Odooオープントーク・2026年7月31日

OCA活動への参加方法

コードを書けなくても、今日から貢献できる

Yoshi Tashiro・コタエル株式会社（Quartile）
tashiro@quartile.co・www.quartile.co

Quartile

Quartile

まず、4つだけ手を挙げてください

A quick show of hands — so I can aim this talk at the right level

当てはまるものに挙手をお願いします

手の挙がり方で、今日どこを厚く話すかを決めます。

1

Odooを触っている

業務でも検証でも

2

OCAモジュールを入れたことがある

1つでも

3

PR・イシューを出したことがある

マージ済みでなくても

4

他人のPRをレビューしたことがある

コメント1つでも

1つも当てはまらなくて構いません。今日はそこが出発点。

Quartile

「使う人」は増えた。「返す人」が増えていない

Japan has users. What it lacks is contributors — and reviewers most of all

1 Odooを触っている

2 OCAモジュールを入れたことがある

ここで切れている

3 PR・イシューを出した

4 レビューした

※ 統計ではなく、日本のコミュニティで見えている傾向

この差を埋めるのは、技術力ではない

3・4に進まないのは、難しいからではなく**入口を知らないから。**今日の話は、その入口だけ。

そして今、一番足りていないのは4番 —— レビュー

AIでコードを書くのは安くなった。レビューは安くなっていない。だから**コードが書けなくても、いちばん喜ばれる貢献ができる。**

Part 1

01
OCAとは何か、なぜ関わるのか

組織の位置づけと、貢献が自分に返ってくる仕組み

前提

OCA(Odoo Community Association)とは

A non-profit association supporting collaborative Odoo development

Odoo S.A.

