

Аляксандр Ворвуль

**АСНОВЫ ТЭСТАВАННЯ  
ПРАГРАМНАГА ДАЧЫНЕННЯ**



# Змест

1. Гісторыя спрыяння якасці праграмнага дачынення.....	5
1.1. 1947–1956 Перыйяд адладкі.....	6
1.2. 1957-1978 Перыйяд дэманстрацыі.....	9
1.3. 1979-1982 Перыйяд разбурэння.....	13
1.4. 1983-1987 Перыйяд ацэнкі.....	17
1.5. 1988-2000 Перыйяд прадухілення.....	21
1.6. 2001-2011 Перыйяд аўтаматызацыі тэсціравання.....	25
1.7. 2012–2021 Перыйяд бесперапыннага тэсціравання.....	27
1.8. 2022-Сёння Тэсціраванне на аснове штучнага інтэлекту.....	30
2. Якасць ПД.....	35
2.1. Сферы якасці ПД.....	37
2.2. Тэставанне якасці ПД.....	39
2.3. Кантроль якасці ПД.....	41
2.4. Спрыянне якасці ПД.....	43
2.5. Характарыстыкі якасці ПД.....	45
2.6. Крытэрыі ўваходу і выхаду.....	49
3. Жыццёвы цыкл распрацоўкі ПД.....	51
3.1. Этап планавання ПД.....	54
3.2. Этап праектавання ПД.....	60
3.3. Этап распрацоўкі ПД.....	62
3.4. Этап тэсціравання ПД.....	65
3.5. Этап разгортання ПД.....	68
3.6. Этап абслугоўвання ПД.....	71
3.7. Мадэлі ЖЦР ПД.....	74
4. Прагнозныя МППД.....	77
4.1. Мадэль Вялікага выбуху.....	79
4.2. Мадэль Вадаспаду.....	82
4.3. Інкрементальная мадэль.....	85
4.4. Ітэрацыйная мадэль.....	89
4.5. Спіральная мадэль.....	93
4.6. V-Model.....	98
5. Адаптыўныя МППД.....	106
5.1. Гнуткая метадалогія.....	108

5.2. Гнуткія метады.....	110
5.3. Маніфест Agile.....	111
5.4. Хуткая распрацоўка ПД.....	113
5.5. Хуткае прататыпаванне.....	116
5.6. Скрам.....	119
6. Фрэймворк Скрам.....	121
6.1. Каманда Скрам.....	122
6.2. Уладальнік прадукта.....	124
6.3. Скрам-майстар.....	126
6.4. Каманда распрацоўшчыкаў.....	127
6.5. Артэфакты Скрам.....	128
6.6. Бэклог прадукта.....	130
6.7. Бэклог спрынта.....	131
6.8. Інкремент.....	133
6.9. Падзеі Скрам.....	134
6.10. Спрынт.....	137
6.11. Штодзённы Скрам.....	139
6.12. Агляд спрынту.....	141
6.13. Рэтраспектыва спрынту.....	142
6.14. Удасканаленне бэклога.....	144
6.15. Працоўны працэс Скрам.....	146
7. Жыццёвы цыкл тэсціравання ПД.....	150
7.1. Этап планавання тэставання.....	152
7.2. Этап праектавання тэстаў.....	154
7.3. Этап распрацоўкі тэстаў.....	155
7.4. Этап выканання тэстаў.....	157
7.5. Этап закрыцця тэставання.....	158
7.6. Падтрымка тэстаў.....	160
8. Тэставая дакументацыя.....	163
8.1. Чэк-ліст.....	166
8.2. Тэст-кейс або тэставы сценар.....	168
8.3. Тэст-скрыпт і тэставыя даныя.....	172
8.4. Тэставы набор.....	176
8.5. План тэставання.....	178
8.6. Стратэгія тэставання.....	180

<i>8.7. Палітыка тэставання.....</i>	182
<i>8.8. Сістэмы кіравання тэставаннем.....</i>	183
<i>9. Дэфекты якасці ПД.....</i>	185
<i>9.1. Класіфікацыя дэфектаў.....</i>	187
<i>9.2. Жыццёвы цыкл дэфекту.....</i>	189
<i>9.3. Справаздача аб дэфекце.....</i>	191
<i>10. Класіфікацыя тыпаў тэсціравання ПД.....</i>	194
<i>10.1. Віды тэставання на этапе планавання ПД.....</i>	195
<i>10.2. Віды тэставання на этапе праектавання ПД.....</i>	200
<i>10.3. Віды тэставання на этапе распрацоўкі ПД.....</i>	203
<i>10.4. Віды тэставання на этапе тэсціравання ПД.....</i>	205
<i>10.5. Віды тэставання на этапе разгортвання ПД.....</i>	212
<i>10.6. Віды тэставання на этапе аблугоўвання ПД.....</i>	214
<i>10.7. Схема класіфікацыі тэсціравання ПД.....</i>	216
<i>11. Практыка.....</i>	219

# I. Гісторыя спрыяння якасці праграмнага дачынення

# 1. Гісторыя спрыяння якасці праграмнага дачынення

Спрыянне якасці праграмнага дачынення — СЯ ПД — як навуковая дысцыпліна развівалася на працягу некалькіх дзесяцігоддзяў, і яе гісторыю можна падзяліць на працяглыя перыяды, якія адлюстроўваюць істотныя змены ў тэхналогіях, метадалогіях і практиках.

Гэтыя перыяды падкрэсліваюць эвалюцыю тэсціравання праграмнага дачынення — ад ручной, спарадычнай дзейнасці да высокааутаматызаванай дысцыпліны, кіруемай штучным інтэлектам.

