Calm Agent：围绕间歇性人类注意力重新构想智能体交互

Dian Jin[1,2],Zixun Fu[3],Guan Wang[3],Yancheng Yuan[3*]

[1]香港理工大学智能可穿戴系统研究所[2]香港理工大学时装及纺织学院[3]香港理工大学应用数学系

arXiv:2611.32009v1 [cs.HC] 24 Nov 2026

摘要

人工智能代理可以在用户专注于其他工作时执行辅助任务。然而，用户仍需了解任务状态、进行调整，并在需要帮助时作出响应。我们认为，针对持续性辅助任务的智能体应围绕间歇性人类注意力进行设计：任务应在交互之间持续执行，同时用户应能够以极少的努力返回任务，了解并控制任务状态。我们将这一立场发展为 Calm Agent。它通过持久化任务表示，连接三种按需交互模式：委派、调整和检查。生成式交互为开放式目标提供适用的控件和反馈，而现有控件则支持常规动作。我们通过 Orvlet 展示这些原则。Orvlet 是一个用于后台执行和紧凑型任务界面的桌面原型。我们指出了若干挑战：在保持连续性的同时调整界面、协调跨任务的注意力、提供响应迅速的交互，以及随着时间推移学习用户习惯。其目标是在保护用户对主要工作注意力的同时，支持对持续任务的控制。

1 引言

人工智能代理目前可以使用工具，并在浏览器和桌面应用程序中完成复杂的多步骤任务[1, 2]。这些能力让用户能够委派辅助任务，同时继续处理主要工作。例如，正在撰写论文的用户可能还希望播放背景音乐，或查看合作者当地的时间。此类任务可以长时间保持运行，尽管用户只需与每项任务进行短暂交互。

委托执行带来了一个重要的交互问题。用户需要打开新的智能体界面，阅读先前的消息以了解当前状态，并用自然语言描述变更需求。这些动作实际上会分散主要工作中有限且宝贵的注意力（图 1）。关于中断的研究 [3, 4]还表明，切换任务会增加额外的努力，并且在注意力回归时需要进行恢复。随着智能体承担

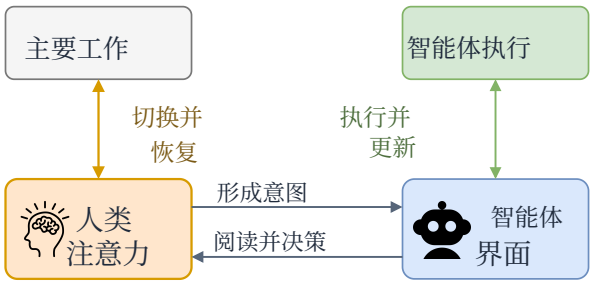


图1：人类注意力瓶颈。主要工作和智能体交互需要同样有限且宝贵的注意力。

更多进行中的任务，用户访问和管理这些任务的方式也成为一个重要的设计考量。

我们的立场是，持续执行辅助任务的智能体应设计为避免占用过多宝贵的人类注意力。用户专注于主要工作时，任务应在委托范围内保持可用并继续执行。当用户回到辅助任务时，智能体的当前状态及其相关控件应可直接访问，而不是通过对话重新构建任务。需要用户输入时，智能体应使这一需求显而易见。该立场关注任务如何在交互之间持续执行，以及用户能否轻松返回任务、了解其状态并对其进行控制。

平静技术 [5, 6] 提供了一个概念基础：技术应根据需要在人类注意力的中心与边缘之间移动。开放式智能体委派提出了进一步的要求：系统需要解读目标，确定用于实现目标的适当动作，从环境中接收反馈，并随着任务的发展，使这些部分与执行保持关联。生成式用户界面 [7–9] 有助于提供特定于任务的界面，而语言和直接操作则提供互补的方式来表达目标并进行精确修改 [10, 11]。这些能力需要支持与持续任务之间连贯的关系。

我们将这一立场发展为 Calm Agent。我们为 Calm Agent 定义了三种交互模式：委派、调整，

*通讯作者：Yancheng Yuan。

和检查。这些模式让用户能够通过语言、直接控制和状态反馈来访问同一任务。共享任务表示将这些交互与后台执行联系起来。该框架指出，当用户专注于其他工作时，哪些任务信息和控件应继续保持可用。我们通过 Orvlet[1]，这一桌面原型来展示该框架，并以其设计为基础提出后续研究问题。本文的贡献包括：

1. 提出一种面向间歇性人类注意力下智能体交互的定位与设计框架，将连续任务执行与用户按需检查和控制任务的能力连接起来。2. 原型 Orvlet 展示了后台执行、生成式任务界面和可复用控件如何支持该框架。3. 分析界面适应与连续性、跨并发任务的注意力请求、响应式生成与执行，以及长期用户适应等方面的开放挑战。

