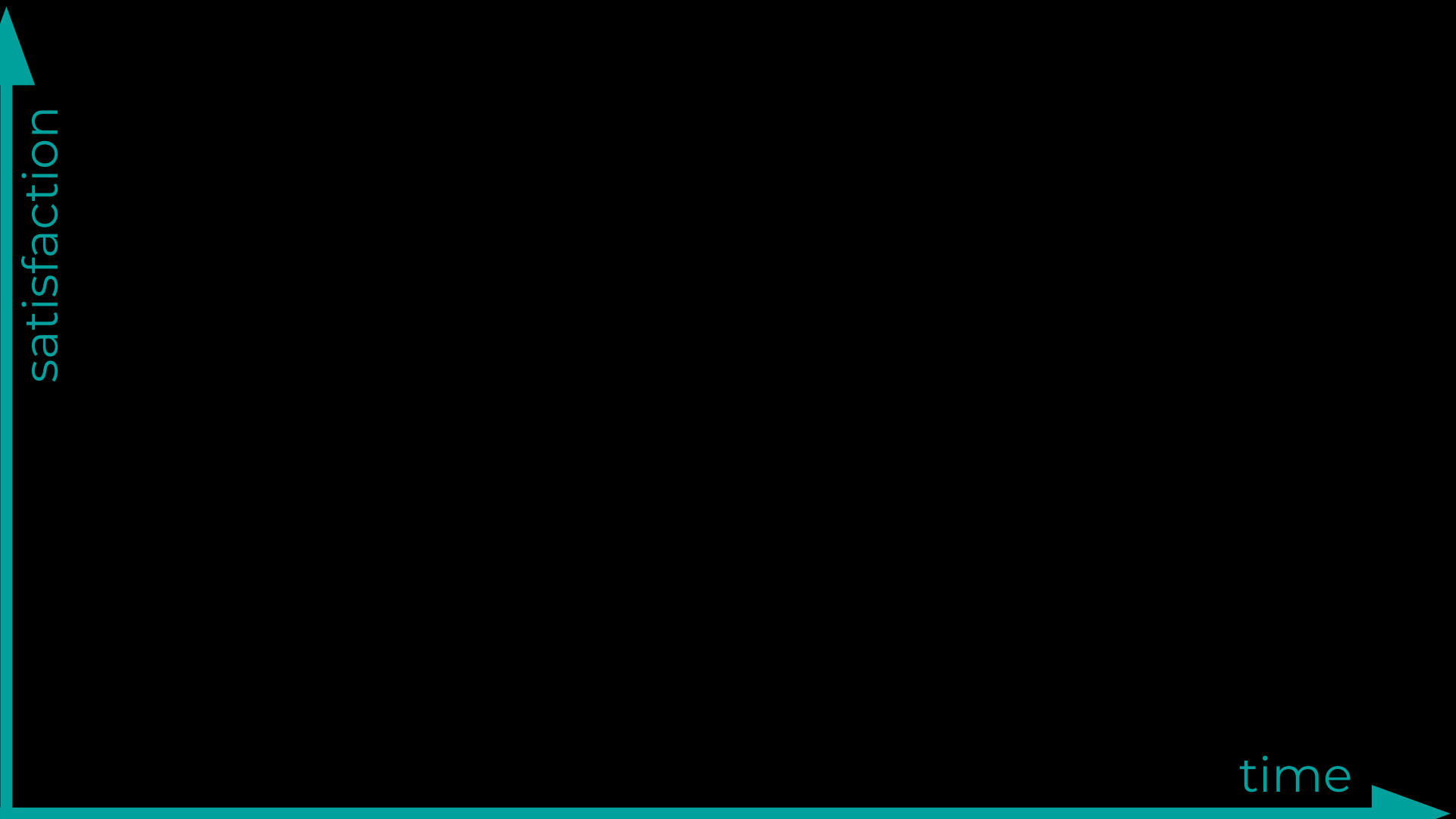
Agenda

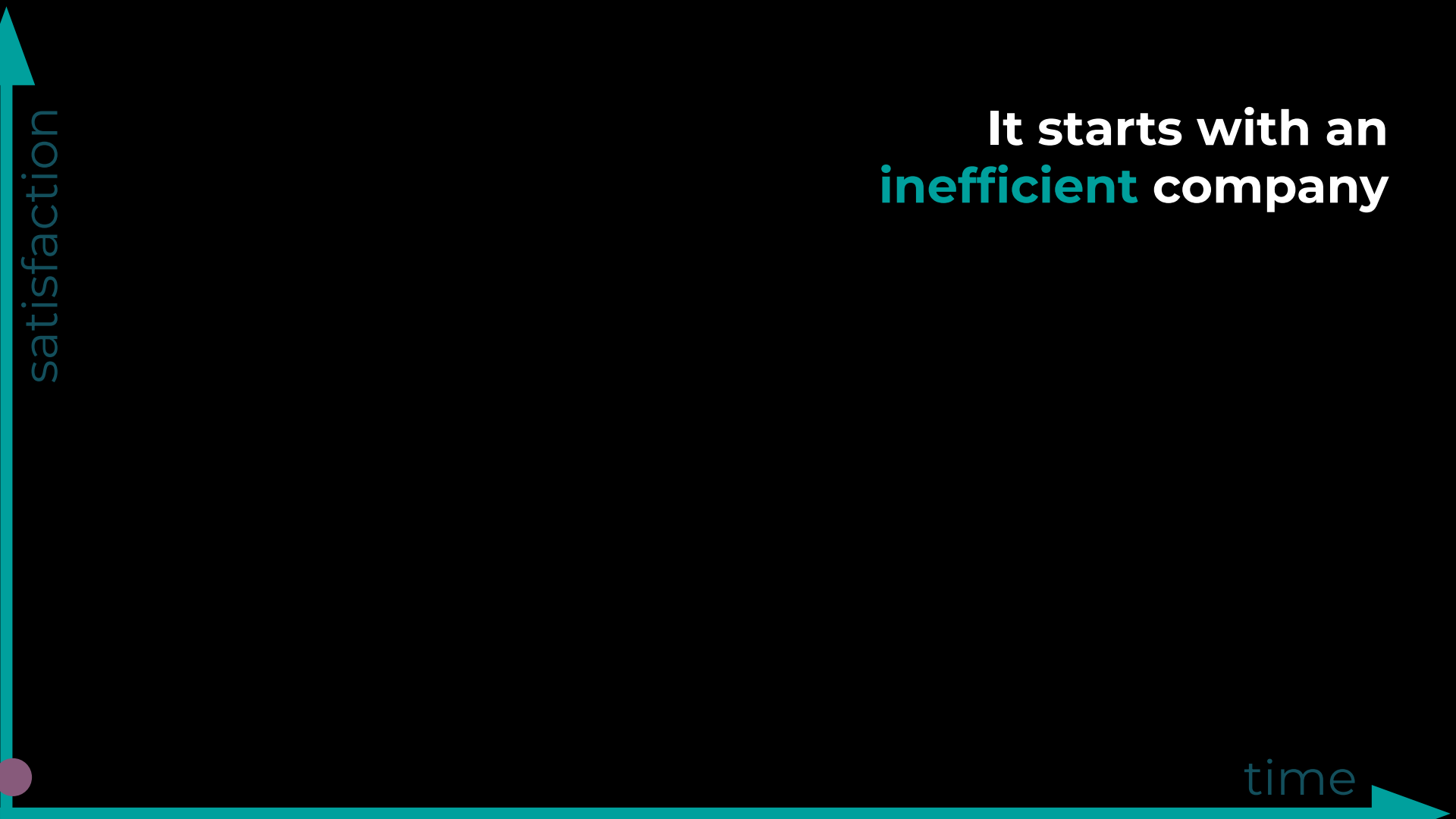
1. Context
2. Business
3. Technical
4. Tips & Tricks
5. Q&A