## Доўгатэрміновыя перыяды

Можна вылучыць пяць важных перыядоў спрыяння якасці праграмнага дачынення — СЯ ПД — да канца XX стагоддзя:

- 1947–1956 — Перыяд адладкі
- 1957–1978 — Перыяд дэманстрацыі
- 1979–1982 — Перыяд разбурэння
- 1983–1987 — Перыяд ацэнкі
- 1988–2000 — Перыяд прадухілення

Працягваючы гэтую тэндэнцыю ў ХХІ стагоддзе, можна асобна вызначыць тры наступныя доўгатэрміновыя эры СЯ ПД:

- 2001–2011 — Перыяд аўтаматызацыі тэсціравання
- 2012–2021 — Перыяд бесперапыннага тэсціравання
- 2022–Сёння — Перыяд тэсціравання на аснове ШІ

## 1.1. 1947–1956 Перыяд адладкі

Асноўная мэта перыяду адладкі заключалася ў выяўленні і вырашэнні недахопаў, або памылак, у праграмным дачыненні.

У адрозненне ад сучасных парадыгм тэсціравання, якія робяць акцэнт на прадухіленні і сістэматычнай праверцы, раннія намаганні былі пераважна сканцэнтраваны на рэактыўнай адладцы — выяўленні і выпраўленні памылак пасля таго, як яны прайвіліся ў кодзе.

### Гістарычны контэкст

Гэты перыяд адпавядае гадам станаўлення вылічальнай тэхнікі — часу, калі электронныя вылічальныя машыны перайшлі ад тэарэтычных канструкцый да практычных інструментаў, здольных выконваць запраграмаваныя інструкцыі.

Распрацоўка праграмнага дачынення знаходзілася ў зачатковым стапені, і структураваныя падыходы да спрыяння якасці яшчэ не з'явіліся.

Тэсціравання, як фармальнай дысцыпліны, не існавала.

Замест гэтага працэс у значнай ступені быў сіонімам адладкі — спецыяльнага метаду спроб і памылак, які гарантаваў, што праграмы функцыянуюць належным чынам.

### Ключавыя характеристыкі

Ключавыя характеристыкі перыяду адладкі наступныя:

- **Адладка, арыентаваная на распрацоўшчыка** — Паколькі спецыялізаванай ролі тэсціроўшчыка яшчэ не было створана, праграмісты самі адказвалі за праверку свайго кода.

Тэставанне было неад'емнай часткай цыклу распрацоўкі, а не асобным этапам.

- **Адсутнасць фармалізаваных інструментаў і метадаў** — У адрозненне ад сённяшніх складаных сістэм тэсціравання, ранняя адладка абапіралася на ручную праверку, вывад логаў і элементарныя метады дыягностыкі.

Не было аўтаматызаваных набораў тэсціравання, стандартызаваных методык або спецыялізаваных каманд па спрыянні якасці.

- **Базавая надзейнасць як галоўная мэта** — Галоўным паказчыкам поспеху было тое, ці выконвалася праграма без збояў або відавочных недакладнасцей у выніку.

Канцэпцыя комплекснай праверкі, напрыклад, аналізу памежных выпадкаў, бенчмаркінгу прадукцыйнасці або ацэнкі спажывецкага досведу, яшчэ не разглядалася.

## Bexi

Ніжэй прыведзены асноўныя вехі перыяду адладкі:

- **Паняцці памылак і адладкі** — У 1947 годзе былі прыдуманы тэрміны "памылка" і "адладка", калі інжынеры, што працавалі над камп'ютарам Harvard Mark II, выявілі сапраўдны моль, які заліп паміж контактамі рэле і замінаў яго нармальнай працы.

**Грэйс Мюрэй Хопер** падрабязна апісала інцыдэнт у працоўным журнале, прыклейшы моль скотчам у якасці доказу і назваўшы яго "жуком", які выклікаў памылку, а дзеянні па ліквідацыі памылкі — "адладкай".

- **Рэактыўны падыход да вырашэння проблем** — Адладка ўспрымалася як карэктыйная, а не прафілактычная мера.

Распрацоўшчыкі вырашалі праблемы толькі пасля іх узнікнення, часта ў адказ на бачныя збоі, замест таго, каб праактыўна распрацоўваць тэсты для выяўлення схаваных дэфектаў.

## **Роля перыяду адладкі**

Гэты пачатковы этап тэсціравання праграмнага дачынення заклаў аснову для будучых дасягненняў у галіне спрыяння якасці.

У той час тэсты былі сканцэнтраваны на абсталяванні, бо яно не было так развіта, як сёння, і яго надзеянасць была неабходнай для належнага функцыянування праграмнага дачынення.

Нягледзячы на прымітыўнасць па сучасных стандартах, акцэнт на адладцы замацаваў фундаментальны прынцып, што праграмнае дачыненне павінна быць старанна праверана на наяўнасць памылак — канцэпцыя, якая пазней ператварылася ў сістэматызаваныя метадалогіі тэсціравання, прыналежныя інструменты і асобныя прафесіі па спрыянні якасці.

---

## **1.2. 1957-1978 Перыяд**

### **дэманстрацыі**

Асноўная мэта дэманстрацыйнага перыяду зрушылася ў бок доказу таго, што праграмнае дачыненне працуе ў строгай адпаведнасці з запланаваным дызайнам і спецыфікацыямі.

У адрозненні ад папярэдняй эпохі, калі тэставанне было сіонімам рэактыўнай адладкі, у гэты перыяд акцэнт рабіўся на праверцы, якая гарантавала, што праграмнае дачыненне не толькі працуе без памылак, але і адпавядае загадзя вызначаным функцыянальным патрабаванням.

### **Гістарычны контэкст**

Па меры пашырэння маштабу, функцыянальнасці і крытычнасці праграмных сістэм абмежаванасць адладкі становілася ўсё больш відавочнай.

Арганізацыі зразумелі, што простай ліквідацыі збояў недастаткова — праграмнае дачыненне цяпер, безумоўна, неабходна для задавальнення бізнес- і аперацыйных патрэб.

У гэтую эпоху адбыўся паступовы пераход ад нефармальнага выпраўлення памылак пад кірауніцтвам распрацоўшчыкаў да больш структураваных — хоць і ўсё яшчэ зачатковых — намаганняў па праверцы правільнасці праграмнага дачынення.

