HelixSpecifier

HelixSpecifier

توسعه مبتنی بر مشخصات که تشریفات خودرا با کار تطبیق می‌دهد.

خلاصه

HelixSpecifier موتوری از نوع Go است که سه روش‌شناسی توسعه‌را — گردش کار مبتنی بر مشخصات SpecKit، نظم تست‌محور Superpowers و چرخه حیات نقاط عطف در — GSD یک جریان تطبیقی ادغام می‌کند. این موتور هر وظیفه‌را بر اساس میزان تلاش طبقه‌بندی کرده و حجم فرآیندرا متناسب با آن افزایش یا کاهش می‌دهد.

توضیح کوتاه

HelixSpecifier موتوری ترکیبی برای توسعه مبتنی بر مشخصات است که برای عامل‌های AI را در هم می‌آمیزد، کارهارا بر GSD و Superpowers، SpecKit طراحی شده است. این موتور اساس سطح تلاش طبقه‌بندی می‌کند، مراحل تدوین مشخصات‌را با بحث و تبادل نظر پیش می‌برد، حداقل نسبت تست به کدرا اعمال می‌کند و از هر جریان تکمیل‌شده یاد می‌گیرد.

توضیح مفصل

HelixSpecifier موتوری ترکیبی برای توسعه مبتنی بر مشخصات (SDD) است که به زبان Go نوشته شده و به عنوان جزئی از مجموعه (`digital.vasic.helixspecifier` ماژول) ساخته شده است. این موتور سه روش توسعه‌را که معمولاً در سه AI برای عامل‌های HelixAgent در SDD ابزار و سه ذهنیت جداگانه قرار دارند — فرآیند هفت‌مرحله‌ای SpecKit (مشخص‌سازی، شفاف‌سازی، برنامه‌ریزی، وظایف، تحلیل، پیاده‌سازی)، نظم تست‌محور Superpowers (Constitution، با اجرای موازی زیرعامل‌ها، و مدیریت چرخه حیات نقاط عطف در یک جریان تطبیقی ادغام می‌کند. هر یک از این سه ستون به کاری که در آن مهارت دارد — GSD ادامه می‌دهد؛ موتور است که آن‌هارا به جای یک خط لوله دستی‌دوز شده، به صورت یک جریان منسجم به اجرا درمی‌آورد.

است: موتور کارهای ورودی‌را بر اساس سطح تلاش طبقه‌بندی تشریفات تطبیقی ایده مرکزی این موتور کرده و حجم فرآیندرا متناسب با آن تنظیم می‌کند، به طوری که یک اصلاح یک‌خطی همان تشریفات سنگین یک قابلیت بزرگ‌را طی نمی‌کند — و یک قابلیت بزرگ نیز با دقت یک غلط املایی از فیلتر رد نمی‌شود. ده قابلیت کلیدی بر این ستون فقرات بنا شده‌اند: اجرای موازی وظایف با همزمانی محدود، به صورت کد» که قوانین اجباری‌را به صورت خودکار اعمال می‌کند Constitution»،

تست‌محوری نیکوئیست» که نسبت حداقل تست به پیاده‌سازی‌را رصد و اعمال می‌کند، پالایش» مشخصات از طریق بحث‌های چندمرحله‌ای و چندعاملی، یادگیری تطبیقی مهارت‌ها، تحلیل کدهای قدیمی، پیش‌بینی مشخصات بر اساس الگوهای تاریخی، انتقال دانش میان پروژه‌ها، تنظیم تشریفات در زمان اجرا که در میانه کار دوباره تنظیم می‌شود، و حافظه پایدار مشخصات با قابلیت جستجوی معنایی

