Гимназија „Бора Станковић"

Врање

МАТУРСКИ РАД

Предмет:

Рачунарски системи

Тема:

Стандардизација терминологије десктоп окружења и предлог нове архитектуре менаџера прозора

Ментор:

Јасмина Младеновић

Ученик:

Лазар Алексић

Врање, мај 2026.

Стандардизација терминологије десктоп окружења и предлог нове архитектуре менаџера прозора

Садржај

1. Увод 3

2. Стандардизација терминологије и архитектура модерних графичких окружења 5

2. Когнитивни темељи просторног управљања 5

2.1. Историјска еволуција менаџера прозора 6

2.2 Графички стекови по платформама 9

2.3. Архитектонска ограничења комерцијалних система (Windows и macOS) 10

2.4. Преглед сродних решења 11

2.5. Унификована терминолошка мапа 11

2.6. Званична терминологија предложене 2D архитектуре 12

2.7. Формални координатни модел 13

3. Проблем постојећих модела и ограничења хардвера 14

3.0. Квантификација продуктивносног трошка 14

3.1. Проблем простора и 1D ограничења 15

3.2. Хоризонтални парадокс и избор монитора 15

3.3. Анализа корисничких профила 16

4. Предложени модел: 2D Картезијански простор и двослојна бесконачност . . . 18

4.1. Декларативно инстанцирање и модел власништва 18

4.1.1. Динамички мозаички распоред и просторна континуираност . . . 19

4.2. Навигација, фокус и организација простора 20

4.3. Viewport Culling Протокол и управљање ресурсима 21

4.3.1. Алгоритам праћења оштећења приказа и просторно хеширање . . 22
4.4. Архитектура подршке за више монитора 23
4.5. Хијерархија семантичких Зона 23
4.6. Интеграција гестова на платну 24
4.7. Логика уноса и механика пречица 24
4.8. Серијализација стања система 24
4.9. Теоријска анализа сложености 25
4.10. Приступачност (Accessibility) и агентни радни токови 25
4.11. Толеранција на грешке и критични случајеви 26
5. Међуплатформски слој апстракције, `libkrozor` 27
5.1. Архитектонска улога и преносивост 27
5.2. Унификовани Ц интерфејс и системске примитиве 28
5.3. Платформски backend-и матрица деградације 28
5.4. Конфигурација и IPC комуникација 29
5.4.1. Јединствена конфигурација 29
5.4.2. IPC Протокол (Канал проширивости) 30
5.5. Стандарди и унификација протокола 30
6. Упоредна анализа и евалуација 32
6.1. Ограничења предложене архитектуре и правци развоја 32
6.2. Нацрт методологије за емпиријску евалуацију 33
6.3. Агентни системи као непредвиђена архитектонска предност 34
7. Закључак 35
8. Литература 37

1. Увод

Између 2010. и 2026. године, тржиште монитора је прошло кроз три дистинктне фазе хардверске револуције: стандардизација формата 16:9 и резолуције 1080p, пролиферација ултра-високих резолуција до 4K, и тренутна активна фаза, масивна адопција ултра-широких формата 21:9 и 32:9 са често закривљеним панелима и фреквенцијом освежавања од 120 и више херца.

Паралелно са еволуцијом монитора, архитектуре кључних рачунарских компоненти су доживеле трансформацију која је још драматичнија. Савремене потрошачке графичке картице поседују капацитете графичке меморије (VRAM) од 8 до 24 гигабајта, и могу истовремено рендеровати стотине независних површина. Са друге стране, процесори од 16 до 32 језгра доступни су и у средњекласном сегменту тржишта.

Овај енормни раст капацитета паралелног рачунарства директно је утицао на радне навике професионалаца. Програмери, инжењери и стручњаци за сајбер-безбедност сада имају хардверску моћ да истовремено покрећу и прате десетине виртуелних машина, контејнера, развојних окружења и алата за анализу мрежа. Последично, потреба за менаџментом и визуелизацијом великог броја прозора у различитим контекстима експоненцијално је порасла. Модеран хардвер са лакоћом обрађује овај паралелизам, али софтвер који њиме визуелно управља постаје најслабији део система.

Упркос овом напретку, софтверске парадигме које управљају графичким интерфејсима остале су концептуално укорењене у епохи првих графичких радних станица касних 1980-их, и то на свим главним платформама подједнако. Windows, macOS, Linux и BSD менаџери прозора деле исте фундаменталне претпоставке: екран је апсолутни радни простор, прозори постоје унутар фиксних граница монитора, а навигација између контекста врши се пребацивањем између изолованих виртуелних простора. Овај јаз између могућности хардвера и експресивности софтвера централни је проблем којим се овај рад бави.

Поред проблема простора, постоји и проблем когнитивног оптерећења при промени контекста (*context-switching*). Истраживања продуктивности у области когнитивне пси-

хологије показују да сваки недобровољни прекид радног контекста захтева просечно 23 минута за потпуни повратак на претходни ниво дубоке концентрације (Марк ет ал., 2008). Системи који приморавају корисника да ментално мапира процесе на нумерисане тагове или апстрактне виртуелне десктопе директно повећавају учесталост оваквих прекида.

Практични случајеви употребе који мотивишу ову архитектуру укључују три дистинктна профила: програмера који жели стабилан детерминистички распоред прозора по контекстима задатка (едитор, терминали, документација, база); системског администратора који организује прозоре по идентитету сервера уместо по типу; и истраживача који паралелно прати вишеструке алате без ризика нежељене суспензије активних скенера. Исти архитектонски принципи идентификовани су као природна платформа за аутономне софтверске агенте који управљају десктоп апликацијама.

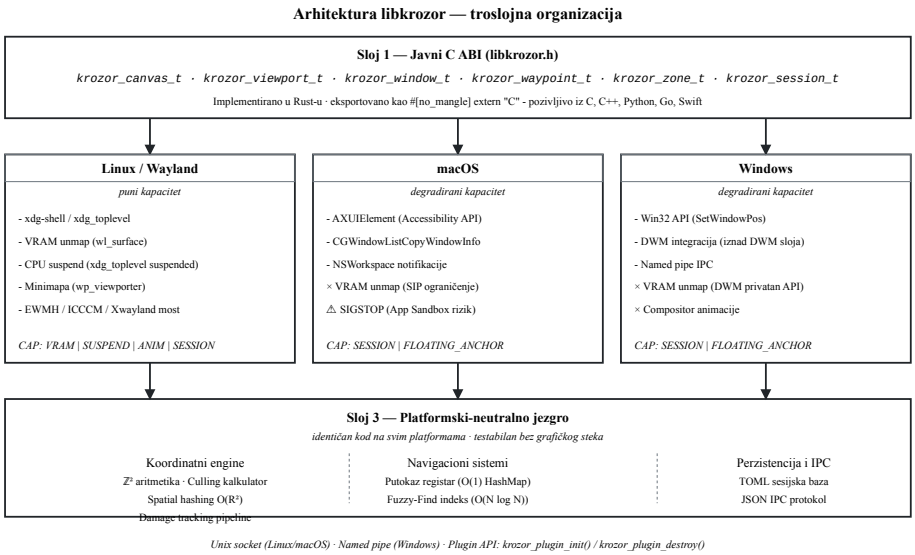
Циљ овог рада је четворострук. Успостављање стандардизоване терминологије која обједињује фрагментисани речник постојећих система на свим платформама; предлог потпуно нове системске архитектуре, 2D Картезијанског менаџера прозора са бесконачним платном; детаљна спецификација унутрашњег функционисања те архитектуре кроз формални координатни модел, *Viewport Culling* протокол и *damage-based render pipeline*; и спецификација међуплатформског слоја компатибилности (`libkrozor`) који чини ову архитектуру преносивом на X11/Wayland (за Linux и BSD), macOS и Windows кроз заједнички C ABI и јасно дефинисану матрицу постепене деградације функционалности (*graceful degradation*).

Централно питање овог рада је: да ли је могуће пројектовати архитектуру менаџера прозора која симултано решава проблем просторног ограничења, когнитивног оптерећења и алокације ресурса, и која се може имплементирати на свим главним оперативним системима кроз заједнички појмовни и технички оквир?

2. Стандардизација терминологије и архитектура модерних графичких окружења

Најпре ћемо саставити заједнички речник. Кроз историју, разни менаџери прозора и оперативни системи су користили различите називе за идентичне концепте. Ово чини међусобна поређења отежаним и често конфузним, што решавамо успостављањем фиксних и дефинитивних појмова преносивих између платформи и екосистема.

Ради бољег дефинисања обима овог рада, прво морамо погледати визуализацију графичког стека.



Слика 1: Хијерархија графичког стека; имплементација сваког слоја се разликује по платформи, а примарни фокус овог рада је слој 4.

Напоменућемо да у модерним системима граница између слоја 4 и слоја 3 често не постоји, зато што су ова два слоја блиско интегрисана. На Wayland-у (Linux, BSD) сваки менаџер прозора је практично приморан да буде уједно и свој композитор. На Windows-у `DWM.exe` такође преузима обе улоге, као и на macOS-у `Quartz`. Ово поседује релевантност за предложени модел.

2. Когнитивни темељи просторног управљања

Пре анализе техничких решења, изузетно је битно разумети зашто просторно управљање прозорима има инхерентну когнитивну предност над апстракцијама.

Просторна меморија и супериорност локацијске енумерације Истраживања когнитивне психологије конзистентно показују да људи навигирају физичким простором са драматично мањим когнитивним напором него што користе апстрактне категорије. Техника познатија као "меморијски дворац", документована још у античкој реторици и потврђена је савременим студијама, демонстрира да је просторно енкодирање информација ефикаснији и ергономичнији меморијски систем од категоријално енкодованих информација.