Тэсціраванне заставалася адносна нефармальным у параўнанні з сучаснымі стандартамі, але яго мэта змянілася — зрушылася з выпраўлення дэфектаў да праверкі патрабаванняў.

Увага ўжо была сканцэнтравана не толькі на тым, каб "прымусіць праграму працеваць", але і на тым, каб прадэманстраваць, што яна працуе належным чынам пры пэўных умовах.

## Ключавыя характеристыкі

Ключавыя характеристыкі дэманстрацыйнага перыяду наступныя:

- **Падыход, арыентаваны на валідацыю** — Тэставанне ў першую чаргу прымянялася для пацверджання таго, што праграмнае дачыненне правільна выконвае свае функцыі, а не для агрэсіўнага выяўлення схаваных недахопаў.

Менталітэт схіляўся да пацверджання "Ці працуе гэта так, як належыць?" замест даследавання "Дзе гэта не працуе?".

- **Мысленне, накіраванае на пазбяганне дэфектаў** — Акцэнт рабіўся на доказе адсутнасці крытычных дэфектаў, што часта прыводзіла да аптымістычных ацэнак.

У адрозненні ад пазнейшых метадалогій, якія актыўна шукалі слабыя месцы — напрыклад, стрэс-тэставанне, памежны аналіз і г.д. — гэты падыход меркаваў правільнасць, пакуль доказы не сведчылі на адваротнае.

- **Тэставанне на позніх стадыях** — Тэставанне часта праводзілася бліжэй да канца распрацоўкі, часта ў якасці канчатковага кантрольнага пункта перад выпускам.

Гэты падыход "тэстуй апошнім" кантраставаў з сучасным ітэратыўным тэсціраваннем, дзе ацэнка адбываецца бесперапынна на працягу ўсяго жыццёвага цыклу распрацоўкі.

- **Ручное і адмысловое выкананне** — Нягледзячы на ўзрастающую патрэбу ў праверцы, тэсціраванню не хапала стандартызаваных метадалогій або аўтаматызацыі.

Большасць праверак праводзілася ўручную, абапіраючыся на інтуіцыю распрацоўшчыкаў або элементарныя тэставыя выпадкі.

## Bexi

- **Распрацоўка тэстаў** — У 1957 годзе Чарльз Бэйкер раслумачыў неабходнасць распрацоўкі тэстаў, накіраваных на дасягненне адпаведнасці праграмнага дачынення загадзя распрацаваным патрабаванням.

Такім чынам, было ўведзена адрозненне паміж контролем функцыянальнасці праграмнага дачынення — адладкай — і падтрымкай якасці праграмнага дачынення — тэсціраваннем — для таго, каб тэсціраванне праводзілася як асобная дзейнасць.

- **Важнасць тэстаў** — Распрацоўка тэстаў стала больш важнай, паколькі распрацоўваліся больш дарагі і складаныя праграмы, і кошт вырашэння ўсіх гэтых недахопаў уплываў на відавочную рызыку прыбытковасці праекта.

Асаблівая ўвага была надана павелічэнню колькасці і якасці тэстаў. Упершыню якасць дачынення пачала ўвязвацца са станам этапу тэставання.

Мэтай тэставання было прадэманстраваць, што ПД выконвае тое, для чаго яно пачаткова было прызначана, ужываючы чаканыя і пазнавальныя параметры.

- **З'яўленне верыфікацыі і валідацыі** — ВiB — Пачало выяўляцца адрозненне паміж праверкай "Ці правільна мы ствараем прадукт?" і валідацыяй "Ці правільны прадукт мы ствараем?".

Гэтая структура заклада аснову для пазнейшых дысцыплін спрыяння якасці.

- **Пастаянная залежнасць ад ручных працэсаў** — Нягледзячы на тое, што філософія тэсціравання развівалася, практыка заставалася працаёмкай і неструктураванай.

Адсутнасць сістэматычнага дызайну тэстаў — напрыклад, планаў тэставання, паказчыкаў пакрыцця — азначала, што эфекты ўнасць моцна адрознівалася ў розных праектах.

## Роля перыяду дэманстрацыі

Дэманстрацыйны перыяд прадстаўляў сабой крытычны пераходны этап, які зрушыў спрыянне якасці ПД з **рэактыўнай адладкі да наўмыснай праверкі**.

Аднак адсутнасць фармалізаваных працэсаў і ранній інтэграцыі тэсціравання азначала, што многія схаваныя дэфекты ўсё яшчэ не былі выяўленыя да разгортвання.

Гэтыя прабелы пазней прывялі да распрацоўкі метадычных сістэм тэсціравання, аўтаматызаваных інструментаў і праактыўных мер якасці, каб пераадолець недахопы перыяду адладкі ў наступныя перыяды тэсціравання.

---

## **1.3. 1979-1982 Перыяд**

### **разбурэння**

Перыяд разбурэння ўвасабляў сабой фундаментальную трансфармацыю ў падыходзе да спрыяння якасці праграмнага дачынення, дзе асноўная мэта змянілася ад доказу правільнасці да актыўнага пошуку і выяўлення дэфектаў.

Тэставанне больш не разглядалася праста як этап праверкі, а як сістэматычная спроба праверыць, аспрэчыць і наўмысна парушыць праграмнае дачыненне ў кантролюваных умовах.

Асноўная перадумова заключалася ў тым, што паляпшэнне якасці праграмнага дачынення патрабуе актыўнага выяўлення слабых месцаў перад выпускам, а не пасіўнага пацвярджэння функцыянальнасці.

### **Гістарычны кантэкст**

Па меры таго, як праграмныя сістэмы становіліся ўсё больш складанымі і крытычна важнымі, абмежаванні традыцыйнага тэсціравання, арыентаванага на верыфікацыю, сталі відавочнымі.