2 相关工作

2.1 平静技术与注意力

Weiser 和 Brown 的平静技术 [5] 描述了注意力在中心与外围之间的移动。外围交互 [12] 和平静设计模式 [6] 探讨了信息和简单动作如何在注意力有限的情况下保持可访问。近期关于环境分析的研究将外围感知与进行更仔细检查的机会联系起来 [13]。

SEEV 模型 [14] 通过显著性、费力程度、预期和价值解释视觉注意力分配。该模型将信息的视觉突出度及关注信息所需的努力，与新信息预期出现的可能性及其与任务的相关性联系起来。这些因素为访问后台任务提供了有益的视角。

以往研究说明了交互如何围绕主要活动展开。然而，开放式委派带来了一个问题：外围界面应当包含什么内容。相关状态和动作取决于委派的目标，并且可能在执行过程中发生变化。因此，智能体界面必须将外围呈现与目标及可用操作联系起来。Calm Agent 通过围绕一项可修订、可控制、可检查的持续任务来组织外围访问，解决了这一关联问题。

2.2 生成式与混合式界面

生成式用户界面 [7, 8] 利用语言模型为用户请求生成界面。DynaVis [15] 生成更小的组件，用于可视化编辑。Jelly [9] 采用一种不断演进的任务驱动数据模型来生成界面，并将自然语言和直接操作转化为模型变更。其基于模型的方法展示了如何围绕持久结构组织界面适应。

直接操作和混合主动交互为结合图形操作与语言提供了更广泛的基础 [16, 17]。研究 [11] 指出了通过提示词表达意图的困难，而 DirectGPT [10] 展示了直接操作如何减少交互时间和撰写提示词的精力。

这些方法阐明了用户如何创建和更改任务界面。对于一项正在进行的辅助任务，界面还必须支持用户短暂查看，同时让任务在其他地方继续执行。生成式界面可能符合目标，但仍可能要求用户重新打开工作区、查找已更改的控件并等待重新生成，这会妨碍其实用性。设计上的空白在于如何确保用户每次查看时都能连续访问：哪些控件应当保留，哪些更改需要生成，以及显示的状态如何与执行保持关联。Calm Agent 通过“委派”“调整”和“检查”围绕同一项进行中的任务来组织这种关系。

2.3 智能体反馈与控制

图形用户界面智能体引发了关于人们在自动执行期间如何保持意识和控制的问题 [18]。AgentLens [19] 将后台移动端执行与选择性叠加层相结合，并在需要交互时，在完整、部分和生成式界面之间切换。它提供了一个尽量减少干扰、适时沟通的重要范例。然而，在干预点选择合适的呈现方式，还留下了一个设计问题：用户应如何在干预间隙访问同一任务，包括由用户主动发起检查或调整时？Calm Agent 通过一种任务表示，将目标修订、直接控制和状态检查联系起来，明确支持这种持续访问。

主动式编程辅助研究考察了有用支持与打断之间的权衡[20]。监听型智能体的相关研究探讨智能体如何获取上下文并决定何时介入 [21]。对于明确委派的辅助任务，介入还必须与用户现有目标保持关联。澄清问题、请求用户决策以及未经请求的建议，对注意力提出了不同要求。Calm Agent 将必要的注意力请求关联到同一任务，用户也可以主动返回该任务。

这些研究领域共同促成了一个贯穿任务生命周期、连接开放式委派、持久控制和选择性注意力请求的框架。Calm Agent 针对用户主要忙于其他事务的时段，构建了这一框架。其重点在于持续执行与间歇性参与之间的关系，其中既包括用户发起的访问，也包括代理发起的请求。

[1] https://github.com/CLaSLoVe/Orvlet

3 CalmAgent 的立场

一项任务可能运行数分钟或数小时，而用户每次只投入几秒钟的注意力。我们认为，智能体界面应支持这种持续时间上的差异。用户既需要能够离开任务，也需要一种有效的返回方式。基于这一立场，我们提出了三项设计原则，随后介绍支持这些原则的交互模式和任务表示。

3.1 设计原则

P1：在交互之间维持任务。无论界面是否打开，任务执行和上下文都应独立持续。在委托范围内，智能体应维护目标、更新可观测状态，并记录与后续交互相关的变化。如果进展需要用户输入，则应保留任务并明确显示这一需求。用户不应为了让任务保持可用而必须持续进行对话。

P2：支持简短且有目的的交互。交互方式应符合用户想要完成的事情。语言支持开放式目标；可用控件支持特定调整；状态显示支持检查。生成式交互应为任务选择有用的信息和动作。一旦控件可用，日常使用就应保留其含义，并避免不必要的界面重新生成。