Просторна навигација је примарно домен хипокампуса и енториналне коре, за које такође знамо да су еволуирали за брзу и ефикасну обраду локацијских информација. Можемо закључити да је систем који од корисника захтева да запамти *gde* се нешто налази далеко природнији од система који захтева апстракције и категорисање. Картезијанско платно је директна примена ових принципа.

Милеров закон и ограничења радне меморије Милер (1956) је експериментално утврдио да је капацитет људске радне меморије ограничен на 7 ± 2 истовремених ставки. Системи који користе нумерисане тагове (`dwm` на Linux/BSD-у) или виртуелне десктопе (Windows, macOS), директно оптерећују радну меморију зато што корисник мора активно памтити који су прозори тачно на ком виртуелном десктопу или тагу. За N простора од којих сваки садржи M прозора, корисник мора да задржи $N \times M$ асоцијација у радној меморији. Са друге стране, просторна позиција се енкодира имплицитно и без свесног напора.

Фитсов закон и $O(1)$ навигација Фитс (1954) је формализовао временски утрошак доласка до циља као $T = a + b \log_2\left(\frac{2D}{W}\right)$, где је D удаљеност до циља, а W величина циља. Класични Alt+Tab (или Cmd+Tab) механизми крше овај закон због захтева N узастопних уноса тастера за долазак до N-тог прозора. Систем Путоказа предложен у овом раду би колапсирао D на нулу, што би довело временску комплексност овог учесталог задатка са $O(N)$ на $O(1)$ у свим случајевима.

2.1. Историјска еволуција менаџера прозора

Да би се разумела дубина проблема који овај рад обухвата, корисно је имати предзнање о еволуцији менаџера прозора од њихових почетака. Свака генерација система увела је нове напретке који су решавали ограничења претходника, али најчешће у само једном смеру.

Генерација 0: Рани GUI Xerox Alto (1973) увео је графички кориснички интерфејс, са концептима прозора и њиховог преклапања, и практично је постао стандард за све што је уследило у наредном низу година. Xerox Star (1981) и Apple Lisa (1983) су све више популаризовали преклапајући модел са слободним позиционирањем, поставивши шаблон који ће доминирати наредне четири деценије, и усмеривши сав развој у том правцу.

Генерација 1: Преклапајући менаџери (1984–2000) `twm` (Tab Window Manager, 1987) је био један од првих менаџера прозора за X11 (касније коришћен за Linux и BSD), и успоставио је парадигму која се до данас није суштински променила у најпопуларнијим системима. Прозори могу бити било где, а корисник их позиционира, увеличава, и фокусира се на њих помоћу миша. `mwm` (Motif Window Манагер, 1989) и `Enlightenment` (1997) развили су већу визуелну софистицираност, али је просторни модел остао идентичан.

Генерација 2: Композиторски менаџери (2004–2010) `Compiz` (2006) је увео хардверско убрзање користећи графичке картице за ефекте попут сенки, транспарентности, анимација и транзиција. `Beryl` и каснији `Compiz Fusion` додали су `Expo` и `Cube` додатке који су виртуелне десктопе приказивали као странице 3Д коцке или као умањени прозор. Ово је био први озбиљан покушај просторне визуализације вишеструких десктопа и велики напредак, али без промене фундаменталног модела.

Генерација 3: Мозаички менаџери (2006–данас) `dwm` (Dynamic Window Манагер, 2006) и `xmonad` (2007) увели су аутоматско распоређивање прозора као одговор на проблем продуктивности код преклапајућих модела, елиминацијом миша и аутоматским додељивањем простора. Мозаички менаџери прозора су драматично повећали ефикасност за кориснике који раде са великим бројем апликација. `i3` (Импровед Тилинг WM, 2009), `bspwm` (2013) и `Sway` (2015) су даље популаризовали овај модел и проширили га са X11 на Wayland протокол. Колика је била потреба за оваквим радним током доказује и чињеница да је заједница уложила огроман инжењерски напор да ову UNIX парадигму пренесе и на комерцијалне, искључиво преклапајуће оперативне системе. На macOS-у, овај прелаз обележили су алати попут `Amethyst`-а (2013) и архитектонски напреднијег `yabai`-ја (2019). На Windows платформи, сличан ниво организације донели су модерни пројекти као што су `komorebi` (2021) и `GlazeWM` (2022). Ипак, иако је ова ге-

нерација успешно решила проблем зависности од миша широм различитих екосистема, она је са собом универзално пресликала и један фундаментални недостатак: проблем 1D ограничења радних простора и круту логичку класификацију базирану на таговима.

Генерација 4: Скролујући менаџери (2017–данас) `PaperWM` (оквирно 2017) и `Niri` (развој од 2023, стабилна верзија 2025), увели су хоризонтално скроловање као додатак таговима; прозори се распоређују у бесконачни хоризонтални низ, а корисник може да скролује између њих. Ово је била прва експлицитна имплементација идеје да радни простор може бити већи од екрана. Међутим, оба система су остала ограничена на кретање у једној димензији.

Генерација 5: 2D Картезијански менаџери (Предлог овог рада) Предложена архитектура представља управо следећи логичан корак у еволуцији менаџера прозора. Проширење из 1D бесконачног низа у 2D бесконачну матрицу, са формалним координатним моделом, семантичким слојевима организације и интегрисаним управљањем ресурсима. Свака претходна генерација решавала је само један проблем претходне, док овај рад аргументује да је неопходно истовремено решити проблеме простора, когнитивног оптерећења и ресурса у јединственој архитектури.

Табела 1: Еволуција менаџера прозора по генерацијама.

Генерација	**Период**	**Примери**	**Кључни допринос**	**Преостало ограничење**
0. Рани GUI	1973–1984	Xerox Alto, Star, Lisa	Преклапајући прозори, миш	Само миш, без организације
1. Пливајући	1984–2000	twm, MWM, E17	Слободно позиционирање	Миш-зависност, без организације
2. Композиторски	2004–2010	Compiz, Beryl	GPU ефекти, визуална навигација	Исти просторни модел
3. Тилинг	2006–данас	dwm, i3, bspwm, Sway	Аутоматски layout, без миша	1D тагови, хоризонтални парадокс
4. Скролујући	2017–данас	PaperWM, Niri (2025)	Простор $>$ екран, 1D	Само једна оса
5. 2D Картезијански	предложено	—	2D грид, семантика, culling	Нема прототипа

2.2 Графички стекови по платформама

Да би се нова архитектура могла позиционирати у односу на све релевантне системе данашњице, неопходно је прегледати архитектуру графичких стекова на доминантним платформама.

Linux графички стек На Linux системима постоје два паралелна протокола сервера приказа. `X11` (X Window System), настао 1984. године, заснован је на мрежно-транспарентном клијент-сервер моделу у коме X сервер функционише као посредник између апликација и хардвера. Менаџер прозора је у овом моделу посебан клијент који декорисањем прозора и управљањем распоредом допуњује X сервер. Овај модел омогућава потпуну замењивост менаџера прозора без рестартовања система. `Wayland`, настао 2008. као замена за X11, елиминише посредника. Сваки клијент директно комуницира са композитором који обједињује улоге сервера приказа, композитора и менаџера прозора. Ова архитектура елиминише латенцију X11 протокола али отежава замену композитора. Менаџери приказа (Display Манагерс) као GDM, SDDM, LightDM и Ly, су посебна класа програма који управљају аутентификацијом и покретањем сесије; они су ирелевантни за дизајн менаџера прозора.

Windows графички стек На Windows системима, управљање прозорима историјски је било интегрисано у језгро оперативног система кроз `User32.dll` и `GDI` подсистеме. Свака апликација директно комуницира са `Win32 API`-јем за креирање и позиционирање прозора. Увођењем `Desktop Window Manager (DWM.exe)` у Windows Vista (2007), Microsoft је додао композитор који пресреће сво цртање и комбинује прозоре пре приказивања, аналогно Wayland композитору. Међутим, за разлику од Wayland-а, DWM **није заменљив**; он је системска компонента која се не може угасити на Windows 8 и каснијим верзијама. Ова непроменљивост значи да треће стране које желе да имплементирају алтернативне менаџере прозора на Windows-у морају радити *iznad* DWM-а користећи Win32 API, без директног приступа композиторској обради.

macOS графички стек На macOS-у, `Quartz Compositor` (део `Core Graphics` framework-а) преузима све улоге: Сервера приказа, композитора и управљања прозорима. Апликације комуницирају са подсистемом прозора искључиво кроз `AppKit` и `Core Graphics` API-је; директан приступ нижим слојевима није дозвољен без заобилажења *System Integrity Protection* (SIP) механизма. `WindowServer` је приватни системски сер-

вис који имплементира стварно позиционирање прозора, а тај API није јавно документован. Треће стране које имплементирају менаџере прозора на macOS-у (Yabai, Rectangle, Moom) користе `Accessibility API` (`AXUIElement`) који је оригинално дизајниран за приступачност, а не за управљање прозорима, што уводи значајна ограничења у перформансама.

2.3. Архитектонска ограничења комерцијалних система (Windows и macOS)

Иако су заједнице уложиле огроман инжењерски напор да мозаички модел пренесу на комерцијалне оперативне системе, ти покушаји су оголили фундаментална ограничења постојећих архитектура затвореног кода.