З'явіўся праактыўны настрой на пошук дэфектаў, які ўсведамляў наступнае:

- Надзейнасць праграмнага дачынення не магла быць выказана меркавана — яе трэба было сур'ёзна праверыць
- Адсутнасць назіраных збояў не азначала правільнасць — недахопы часта хаваліся ў неправераных сценарыях
- Псіхалогія чалавека паўплывала на эфектыўнасць тэсціравання — распрацоўшчыкі натуральным чынам пазбягалі тэстаў, якія маглі бы "узламаць" іх код

Перыяд разбурэння адзначыў прафесіяналізацыю тэсціравання як асобнай дысцыпліны, здзейнішы пераход ад

аптымістычнай дэмманстрацыі функцыянальнасці да песімістычнага, расследавальнага працэсу, накіраванага на выяўленне недахопаў.

## Ключавыя характарыстыкі

Ніжэй прыведзены ключавыя характарыстыкі перыяду разбурэння:

- **Фармалізацыя тэсціравання** — Тэсціраванне перайшло з нефармальной дзейнасці пасля распрацоўкі ў структураваны этап, інтэграваны ў жыццёвы цыкл распрацоўкі праграмнага дачынення.

Спецыяльныя тэставыя выпадкі, планы і дакументацыя сталі стандартнымі, што забяспечвала сістэматычную ацэнку, а не адмысловыя праверкі.

- Метадычныя метады тэсціравання — некаторыя метады ўяўлялі сабой раннія спробы сістэматызацыі дызайну тэстаў, якія выходзяць за рамкі метаду спроб і памылак і былі ўпершыню ўведзены ў перыяд знішчэння:
  - **Аналіз памежных значэнняў** — Boundary value analysis. BVA — Арыентаваны на тэставанне значэнняў уводу на межах яго дыяпазону, дзе часта збіраюцца дэфекты
  - **Эквівалентнае размежаванне** — Зніжэнне лішку тэставання шляхам групавання ўваходных данных, якія павінны выклікаць падобныя паводзіны, дзеля аптымізацыі тэставага пакрыцця
- **Мэта максімізацыі дэфектаў** — Поспех тэсціравання вымяраўся яго здольнасцю выяўляць недахопы, а не проста пацвярджаць чаканыя паводзіны.

Гэта патрабавала крэатыўнага, спаборніцкага мыслення — стварэння сцэнарыяў, якія ўжываюць патэнцыйныя недахопы ў логіцы, апрацоўцы ўводу або памежных выпадках.

- **Праактыўны настрой** — Тэсціраванне больш не заключалася ў чаканні збояў у прадукцыйнай частцы, а ў прэвентыўным прымушэнні да збояў у распрацоўцы.

Гэты зрух знізіў выдаткі і рызыку выяўлення дэфектаў на позніх стадыях.

## Bexi

- У 1979 годзе **Гленфард-Майерс**, з прыведзеным ніжэй вызначэннем, радыкальна перагледзеў працэдуру выяўлення памылак у праграме:

*"Тэсціраванне праграмнага дачынення — гэта працэс запуску праграмы з мэтай пошуку памылак."*

Майерс хваляваўся тым, што, імкнучыся прадэманстраваць бездакорнасць праграмы, можна падсвядома выбраць тэставыя даныя, якія маюць нізкую верагоднасць выкліку збою праграмы, тады як калі мэта складаецца ў тым, каб прадэманстраваць, што праграма мае недахопы, тэставыя даныя будуть мець большую верагоднасць іх выяўлення, і мы будзем больш паспяховымі ў тэсціраванні і, такім чынам, у спрыянні якасці праграмнага дачынення.

З тых часоў тэсты спрабуюць прадэманстраваць, што праграма **не працуе як трэба**, насуперак таму, як гэта рабілася да таго часу.

Гэтая пераарыентацыя заклала аснову для сучаснай тэорыі тэсціравання, аддаючы прыярытэт **выяўленню дэфектаў** перад **пацверджаннем якасці**, што прывяло да з'яўлення новых метадаў тэсціравання і аналізу.

- **Метадалогіі разбуразльнага контролю** — Філасофія Майерса стымулявала распрацоўку новых метадаў, спецыяльна прызначаных для:

- Выяўлення схаваных здагадак у кодзе
- Праверкі надзейнасць апрацоўкі памылак
- Выяўлення ўразлівасця ў памежных выпадках

Тэставанне больш не было пасіўным контрольным пунктам, а актыўным механізмам паляпшэння якасці.

## Роля ў перыядзе знішчэння

Перыяд разбурэння назаўсёды змяніў практыку распрацоўкі праграмнага дачынення, усталяваўшы асноўныя прынцыпы, якія захоўваюцца і сёння:

- **Тэставанне павінна быць скептычным** — мяркуючы, што дэфекты існуюць, пакуль не будзе доказана адваротнае
- **Якасць дасягаецца праз жорсткія выпрабаванні** — а не проста падцверджанне
- **Дызайн тэстаў — гэта асобны навык** — які патрабуе спецыялізаваных ведаў, акрамя навыкаў кадавання

Укараненне структурованых метадаў — BVA, эквівалентнага падзелу і г.д. — і псіхалагічныя ідэі Майерса заклалі аснову для наступных дасягненняў у аўтаматызаваным тэсціраванні, тэсціраванні на аснове рызыкі і бесперапынным спрыянні якасці.

---

## 1.4. 1983-1987 Перыяд ацэнкі

Перыяд ацэнкі прадстаўляў сабой фундаментальную трансфармацыю ў тым, як індустрыйя праграмнага дачынення канцэптуалізавала і ўкараняла кантроль якасці.

Замест таго, каб разглядаць тэсціраванне як канчатковую дзейнасць па кантролі доступу, арганізацыі пачалі ўжываць усеабдымную філасофію бесперапыннай ацэнкі якасці, якая ахоплівае ўесь жыццёвы цыкл распрацоўкі праграмнага дачынення.