P3：让用户返回时能够获取信息并采取行动。用户返回时，应能够了解当前情况，并确定下一步可以做什么。界面应显示观测状态、可用操作以及任何需要输入的事项。界面的突出程度应与注意力需求相匹配。紧凑的显示可以支持这一原则，但其价值取决于用户能否以较少的努力理解任务并采取行动。

这些原则共同为评估设计设定了具体标准：用户离开期间，任务应保持可用；无需查看完整历史记录即可清晰了解其状态；应直接支持例行调整；并且应及时注意到必要的干预。第5 节将探讨满足这些要求时面临的开放挑战。

3.2 三种按需交互模式

委派、调整和检查描述了用户与进行中的任务交互时的目的（图2）。初始委派用于建立任务。之后的交互可根据用户的需求采用任意模式，顺序不限。

委派：表达或修改目标。用户使用语言表达意图，而可用控件不进行捕获。智能体解读目标或修订后的约束并确定所需的动作。修订可以在改变任务目标的同时，继续执行现有任务。

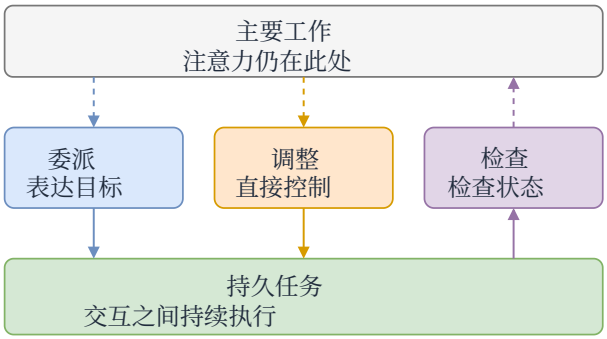


图2：与进行中的任务进行按需交互。委派表达目标，调整更改任务参数，而检查显示任务状态。

调整：使用可用控件。用户通过直接操作调用动作或更改参数。控件的含义由当前任务定义，例如暂停播放或更改计时器值。调整指的是用户动作；自动重试和运行时更新属于后台执行。

检查：检查当前状态。用户阅读任务反馈，但不发出新指令。简要视图可能足以确认任务正在运行、等待输入或已完成。当用户需要了解变化或决定如何响应时，应提供更多详细信息。

智能体发起的关注与澄清。三种模式描述用户的操作；关注请求描述交互如何开始。当缺少必要信息或任务需要作出决策时，智能体可能会重新请求参与。随后，用户可以通过检查查看请求，通过委派澄清目标，或通过调整使用可用控件作出响应。因此，智能体发起的关注连接到全部三种模式，而不是形成第四种模式。澄清应询问继续执行所需的信息，并保持与现有任务的关联。如果决策可以稍后再作出，任务应保留该请求，直到用户返回。这些请求的时机和醒目程度属于设计选择，将在第5.2节讨论。

3.3 将交互与执行联系起来

图 3 将交互模式与任务环境连接起来。用户仍处于主要工作区中。生成式交互层会根据解释后的目标选择控件和反馈，而智能体则通过网页、本地应用程序或系统工具执行操作。持久化任务表示将这些活动连接起来。