На Windows платформи, еволуција организације прозора ишла је од пливајућег модела, преко 1D виртуелних десктопа, до Снап Layouts и FancyZones механизама. Међутим, ови системи нуде само предефинисане, нефлексибилне зоне ограничене на физички екран (0D просторни модел). Сви покушаји трећих страна да емулирају напреднији радни ток функционишу искључиво хватањем Win32 порука и преписивањем позиција прозора *post-factum*. Фундаментално ограничење је одсуство интеграције на нивоу композитора (DWM.exe), због чега ови алати не могу да контролишу VRAM алокацију нити да оптимизују ресурсе.

Филозофија macOS-а ослања се на системске гестове и визуелно богате приказе (Спацес, Стаге Манагер), који су доказано неефикасни за рад оријентисан на тастатури. Покушаји имплементације правих мозаичких менаџера, попут yabai-ја, захтевају потпуно искључивање заштите интегритета система (SIP) ради убризгавања кернел екстензија. Алати који то не раде (Rectangle, Moom) ослањају се на Accessibility API, који није пројектован за високо-фреквентне операције и уводи мерљиву латенцију. Као и код Windows-а, композитор остаје потпуно закључан за треће стране.

Матрица постепене деградације (Graceful Degradation) Ова системска ограничења комерцијалних платформи не елиминишу могућност имплементације предложеног 2D Картезијанског модела, али јасно дефинишу путању деградације за поменути међуплатформски слој компатибилности (`libkrozor`, објашњен у каснијим секцијама).

Док ће на Linux/Wayland стеку предложена архитектура користити пун капацитет композитора за имплементацију Viewport Culling протокола (ослобађање VRAM-а и слање

`SIGSTOP` сигнала за суспензију процеса), на macOS-у и Windows-у се функционалност намерно деградира. На овим платформама, `libkrozor` испоручује пуну логику Картезијанске 2D навигације и систем Путоказа, али се одриче CPU/VRAM суспензије како би се очувала стабилност система и поштовале рестриктивне безбедносне полисе.

2.4. Преглед сродних решења

Следећи менаџери прозора представљају директно релевантне претке или парцијална решења проблема којим се овај рад бави:

- **PaperWM:** Уводи хоризонтално скроловање унутар једног Крозора, али остаје 1D.
- **Niri:** Скролабилни мозаички Wayland композитор (стабилан од јануара 2025) по једној хоризонталној оси са фиксним колонама; написан у Rust-у на основу Smithay framework-а, истог који предлажемо за имплементацију `libkrozor`-а.
- **Hyprland:** Модерни Wayland композитор са просторним распоредом, базиран на изолованим радним просторима по монитору.
- **Awesome WM:** Конфигурабилан у Lua језику, подржава експерименталне 2D распореде, али без нативне просторне апстракције.
- **Herbstluftwm:** Експлицитно управљање фраме стаблом, без просторне континуалности.
- **EXWM:** Emacs-базирани WM са слободним позиционирањем, без грид система.
- **River и LeftWM:** Минималистички Wayland WM-ови у `i3`/`dwm` парадигми.
- **Xmonad:** Haskell-базирани WM са прилагођеним алгоритамским layout-има, без нативне 2D апстракције.
- **KDE Plasma (KWin):** Једини популарни DE са рудиментарним 2D грид распоредом виртуелних десктопа, без просторне континуалности или culling-а.

Ниједан од наведених система не комбинује дискретну Крозор-мрежу, континуално унутрашње скроловање, декларативно инстанцирање и Culling протокол у јединственој архитектури.

2.5. Унификована терминолошка мапа

Следећа табела мапира термине различитих екосистема на еквивалентне појмове и на термине предложене у овом раду, чиме се експлицитно демонстрира терминолошки допринос стандардизације.

Табела 2: Терминолошка мапа између платформи и предложене архитектуре

Концепт	**Windows**	**macOS**	**Linux (i3/dwm)**	**Linux (KDE)**	**Овај рад**
Изоловани радни простор	Virtual Desktop	Space	Workspace / Tag	Virtual Desktop	— (замењено Крозор-мрежом)
Приказ свих простора	Task View	Mission Control	—	Overview	Минимапа
Брзи скок на простор	Win + Ctrl + N	Ctrl + N	Пречица за таг	Пречица	Путоказ ($O(1)$)
Зона прозора на екрану	Snap Zone / FancyZone	—	Container	—	Ћелија + Зона
Привремени прозор	—	—	Scratchpad	—	Привремени простор
Групни контекст	Snap Group	Stage Group	Workspace	Activity	Зона
Кретање између простора	Ctrl + Win + стрелице	Ctrl + стрелице	Пречица	Пречица	Крозор навигација

2.6. Званична терминологија предложене 2D архитектуре

Овај рад успоставља следеће формалне термине који су супериорни свим претходним у смислу прецизности, експресивности и преносивости између платформи:

1. **Картезијанско платно:** Логички бесконачна дводимензионална целобројна мрежа $\mathbb{Z}^2$ на којој су распоређене ћелије Крозора. Замењује све концепте изолованог радног простора.

2. **Крозор (Viewport):** Логичка репрезентација физичког екрана; прозор кроз који корисник посматра део Картезијанског платна. Има фиксне димензије и целобројну (x, y) позицију на платну.

3. **Путоказ (Waypoint):** Статички меморисана (x, y) координата на платну којој је додељена пречица за $O(1)$ телепортацију Крозора.

4. **Декларативно инстанцирање (Declarative Spawning):** Отварање новог прозора експлицитном декларацијом смера од стране корисника.

5. **Зона:** Именовани правоугаони регион који обухвата више ћелија платна; органи-

зациони ниво између појединачне ћелије и платна у целини.

6. **Culling праг (Culling Тхресхолд):** Максимална Крозор-раздаљина изван које се прозори подвргавају суспензији и ослобађању графичке меморије.
7. **Миграторна транзиција (Migratory Transition):** Пренос власништва прозора са једне ћелије на суседну при преласку границе.

2.7. Формални координатни модел

Дефиниција Картезијанског платна: Картезијанско платно $\mathcal{C}$ дефинише се као целобројна мрежа $\mathbb{Z}^2$. Свака тачка $(i, j) \in \mathbb{Z}^2$ одговара једној ћелији мреже чије су физичке димензије $W_{phys} \times H_{phys}$ пиксела.

Дефиниција прозора: Сваки прозор w карактеришу:

- Ћелијска позиција: $\text{cell}(w) \in \mathbb{Z}^2$
- Унутар-ћелијска позиција: $\text{pos}(w) \in \mathbb{R}^2$ (пиксел-прецизно)
- Димензије: $\text{dim}(w) \in \mathbb{R}^2$

Власништво ћелије одређује центроид:

$$\text{cell}(w) = \left\lfloor \frac{\text{pos}_{global}(w)}{(W_{phys},\ H_{phys})} \right\rfloor$$

Culling услов:

$$\text{cull}(w, \mathcal{V}) = (|\text{cell}(w)_x - \text{grid}(\mathcal{V})_x| > R) \vee (|\text{cell}(w)_y - \text{grid}(\mathcal{V})_y| > R)$$

за culling праг $R \in \mathbb{N}$ (подразумевано $R = 2$).

Дефиниција Путоказа: Скуп Путоказа: $\Phi : \text{String} \to \mathbb{Z}^2$ (хеш мапа, $O(1)$ приступање/прегледање).

Дефиниција Зоне: Зона $Z = (\text{naziv}, p_{min}, p_{max})$ где су $p_{min}, p_{max} \in \mathbb{Z}^2$ углови правоугаоног региона. Центроид: $\text{centroid}(Z) = \lfloor (p_{min} + p_{max})/2 \rfloor$.

---

3. Проблем постојећих модела и ограничења хардвера

Постојећи системи управљања прозорима на свим платформама дизајнирани су са претпоставком да је радни простор ограничен физичким димензијама монитора. Овај приступ данас ствара два фундаментална проблема: проблем простора и проблем когнитивног контекста.

3.0. Квантификација продуктивносног трошка

Пре анализе техничких ограничења, корисно је квантификовати економски и временски трошак који лош менаџер прозора намеће професионалном кориснику. Ове процене су конзервативне и засноване на објављеним истраживањима.

Трошак једне промене контекста Марк ет ал. (2008) су емпиријски измерили да недобровољни прекид радног контекста захтева просечно 23 минута за потпуни повратак на претходни ниво концентрације. Чак и добровољна промена контекста (када корисник сам одлучује да пређе на други задатак), захтева 64 секунде само да се визуелни фокус стабилизује на новом садржају (Iqbal & Bailey, 2007). У радном дану који укључује интензиван рад са 15–20 прозора, просечан професионалац изводи између 80 и 150 навигационих операција дневно.

На 1D тилинг систему са 9 тагова и 40 отворених прозора:

- Сваки прелаз који захтева проналажење погрешног тага пре исправног: ~ 8 секунди губитка.
- Проценат "погрешних тагова" при пуном капацитету (емпиријска процена): 15–25%.
- Дневни губитак при 100 навигационих операција: $100 \times 0.20 \times 8\text{s} = 160\text{s} \approx 2.7$ min/dan.

Ово делује безазлено док се не укалкулише прекид концентрације који прати сваки погрешан прелаз, чак и кратки визуелни прекид од 2–3 секунде може иницирати контекстуални ресет који кошта до 64 секунде опоравка. При конзервативној стопи од једног таквог прекида сваких 10 навигационих операција, од 100 дневних операција настаје $100/10 = 10$ таквих прекида дневно. Трошак опоравка је $10 \times 64\text{s} = 640\text{s} \approx 10.7$ min/dan

непродуктивног времена узрокованог искључиво навигацијом.