یا دستور جایگزینی محلی — در پشت یک `go get` از طریق — Go این موتور به صورت یک ماژول مصرف می‌شود: سه ستون به علاوه یک مقیاس‌دهنده تشریفات و حافظه API موتور عمداً کوچک مشخصات ثبت می‌شوند، میزان تلاش کار طبقه‌بندی می‌شود، سپس کل جریان اجرا شده و نتیجه‌ای با امتیاز کیفیت دریافت می‌گردد. سطح آن ساده است؛ هماهنگی پشت آن پیچیده است. مانند سایر این موتور تحت رژیم تأیید ضدفریب توسعه یافته است، با یک اجراگر چالش ،Helix اعضای خانواده .درون‌فرآیندی که به جای ماکت‌ها، کد واقعی‌را آزمایش می‌کند

چرا آن‌را ساختیم

توسعه مبتنی بر مشخصات، تست‌محوری دقیق و مدیریت نقاط عطف معمولاً سه روش جداگانه با سه بتواند هر (HelixAgent) AI ساخته شد تا یک عامل HelixSpecifier .ابزار جداگانه هستند سه‌را به صورت یک جریان منسجم و خودمقیاس‌دهنده اجرا کند، به جای آنکه آن‌هارا به صورت دستی .به هم بدوزد

محتوا

چرا یک تغییردهندهٔ بازی است

این سیستم، فرایندرا متناسب با کار تنظیم می‌کند — به‌طور خودکار. تیم‌ها معمولاً در یکی از دو سر طیف نامطلوب گیر می‌کنند: تشریفات سنگین برای همه چیز (امن اما کند و در سکوت مورد نفرت) یا این معامله‌را از بین HelixSpecifier .تشریفات برای هیچ چیز (سریع تا زمانی که دیگر نباشد) ،می‌برد؛ تشریفات‌را متناسب با میزان تلاش طبقه‌بندی‌شدهٔ هر وظیفه اندازه‌گیری می‌کند و در زمان اجرا با آشکار شدن کار، آن‌را دوباره تنظیم می‌کند. قابلیتی که پیش از این عملی نبود، فرایندی است که خود را برای هر وظیفه به‌اندازهٔ مناسب درمی‌آورد — و علاوه بر آن، تصمیمات مشخصات با بحث .چندمرحله‌ای و چندعاملی همراه با امتیازدهی به مواضع، به‌جای حدس اولیه یک عامل واحد

نوآوری‌ها

- سطح فرایند بر اساس معیارهای کیفیت بلادرنگ هدایت می‌شود و در زمان — **تشریفات تطبیقی** .اجرا تنظیم می‌گردد، نه اینکه از پیش ثابت باشد
- دروازه‌ای برای نسبت تست به پیاده‌سازی (حداقل دو برابر)، با الهام — **تست‌محوری نی‌کوییست** از قضیهٔ نمونه‌برداری نی‌کوییست: برای ثبت دقیق رفتار، باید آن‌را بسیار بالاتر از نرخش .نمونه‌برداری کرد؛ بنابراین تست‌ها باید بیش از کدی که پوشش می‌دهند، اندازه‌گیری شوند
- پالایش مشخصات در چند مرحله و با مشارکت چند عامل، که در آن مواضع — **معماری بحث** .پیشنهاد، امتیازدهی و همگرا می‌شوند و جایگزین یک نظر واحد با رویکردی تقابلی می‌گردند
- ،موتور با استخراج جریان‌های انباشته‌شده — **انتقال بین‌پروژه‌ای** و **مشخصات پیش‌بینی‌کننده** مشخصات‌را پیش‌بینی می‌کند و دانش سخت به‌دست‌آمده از یک پروژه‌را به پروژه بعدی منتقل .می‌سازد
- قوانین اجباری پروژه به‌صورت خوانا برای ماشین درآمده و — **به‌مثابه کد Constitution** .توسط موتور اجرا می‌شوند، نه اینکه به هوشیاری بازبینان واگذار گردند