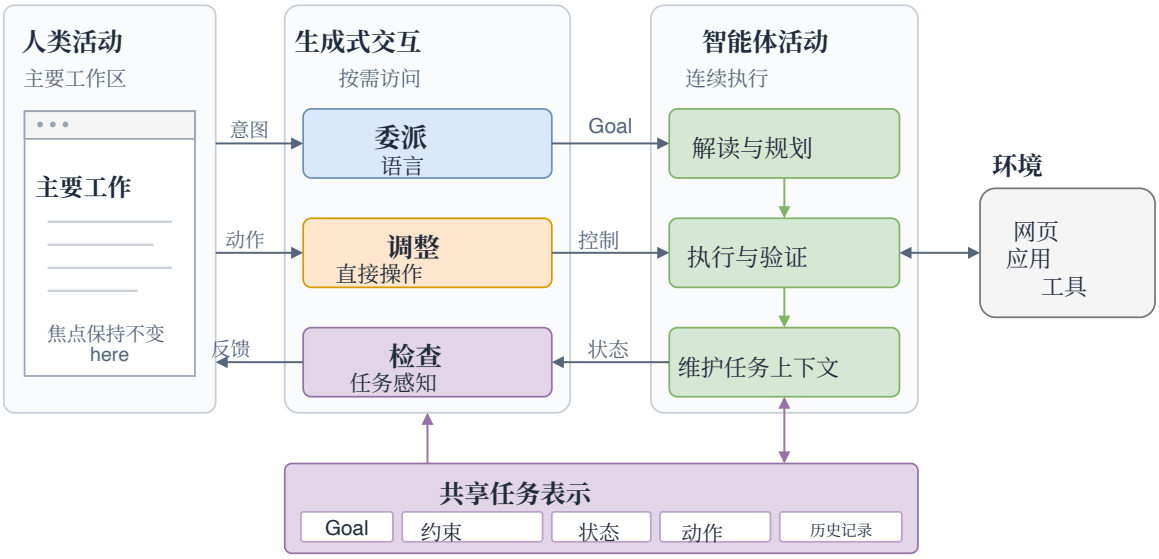


图3：Calm Agent 的概念架构。共享任务表示使界面与执行保持连接，同时用户可以专注于主要工作。

我们将每项任务描述为

$$T_i = \langle \mathcal{G}, \mathcal{C}, \mathcal{S}, \mathcal{A}, \mathcal{H} \rangle, \tag{1}$$

其中，$\mathcal{G}$ 是目标，$\mathcal{C}$ 包含用户约束和环境约束，$\mathcal{S}$ 存储可观测状态，$\mathcal{A}$ 包含可用动作及其处理程序，$\mathcal{H}$ 记录执行历史。委派可以修改目标和约束；调整会调用一个动作；检查会读取状态。执行可以在没有新的用户请求的情况下更新状态和历史记录。

这种表示方式区分了需要进行界面适应的变化，以及可以使用现有控件的变化。当修改后的目标引入新的交互需求时，生成式交互可以提供合适的控件和反馈。当用户调用可用动作时，系统可以复用其绑定。运行时更新可能只需要更改显示值。设计原则是：在交互需求发生变化时适应界面，同时在日常使用过程中保持熟悉的控件稳定。

例如，将背景音乐切换为器乐爵士乐会改变选择标准，但不一定会改变对播放控件的需求。智能体可以解释修改后的偏好，同时保留这些控件的含义和位置。随后，曲目变更可能只需更新显示标题，而无需改变布局。对于间歇性交互而言，这种连续性很重要：用户应能够重新使用他们在之前访问时学会的操作方式。如果某个动作不再可用，界面应明确显示这一变化，而不是悄悄地赋予同一控件不同的含义。

随着环境变化，这种关联也必须保持其意义。显示控件应对应当前可用的动作，而反馈应反映观察到的执行情况。如果智能体无法确定当前状态，界面应将这种不确定性清晰地呈现出来。这些要求有助于用户返回并继续处理任务。

4 Orvlet 原型

Orvlet 在桌面上具体展示了 Calm Agent 原则。它结合了隔离的后台浏览器、紧凑型任务界面以及存储的任务上下文。我们通过其设计和若干已实现的场景，展示这些原则如何塑造交互。

图 4 展示了作者在编辑手稿时如何访问 Orvlet 的输入栏和音乐控件。

4.1 后台任务执行

Orvlet 在后台操作相关网页或工具。对于媒体请求，它会执行网页上的动作，并在呈现结果前检查播放情况。原始服务提供任务能力，而紧凑型界面则提供对选定控件和状态的访问。折叠此界面后，后台活动仍会继续运行，从而支持 P1。

图 5 展示了执行路径。新目标可能需要规划；现有控件则会调用其动作绑定。结果检查和运行时更新会反馈到任务上下文中，为后续请求和可见界面提供支持。实现还可选择使用 Jev [22]，以便快速将请求路由至可用能力。