На годишњем нивоу (230 радних дана) то износи: $10.7 \times 230 \approx 41$ сат годишње по професионалцу. За тим од 10 програмера са просечном сатницом од 50 ЕУР/х, то је **~20.500 ЕУР годишњег трошка** који се може директно приписати неефикасности менаџера прозора. Ове процене не укључују когнитивни трошак менталног мапирања прозора на тагове (Милеров закон), који је по природи теже монетизовати, али није нимало мање реалан.

3.1. Проблем простора и 1D ограничења

Динамички тилинг менаџери прозора, попут `dwm`-а, моделирају простор као 1D структуру. Корисник има 9 ознака. Математички гледано капацитет је:

$$\text{min: } 4 \times 9 = 36 \text{ прозора}, \quad \text{max: } 8 \times 9 = 72 \text{ прозора}$$

Емпиријски, горња граница пада знатно ближе доњој вредности услед неуједначене дистрибуције задатака. Исти проблем, у различитој форми, постоји на свим платформама. Windows 11 Снап Layouts нуде 6 предефинисаних распореда по екрану; macOS Спацес су само хоризонтална секвенца без капацитетног проширења по дубини. Ниједан систем не третира монитор као пролаз (крозор) у веће платно.

3.2. Хоризонтални парадокс и избор монитора

Закривљени 34-инчни (21:9) монитор са резолуцијом 3440×1440 пиксела представља оптималан инжењерски алат. Његова закривљеност математички одговара фокалном радијусу људског ока, а ширина омогућава приказ три стандардна прозора (80–120 карактера) без деформације. Међутим:

- Подела на **3 ступца**: $3440/3 \approx 1147$ px, прихватљиво за терминал.
- Подела на **4 ступца**: $3440/4 = 860$ px, гранично и често преуско за писање кода.
- `dwm` Мастер/Стацк распоред при H=3 Стацк прозора: Мастер зона заузима 55% ширине по подразумеваним вредностима ($3440 \times 0.55 \approx 1892$ px), што оставља $3440 - 1892 = 1548$ px за Стацк колону. Подељено на 3 прозора: $1548/3 = 516$ px по прозору, што је далеко испод минимума читљивости за интегрисана развојна окружења (IDE).

- Windows Снап на 21:9: Стандардни 50/50 Снап даје по 1720 px сваком од два прозора, што је коректно. Међутим, постављање трећег прозора нема адекватне системске подршке, корисник мора ручно бирати између 33/33/33 поделе (што је нативно подржано тек у Windows 11) или применити конфузно преклапајуће позиционирање.
- macOS Спацес: Све апликације су максимизоване на цео екран по *default*-у, губећи целокупни бенефит широког формата.

Овај јаз између физичких капацитета монитора и онога што постојећи менаџери прозора од њега успевају да направе назива се **хоризонтални парадокс**.

3.3. Анализа корисничких профила

Профил 1: Софтверски инжењер (Програмер) Типичан радни ток: едитор, терминали (буилд, тест, лог), претраживач, база података, најчешће између 6 и 10 активних процеса. На Windows-у, корисник се ослања на комбинацију *Snap* функције и Alt+Tab, што захтева континуирану, мишем интензивну реорганизацију. На macOS-у, *Stage Manager* губи ефикасност већ при више од 5 апликација. Код 1D менаџера на Linuxu, тагови постају загушени. На предложеном 2D систему: едитор се позиционира на $(0, 0)$, терминали на $(1, 0)$, документација на $(-1, 0)$, а база на $(0, -1)$. Добија се стабилан распоред где је сваки контекст доступан једном пречицом, уз нулто когнитивно преклапање.

Профил 2: Системски администратор Надзор више сервера истовремено: SSH сесије, токови логова (*log streams*), мониторинг системи. Кључна специфичност овог профила је у томе што би се прозори организовали по *идентитету сервера*, а не по типу алата (што представља разлику у односу на програмере). Windows *Task View* и macOS *Spaces* не подржавају овај тип изолације нативно нити ефикасно. Концепт Картезијанског платна директно решава ову потребу на свим платформама кроз `libkrozor`, омогућавајући да Y-оса представља специфичан сервер, а X-оса алате покренуте на њему.

Профил 3: Истраживач сајбер-безбедности Захтева паралелно извршавање алата попут мрежних скенера и *fuzzer*-а, који апсолутно не смеју бити суспендовани док раде у позадини. Контекстуални *Whitelist* механизам унутар *Viewport Culling* протокола решава ово унапред. Поред тога, просторно груписање по фази напада (нпр. фаза изви-

ђања мреже мапирана на Y=1, фаза експлоатације на Y=0) мапира се сасвим природно на 2D мрежу.

Профил 4: Аутономни софтверски агенти (Desktop Агенти) Иако машински ентитети и аутономни AI агенти првобитно нису разматрани као циљна публика овог рада, пажљива архитектонска анализа указује да је предложени модел заправо примарна платформа за њихову интеграцију у оперативне системе. Софтверски агенти (који решавају комплексне задатке самостално комуницирајући са десктоп апликацијама) захтевају инстанцирање огромног броја прозора за парсирање података. У традиционалним пливајућим системима, визуелна оклузија (преклапање прозора) доводи до губитка контекста за моделе рачунарског вида (*Computer Vision*), док 1D платформе отежавају програматску навигацију. У 2D Картезијанском систему, агент користи декларативно инстанцирање да математички лоцира процесе на експлицитне (x, y) координате. Захваљујући таквој детерминистичкој природи, агент у $O(1)$ времену располаже тачним стањем и локацијом сваког GUI елемента на целокупном систему, распоређеном на бесконачном платну без иједног преклопљеног пиксела.

4. Предложени модел: 2D Картезијански простор и двослојна бесконачност

Као решење наведених ограничења, овај рад предлаже нову архитектуру засновану на јединственом моделу **двослојне бесконачности** у Картезијанском координатном систему, описаном формалним моделом из Одељка 2.7. Архитектура системски раздваја организацију контекста од организације самих прозора, оперишући на два потпуно независна нивоа:

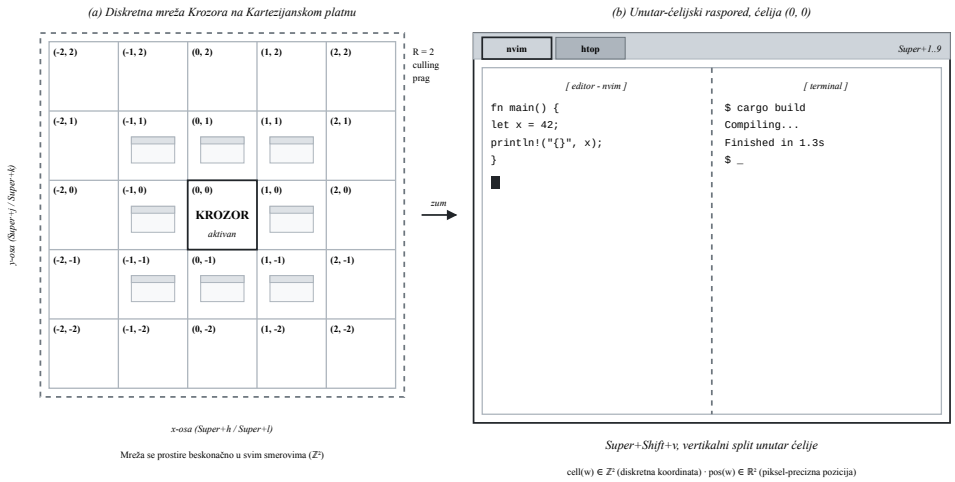
1. **Макро-простор (Дискретна мрежа Крозора):** Први ниво представља бесконачну 2D матрицу Крозора (*Viewports*). Сваки Крозор се налази на целобројној ћелијској координати ($\mathbb{Z}^2$). Прелазак монитора са једног на други Крозор (нпр. са координате $(0, 0)$ на $(0, 1)$) је тренутан и дискретан (*snap*). Овај ниво служи искључиво за изолацију радних контекста (нпр. прелазак са програмирања на базу података).
2. **Микро-простор (Бесконачно скролујуће платно):** Унутар сваког Крозора налази се други ниво, Картезијанско платно. Платно оперише на пиксел-прецизним координатама ($\mathbb{R}^2$) и такође је просторно бесконачно. Додавање прозора унутар истог Крозора не сужава постојеће прозоре до неупотребљивости, већ шири радно платно. Кретање монитора кроз ово платно је континуално (*smooth scrolling*).

Овакав дизајн гарантује да корисник никада не остаје без простора: Макро-простор обезбеђује бесконачно много контекста, док Микро-простор обезбеђује бесконачно много места за апликације унутар једног контекста.

4.1. Декларативно инстанцирање и модел власништва

Присуство две бесконачности захтева јасну разлику између команди за управљање Мрежом (Макро) и команди за управљање Платном (Микро). Корисник експлицитно диктира смер отварања и померања прозора путем тастатуре, чиме систем не доноси предиктивне (често погрешне) одлуке:

1. Операције на Микро-простору (Унутар истог платна): Када корисник захтева отварање новог алата у *истом радном контексту*, користи стандардно инстанцирање



Слика 2: Двослојна бесконачност предложене архитектуре. Лево (Макро-простор): дискретна Картезијанска мрежа Крозора ($\mathbb{Z}^2$) са активним монитором на координати $(0, 0)$, приказом навигационих пречица и culling прагом $R = 2$ (испрекидани оквир). Прелазак између ових ћелија је дискретан. Десно (Микро-простор): зум ћелије $(0, 0)$ који демонстрира бесконачно скролујуће платно унутар тог Крозора. Позиција прозора на платну је континуална и пиксел-прецизна (pos$(w) \in \mathbb{R}^2$). Приказан је таб-бар са више прозора и активиран вертикални сплит, са назнаком да се платно шири изван граница физичког екрана.