Business

Today's reality

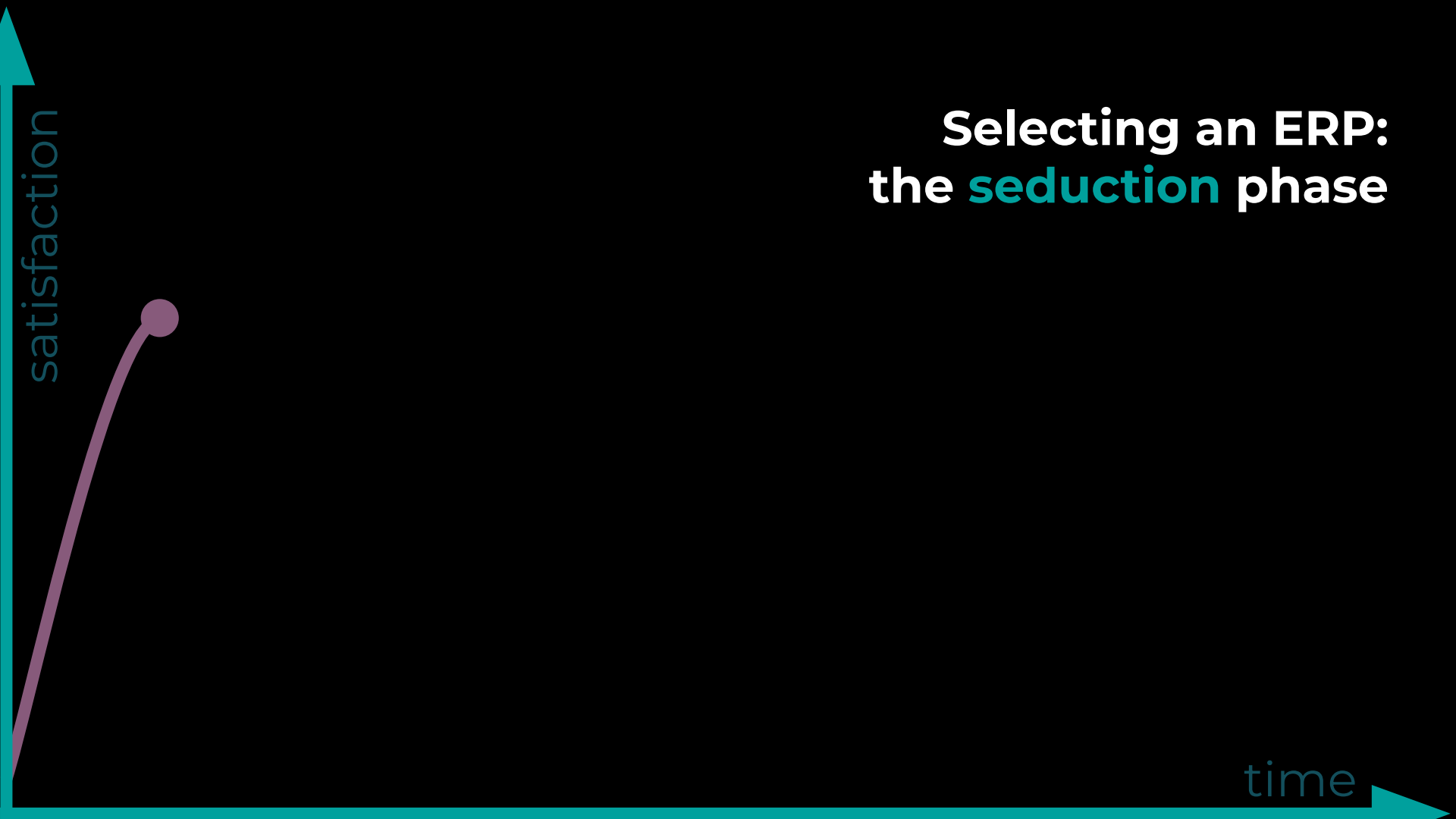
The life of an ERP implementation.