Гэтая змена парадыгмы змяніла меркаванні аб якасці:

- Ад **рэактыўнасці** — да **праактыўнасці**
- Ад **фрагментаванасці** — да **інтэграванасці**
- Ад **увагі на дэфект** — да **увагі на якасць**

### Гістарычны контэкст

Да пачатку 1980-х некалькі збліжаных фактараў абумовілі наступную эвалюцыю:

- Павелічэнне складанасці праграмнага дачынення ў бізнес-сістэмах і крытычна важных сістэмах
- Рост выдаткаў, звязаных з ліквідацыяй дэфектаў пасля рэліза
- Удасканаленне разумення таго, што якасць нельга "праверыць", але неабходна ўбудаваць у працэс
- Паглыбленне разумення праграмнай інжынерыі як дысцыплінарнай практикі

Праграмная індустрыйя змяніла свой погляд на тэсціраванне:

- ад **асобнай сцэны** да **інтэграванага працэсу**
- ад **выяўлення дэфектаў** да механізма **спрыяння якасці**
- ад **канчатковай праверкі** да практикі **бесперапыннай ацэнкі**

## Ключавыя характеристыстыкі

Ключавыя характеристыстыкі перыяду ацэнкі наступныя:

- **Ацэнка якасці на працягу ўсяго жыццёвага цыклу** — Ацэнка якасці стала неад'емнай часткай кожнага этапу жыццёвага цыклу распрацоўкі праграмнага дачынення:

- Аналіз патрабаванняў — праверка тэставанасці
- Дызайн — архітэктурныя агляды
- Распрацоўка — модульнае тэставанне
- Разгортванне — тэставанне сістэмы
- Тэхнічнае абслугоўванне — рэгрэсійнае тэставанне

Распрацоўка атрымала V-вобразную форму, выразна адпавядаючы тэставым дзеянням кожнаму этапу распрацоўкі.

- **Фармалізацыя практикі тэсціравання** — З'явіліся стандартызаваныя метадалогі, напрыклад, IEEE 829:
  - Документацыя — планы тэставання, выпадкі, працэдуры — стала абавязковай
  - Былі ўведзены метрыкі і распрацаваны праграмы вымярэнняў
  - Распрацаваны спецыялізаваныя ролі ў галіне якасці і арганізацыйныя структуры
- **Спрыянне якасці як арганізацыйная функцыя** — Спрыянне якасці стала адрознівацца ад тэсціравання і ахопліваць:
  - Вызначэнне і ўдасканаленне працэсу
  - Мерапрыемствы, арыентаваныя на прафілактыку
  - Культура якасці арганізацыі
  - Метрыкі і бенчмаркінг

Канцэпцыя **Брамы якасці** ўсталявала контрольныя пункты на працягу ўсяго жыццёвага цыклу распрацоўкі праграмнага дачынення, дзе павінны быць выкананы пэўныя крытэрыі якасці, перш чым праект зможа перайсці да наступнага этапу.

- **Падыход да тэсціравання, арыентаваны на працэс:**
  - Паўторныя, вызначаныя працэсы тэсціравання замянілі спецыяльныя метады
  - Планаванне тэставання стала хутчэй прагназуемым, чым рэактыўным
  - Трасіровачныя матрыцы звязалі патрабаванні з тэставымі выпадкамі
  - Этапы тэсціравання рэгуляваліся фармальнымі крытэрыямі ўваходу і выхаду

## Bexi

- У 1983 годзе **Стандарт документацыі па тэсціраванні праграмнага дачынення IEEE 829** усталяваў адзіныя патрабаванні да документацыі, у тым ліку наступных:
  - Структура плана тэставання
  - Спецыфікацыі тэставых выпадкаў
  - Вызначэнне працэдуры выпрабаванняў
  - Патрабаванні да тэставага журнала
  - Фарматы паведамленняў аб інцыдэнтах
  - Справаздача аб выніках тэставання
- Стандарт увёў агульную мову для спецыялістаў па тэсціраванні, што дазволіла забяспечыць узгодненасць працэсаў ва ўсіх арганізацыях.
- З'яўленне **Аўтаматызаваных інструментаў тэсціравання** стала наступным значным дасягненнем перыяду ацэнкі, калі:
  - З'явіліся першыя фрэймворкі аўтаматызацыі тэсціравання
  - Рэгрэсійнае тэставанне набыло вялікія маштабы
  - Закладзены падмурак для сучаснага бесперапыннага тэсціравання
- Інстытуцыяналізацыя **Метрык якасці** дадала:

- Вымярэнне шчыльнасці дэфектаў
- Вызначэнне працэнта пакрыцця тэстамі
- Матрыцу адсочвання патрабаванняў
- Аналіз дэфектаў уцёкаў

Яны дазволілі колькасную ацэнку якасці.

- Спадчына і пераход да **Сучасных практик** — Перыйяд ацэнкі замацаваў асноўныя канцэпцыі, якія працягваюць фарміраваць практику якасці і сёння:
  - Прынцып, што якасць — гэта працэс, а не падзея
  - Разуменне, што тэставанне павінна быць спланаваным і кіраваным
  - Прызнанне, што якасць патрабуе арганізацыйных высілкаў
  - Неабходнасць супольнага падыходу для інтэграцыі тэсціравання на працягу ўсёй распрацоўкі

## Роля ў перыйядзе ацэнкі

Акцэнт гэтай эпохі на стандартызацыю, документацыю і арыентацыю на працэсы даў шлях наступным інавацыям, такім як:

- Інтэграцыя мадэлі сталасці магчымасцей
- Гнуткія метадалогіі тэсціравання
- Бесперапынная інтэграцыя/бесперапынная пастаўка — CI/CD
- Практикі якасці DevOps

Інстытуцыяналізацыя спрыяння якасці ў перыйяд ацэнкі ператварыла тэсціраванне ПЗ з тэхнічнай другараднай задачы ў прафесійную інжынерную дысцыпліну, усталяваўшы мадэлі і практикі, якія застаюцца актуальнымі і на працягу наступных дзесяцігоддзяў.