(нпр. пречица за вертикалну поделу). Прозор се додаје на платно на прецизним пиксел--координатама. Уколико прозори премаше димензије физичког екрана, систем омогућава глатко скроловање камере унутар тог Крозора.

2. Операције на Макро-простору (Кроз дискретну мрежу): Када корисник жели да отвори апликацију у *новом контексту*, користи усмерено декларативно инстанцирање кроз мрежу (нпр. `Super + l + w`). Систем креира инстанцу апликације, додељује је Картезијанском платну које се налази на координати $(X + 1, Y)$, и врши дискретни прелазак монитора на ту целобројну координату у $\mathbb{Z}^2$ простору.

Исти модел примењује се и на померање већ постојећих прозора, где се мења "власништво" над процесом, прозор се премешта са платна једне ћелије на платно друге ћелије.

4.1.1. Динамички мозаички распоред и просторна континуираност

Одељак 4.1 дефинише како прозори долазе на платно, али не и како се распоређују унутар њега. Архитектура у потпуности одбацује концепте картица (табова) и преклапајућих прозора јер они сакривају стање система и нарушавају просторну меморију

корисника.

Унутар сваке ћелије (Крозора), сви прозори су **увек распоређени у динамички мозаик (тилинг)**, изузеци су нотификације, обавештења, врсте статуса и слично.

Када корисник отвори нови прозор у активном Крозору, систем примењује алгоритам аутоматског дељења простора (нпр. бинарна подела или колоне). Међутим, за разлику од традиционалних тилинг менаџера који сабијају прозоре унутар фиксних граница монитора (стварајући нечитљиве и преуске колоне), предложени систем интегрише тилинг са **бесконачним скроловањем**:

- **Правило очувања минималне ширине:** Сваки прозор има дефинисану минималну употребљиву ширину (нпр. 80 карактера или 800 пиксела).
- **Аутоматско проширење платна:** Уколико додавање новог прозора у мозаик захтева да постојећи прозори падну испод своје минималне ширине, систем не дозвољава њихово даље сабијање. Уместо тога, нови прозор се додаје у мозаик у пуној величини, гурајући суседне прозоре удесно или надоле.
- **Скроловање као решење за преливање:** Прозори који су "гурнути" преко физичке ивице екрана остају део исте мозаичке структуре на бесконачном платну. Корисник до њих долази природним, глатким скроловањем камере дуж те осе.

Ово дизајнерско правило обезбеђује нулту когнитивну ентропију, корисник не мора да управља табовима нити да размишља о сакривеним прозорима. Сви прозори задржавају своје фиксно место у мозаику на платну, а просторна меморија остаје апсолутно очувана.

Табела 3: Пречице за управљање унутар-ћелијским мозаиком.

Пречица	**Акција**
`Super + Shift + v`	Изврши вертикалну поделу (сплит) активног прозора
`Super + Shift + b`	Изврши хоризонталну поделу (сплит) активног прозора
`Super + Shift + h/j/k/l`	Помери фокус између подељених прозора у мозаику
`Super + Ctrl + h/j/k/l`	Ручно прошири/скупи фокусирани прозор на платну

4.2. Навигација, фокус и организација простора

Фокус прати Крозор, при навигацији на нову ћелију у мрежи, фокус добија прозор чији је центроид најближи центру Крозора. Унутар ћелије фокус се мења пречицом између

подељених прозора у мозаику.

Дијалошки оквири и менији везани су за родитељски прозор релативним координатама; прате га при скроловању на платну или миграцији између ћелија. Класификација је аутоматска путем стандардних системских атрибута.

Механизми навигације:

1. **Минимапа (`Super + Tab`):** Удаљавање камере (зоом-оут) изнад целокупне мреже. Композитор рендерује све активне бафере умањено, приказујући Зоне као обојена поља, пружајући брзу просторну оријентацију.
2. **Путокази (`Super + m + [taster]` / `' + [taster]`):** $O(1)$ телепортација. Хеш-мапа која асоцира тастер са тачном координатом на мрежи или платну, омогућавајући тренутан повратак у сачуван контекст.
3. **Fuzzy-Find претрага (`Super + /`):** Текстуална претрага по насловима и процесима апликација; аутоматски и глатко помера Крозор до проналажења тражене ставке.
4. **Историја навигације (`Super + [` / `Super + ]`):** Стек структура која бележи редослед посећених координата, омогућавајући повратак уназад.
5. **Привремени простор (Сцратцхпад):** Резервисани регистар изван 2D координатног система за привремено складиштење алата који се могу позвати на било који део платна.

4.3. Viewport Culling Протокол и управљање ресурсима

Viewport Culling Protokol решава проблем експоненцијалног раста потрошње хардвера на бесконачном платну кроз две фазе интеграције:

Фаза 1: Ослобађање графичке меморије (VRAM Unmapping) Систем прати радијус R око активног Крозора. Сви прозори чији се центроид налази ван овог радијуса губе приступ визуелном рендеровању. Композитор их уклања из активне видео меморије, чиме се GPU у потпуности растерећује без обзира на укупан број отворених алата на бесконачном платну.

Фаза 2: Кооперативна суспензија процеса Поред ослобађања графике, архитектура уводи процесорску суспензију. Уместо небезбедног насилног гашења, менаџер прозо-

ра користи IPC сигнале како би обавестио апликације да су изашле из видног поља. Компатибилне апликације тада заустављају позадинске анимације и тешке прорачуне, сводећи CPU потрошњу на минимум. За процесе који обављају мрежни рад или позадинску обраду података, архитектура предвиђа белу листу (*whitelist*) која заобилази ову фазу.

```
# Konfiguracija Viewport Culling protokola
[culling]
threshold = 2
whitelist = ["mpd", "ncspot", "cargo", "nmap", "tcpdump"]
blacklist_patterns = [".*-scan.*", ".*build.*"]
```

4.3.1. Алгоритам праћења оштећења приказа и просторно хеширање

Ефикасно исцртавање на бесконачном платну захтева рендер који процесира искључиво онај део екрана који је претрпео визуелну промену (оштећење). Наивна рекреација платна би изазвала перцептуално цепање слике.

Архитектонски ток рендеровања спроводи се кроз следећу рутину:

```
1. Osluškuj signal za promenu frejma od grafičkog podsistema.
2. Prikupi informacije o vizuelnim promenama od svih aktivnih prozora.
3. Ukoliko je Krozor promenio koordinate u ovom frejmu:
      Označi čitav ekran kao promenjen
                        (neophodno osvežavanje celog prikaza).
   U suprotnom:
      Označi samo one delove ekrana gde su prozori prijavili promene.
4. Za svaki prozor u zoni vidljivosti:
      Izračunaj relativnu poziciju prozora na fizičkom monitoru.
      Sprovedi matematičko klipovanje (odbaci piksele van granica ekrana).
      Ako prozor ima promene, primeni efekte kompozitora (senke, ivice).
5. Isporči finalni prikaz hardveru.
```

Да би се Culling протокол извео ефикасно при сваком помаку камере, архитектура одбацује итерацију кроз све отворене прозоре ($O(W)$). Уместо тога, имплементира се **просторно хеширање** (Spatial Hashing). Сви прозори се индексирају на основу својих Макро-координата. Приликом транслације Крозора из координате A у координату B, систем спроводи разлику скупова:

```
Ulaz: Stara pozicija (X0, Y0), Nova pozicija (X1, Y1), Prag R
1. Skup S_OUT = Celije koje su bile u radijusu (X0,Y0),
        a nisu u radijusu (X1,Y1)
2. Skup S_IN  = Celije koje nisu bile u radijusu (X0,Y0),
```

```
       a jesu u radijusu (X1,Y1)

3. Za svaku celiju C u S_OUT:
      Za svaki prozor u C: Suspenduj iscrtavanje i oslobodi VRAM
4. Za svaku celiju C u S_IN:
      Za svaki prozor u C: Obnovi VRAM alokaciju
```

Овај приступ своди комплексност провере са обраде читавог система прозора на обраду искључиво ивичних ћелија које прелазе праг видљивости, гарантујући константне перформансе процесора.

4.4. Архитектура подршке за више монитора

Предложена архитектура радикално мења парадигму вишеструких монитора. Уместо да сваки монитор представља изоловани координатни систем, сви физички монитори делују као **засебни Крозори који деле исто Картезијанско платно**.

Ово омогућава два главна мода рада:

1. **Дељени мод:** Монитори посматрају различите делове платна. На пример, леви монитор фиксира фокус на координату $(0, 0)$, док десни монитор фиксира $(1, 0)$. Корисник перципира континуални радни простор, а премештање прозора са левог на десни монитор своди се на промену његових математичких координата на јединственом платну.
2. **Дупликатни мод:** Монитори деле идентичну фокусну координату на платну, што је намењено ситуацијама презентовања и аналитике. Крозор на пројектору прати исте операције претраге простора као и Крозор на примарном уређају.

4.5. Хијерархија семантичких Зона

Да би се додатно умањило когнитивно оптерећење навигације, платно је надограђено семантичким слојем, односно **Зонама**. Зона је архитектонски конструкт који обухвата блок ћелија (нпр. матрица 2×2) и додељује им логичко име (например "Backend систем" или "Мрежна фаза напада").

Тако систем формира јасну онтолошку хијерархију:

- `Прозор`: Засебни процес унутар мозаика.