satisfaction
time



It starts with an
inefficient company

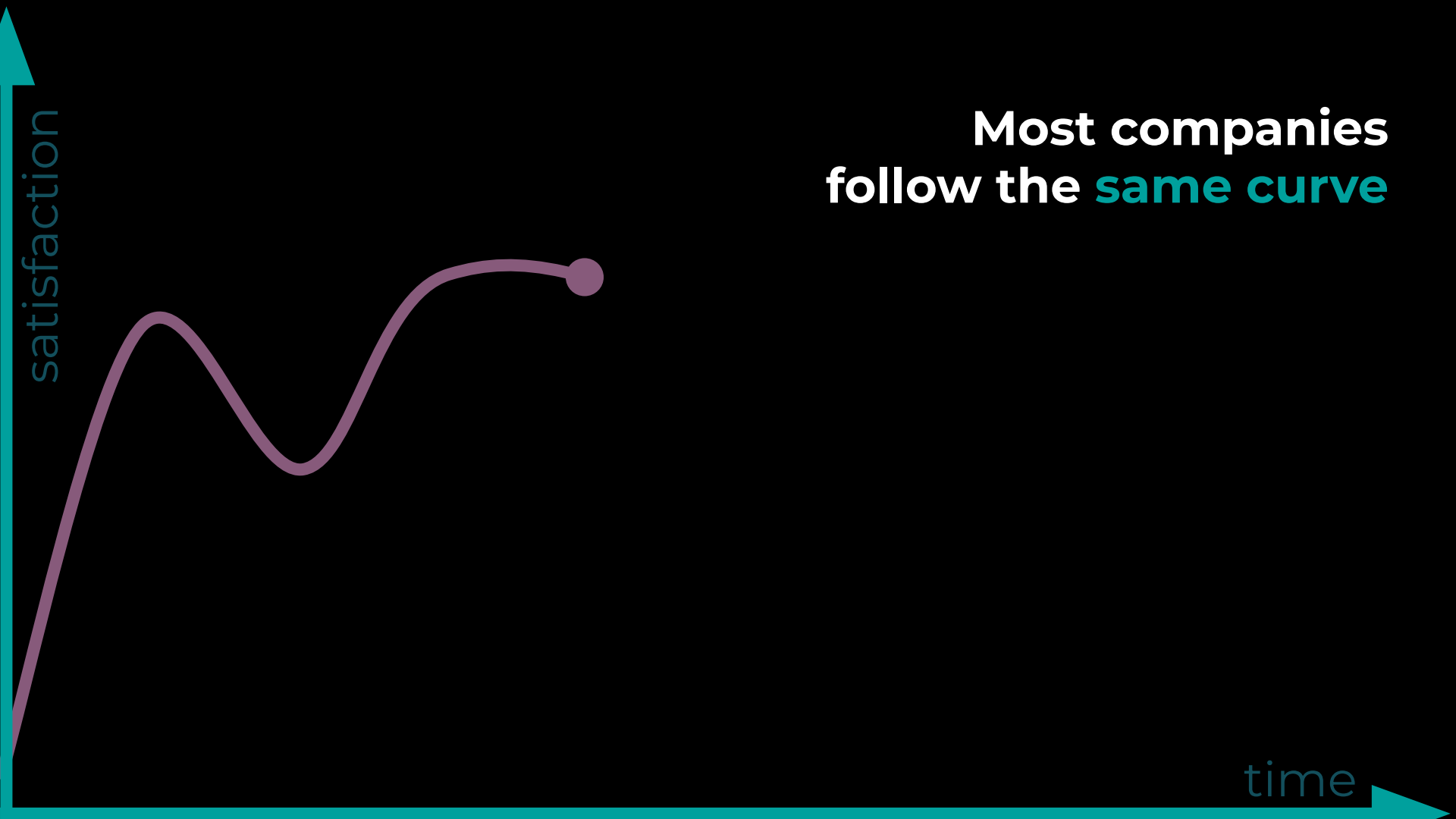
Selecting an ERP: the seduction phase



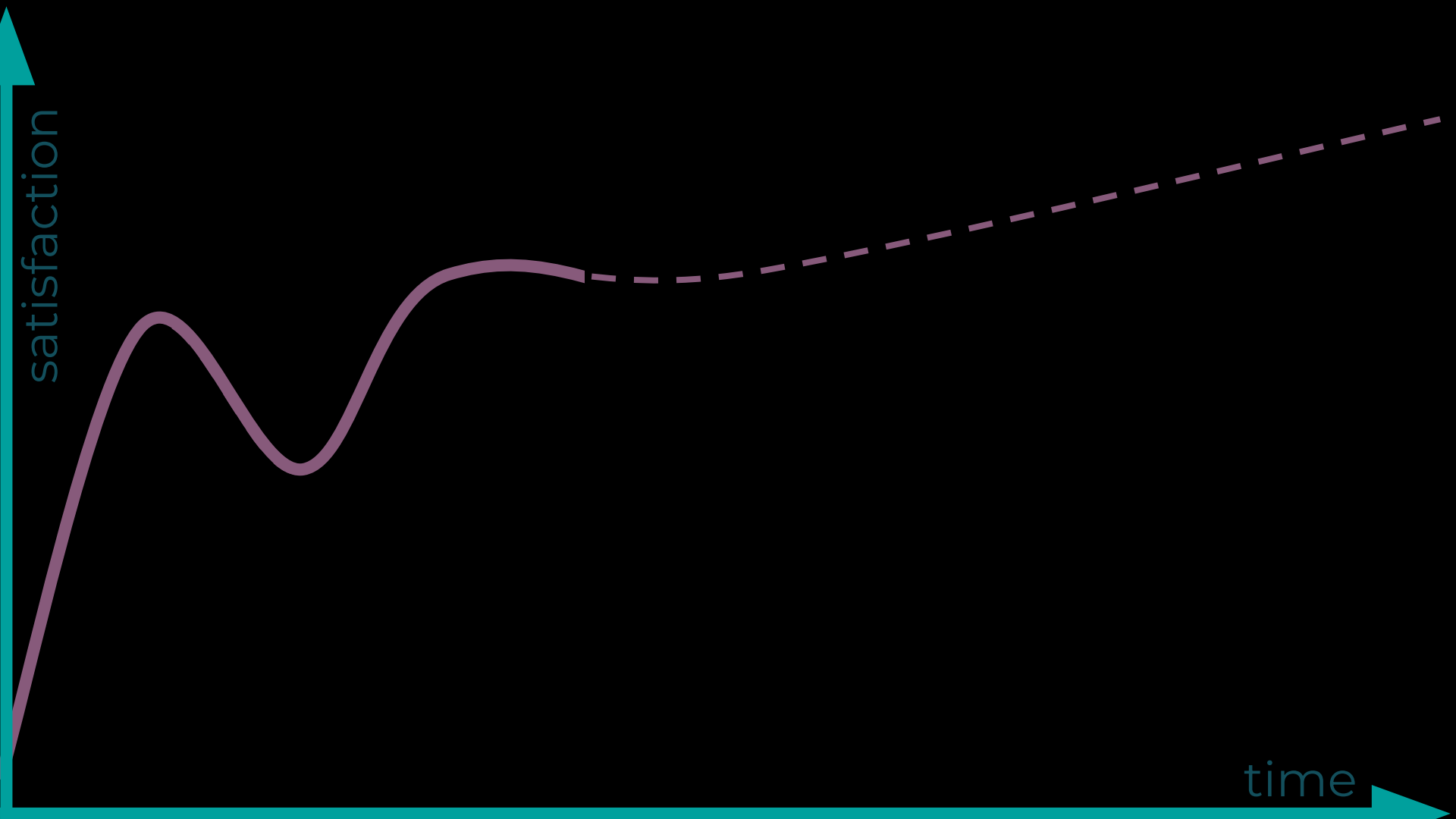
The implementation: living together

Going live: the wedding