## **1.5. 1988-2000 Перыяд**

### **прадухілення**

Перыяд прадухілення адзначыў змену парадыгмы ў кіраванні якасцю праграмнага дачынення, пераходзячы ад рэактыўнага выяўлення дэфектаў да праактыўнага прадухілення дэфектаў.

Прамысловасць прызнала, што пошук памылак на позніх этапах цыклу распрацоўкі праграмнага дачынення з'яўляецца дарагім і неэфектыўным, што прывяло да стратэгічнага акцэнту на раннєе тэсціраванне, строгі контроль працэсаў і пастаяннае ўдасканаленне.

Тэсціраванне ператварылася з контрольнага пункта пасля распрацоўкі ў інтэграваную, ітэратыўную практыку, убудаваную на працягу ўсяго жыццёвага цыклу распрацоўкі праграмнага дачынення.

Мэта была ўжо не проста ў выяўленні недахопаў, а ў прадухіленні іх уznікнення шляхам сістэматычнага ўдасканалення працэсаў і ранній праверкі.

### **Гістарычны контэкст**

Такое пераутварэнне абумовілі некалькі ключавых фактараў:

- **Павелічэнне складанасці праграмнага дачынення** — Па меры таго, як праграмы становіліся ўсё больш маштабнымі і ўзаемазвязанымі, вырашэнне дэфектаў на позніх стадыях стала занадта дарагім
- **Кошт позніх дэфектаў** — Даследаванні паказалі, што выпраўленне памылак, выяўленых пасля выпуску праграмнага дачынення, абыходзілася ў 10–100 разоў даражэй, чым тых, якія былі выяўлены раней

- **Попыт на больш хуткія рэлізы** — З'яўленне бізнес-мадэляў, арыентаваных на Інтэрнэт, запатрабавала скарачэння цыклаў распрацоўкі, што зрабіла традыцыйныя падыходы "тэсціраваць да канца" састарэлымі
- **Сталасць праграмнай інжынерыі** — Дысцыпліна пачала ўкараняць фармалізаваныя перадавыя практыкі, натхнёныя контролем якасці вытворчасці

У гэты перыяд прафесіяналізацыя тэсціравання разглядалася як стратэгічная функцыя, а не як тактычная другарадная думка.

## Ключавыя характарыстыкі

Ключавыя характарыстыкі перыяду прафілактыкі наступныя:

- **Раннєе і бесперапыннае тэставанне** — або Shift-Left, зрух улева — Тэстывыя мерапрыемствы ў жыццёвым цыкле распрацоўкі ПЗ перанесены як мага раней з увядзеннем адпаведных практык:
  - **Агляды патрабаванняў** — Фармальныя праверкі для дасягнення яснасці, паўнаты і тэставанасці да пачатку кадавання
  - **Праверкі праектавання** — Дбайная ацэнка архітэктурных рашэнняў на прадмет патэнцыйных недахопаў
  - **Агляд кода і Рэв'ю кода** — Сумеснае прадухіленне дэфектаў з дапамогай ручнога аналізу кода

Дзякуючы гэтым метадам, да рэалізацыі ўдалося выявіць неадназначнасці і недахопы дызайну, што дазволіла скараціць колькасць пераробак.

- **Працэсна-арыентаванае спрыянне якасці:**
  - **Брама якасці** — Абавязковыя контрольныя пункты напрыканцы кожнага з этапаў для контролю выканання стандарттаў

- **Стандартызаваныя метадалогіі** — Укараненне такіх фрэймворкаў, як мадэль сталасці магчымасцей і гнуткая ітэрацыйная распрацоўка, якія дазволілі атрымліваць пастаянную зваротную сувязь праз паступовае тэсціраванне
  - **Паляпшэнне, заснаванае на метрыках**— Шчыльнасць дэфектаў, узровень выхаду з-пад контролю і пакрыццё тэстамі сталі ключавымі паказчыкамі эфектыўнасці
- **Паляпшэнні ў галіне інструментаў і аўтаматызацыі:**
  - **Інструменты статычнага аналізу** — Аўтаматызаваная праверка кода для ранняга выяўлення дэфектаў, напрыклад, інструменты ачысткі кода ад памылак
  - **Фрэймворкі для аўтаматызацыі тэсціравання** — З'явіліся спецыялізаваныя прадукты для аўтаматызацыі тэсціравання праграмнага дачынення і спрыяння якасці праграмнага дачынення
  - **Папярэднікі CI/CD** — Раннія інструменты бесперапыннай інтэграцыі і бесперапыннай распрацоўкі дазволілі рэгулярную валідацыю
- **Культурны зрух у бок адказнасці за якасць:**
  - **Сумесная адказнасць** — Распрацоўшчыкі, тэсціроўшчыкі і бізнес-аналітыкі сумесна працуюць над якасцю
  - **Прэвентыўны лад мыслення** — Каманды пільнуюць на тое, каб зрабіць усё слушна з першага разу, не спадзеючыся на далейшае тэсціраванне

## Bexi

- **Фармалізацыя тэставання са зрухам улева** — Прынцып ранняга і частае тэсціравання стаў краевугольным каменем сучасных методых распрацоўкі і паўплываў на пазнейшыя метадалогіі, такія як бесперапыннае тэсціраванне
- **Уздым ітэрацыйной Agile-распрацоўкі:**

- Метадалогії **Скрам**, 1995, і **Экстрэмальнаага праграмаванне**, 1996, ўкаранялі тэсціраванне ў кароткія цыклы
  - **Распрацоўка праз тэставанне** — з'явілася як практыка, арыентаваная на прадухіленне памылак
- **Развіццё інструментаў і фрэймворкаў для тэсціравання**
  - **Пашырэнне галіновых стандарттаў:**
    - **ISO 9000-3** — 1997 — Распрацаваў рэкамендацыі па спрыянні якасці падчас распрацоўкі ПЗ
    - **IEEE 1012** — 1998 — Стандартызаваў працэсы верыфікацыі і валідацыі