- `Ћелија`: Фокусирани контекст задатка.
- `Зона`: Општи пројекат или радни простор који групише ћелије.

Приликом зумирања на Минимапу, зоне се рендерују као обојена геометријска поља, обезбеђујући инстинктивно разумевање радне структуре простора.

4.6. Интеграција гестова на платну

Како би се очувала ефикасност на преносивим уређајима, архитектура предвиђа превођење покрета (гестова) у математичке помаке на мрежи. Гестови се директно пресликавају у исту навигациону логику са тастатуре. Двопрсто превлачење по додирној плочи израчунава се на основу вектора брзине. Уколико вектор пређе задати инжењерски праг, Крозор се на мрежи аутоматски помера у том смеру за целобројну вредност 1. Тропрсти гестови се везују за z-осу, омогућавајући "подизање" Крозора у Минимапа преглед.

4.7. Логика уноса и механика пречица

Кориснички интерфејс заснива се на композабилности, правци се комбинују са акцијама. Примењени су принципи мнемоничности и компатибилности са VIM окружењем.

Табела 4: Архитектонски преглед навигационих интенција.

Категорија интенције	**Акција и правац**
Транзиција Макро-простора	Скок на суседну ћелију мреже (`Super + Smer`)
Инстанцирање објеката	Креирање прозора у одабраној ћелији (`Super + Smer + w`)
Транзиција Микро-простора	Подела платна (сплит) унутар активне ћелије (`Super + Shift + Smer`)
Скокови сложености $O(1)$	Телепортација на меморисане Путоказе и Зоне
Управљање хијерархијом	Промена фокуса унутар угњежденог стабла ћелије

4.8. Серијализација стања система

Зрелост архитектуре огледа се у њеној способности обнављања простора. Стање система $\mathcal{S}$ се периодично серијализује из радне меморије у трајни формат података (попут TOML структуре). Модел података који се серијализује изолује искључиво релевантне податке о платну:

```
# Pseudokod modela stanja sistema
[viewport]
```

```
aktivna_koordinata = (x, y)

[[windows]]
vlasnik_celija = (x, y)
piksel_pozicija = (px, py)
dimenzije = (w, h)
komanda = "putanja/do/izvrsnog/fajla"

[waypoints]
"Taster A" = (x, y)
```

Овакав детерминистички модел гарантује да ће се након гашења система радно окружење рестаурирати са пикселском тачношћу.

4.9. Теоријска анализа сложености

Систем је архитектонски прилагођен масивном паралелном раду, што се огледа у оптимизацији рачунарске комплексности.

Табела 5: Сложеност операција на предложеној структури података.

Операција	**Очекивана сложеност**	**Структурно објашњење**
Коришћење Путоказа	$O(1)$	Очитавање директно из хеш мапе
Померање Крозора	$O(1)$	Скаларна адиција на координати
Извршавање Culling евалуације	$O(1)$ у просеку	Употреба просторног хеширања
Претрага инстанци по имену	$O(N \log N)$	Сортирање стринг-мечева
Серијализација на диск	$O(N)$	Итерација листе свих објеката

(Напомена: N представља укупан број алата односно прозора на платну).

4.10. Приступачност (Accessibility) и агентни радни токови

Архитектура уважава чињеницу да чисто ослањање на уносе са тастатуре може бити ограничавајуће. Дизајн интегрише принципе моторичке приступачности кроз инхерентну подршку за лепљиве тастере (*Sticky Keys*) и мод скенирања (*Scan Mode*), омогућавајући корисницима са ограниченим моторичким способностима да итерирају кроз Крозор помоћу једноставног прекидног механизма.

Такође, као што је претходно наведено у Одељку 3.3, оваква егзактна просторна архитектура ($O(1)$ лоцирање инстанци) претвара систем у идеалну полазну тачку за AI агенте, који комуницирају са оперативним системом преко истог интерфејса (IPC) који је доступан и кориснику, замењујући визуелну навигацију просторном математиком.

4.11. Толеранција на грешке и критични случајеви

Планирање архитектуре обухвата моделовање опоравка од следећих критичних случајева:

- **Завршетак процеса у суспендованом стању:** Уколико процес претрпи прекид рада док је у *Culling* (суспендованој) фази, менаџер архитектонски развезује његове координате, ослобађајући ћелију без потребе да се процес пребаци у фокусну раван корисника.
- **Неуправљани процеси:** Системи који изричито захтевају фиксно позиционирање (УАЦ дозволе на Windows-у или инсталациони дијалози) третирају се изузетком који их привремено одваја од Макро-мреже, дозвољавајући плутање ван платна.
- **Ексклузивни екрани (Fullscreen):** Софтвер који иницира преузимање целог екрана (као код игара) изазива заобилажење стандардног рендеровања. Системска архитектура привремено "замрзава" операције на Крозору док се не прими сигнал за излазак, када се целокупан просторни модел рестаурира на своје примарно стање.

5. Међуплатформски слој апстракције, `libkrozor`

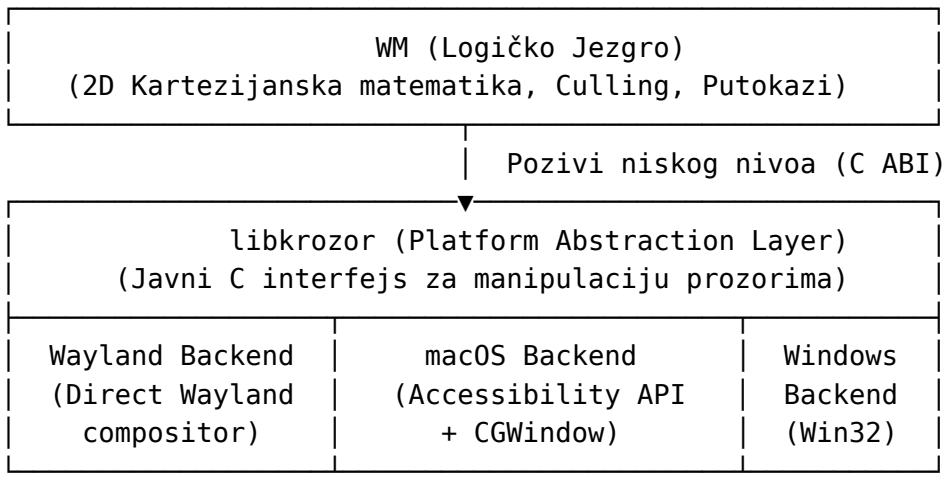
Формални модел и терминологија предложени у овом раду (Крозор, Путоказ, Зона, Картезијанско платно) су по природи мултиплатформски и не зависе од специфичног оперативног система или графичког протокола. Да би се омогућила преносивост целокупног радног тока корисника, системска архитектура мора раздвојити логику менаџера прозора од конкретног кода за интеракцију са хардвером/системом.

У ту сврху пројектована је библиотека `libkrozor`, која делује као **Слој платформске апстракције** (Platform Abstraction Layer, PAL). Њен задатак је да сакрије специфичности различитих оперативних система иза јединственог, преносивог програмског интерфејса.

5.1. Архитектонска улога и преносивост

Слој апстракције `libkrozor` је пројектован као динамичка библиотека реализована у програмском језику Rust, која излаже стабилан и унификован јавни C ABI. Овакав дизајн обезбеђује нулти утицај на перформансе и максималну преносивост.

Архитектонска позиција библиотеке у оквиру система приказана је на Слици 3:



Слика 3: Слој платформске апстракције (libkrozor) у односу на језгро менаџера и backend-е.

Менаџер прозора доноси одлуке на основу просторне математике, док `libkrozor` служи искључиво као извршитељ који те одлуке преноси на оперативни систем.

5.2. Унификовани Ц интерфејс и системске примитиве

Библиотека `libkrozor` излаже минималан и стабилан скуп функција које апстрагују три кључне системске операције:

1. **Управљање прозорима:**
    - `krozor_get_windows`: Враћа листу активних прозора и њихове базичне податке (наслов, класа, јединствени ИД).
    - `krozor_window_move`: Мења позицију и димензије прозора на екрану.
    - `krozor_window_focus`: Пребацује фокус на одабрани прозор.
2. **Управљање процесима:**
    - `krozor_process_suspend`: Шаље сигнал за замрзавање CPU активности одређеног процеса (ПИД).
    - `krozor_process_resume`: Шаље сигнал за одмрзавање процеса.
3. **Ослушкивање догађаја (Event Polling):**
    - `krozor_event_poll`: Јединствена системска петља која прима хардверске и софтверске догађаје (притисак тастера, покрете, отварање прозора) и прослеђује их логичком језгру на обраду.

5.3. Платформски backend-и матрица деградације

Имплементација унификованих функција се драстично разликује међу оперативним системима и графичким протоколима. Због рестриктивних сигурносних полиса комерцијалних платформи и различитости графичких сервера, предложена архитектура уводи принцип **постепене деградације функционалности** (*graceful degradation*):

- **Linux/BSD Wayland Backend (Пуни капацитет):** Делује као нативни Wayland композитор. Има директан приступ видео меморији и графичким баферима. Имплементира пуни капацитет *Viewport Culling* протокола (ослобађање VRAM-а кроз Wayland протоколе и CPU суспензију преко POSIX `SIGSTOP` сигнала који су нативно подржани на Linux и BSD кернелима).
- **Linux/BSD X11 Backend (Средњи капацитет):** Оперише као традиционални X11

менаџер прозора (WM клиент). Подржава унификовану 2D навигацију и пренос прозора. Culling прве фазе се имплементира кроз сакривање прозора (`XUnmapWindow` команда која сигнализира X серверу да ослободи ресурсе), док се друга фаза ослања на нативне POSIX сигнале. Главна деградација је одсуство директне контроле над модерним композитним исцртавања без спољних алата (попут *Picom*-а).