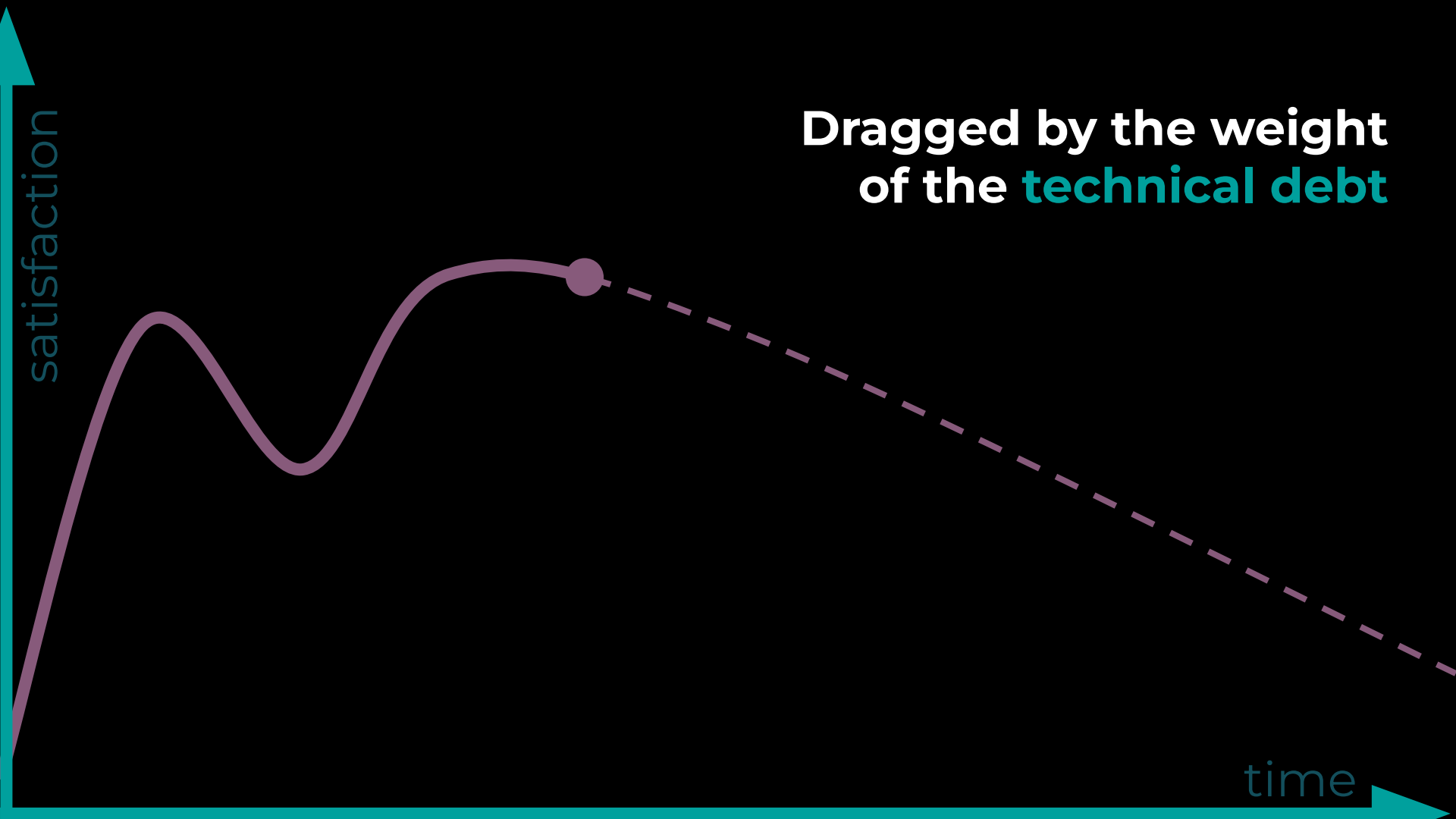
An efficient tool: a happy life



Most companies follow the same curve


satisfaction
time

Dragged by the weight of the technical debt



Why are some companies dragged by the weight of the technical debt?