4.2 快速访问与任务返回

每个活动任务都会以一个小型胶囊标签的形式显示在工作区边缘，其中包含标签、状态和主要操作（图 6e）。当界面折叠且不需要用户进行交互或关注反馈时，

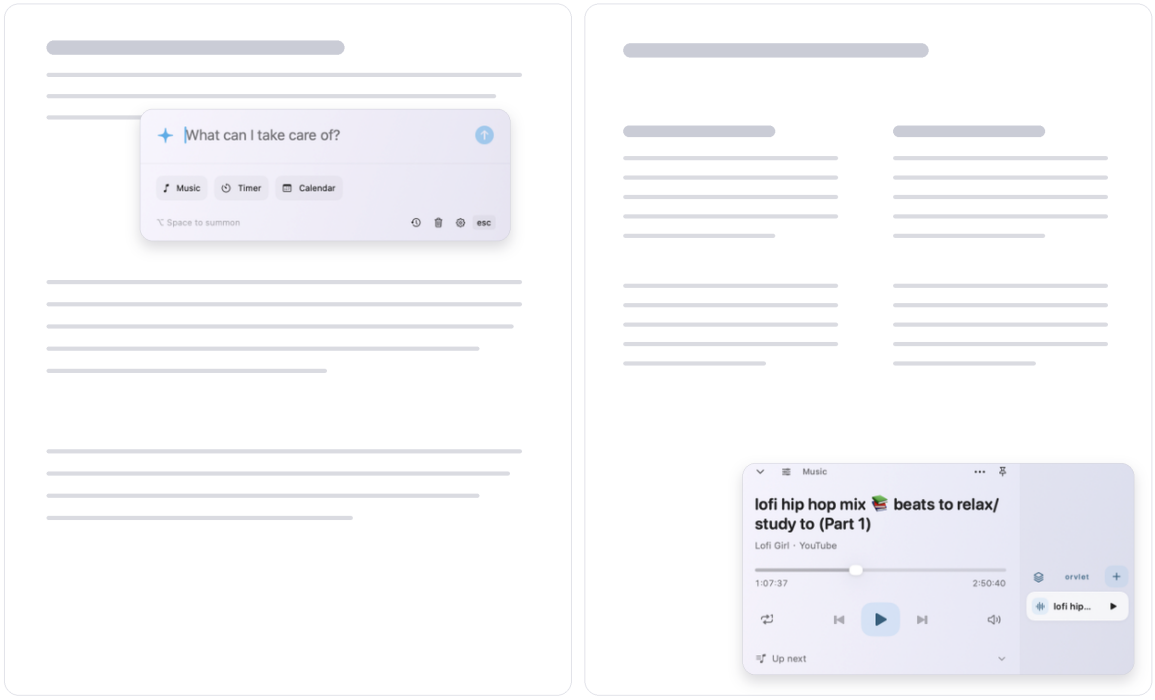


图 4：手稿编辑期间的 Orvlet。浮动输入栏允许作者委派任务，右下方展开的音乐卡片提供播放控件。折叠后，卡片会变成工作区边缘处的一个小型胶囊标签。闲置三秒后，胶囊标签会淡化为半透明状态，同时音乐任务继续在后台运行。

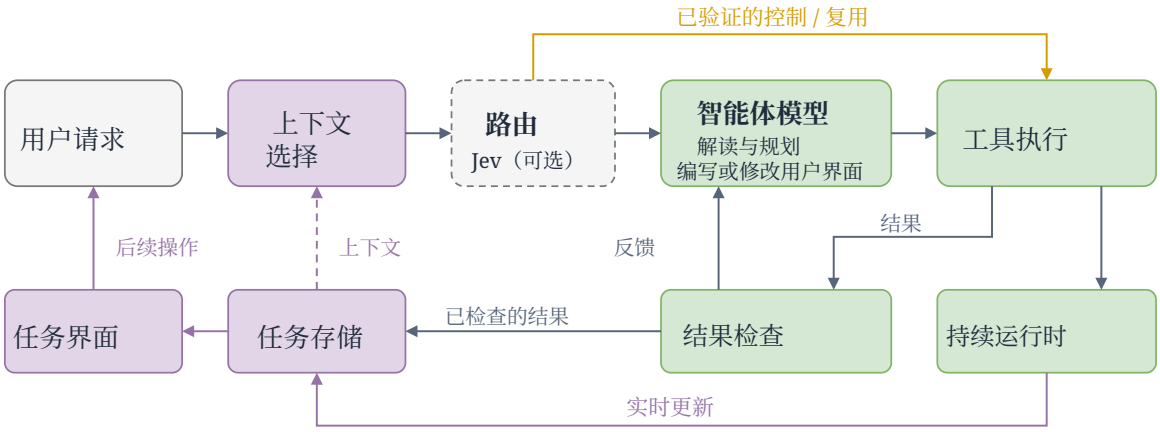


图 5：Orvlet 中的任务执行与交互。请求使用当前任务上下文，已验证控件可以直接重复使用。结果检查提供反馈并更新任务状态。持续运行时还会提供实时更新。

这些胶囊标签会在三秒后淡化为半透明状态。这会降低其视觉突出度，减少对主要工作的干扰。当指针进入或交互恢复时，它们会恢复为完全不透明。将鼠标悬停在胶囊标签上或将其打开会显示一张紧凑卡片。这提供了不同层级的访问方式：用户可以查看状态、执行主要操作，或根据当前需要打开附加控件。

系统会为任务生成信息与可复用控件的组合。对于 P2，用户可以通过现有控件暂停播放，或从任务界面读取倒计时，而无需再次编写请求或等待新的界面。这些动作使用当前任务的控件和显示值。对于 P3，

播放状态和剩余计时器值会在用户返回时提供具体信息，同时还会提供用于继续交互的控件。任务历史视图则为访问现有任务提供了另一个入口（图 6f）。

当缺少必要的用户信息时，Orvlet 还支持澄清。智能体可以提出简洁的后续问题，并将请求标记为需要输入。问题和回复会与同一任务保持关联。当对话未打开时，问题会以反馈的形式显示在工作区边缘，而不会自动将焦点置于输入字段。反馈会一直保留，直到被阅读或关闭。这通过展示用户需要提供哪些信息才能继续任务，进一步支持了 P3。