ベルギーの営利企業

- Odoo本体(Community / Enterprise)の開発
- 公式パートナー制度・SaaSの提供
- 製品ロードマップの決定

OCA

非営利団体・ボランティア主体

- コミュニティモジュールの開発と維持
- ローカライゼーション・翻訳基盤の運営
- コーディング規約・CI・レビュー体制

別組織だが、同じエコシステムの上にいる

OCAは「本体に足りない部分」を集まって埋める場。成果物はすべてAGPL/LGPLで github.com/OCA に公開。

メンバーは機能コンサルタント(業務要件)と技術コンサルタント(Python)が混在する構成。

Quartile

貢献は「奉仕」ではなく、自分に返ってくる

Not altruism — it sharpens quality, builds skills, and earns trust

保守コストが下がる

自社カスタマイズとして抱え続けると、**バージョンアップのたびに自分で直す**。OCAに出しておけば、世界の誰かが移行してくれる側に回る。

品質とスキルが上がる

世界のレビュアーに見られる前提で書くと、コードは確実に良くなる。**レビューで学べる量が、社内だけの開発とは桁違い**。

顧客の信頼につながる

「標準とOCAで作れる範囲」を判断できる根拠になる。貢献実績は、提案の説得力そのもの。

AI時代の「正解」を残せる

AIは日本の税・帳票ルールを平然と間違える。レビュー済みのOCAモジュールが、AIの出力を**検証できる唯一の基準**になる。

AIでコードを書くのは安くなった。レビューは安くなっていない

結果として、AIが書いたPRが少数のメンテナに集中する。だから今、コードよりも**レビューする人**の価値が上がっている —— ここが日本から入る一番の隙間。

Part 2

02
どこから入るか

非技術者ルートと、技術者ルート

Quartile

貢献は「コードを書く」だけじゃない

Four entry points that need no Python and no local setup

翻訳

Weblate上で日本語訳を改善。**翻訳PRは直接投げない**のがルール。

translation.odoo-community.org

Issueを立てる

バグでも要件でもよい。該当リポジトリのIssueに**再現手順**を書く。それだけで十分に価値がある（次ページ）。

github.com/OCA

機能レビュー

Runboatの検証インスタンスでPRを試し、PRにコメント。**ローカル環境不要。**

PRのGitHub checksからリンク

ドキュメント

README・モジュール説明・How-toの改善。**最も通りやすい**入口。

oca-for-functional-consultants

いま一番不足しているのは、コードではなくレビューと翻訳

機能コンサルタントがRunboatでPRを触ってフィードバックを残すだけで、マージまでの詰まりが解ける。Python も Git も要らない。

Quartile

困っていることは、Issueにする

Open an issue — the cheapest contribution, and the only way others can join you

✗ 社内で解決して終わる

- 同じ問題を、他社も別々に踏み続ける
- 直したコードが自社の中に閉じる
- 「Odooは日本だと厳しい」で話が止まる

解決したこと自体は成果。ただし**誰にも見えない**ので、次の人の役には立たない。

✓ Issueとして公開の場に置く

- 検索で見つかる —— 半年後の誰かが助かる
- 「うちも同じ」が集まり、**優先度が上がる**
- パートナー同士で**分担・共同開発**の起点になる

再現手順と「日本ではこう回らない」が書けていれば成立する。**コードは要らない。**

Issueは答えを出す場ではなく、人が集まる場

まず検索し、あれば「うちも踏んだ」と**一言足す**。無ければ立てる。**数が見えないと、要件は要件として扱われない。**

どのリポジトリか分からなければ、Discordかこのオープントークで聞く。**賛同する会社が集まれば**、設計もモジュール化も分担できる。

Quartile

技術者ルート:とにかく小さく始める

Start small ・ your first PR is about learning the workflow

✗ 最初のPRで避けたいこと

- 大きな新機能の提案
- 大規模なリファクタリング
- 複数モジュールにまたがる変更

新規貢献者の場合、これらは採用率が明確に低い。技術力の問題ではなく、レビュー負荷とリスクの問題。

✓ おすすめの入口

- **自分が既に使っているモジュール**の小さな不具合修正
- ドキュメント(README・docstring)の改善
- 他人のPRのレビュー

1回目の目的は「成果を出す」ことではなく、**ワークフローに慣れる**こと。

前提知識:Git / Python / Odoo技術ドキュメント

流れは OCA Days 2025 の Daniel Reis の解説動画が一番早い： youtu.be/bX4AXbqQwJU

Part 3

03
出してから、マージされるまで

流れ・期待値・実例・落とし穴・AIの扱い

Quartile

PRがマージされるまでの流れ

CLA → PR → CI → review → /ocabot merge

1 準備

CLA(貢献者ライセンス同意)に署名。既存のIssue・PRを確認して重複を避ける。

2 PR作成 — ここでのミスが一番多い

対象は**バージョンブランチ**(`19.0` / `18.0`)。`main` や `master` ではない。タイトルは `[18.0][FIX] module_name: description`、コミットは `[TAG] module_name: description`。原則 **1 PR = 1 コミット**。AIを使ったなら `Assisted-by:` を付ける(後述)。

3 CI が自動で回る

pre-commit(整形・lint)／ GitHub Actions(テスト)／ Runboat(実機検証用インスタンス)／ Codecov(カバレッジ・参考情報)。

4 レビューと修正の往復

指摘への**反応の速さ**が採用率を大きく左右する。往復が発生するのは普通のこと。

5 マージ

メンテナまたはPSCメンバーが `/ocabot merge minor` とコメント。**自分では押せない**。あとはbotが取り込む。

Quartile

初めてのPRを出した後、何が起きるか

What to expect after your first PR

3

5日以内なら、必要な承認レビュー数

2

5日経過後は2件でよい（うち1件以上はPSCメンバー）

120日

動きがないとstale botがクローズ。コメント1つで復活する

マージを待たなくていい

未マージのPRは `git-aggregator` で取り込める（`repos.yml` に1行）。**今日から使える**ので、マージは後片付け。ただし自社で持つ差分は、バージョンアップのたびに払い直す —— **上流に入れるほうが安い。**

レビューを待つ最短ルートは、他人のPRをレビューすること

レビューは全員ボランティア。期間の保証はない。相互性が効く世界で、「出す人」より「見る人」になるほうが自分のPRも早く進む。

- CIは通常、PRを開いた時点で自動実行される（GitHubアカウントが新規の場合のみPSCの承認が必要）。
- 指摘は主に整形（pre-commit / ruff）、カバレッジ、Odooの作法。**敵対ではなく協調** —— レビュアーはマージさせたい。

Quartile

実例：2行のPRが、どう進んだか

A real merged PR — 2 lines, 2 approvals, merged the same day

`[18.0][FIX] account_billing: Change 'invisible' to 'column_invisible'` —— OCA/account-invoicing#2201（18.0移行で見落とした列表示の不具合）

2
追加2行・削除2行・1ファイル

2
承認レビュー。**2件とも社外**から

8時間
PR作成からマージまで（同日）

0分	PR作成。**bot が自動でモジュールのメンテナにメンション**（自分で探して声をかける必要はない）
＋3時間	社外の開発者が approve。コメントは `LGTM` の一言だけ
＋8時間	PSCメンバーが2件目の approve。`trivial` と書いて `/ocabot merge minor`
＋8時間	**マージ完了。**「Congratulations, your PR was merged. Thanks a lot for contributing to OCA. ❤️」