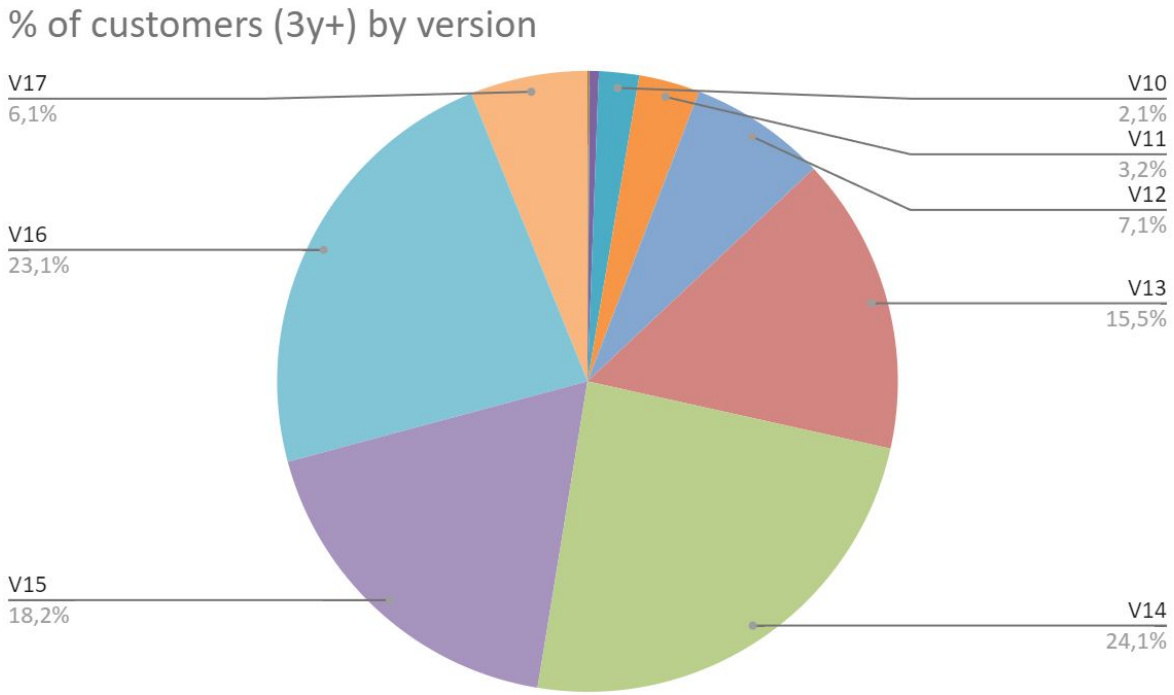
Why are some company dragged down by the technical debt ?

- Stick to Older Version
- Not integrator's priority
- Nobody wants to fix issues

This leads to the following issues

- Customer stuck on unsupported versions

This leads to the following issues



More than 52% of our customers older than 3 years are on unsupported versions

This leads to the following issues

- Customer stuck on unsupported versions
- Unsatisfied customers ⇒ Higher risk of churn
- Missed opportunities to sell more services

Important to have maintenance/upgrade contract !

What type of contract should you sell ?

Any contract is better than nothing ...

Key elements

- Internalize costs
- Contract should cover maintenance and upgrade.
- Insurance that is prepaid
- Recurrent vs T&M

Benefits of maintenance contract

1) Keeps customers on updated versions
2) Avoid bad surprises for customer and issues getting bigger
3) Incentive for integrators to maintain old customers (opportunity to sell services)
4) Customer have a guarantee

How to evaluate the value of the contract ?

3 years ago: launch of the "**Standard maintenance contract**" based on the amount of Lines of Code (LoC)

Purpose:

- Guarantee of Odoo as a back up (support, bug fixes and upgrade)
- Simple for the customer (1 contract to cover everything)
- Good tool to limit increase in LoC
- Easily auditable and transparent pricing (can be automated)
- Monthly installment possible = affordable

⇒ 16€ / 100 Lines of Code

Different types of contracts

Based on LoC

- Good tool to limit development
- Easily auditable and transparent
- Reward the code refactoring
- Good for community modules

Based on a % of the budget

- Known market practice
- ~ 20% of development cost to pay on a yearly basis.

T&M (on demand)

- No insurance
- No provisioning

Best Practices

- Limit the amount of upgrades
- Define what is included vs what needs to be paid

  Technical vs Functional
- Upgrade Support via DoD services
- Give a budget/timing to your maintenance team

Technical

Technical Agenda

1. Documentation
2. Studio & Upgrades
3. Scripts vs Manual
4. Tips & Tricks
5. Q&A

Documentation update

Standard

- Getting an upgraded database
- Testing
- Going in production

Custom

- Steps we follow
- Good practices

Documentation update

odoo docs

What are you looking for?