- **macOS Backend (Деградирани капацитет):** Оперише изнад затвореног Quartz композитора користећи системски *Accessibility API*. Позиционирање има мерљиву латенцију, док је ослобађање VRAM-а немогуће без кршења безбедносних полиса система (SIP). Функционалност се аутоматски деградира на испоруку само 2D навигације и Путоказа.
- **Windows Backend (Деградирани капацитет):** Оперише изнад DWM композитора преко стандардног Win32 API-ја. Позиционирање је високо ефикасно, али DWM не дозвољава директну манипулацију VRAM-ом, због чега се culling такође делимично суспендује.

Табела 6: Матрица постепене деградације libkrozor библиотеке по платформама.

Функционалност / Протокол	**Wayland (Linux/BSD)**	**X11 (Linux/BSD)**	**macOS (AppKit/AX)**	**Windows (Win32/DWM)**
2D Картезијанско платно	Да	Да	Да	Да
Путокази (Waypoints)	Да	Да	Да	Да
VRAM Culling (Фаза 1)	Да	Делимично (`XUnmapWindow`)	Не (системски лимит)	Не (системски лимит)
CPU Суспензија (Фаза 2)	Да (`SIGSTOP`)	Да (`SIGSTOP`)	Опционо (`SIGSTOP`)	Опционо (API fallback)
Compositor анимације	Да (нативно)	Не (захтева Picom)	Не	Не

5.4. Конфигурација и IPC комуникација

Да би се обезбедила преносивост радног тока, `libkrozor` омогућава да се идентичан конфигурациони фајл (TOML формат) и исти IPC протоколи користе на свим платформама.

5.4.1. Јединствена конфигурација

Сви параметри (пречице, culling праг, зоне) дефинишу се у једном фајлу. Системи на којима су неке функције хардверски онемогућене (нпр. culling на macOS-у) једноставно

игноришу те кључеве без рушења система:

```
# Primer konfiguracionog fajla kompatibilnog sa svim sistemima
[canvas]
culling_threshold = 2

[keybindings]
move_left  = "Super+h"
move_right = "Super+l"
fuzzy_find = "Super+slash"

[waypoints]
code = [0, 0]
web  = [-1, 0]
```

5.4.2. IPC Протокол (Канал проширивости)

Комуникација са спољним скриптама и статусним тракама обавља се преко Unix доменских сокета (Linux/macOS) или именованих фифо-а (*Named Pipes* на Windows-у). Размена података се врши путем једноставних JSON порука у реалном времену:

```
# Klijent salje komandu za navigaciju
{"cmd": "waypoint_goto", "key": "web"}

# Server (libkrozor) odgovara uspehom
{"ok": true, "moved_to": [-1, 0]}
```

Овај преносиви IPC интерфејс омогућава корисницима да пишу скрипте за аутоматизацију које ће радити идентично без обзира на оперативни систем под којим се извршавају.

5.5. Стандарди и унификација протокола

Како би систем био компатибилан са постојећим апликацијама које су дизајниране за класичне оперативне системе, `libkrozor` у оквиру својих backend слојева врши превођење стандардних системских сигнала (*hints*):

- **Класификација прозора:** На Linux/X11 системима користи се стандардни `_NET_WM_WINDOW_TYPE` (EWMH протокол) како би се аутоматски препознали дијалошки и привремени прозори (који се остављају да плутају). На macOS-у се иста класификација постиже превођењем `kAXWindowRole` атрибута, док Windows backend за то користи Win32 `WS_EX_TOOLWINDOW` стилове.

- **Гранични простори:** Статусне траке (попут *Waybar*-а) користе EWMH протоколе да резервишу простор на екрану. `libkrozor` препознаје ове захтеве на свим платформама и аутоматски смањује димензије активног Крозора, спречавајући да тилинг прозори прекрију системске траке.

Кроз овакву унификацију, апликације попут интернет претраживача или терминала аутоматски добијају исправан третман на бесконачном платну, без потребе да програмери прилагођавају свој софтвер предложеном 2D Картезијанском моделу.

6. Упоредна анализа и евалуација

Како би се предложена 2D Картезијанска архитектура могла објективно вредновати, неопходно је спровести упоредну анализу са доминантним историјским и модерним парадигмама управљања прозорима.

Табела 7: Поређење архитектуралних парадигми менаџера прозора.

Карактеристика	**Преклапајући WM (Win/macOS)**	**1D Тилинг WM (dwm/и3)**	**Скролујући 1D WM (Niri)**	**Предложени 2D Картезијански WM**
Искоришћеност екрана	Ниска (преклапање)	Висока на 16:9 формату	Висока на хоризонталној оси	**Апсолутна** (Viewport на платну)
Комплексност навигације	$O(N)$ линеаран Alt+Tab	$O(1)$ уз лимит од 9 тагова	$O(N)$ хоризонтално скроловање	$O(1)$ **уз неограничене Путоказе**
Груписање по намени	Лоше (захтева миш)	Тагови (ограничен простор)	Линеарне секвенце	**Зоне** (неограничено 2D груписање)
Менаџмент ресурса	Активни рендер свих прозора	Рендер искључиво активног тага	Рендер видљивих колона	**Viewport Culling невидљивог простора**
Димензионалност	0D (један екран)	1D (линеарни низ тагова)	1D (бесконачна трака)	**2D** (дискретни грид + континуални скрол)
Интерфејс уноса	Доминантно миш	Тастатура (*keyboard-centric*)	Делимично миш/тастатура	**Тастатура** (Vim-style граматика уноса)
Ultrawide оптимизација	Лоша (захтева ручни рад)	Лоша (хоризонтални парадокс)	Добра (скроловање у 1D)	**Одлична** (2D Кровзор над платном)
Когнитивни модел	Визуелно призивање	Призвање апстрактног тага	Призвање редоследа на траци	**Препознавање апсолутне позиције**
Преносивост	Немогућа (системски лок)	Не (чврста веза са X11/Way)	Не (Smithay/Wayland лок)	**Да** (Агностички Rust уз `libkrozor` ПАЛ)

(Напомена: Табела пореди теоријске парадигме, а не специфичне софтверске верзије. Свака категорија обухвата скуп репрезентативних система).

6.1. Ограничења предложене архитектуре и правци развоја

Иако предложена архитектура нуди решење за просторна и когнитивна ограничења, она са собом носи специфичне системске изазове и лимите:

Све предности напредног *Viewport Culling* протокола (нарочито Фаза 1 са унмаповањем

VRAM бафера) доступне су искључиво на Linux/BSD Wayland backend-у. Затвореност композитора на Windows и macOS платформама приморава `libkrozor` на деградацију функционалности, сводећи оптимизацију ресурса на ниво CPU суспензије, без директне контроле над видео меморијом. Ефикасност Фазе 2 Culling протокола у великој мери зависи од тога да ли апликације трећих страна исправно реагују на системске поруке о суспензији. Уколико апликација игнорише сигнал композитора, она наставља да троши CPU циклусе у позадини. Иако се графички бафер и даље ослобађа (Фаза 1), потпуна уштеда процесорског времена захтева адаптацију модерног софтверског екосистема. Све изнете тврдње о уштеди времена и смањењу когнитивног оптерећења су теоријске пројекције изведене из модела когнитивне психологије. Валидација предложених решења у реалном времену захтева комплетну имплементацију наменског композитора (нпр. преко *Smithay* развојног оквира у Rust-у) и спровођење контролисане корисничке студије.

6.2. Нацрт методологије за емпиријску евалуацију

Да би се теоријске предности 2D Картезијанског модела могле научно доказати, предлаже се нацрт ригорозне корисничке студије која пореди предложени модел са традиционалним преклапајућим окружењима (KDE Plasma) и 1D тилинг менаџерима (`i3wm`).

Евалуација се фокусира на мерење две кључне зависне варијабле:

1. **Време до циља (Time-to-Target, TTT):** Време изражено у секундама које протекне од тренутка давања задатка кориснику (нпр. „пређи на прозор са мрежним логовима") до тренутка када тај прозор добије фокус и постане визуелно доступан.
2. **Субјективно когнитивно оптерећење:** Вредновано помоћу стандардизованог NASA-TLX упитника (Mental Task Load Index) непосредно након завршетка стандардизованих програмерских, администраторских и безбедносних задатака.

Кроз мерење TTT-а при великом броју паралелних процеса (између 10 и 40 активних прозора), студија треба да потврди хипотезу да прелазак са Alt+Tab навигације и апстрактних 1D тагова на 2D Картезијанске Путоказе (Waypoints) колапсира временску сложеност проналажења прозора са линеарне $O(N)$ на константну $O(1)$ брзину, уз статистички значајно смањење менталног напора и когнитивног умора корисника.

6.3. Агентни системи као непредвиђена архитектонска предност

Иако је предложена 2D архитектура оригинално пројектована како би решила когнитивне лимите људских оператера, детаљна анализа њених структурних особина открива да је она заправо идеално радно окружење за аутономне софтверске агенте који управљају десктоп апликацијама.

Све већи број модерних система приморава агенте да контролишу графичке апликације очитавајући визуелни садржај екрана и емулирајући кликове миша. На традиционалним оперативним системима, агенти се суочавају са огромним проблемом визуелне оклузије, преклапање прозора сакрива информације, приморавајући агенте да троше драгоцене ресурсе и време на рачунарски вид (Computer Vision) и OCR анализу како би уопште лоцирали прозор. Навигација кроз Alt+Tab или 1D тагове захтева континуирани визуелни *feedback loop* који је спор и подложан грешкама. Јасно је да би овакав модел менаџера прозора решио ове проблеме због својих предности.