## Роля перыяду прадухілення

Перыяд прадухілення заклаў аснову для будучых удасканаленняў працэсу спрыяння якасці ПЗ:

- **DevOps** — Аб'яднанне распрацоўкі і аперацый з агульнай мэтай якасці
- **Зруч управа** — Shift-Right — Дапаўненне ранняга тэсціравання маніторынгам выніковай працы
- **Інжыніринг якасці** — Выходзячы за межы тэставання, ён ахоплівае прынцыпы "праектавання дзеля якасці"

Акцэнт гэтага перыяду на прадухіленні, аўтаматызацыі і дысцыпліне працэсаў ахінаецца цэнтральным у сучаснай праграмнай інжынерыі, што даказвае — праактыўнае спрыянне якасці значна больш эфектыўнае, чым рэактыўная адладка.

---

# 1.6. 2001-2011 Перыяд

## аўтаматызацыі тэсціравання

Першае дзесяцігоддзе ХХІ стагоддзя стала перыядам пераменаў у спрыянні якасці ПД, які характарызуецца шырокім распаўсюджваннем аўтаматызацыі і з'яўленнем новых парадыгм тэсціравання.

Па меры таго, як праграмныя сістэмы становіліся ўсё больш складанымі, а цыклы выпуску паскарабаліся, ручное тэсціраванне становілася ўсё больш непрактичным.

У гэты перыяд адбылося індустрыйлізацыя тэсціравання з дапамогай аўтаматызаваных фрэймворкаў, адаптаваных да Agile метадалогіі і ранніх інструментаў на базе штучнага інтэлекту, што заклала аснову для сучасных практык DevOps і бесперапыннага тэсціравання.

### Гістарычны контэкст

Некалькі ключавых фактараў спрыялі пераходу да сістэматычнага і інтэлектуальнага тэсціравання, арыентаванага на эфектыўнасць і маштабаванасць:

- **Вэб-выбух** — Бум дот-комаў і Web 2.0 запатрабавалі падтрымку кросбраўзернасці і кругласутачную надзейнасць
- **Гнуткая адаптацыя** — Карацейшыя цыклы распрацоўкі запатрабавалі больш хуткіх цыклаў зваротнай сувязі
- **Ціск на выдаткі** — Даследаванні паказалі, што аўтаматызацыя можа скараціць намаганні па рэгрэсійным тэсціраванні больш чым напалову
- **API-арыентаваная архітэктуры** — Сэрвісна-арыентаваная архітэктура зрабіла тэставанне API крытычна важнай задачай

## Ключавыя характеристыстыкі

Ніжэй прыведзены ключавыя інавацыі і метадалогіі эпохі аўтаматызацыі:

- **Распрацоўка праз тэставанне** — Test-Driven Development, TDD — Напісанне тэстаў **перед** кодам значна знізіла ўзровень дэфектаў і прымусіла распрацоўваць ПЗ модульна і тэставана
- **Распрацоўка праз паводзіны** — Behavior-Driven Development, BDD — Ліквідацыя разрыва між бізнесам і IT з ужыццём спецыфікацый звычайнай мовы і чытэльнага для чалавека сінтаксісу, а таксама тэсціравання, узгодненага з гісторыямі спажыўцу, і магчымасці праверкі логікі зацікаўленымі бакамі, якія не з'яўляюцца тэхнічнымі спецыялістамі
- **Selenium-рэвалюцыя** — 2004 — Selenium WebDriver вырашыў проблему крос-браўзернай аўтаматызацыі з падтрымкай розных моў распрацоўкі, і стаў фактычна стандартам для тэсціравання вэб-інтэрфейсаў
- **Сталасць тэсціравання API** — Переход ад арыентацыі на інтэрфейс да праверкі API дазволіў ранняе — са зрухам улева — інтэграцыі тэсціраванне і бенчмаркінг якаснасці
- **Воблачныя платформы тэсціравання** — Ліквідацыя выдаткаў на абслугоўванне лабараторый і мажлівасць тэсціравання на рэальных прыладах мела вырашальнае значэнне для распрацоўкі ПЗ для мабільных прылад

## Роля перыяду аўтаматызацыі

Перыяд аўтаматызацыі ператварыў тэсціраванне з ручной працы ў стратэгічную, тэхналагічна арыентаваную дысцыпліну, даказаўшы, што якасць і хуткасць дасяжныя дзякуючы інавацыям.

# **1.7. 2012–2021 Перыяд бесперапыннага тэсціравання**

Перыяд паміж 2012 і 2021 гадамі адзначыўся рэвалюцыйным зрухам у распрацоўцы і тэсціраванні ПД, абумоўленым шырокім распаўсюджваннем прынцыпаў DevOps і бесперапыннага тэсціравання.

Гэтая эпоха была вызначана канвергенцыяй распрацоўкі — **Dev**, тэставання — **QA**, і апераціўнае — **Ops**, пераўтвараючы традыцыйна ізаляваныя функцыі ў сумесны, аўтаматызаваны канвеер.

Асноўнай мэтай было паскарэнне распрацоўкі ПЗ пры падтрымцы высокай якасці, дасягнутай дзякуючы аўтаматызацыі, інфраструктурным інавацыям у і культурным зменам.

## **Гістарычны кантэкст**

Гэтая эвалюцыя была абумоўлена некалькімі тэндэнцыямі ў галіне:

- **Попыт на больш хуткія рэлізы** — Рост воблачных вылічэнняў і мадэляў "праграмнае дачыненне як паслуга" патрабаваў бесперапыннага разгортання, гэта значыць штодзённых або штогадзінных рэлізаў
- **Архітэктура мікрасэрвісаў** — Размеркаваныя сістэмы павялічылі складанасць, што ўскладніла скразное тэсціраванне
- **Гнуткая і маштабная** — Буйныя арганізацыі ўкаранілі Скрам, Kanban і Скрам Agile Framework — SAFe, — патрабуючы тэсціравання, каб ісці ў нагу з хуткімі ітэрацыямі
- **Кошт ручных працэсаў** — Вузкія месцы ў ручным тэсціраванні відавочна затрымлівалі выпуск рэлізаў

У гэты перыяд знікла мысленне "Перакінь гэта праз сцену" — яго замяніла сумесная адказнасць за якасць.