2行で十分。そして小さいほど早く通る

別のPRは同じ1ファイルで**76日**。この幅がどちらも「普通」。**マージ前でもコードは使える**（前ページ）ので、待ち時間は止まっている時間ではない。

Quartile

新規貢献者がやりがちなミス

Common newcomer mistakes・check before you submit

✗ 翻訳PRを直接投げる

→ Weblate を使う。PRでの翻訳更新は受け付けられない。

✗ 履歴を残さない移行PR

→ 公式のmigration guideに従い、**git履歴を保持**する。ゼロから作り直さない。

✗ 複数トピックを1つのPRに詰める

→ 1 PR = 1 の課題・機能。混ざっているなら分割する。

✗ AI利用を申告せず投稿

→ `Assisted-by:` を付ける。2026年7月からOCAの明文ルール（次ページ）。

✗ pre-commitを飛ばす

→ push前に `pre-commit run -a`。CIで同じ指摘を受ける前に潰す。

✗ PRタイトル・コミット形式のミス

→ `[18.0][FIX] module_name: description` の形を守る。

最新 ・ 2026年7月に明文化

AIを使って貢献するときのルール

OCA’s new Generative AI / LLM Policy — in effect now

AIの使用は禁止ではない。求められるのは「開示」と「責任」だけ

原文の要点：**「AIへの責任の放棄は受け入れない」**。使ったこと自体は品質評価に無関係。

やること：コミットに1行足す

```
[IMP] my_module: add foo

Assisted-by: Claude Opus 4.6
Assisted-by: GitHub Copilot:gpt-5
```

- **「使ったか / 使っていないか」の二値。**助言レベルから全自動生成まで、量の閾値はない。
- モデルごとに1行。PRタイトルには不要だが、**説明欄には書く。**

```
Co-authored-by: Claude    ← 使ってはいけない
```

著作者フィールドは法的に未定義のため、AIには使わない。

越えてはいけない線

- **無監督のエージェントは不可。**PRやレビューの自動大量生成は**永久BAN**（人間のアカウントでも）。
- **全行を説明・弁護できないなら出さない。**「AIが書いた」は指摘への回答にならない。
- レビューは会話。**指摘に答えず再生成して出し直すのは回答ではない**（rework指標に加算される）。
- レビューする側も、**AI生成のレビューコメントを投稿しない。**AI要約は自分で検証してから。

レッドライン：1つで精査、2つで却下

24時間に5PRまたは10レビューを超える／事前合意なく1貢献500行超／指摘に応えず再生成の反復。
参照点は**「1ファイル30行未満のパッチ」**—— つまり**小さく出せ**が公式基準になった。

考え方は `レビュアーの負荷 = 量 × 頻度`。全文：github.com/OCA/.github/blob/master/AI_POLICY.md

Quartile

個人の貢献の先にあるもの

Governance, membership, sponsorship, paid opportunities

PSC（Project Steering Committee）

リポジトリごとの品質管理とマージ権限。**実力主義**で、継続的な貢献とレビューが唯一の道。登録は `repo-maintainer-conf` にPR。

oca.github.io/repo-maintainer-conf（PSC一覧）

OCA会員（Membership）

有料会員として財団を支援。Delegate Membersが年次で選出され、戦略と予算を承認し、理事会（Board）を選出する。

odoo-community.org/get-involved/become-a-member

スポンサーシップ

企業として財政支援。Bronze / Silver / Gold / Platinum の各ティアで、OCAサイト・イベント・刊行物での露出が得られる。

odoo-community.org/get-involved/become-a-sponsor

有償案件（RFQ）

移行やインフラ更新など、OCA自体のプロジェクトに対して見積依頼（RFQ）が出ることがある。SI事業者にとっては実務的な接点。

odoo-community.org/about/rfq-process

続けるには、会社としてのコミットが要る

承認は**実際には自社チームから出ることが多い**。個人の善意では続かない —— **会社として複数人で関わる**形にして初めて回る。他社と組めば、レビューを自社だけで抱えずに済む。

Quartile

日本コミュニティとしての現実的な入口

Where Japanese contributors can make the most difference

Weblateで日本語訳を直す

訳語の統一と品質は、日本語ユーザーが**最も早く効果を体感できる**領域。自分が業務で使っている画面の違和感を直すのが一番続く。

l10n-japan に関わる

日本固有の帳票・税制・商習慣。まだ空白が多く、**機能要件を言語化できる人**の価値が高い。会計周りはOdoo S.A.側も動き始めている（次ページ）。

Discord・メーリングリスト

OCA公式は英語。日本語なら**非公式日本Odooコミュニティ**（`discord.gg/pDCzBjJN3X`）。相談相手を増やすことが、継続の条件。

このオープントーク

詰まっていることを持ち込む場。「反応がない」「どのリポジトリか分からない」も歓迎。ここで出た話は**Issueにして残す**と、来ていない人にも届く。

日本語圏の課題は、日本語圏の人しか報告しない

日本の帳票要件や訳語の不自然さは、グローバルのレビュアーには見えない。「誰かが直すはず」の「誰か」は**報告する側**にはいない。報告先はOCAにずっとあり、今年はOdoo S.A.にも増えた（次ページ）。

Quartile

Odoo S.A. 側にも、受け取る人ができた

Odoo S.A. now has a product manager for Japan — the community's reports have somewhere to land

Odoo S.A.