Комбинација ових фактора показује да конвергенција између оптималног окружења за људску продуктивност и оптималног окружења за аутономне машинске агенте **можда** није случајна? С обзиром да обе групе корисника захтевају строги просторни детерминизам, нулту оклузију и минимално когнитивно, односно рачунарско оптерећење при навигацији.

7. Закључак

Закључна разматрања овог рада указују на то да се налазимо на прагу неизбежне промене у начину на који људи комуницирају са рачунарским простором. Док је хардверска револуција обезбедила процесорску снагу и просторне капацитете екрана који превазилазе потребе појединачних процеса, наш софтвер је остао заробљен у дводимензионалним кавезима с краја прошлог века. Предложени модел Картезијанског менаџера прозора не представља само још један у низу покушаја естетске оптимизације распореда на екрану, већ фундаментални преокрет у схватању радног простора. Раздвајањем организације на макро-контексте и микро-платна, овај рад поставља темеље за интерфејс који се природно прилагођава људским когнитивним мапама, уместо да приморава корисника на неприродну апстракцију и ментално преоптерећење.

Кроз формализацију двослојне бесконачности, рад успешно спаја когнитивне законитости. Попут Милеровог и Фитсовог закона, са строгим инжењерским решењима. Увођење *Viewport Culling* протокола и просторног хеширања доказује да бесконачни просторни капацитет не мора долазити по цену перформанси хардвера. Спецификација слоја платформске апстракције кроз `libkrozor` успешно премошћује дубоке архитектонске разлике између отвореног Wayland стека и затворених комерцијалних платформи попут Windows-а и macOS-а. Тиме је показано да предложена решења нису утопијски дизајн, већ технички изводљива архитектура која поштује и интегрише наслеђене протоколе кроз унифициране системске примитиве и матрицу постепене деградације.

Иако одсуство функционалног софтверског прототипа остаје примарно ограничење овог истраживања, развијени теоријски и формални координатни оквир нуди кохерентан и детаљно испланиран нацрт за будући развој. Системска решења развијена у овом раду, од двослојне бесконачности и *Viewport Culling* протокола, па све до међуплатформског слоја апстракције кроз библиотеку `libkrozor`, чине комплетан технички пројекат спреман за реализацију у коду.

Крајњи циљ овог рада јесте успостављање новог стандарда у дизајну графичких окружења; стандарда који више не приморава корисника да своје мисли прилагођава физичким ограничењима екрана, већ радни простор претвара у природно, просторно проду-

жење људског когнитивног апарата.

8. Литература

- Гарбе, А. Р. ет ал. (2006–данас). *dwm — dynamic window manager*. suckless.org. (Оригинални аутор: Анселм Р. Гарбе).
- Хøгсберг, К. ет ал. (2008–данас). *The Wayland Protocol*. freedesktop.org. (Оригинални аутор: Кристиан Хøгсберг, Ред Хат).
- Стевенс, W. Р., & Раго, С. А. (2013). *Advanced Programming in the UNIX Environment* (3rd ed.). Addison-Wesley.
- Ханссен, П. Б. ет ал. (2017–данас). *PaperWM — Scrollable window manager for GNOME Shell*. GitHub Repository. https://github.com/paperwm/PaperWM
- Мооленаар, Б. ет ал. (1991–2024). *Vim Reference Manual: Marks and motions*. vimhelp.org. https://vimhelp.org/motion.txt.html
- Миллер, Г. А. (1956). The magical number seven, plus or minus two. *Psychological Review*, 63(2), 81–97.
- Фиттс, П. М. (1954). The information capacity of the human motor system. *Journal of Experimental Psychology*, 47(6), 381–391.
- Пике, Р. (1991). *8½, the Plan 9 window system*. USENIX Summer Conference Proceedings.
- Раскин, J. (2000). *The Humane Interface*. Addison-Wesley.
- vaxry ет ал. (2022–данас). *Hyprland — A dynamic tiling Wayland compositor*. hyprland.org. https://github.com/hyprwm/Hyprland
- Смитх, В. ет ал. (2017–данас). *Smithay — A smithing ground for Wayland compositors*. GitHub Repository. https://github.com/Smithay/smithay
- *xdg-shell: Add suspended toplevel state (MR !201)*. freedesktop.org GitLab — wayland/wayland-protocols. https://gitlab.freedesktop.org/wayland/wayland-protocols/-/merge_requests/201
- Марк, Г., Гудитх, Д., & Клоцке, У. (2008). The cost of interrupted work. *Proceedings of CHI 2008*, 107–110.
- О'Кеефе, J., & Надел, Л. (1978). *The Hippocampus as a Cognitive Map*. Oxford University Press.
- Ushkaryov, И. (YaLTeR) ет ал. (2023–2025). *Niri — A scrollable-tiling Wayland compo-*

sitor (v1.0, јануар 2025). GitHub Repository. https://github.com/YaLTeR/niri

- Лие, К. (koekeishiya) ет ал. (2019–данас). *Yabai — A tiling window manager for macOS*. GitHub Repository. https://github.com/koekeishiya/yabai
- Цхеунг, Ј. (ЛГУГ23) ет ал. (2021–данас). *komorebi — A tiling window manager for Windows*. GitHub Repository. https://github.com/LGUG2Z/komorebi
- Microsoft Corporation. (2021–данас). *Microsoft PowerToys FancyZones documentation*. Microsoft Learn. https://learn.microsoft.com/en-us/windows/powertoys/fancyzones
- Apple Inc. (2023). *Accessibility API (AXUIElement) Reference*. Apple Developer Documentation. https://developer.apple.com/documentation/applicationservices/axuielement
- Microsoft Corporation. (2023). *Win32 API: SetWindowPos, EnumWindows*. Microsoft Learn. https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-setwindowpos
- Apple Inc. (2023). *CGWindowListCopyWindowInfo Reference*. Apple Developer Documentation. https://developer.apple.com/documentation/coregraphics/1455137-cgwindowlistcopywindowinfo
- Харт, С. Г., & Ставеланд, Л. Е. (1988). Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. *Human Mental Workload*, 1(3), 139–183.
- МацКензие, И. С. (1992). Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7(1), 91–139.
- *xdg-shell protocol specification — xdg_toplevel suspended state*. Wayland protocols repository, freedesktop.org. https://gitlab.freedesktop.org/wayland/wayland-protocols
- Ирсаид, А. ет ал. (2020–данас). *Waybar — Highly customizable Wayland bar for Sway and Wlroots based compositors*. GitHub Repository. https://github.com/Alexays/Waybar
- *NtSuspendProcess — Неформална документација недокументованог Win32 API-ја*. NTInternals.net. (Напомена: неформални, reverse-engineered API без званичне Microsoft документације, наведен искључиво ради потпуности анализе алтернативних механизама суспензије).
- Microsoft Corporation. (2023). *RegisterSuspendResumeNotification — Windows power management API*. Microsoft Learn. https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-registersuspendresumenotification
- Wobbrock, J. O., & Киентз, J. A. (2016). Research contributions in human-computer interaction. *Interactions*, 23(3), 38–44. https://doi.org/10.1145/2907069

- *ICCCM — Inter-Client Communication Conventions Manual*. X Consortium, 1994. freedesktop.org. https://www.x.org/releases/X11R7.6/doc/xorg-docs/specs/ICCCM/icccm.html
- *Extended Window Manager Hints (_NET_WM_*)*. freedesktop.org спецификација, 1999–данас. https://specifications.freedesktop.org/wm-spec/wm-spec-latest.html
- *Motif Window Manager (MWM) Hints*. Open Group, 1997.
- freedesktop.org. (2008–данас). *xdg-shell Wayland protocol*. freedesktop.org GitLab. https://gitlab.freedesktop.org/wayland/wayland-protocols
- wlroots Contributors. (2018–данас). *wlr-layer-shell-unstable-v1*. wlroots GitHub Repository. https://github.com/swaywm/wlr-protocols
- freedesktop.org. (2015–данас). *zwp-pointer-gestures-v1 Wayland protocol*. freedesktop.org GitLab. https://gitlab.freedesktop.org/wayland/wayland-protocols
- Anthropic. (2024). *Computer Use — Enabling Claude to interact with desktop applications*. Anthropic Developer Documentation. https://www.anthropic.com/news/computer-use
- *OpenAdapt — Open-source generalist process automation*. GitHub Repository. https://github.com/OpenAdaptAI/OpenAdapt
- *screenpipe — AI-powered screen capture and automation*. GitHub Repository. https://github.com/mediar-ai/screenpipe
- Iqbal, C. T., & Bailey, Б. П. (2007). Understanding and developing models for detecting and differentiating breakpoints during interactive tasks. *Proceedings of CHI 2007*, 697–706. https://doi.org/10.1145/1240624.1240732
- *Xerox Alto Technical Reference Manual*. Xerox PARC, 1979.
- *twm — Tab Window Manager manual*. X Consortium, 1987.
- *AT-SPI2 — Assistive Technology Service Provider Interface*. freedesktop.org. https://specifications.freedesktop.org/at-spi-spec/at-spi-spec-latest.html
- *Web Content Accessibility Guidelines (WCAG) 2.1*. W3C Recommendation, 2018. https://www.w3.org/TR/WCAG21/
- *TOML — Tom's Obvious, Minimal Language specification v1.0*. toml.io, 2021. https://toml.io/en/v1.0.0