Try Odoo for FREE

EN Odoo 17

- User Docs
  - Odoo essentials
  - Finance
  - Sales
  - Websites
  - Inventory & MRP
  - Human resources
  - Marketing
  - Services
  - Productivity
  - Studio
  - General settings
- Database management
  - Hosting
  - Odoo Online
  - Odoo.sh
  - On-premise
  - Upgrade
  - Neutralized database
  - Supported versions
  - Odoo mobile apps
  - Odoo.com accounts

Odoo Documentation

User Docs

Discover our guide to help you use and configure the platform, by applications.

TOP APPS

- Accounting
- Inventory
- Manufacturing
- Point of Sale

Install and Maintain

Learn how to install, deploy and upgrade Odoo on premise or on Odoo.sh.

TOP LINKS

- Installing Odoo
- Bugfix updates
- Upgrading Odoo
- Odoo.sh: The Odoo Cloud Platform

Developer

Learn to develop in Odoo with the developer tutorials and framework references.

TOP LINKS

- Tutorials
- How-to guides
- Reference

Contributing

You want to contribute to Odoo but don't know where to start? The tutorials and guidelines are there to help you make Odoo even better.

TOP LINKS

- Coding guidelines
- Documentation

Documentation update

Upgrade

Edit on GitHub

An upgrade is the process of moving your database from an older version to a newer **supported version** (e.g., Odoo 14.0 to Odoo 16.0). Frequently upgrading is essential as each version comes with new and improved features, bug fixes, and security patches.

> **Automatic upgrades: Odoo Online's Rolling Release process**

An upgrade does not cover:

- Downgrading to a previous version of Odoo
- **Switching editions** (e.g., from Community to Enterprise)
- **Changing hosting type** (e.g., from on-premise to Odoo Online)
- Migrating from another ERP to Odoo

> **Warning**
> If your database contains custom modules, it cannot be upgraded until a version of your custom modules is available for the target version of Odoo. For customers maintaining their own custom modules, we recommend to parallelize the process by **requesting an upgraded database** while also **upgrading the source code of your custom modules**.

Upgrading in a nutshell

1. Request an upgraded test database (see **obtaining an upgraded test database**).
2. If applicable, upgrade the source code of your custom module to be compatible with the new version of Odoo (see **Upgrade a customized database**).
3. Thoroughly test the upgraded database (see **testing the new version of the database**).
4. Report any issue encountered during the testing to Odoo by **submitting a ticket for an issue related to my future upgrade (I am testing an upgrade)**.

ON THIS PAGE

Testing: involve the customer

- Define key users: 1 in each team, managers,...
- What are their critical flows?
  - Get them to test what they do the most
  - Complete full flows

Basic test checklist

- Are there views that are deactivated in your test database but active in your production database?
- Are your usual views still displayed correctly?
- Are your reports (invoice, sales order, etc.) correctly generated?
- Are your website pages working correctly?
- Are you able to create and modify records? (sales orders, invoices, purchases, users, contacts, companies, etc.)
- Are there any issues with your mail templates?
- Are there any issues with saved translations?
- Are your search filters still present?
- Can you export your data?

Documentation update

Upgrade a customized database

Edit on GitHub

Upgrading to a new version of Odoo can be challenging, especially if the database you work on contains custom modules. This page intent is to explain the technical process of upgrading a database with customized modules. Refer to **Upgrade documentation** for guidance on how to upgrade a database without customized modules.

We consider a custom module, any module that extends the standard code of Odoo and that was not built with the Studio app. Before upgrading such a module, or before requesting its upgrade, have a look at the **Service-level agreement (SLA)** to make sure who's responsible for it.

While working on what we refer to as the **custom upgrade** of your database, keep in mind the goals of an upgrade:

1. Stay supported
2. Get the latest features
3. Enjoy the performance improvement
4. Reduce the technical debt
5. Benefit from security improvements

With every new version of Odoo, changes are introduced. These changes can impact modules on which customization have been developed. This is the reason why upgrading a database that contains custom modules requires additional steps in order to upgrade the source code.

These are the steps to follow to upgrade customized databases:

1. **Stop the devolopments and challenge them.**
2. **Request an upgraded database.**
3. **Make your module installable on an empty database.**
4. **Make your module installable on the upgraded database.**
5. **Test extensively and do a rehearsal.**
6. **Upgrade the production database.**

ON THIS PAGE

Documentation update

odoo docs

What are you looking for?

Try Odoo for FREE

EN · Odoo 17

Upgrade scripts

Edit on GitHub

An upgrade script is a Python file containing a function called `migrate()`, which the upgrade process invokes during the update of a module.

`migrate(cr, version)`

Parameters: **cr** (`Cursor`) – current database cursor
version (*str*) – installed version of the module

Typically, this function executes one or multiple SQL queries and can also access Odoo's ORM, as well as the **Upgrade utils**.

Writing upgrade scripts

Upgrade scripts follow a specific tree structure with a naming convention which determines when they are executed.