4.3 场景

控制音乐。用户请求播放轻柔的背景音乐，然后继续写作。Orvlet 在预定义的音乐卡片中开始播放，并提供播放和音量控制（图 6a）。

监控并发的时间相关任务。用户正在协调北京和伦敦之间的工作，请求一个双时区时钟和休息倒计时。Orvlet 在卡片中展示详细的时区视图，同时将时区摘要和倒计时以胶囊标签的形式堆叠在主要工作旁边（图6b）。

检查并刷新代码仓库的星标数。用户请求获取某个 GitHub 仓库的星标数。Orvlet 会呈现星标数、代码仓库标识符、数据源和刷新控件，并配有摘要胶囊标签（图 6c）。

比较股价。用户请求比较 Apple 和 Tesla 的股价。Orvlet 会在同一张卡片中呈现两者的股价、变动情况和价差，同时显示交易时段、数据源和摘要胶囊标签（图 6d）。

4.4 交互启示

表1对正在进行的辅助任务所对应的典型交互配置进行了对比。该比较指出了用户访问控件和恢复任务上下文的位置。这些配置可以组合使用；例如，对话式界面也可以提供持久的任务控件。

设计问题在于，一次短暂访问需要完成多少工作：定位任务、了解其状态、选择一个动作，然后返回主要工作。Orvlet 将这些活动整合到一个持久的访问入口中。以下挑战涉及如何将这一方法扩展到不断变化的任务、执行条件和用户习惯中。

5 挑战

Calm Agent 将相互竞争的需求置于同一个用户界面上：它必须简洁而信息丰富、可适应而又熟悉，在持续执行期间保持响应，并能够请求输入，而不会反复打断主要工作。为满足这些要求，我们确定了四项开放挑战。

5.1 具备交互连续性的自适应界面

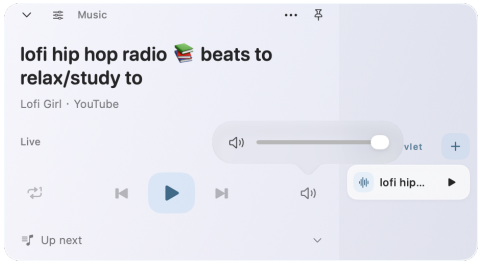
紧凑型界面为包含众多可能状态与动作的开放式任务提供的空间有限。只展示常用控件可能会隐藏罕见但必要的决策，而展示所有选项则会增加用户短暂访问时的操作负担。不断演进的任务驱动型界面为适应提供了基础 [9], ，但要支持用户间歇性地返回，就必须决定哪些元素应该适应变化，哪些应保持稳定。

随着用户意图和执行环境不断变化，系统必须判断某项修订改变的是动作的含义、目标，还是仅仅改变了一个参数。系统既要避免控件外观不变、实际效果却已改变，也要避免不必要的重新设计，以免失去熟悉感。渐进式披露可以让界面保持紧凑，但对于未曾看到状态变化的用户而言，受阻任务或不可用动作等关键状态区分必须始终清晰可见。

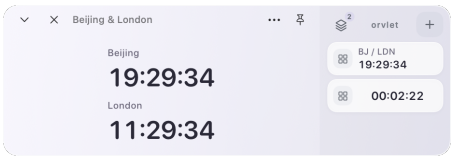
5.2 并发任务中的注意力请求

智能体必须决定是否需要人工输入，以及应以多醒目的方式发出请求。对每一处歧义都征求意见，会把解释工作推回给用户；而基于无依据的假设采取行动，则可能导致用户不希望看到的结果。如何区分会改变智能体动作的缺失信息与可在委托范围内解决的选择，仍有待探讨。即使确实需要输入，请求的时机和醒目程度也应考虑延迟成本及用户当前正在进行的活动 [4, 18]：可以延后的问题可保持低调但易于访问，而时间敏感型决策则需要更强烈的提示。

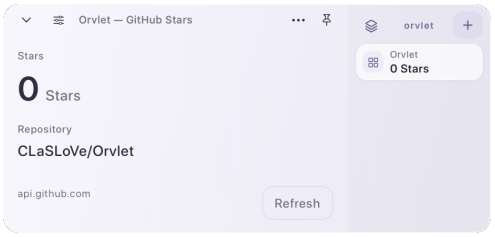
并发会使跨任务的问题更加复杂。单独来看合理的请求可能会同时到达，或接连打断主要工作；Orvlet 中的堆叠式胶囊控件具体体现了这种张力。将请求分组可以降低中断频率，但可能延误紧急决策；而优先处理某一任务，又容易让其他任务被忽视。在不掩盖尚未解决的决策的情况下，协调并发任务之间的时机和显著性，仍是一项关键挑战。