بزرگ‌ترین چالش‌های فنی و راه‌حل‌های ما

- **تلفیق سه روش‌شناسی بدون درگیری آن‌ها با یکدیگر** — SpecKit، Superpowers و GSD هر یک فرض می‌کنند که جریان کار را در اختیار دارند. این مشکل با موتوری تلفیقی حل شد که هر یک از این سه ستون را پشت یک رابط مشترک ثبت می‌کند و آن‌ها را از طریق یک چرخهٔ حیات جریان مشترک هدایت می‌نماید تا به‌جای سه فرایند متضاد، در یک فرایند واحد ترکیب شوند.
- **تعیین میزان فرایند مورد نیاز برای یک وظیفهٔ خاص** — اگر بیش از حد تخمین زده شود، همه چیز کند می‌شود؛ اگر کمتر از حد باشد، کارهای پرخطر بدون بررسی ارسال می‌شوند. این مشکل با یک طبقه‌بندی‌کنندهٔ تلاش حل شد که حجم کار را اندازه‌گیری می‌کند و یک مقیاس‌دهندهٔ تشریفات را تغذیه می‌کند که سطح فرایند را به‌طور پویا در حین اجرا تنظیم می‌نماید.
- **حفظ کیفیت مشخصات بدون نیاز به ناظر انسانی برای هر تصمیم** — این مشکل با جایگزینی مشخصات یک‌باره با پالایش مبتنی بر بحث حل شد؛ در این روش، عوامل در چند مرحله به مواضع رقیب امتیاز می‌دهند و با اجرای نسبت‌های تست‌محوری نی‌کوییست، پیاده‌سازی نمی‌تواند از تست‌های خود پیشی بگیرد.

پشتهٔ فناوری

- **Go** — انتخاب شده تا موتور به‌صورت یک باینری قابل‌واردکردن بدون وابستگی زمان اجرا عرضه شود؛ مدل همزمانی آن است که ارسال وظایف با موازی‌سازی محدود و دورهای بحث چندعاملی را ممکن می‌سازد، نه اینکه به سردرد مدیریت نخ‌ها تبدیل شود.
- **logrus** — ثبت ساختارمند لاگ‌ها در سراسر موتور و هر سه ستون، به‌گونه‌ای که تصمیمات یک جریان (طبقه‌بندی، تغییرات تشریفات، نتایج بحث) پس از اجرا قابل‌خواندن باشند.
- **ستون SpecKit** — فرایند توسعهٔ مشخصات‌محور هفت‌مرحله‌ای (Constitution → مشخص‌سازی → شفاف‌سازی → برنامه‌ریزی → وظایف → تحلیل → پیاده‌سازی)، که ستون فقرات منظم تبدیل یک مشخصه به کد را فراهم می‌آورد.
- **ستون Superpowers** — نظم تست‌محوری با اجرای زیرعامل‌های موازی، که دقت تست‌اول و گسترش آن را تأمین می‌کند تا پیاده‌سازی صادقانه و سریع باقی بماند.
- **ستون GSD** — مدیریت نقاط عطف و چرخهٔ حیات، که به جریان احساس «انجام‌شدن» و پیشروی در مراحل را می‌دهد.
- **مخزن حافظهٔ مشخصات** — فهرستی پایدار و قابل‌جستجوی معنایی از مشخصات گذشته، بستری که مشخصات پیش‌بینی‌کننده و انتقال بین‌پروژه‌ای را ممکن می‌سازد، به‌جای اینکه هر بار از صفر شروع شود.

محتوا

یادداشت‌های وضعیت و صداقت

- **وضعیت: بتا.** HelixAgent از Go به‌عنوان مؤلفهٔ ماژول مورد استفاده قرار گرفته است.
- **مجوز: نامشخص.** هیچ مجوزی از طریق GitHub API اعلام / تأیید نشده — شناسایی نشد.
- نام نمایشی «HelixSpecifier» به مخزن `specifier` نگاشت می‌شود.

سطح اولویت: Helix-اصلی.