The structure of an upgrade script path is `$module/migrations/$version/pre,post,end-*.py`, where `$module` is the module for which the script will run, `$version` is the full version of the module (including Odoo's major version and the module's minor version) and `{pre|post|end}-*.py` is the file that needs to be executed. The file's name will determine the **phase** and order in which it is executed for that module and version.

> **Note**
> From Odoo 13 the top-level directory for the upgrade scripts can also be named `upgrades`. This naming is preferred since it has the correct meaning: *migrate* can be confused with *moving out of Odoo*. Thus `$module/upgrades/$version/` is also valid.

> **Note**
> Upgrade scripts are only executed when the module is being updated. Therefore, the module's minor version set in the `$version` directory needs to be higher than the module's installed version and equal or lower to the updated version of the module.

ON THIS PAGE

Studio & Upgrades

Studio & Upgrade

Studio

- Be **aware** of the **risk**
- Think of it as **development** ⇒ it creates a **technical debt**
- It **increases** the probability of **issues**

Studio & Upgrade

Upgrade

- **Test** it: just like **custom code**
- **Submit** your **issues**: odoo.com/help

Studio & Custom modules

Custom

- **Avoid** mixing
- If you built **studio** on top of custom

  ⇒ you are **responsible**

Scripts vs manual work?

Scripts

- **Avoid** manual actions
- Do the job **once**
- **Migrate** data

Scripts: how?

- `migrations` folder
- **version** folder
- `pre-` , `post-`, `end-` prefixes

Documentation

Scripts: how?

- `custom_module`
  - `17.0.1.0.1`
    - `pre-fix-field.py`
  - `__manifest__.py`

- Version in database `17.0.1.0.0`
- Version in manifest `17.0.1.0.1`

module update

Documentation

Example

One script instead of multiple action

- All those **actions** can be done **manually**
- It **can** require **thousands** of actions

Anything done **manually** can be **scripted**

```python
from odoo.upgrade import util

def migrate(cr, version):

    # Get the environment
    env = util.env(cr)

    # Set a parameter
    env['res.company'].browse(1).filter_allowed_carriers = True

    # Recompute a stored field
    ids = env['sale.order'].search([
            ('delivery_status', '=', 'pending')
        ]).ids

    util.recompute_fields(
        cr, 'sale.order', ['shipping_weight'], ids
    )
```

Util & Custom util

github.com/odoo/upgrade-util

Util

- Used in odoo upgrade scripts
- Helpers

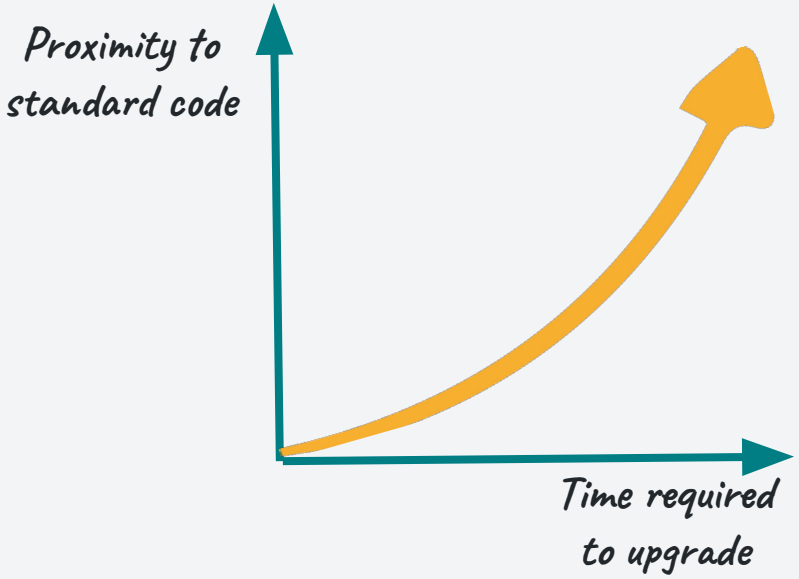
github.com/odoo-ps/custom-util

Custom-Util

- Used in custom upgrades
- Custom modules helpers

Tips & tricks

Avoid developing on Standard



- The more you customize standard feature
- The hardest it is to upgrade

Easier to upgrade

- New models/fields
- New features
- New applications
- Integrations

Common point:

Standard code **changes** are **less** likely to have an **impact** on them

Have an upgrade team

- Build **long term**
- Gain **expertise**
- **Everybody** should have done an **upgrade**

Implementation & upgrade teams

Implementation

- Project specific expertise
- Customer relationship

Upgrade

- Process knowledge
- Good practices

Get the **best** of **both** worlds

Every 3 versions maximum

- Get **new features**
- Don’t **overgrow** the technical **debt**
- **6 months** process every **3 years**

  ⇒ **17%** of the time spent upgrading
- Stay **supported**

Thank you!

Henri Piron
hepi@odoo.com

Donat van Steenbergh
dav@odoo.com

/odoo

@odoo.official