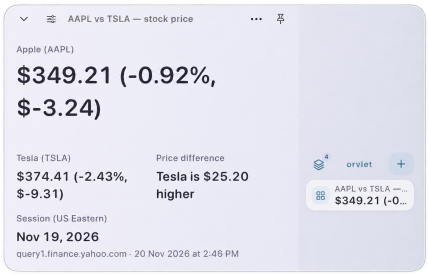
（a）音乐播放控件



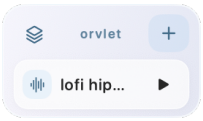
（b）世界时钟和倒计时



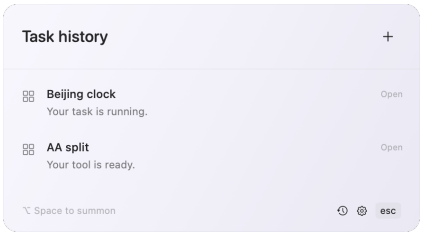
（c）GitHub 星标数



（d）股票价格比较



（e）折叠边缘胶囊



（f）任务历史

图 6：Orvlet 中的任务界面。(a) 带有播放和音量控制的音乐卡片。(b) 双时区时钟卡片。(c) 用于检查 GitHub 仓库星标数的卡片。(d) 用于比较 Apple 和 Tesla 股价的卡片。(e) 折叠边缘胶囊。(f) 用于返回进行中的任务的任务历史视图。

表 1：进行中的辅助任务的代表性交互配置。

交互需求	应用控制	聊天机器人	生成的任务 工作区	Calm Agent
建立任务	导航至相关 功能	描述目标	描述一个目标并 生成其界面	描述一个用于 后台执行的目标
例行 调整	使用应用控制	编写请求并阅读 回复	使用生成的控件	在一个 胶囊标签或卡片
检查状态	打开相关的 应用视图	阅读一条更新或 请求一条	检查任务 工作区	阅读任务指示器或 展开其卡片
在离开后返回	定位相关视图	从以下内容恢复上下文	重新查看生成的	访问持续进行的
缺席	以及状态	消息或摘要	工作区	任务状态和控件

5.3 响应式界面生成与执行

如果界面生成只有在规划和执行完成后才开始，一次简短的交互就会变成长时间等待。并行准备界面及其底层行为，例如在智能体解析数据源的同时起草一张卡片，可以降低延迟，但这些过程共享依赖关系：有效操作和值通常仍然在执行产生证据之前仍然未知。

分阶段界面可以及早呈现已接收的意图和已知信息，并在控件的后台行为准备就绪后启用相应控件。这样做需要明确区分待处理操作与就绪控件，并在用户于执行过程中修改目标时，协调部分结果。若缺乏谨慎的协调，异步更新可能会导致布局偏移或无效操作，恰好发生在用户尝试进行交互之时。

5.4 随时间理解用户并适应用户

反复使用可以让智能体用更少的解释来理解请求，并根据用户习惯定制界面，例如偏好的术语、常用控件或详细程度。然而，简短的交互会提供不完整且含糊的证据：某个控件可能只是因为易于触达而被频繁使用，而被忽略的通知也不一定表示用户偏好保持安静。

因此，长期适应必须将明确偏好与推断出的习惯区分开来，并考虑任务语境。在专注工作期间偏好简洁显示的用户，在详细探索任务时可能并不希望如此；明确的更正应当修正当前假设，但不应抹去在其他场景中仍然有用的习惯。如何决定哪些已学习的偏好可以迁移到新请求，同时跟踪这些偏好的置信度，并确保假设易于更正，仍然是一个开放问题。

6 结论

本文提出了 Calm Agent，这是一种面向间歇性人类注意力下持续运行的辅助任务的设计框架。其贡献在于将任务连续性与三种按需交互模式连接起来：委派、调整和检查。持久化任务表示将这些交互模式与执行关联起来，而代理发起的请求则会在需要澄清或决策时让用户重新参与其中。Orvlet 原型通过后台执行、紧凑的任务控件以及与任务关联的澄清，展示了这一框架。我们确定了四项开放挑战：界面适应与连续性、跨并发任务的注意力请求、响应式生成与执行，以及长期用户适应。总体而言，这些贡献围绕简短且有目的的参与来构建智能体交互，同时保护用户用于主要工作的注意力。