本体(Community / Enterprise)を作る側

- **2026年5月から**、日本のローカライゼーションを担当するプロダクトマネージャーが置かれた
- 当面のフォーカスは**会計ローカライゼーション**
- 見ているのは**本体のコード**。OCAモジュールを取り込む役割ではない

コミュニティ / OCA

現場の要件を見つけて検証する場

- 実際の業務で何が通らないかを**最初に踏む**のは現場
- OCAモジュールとして**先に作って動かして確かめられる**
- 訳語の不自然さは、日本語で使っている人しか気づけない

コミュニティとOdoo S.A.は競合ではない。同じ穴を、別の側から埋めている

PMは「この帳票では実務が回らない」を自分では発見できない。**現場からの入力が前提**の役割。ただし本体とOCAは**別のライン**。つなぐのは**コードではなく要件**。

つまり、いま「日本ではこう回らない」を伝える価値が上がっている

OCAには以前から出せた。一方で**本体**に日本固有の要件を伝える先はなかった —— そこに担当者ができた。別ルートなので**両方に意味がある**。

Quartile

今日からできる一歩

Four things you can do this week

1 CLAに署名しておく

PRを出す前でも構わない。先に済ませておくと、いざ出したときレビューが止まらない。

2 Weblateのアカウントを作り、訳を1つ直す

よく使うモジュールの、気になっていた訳語をひとつ。これで貢献者になれる。

3 気になるPRをRunboatで開いてコメントする

動いた／動かなかった、を書くだけでレビューとして成立する。

4 いま困っていることをIssueに1本立てる

社内の回避策は、そのままIssueになる。既に同じIssueがあれば「うちも踏んだ」と一言足す。

貢献ガイド github.com/OCA/odoo-community.org … CONTRIBUTING.rst

CLA odoo-community.org/page/cla

規約・移行手順 github.com/OCA/maintainer-tools/wiki

Discord（日本語） discord.gg/pDCzBjJN3X

機能コンサル向け odoo-community.org/oca-for-functional-consultants

翻訳（Weblate） translation.odoo-community.org

Discord（OCA・英語） discord.gg/y4ZNg6FVmm

より詳しい日本語ガイドを準備中。公開したら quartile.co でお知らせします。

最後に

その「誰か」は、
あなたです。

OCAの入口は、ずっと開いています。
今年はOdoo S.A.にも日本市場担当のPMができました。
でも「実務が回らない」と言えるのは、日本で使っている人だけです。

2行でいい。訳語ひとつでいい。「動きました」の一言でいい。
差は、入口を知っているかどうかだけ。

Yoshi Tashiro・tashiro@quartile.co・www.quartile.co

Quartile

Quartile

用語集 —— 今日出てきた略語

Glossary of the jargon in this deck

OCA Odoo Community Association。コミュニティモジュールを開発・維持する非営利団体。

PSC Project Steering Committee。リポジトリごとの品質管理とマージ権限を持つ人たち。

Runboat PRごとに自動で立ち上がる検証用Odooインスタンス。ローカル環境なしで実機確認できる。

pre-commit コミット前に整形・lintを自動実行する仕組み。`pre-commit run -a` で手元確認。

CI GitHub Actionsによる自動テスト。PRを開くと自動で回る(Odoo本体のRunbotとは別物)。

l10n localization(ローカライゼーション)の略記。`l10n-japan` は日本向けリポジトリ。

git-aggregator 複数リポジトリ＋未マージPRを`repos.yml`の定義どおりに集めてビルドするツール。マージ待ちのPRを先に使える。

CLA Contributor License Agreement。貢献者ライセンス同意。PRを出す前に一度署名すれば以後不要。

ocabot OCAの自動化bot。PRコメントで `/ocabot merge` 等を受け取り、マージやバージョン更新を代行。

Weblate OCAの翻訳プラットフォーム。翻訳は**必ずここで**行い、PRでは出さない。

stale bot 一定期間動きのないPR・Issueを自動クローズするbot。コメント1つで復活する。

Assisted-by: AIを使って書いた場合にコミットに付ける申告行(2026年7月からのOCAルール)。

LGTM "Looks Good To Me"。レビューでの承認を表す定番の一言。

> **迷ったら日本語で聞いてください**
>
> 用語が分からないこと自体が、参加の障壁。非公式日本Odooコミュニティ Discord —— discord.gg/pDCzBjJN3X