参考文献

[1] Shunyu Yao、Jeffrey Zhao、Dian Yu、Nan Du、Izhak Shafran、Karthik Narasimhan 和 Yuan Cao。React：协同语言模型中的推理与行动。*arXiv preprint arXiv:2210.03629* 2022，。[2] Boyuan Zheng、Boyu Gou、Jihyung Kil、Huan Sun 和 Yu Su。GPT-4 *arXiv* V（ision）*preprint arXiv:2401.01614* 2024若有依据，便是通用型网页智能体。，。[3] Gloria Mark、Daniela Gudith 和 Ulrich Klocke。工作被打断的代价：速度更快，压力更大。载于 *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*，第107–110页，2008年。[4] Shamsi T Iqbal 和 Eric Horvitz。计算任务的中断与恢复：实地研究、分析及方向。载于 *Proceedings of the SIGCHI Conference on Human factors in computing systems* 677 686, 2007，第–页。[5] Mark Weiser、John Seely Brown 等。设计平静技术。*PowerGrid Journal*，1(1):75–85，1996年。[6] Amber Case。*Calm technology: principles and patterns for non-intrusive design* 2015。” O’Reilly Media, Inc.”，。[7] Yaniv Leviathan、Dani Valevski、Matan Kalman、Danny Lumen、Eyal Segalis、Eyal Molad、Shlomi Pasternak、Vishnu Natchu、Valerie Nygaard、James Manyika 等。生成式用户界面：大语言模型是有效的用户界面生成器。*arXiv preprint arXiv:2604.09577* 2026，。[8] Jiaqi Chen、Yanzhe Zhang、Yutong Zhang、Yijia Shao 和 Diyi Yang。面向语言模型的生成式界面。载于 *Findings of the Association for Computational Linguistics: ACL 2026*，第1499–1519页，2026年。[9] Yining Cao、Peiling Jiang 和 Haijun Xia。借助生成且不断演进的任务驱动数据模型，实现生成式、可塑的用户界面。载于*Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* 1 20, 2025，第–页。[10] Damien Masson、Sylvain Malacria、G´ery Casiez 和 Daniel Vogel。DirectGPT：用于与大语言模型交互的直接操纵界面。载于 *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*，第1–16页，2024年。[11] Hari Subramonyam、Roy Pea、Christopher Pondoc、Maneesh Agrawala 和 Colleen Seifert。弥合构想鸿沟：基于提示与大语言模型交互所面临的认知挑战。载于 *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* 1 19, 2024，第–页。[12] Saskia Bakker、Elise Van Den Hoven 和 Berry Eggen。外围交互：特征与考量。 *Personaland ubiquitous computing*，19(1):239–254，2015年。[13] Sebastian Hubenschmid、Arvind Srinivasan、Niklas Elmqvist、Dieter Schmalstieg 和 Michael Sedlmair。环境分析：用于沉浸式可视化与意义建构的平静技术。*IEEE Computer Graphics and Applications* 46 3 110 117, , ( ): –。2026[14] Kelly S Steelman、Jason S McCarley 和 Christopher D Wickens。视觉工作空间中基于理论的注意力模型。*International Journal of Human–Computer Interaction*，33(1):35–43，2017年。[15] Priyan Vaithilingam、Elena L Glassman、Jeevana Priya Inala 和 Chenglong Wang。DynaVis：用于可视化编辑的动态合成用户界面控件。载于 *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*，第1–17页，2024年。[16] Ben Shneiderman。直接操作：超越编程语言的一步。*Computer*，16(08):57–69，1983年。[17] Eric Horvitz。混合主动式用户界面的原则。载于 *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* 159 166, 1999，第–页。[18] Raja Parasuraman、Thomas B Sheridan 和 Christopher D Wickens。人与自动化交互的类型与层级模型。*IEEE Transactions on systems man and cybernetics-Part A: Systems and Humans* 30 3 286 297, 2000, , , ( ): –。[19] Jeonghyeon Kim 、 Byeongjun Joung 、 Junwon Lee 、 Joohyung Lee、Taehoon Min 和 Sunjae Lee。AgentLens：移动端图形用户界面智能体中面向人机交互的自适应视觉模态。载于 *Proceedings of the 39th Annual ACM Symposium on User Interface Software and Technology* 1 18, 2026，第–页。[20] Kevin Pu、Daniel Lazaro、Ian Arawjo、Haijun Xia、Ziang Xiao、Tovi Grossman 和 Yan Chen。协助还是干扰？探索并评估主动式人工智能编程支持的设计与权衡。载于 *Proceedings of the 2025 CHI conference on human factors in computing systems*，第1–21页，2025年。[21] Andrew Zhu 和 Chris Callison-Burch。旁听大语言模型智能体：综述、分类体系与路线图。*arXiv preprint arXiv:2509.16325* 2025，。[22] Feiyu Duan、Jiayu Lin、Jia Wang、Jun Xiang、Jialiang Wu、Xinnong Zhang、Hanqi Yan、Siyuan Wang 和 Zhongyu Wei。通用决策模型：基准测试与超越 Jev 的见解。 *arXiv preprint arXiv:2610.03935*，2026年。