## Ключавыя характеристыстыкі

- **Зрух улева, Shift-Left**, або Раннє тэставанне:
  - Тэставанне ў жыццёвым цыкле ПЗ перамясцілася раней, распрацоўшчыкі сталі пісаць модульныя, інтэграцыйныя тэсты і API-тэсты разам з кодам.
  - Распрацоўка, арыентаваная на тэставанне (TDD), і распрацоўка, арыентаваная на паводзіны (BDD), сталі стандартам у Agile-камандах.
- **Зрух управа, Shift-Right**, або тэставанне ў вытворчасці, паширанае тэставанне рэліза праз маніторынг рэальных спажыўцуў, А/В-тэставанне і іншыя метады
- **Інфраструктура-як-код**:
  - **Інструменты кантэйнерызацыі** — напрыклад, Docker, 2013 — рэвалюцыянізаваў тэсціраванне і надаў:
    - Упакоўку праграм і залежнасцей у лёгкія, партатыўныя кантэйнеры
    - Забеспячэнне паслядоўнасці тэставых асяроддзяў, ліквідацыю проблем "працуе на маёй машыне"
  - **Інструменты аркестроўкі** — напрыклад, Kubernetes, 2014 — аўтаматызавала кіраванне кантэйнерамі, што дазволіла маштабаваць тэставыя кластары
  - **Інструменты Інфраструктуры-як-код** — такія як Terraform, Ansible — аўтаматызavalі стварэнне асяроддзяў, што скараціла час налады з дзён да хвілін

- **Канвееры CI/CD і аўтаматызаванае тэсціраванне** —  
Інструменты бесперапыннай інтэграцыі і бесперапыннай  
распрацоўкі дазволілі аўтаматызаваць:
  - лакальныя і воблачныя цыклы  
зборкі-тэставання-разгортвання
  - паралельнае выкананне тэстаў у розных асяроддзях
- **Інжынерыя надзеінасці сайта** — спалучыла выкананне  
аперацый і спрыянне якасці.

## Роля перыяду бесперапыннага тэсціравання

Гэты перыяд дэмакратызуваў тэсціраванне, зрабіўшы яго  
адказнасцю кожнага інжынера, а не толькі задачай контролю  
якасці.

Гэта ператварыла тэсціраванне з павольнага ручнога працэсу  
ў хуткую, аўтаматызованую і інтэлектуальную практику.

Інтэграцыя тэсціравання ў CI/CD, інфраструктуру і маніторынг  
надала аснову для сённяшніх аўтаномных практик, удасканаленых  
штучным інтэлектам, і адкрыла пачатак **Бесперапыннай якасці**.

Пастаяннае тэсціраванне стала хутчэй неабходнасцю, чым  
варыянтам, гарантуючы, што ПД адпавядае патрабаванням хуткай  
дастаўкі і высокім чаканням кліентаў.

Гэта даказала, што хуткасць і якасць — гэта не кампрамісы,  
але ўзаемна дасяжныя мэты.

# **1.8. 2022-Сёння Тэсціраванне на аснове штучнага інтэлекту**

Бягучы перыяд тэсціравання праз штучны інтэлект — ШІ — і машыннае навучанне — МН — кардынальна перавызначаюць тое, як тэсціраванне задумваецца, выконваецца і аптымізуецца.

**Тэсціраванне на аснове штучнага інтэлекту** — або тэсціраванне праз штучны інтэлект — гэта ўжытак ШІ і МН для аўтаматызацыі і паляпшэння працэсаў тэсціравання ПД.

Тэсціраванне больш не абмяжоўваецца сцэнарнымі праверкамі — яно ператварылася ў інтэлектуальны, прагназуемы і самаўдасканалываючы працэс, які ахоплівае ўвесь жыццёвы цыкл праграмнага дачынення.

## **Гістарычны контэкст**

Некалькі сумежных тэндэнций паскорылі гэтую трансфармацыю:

- **Сталенне штучнага інтэлекту** — Правыя ў глыбокім навучанні знайшлі практычнае прымяnenне за межамі даследчых лабараторый
- **Крызіс складанасці тэсціравання** — Па меры ўскладнення сістэм традыцыйныя метады сталі эканамічна неўстойлівыі
- **Выбух генератыўнага штучнага інтэлекту** — Выпуск ChatGPT у 2022 годзе даказаў патэнцыял ШІ не толькі для аналізу, але і для стварэння тэставых артэфактаў.

## **Ключавыя характеристыстыкі**

Ключавыя тэхналогіі, якія ляжаць у аснове тэсціравання на аснове штучнага інтэлекту:

- **Машыннае навучанне** — МН — Паляпшае тэставыя сцэнарыі з цягам часу, вывучаючы вынікі мінульых выкананняў
- **Апрацоўка натуральнай мовы** — Пераўтварае патрабаванні ў выглядзе звычайнага тэксту ў аўтаматызаваныя тэставыя выпадкі
- **Камп'ютэрны зрок** — Ужывае распознаванне малюнкаў для тэсціравання спажывецкага інтэрфейсу, напрыклад, для ідэнтыфікацыі дынамічных элементаў
- **Прагнастычная аналітыка** — Вызначае зоны высокай рызыкі, якія патрабуюць дадатковага тэсціравання

## Bexi

Ключавыя характеристыстыкі тэсціравання на аснове штучнага інтэлекту наступныя:

- **Аўтаматызацыя самааднаўлення тэстаў** — Традыцыйныя тэсты не спрацоўвалі з-за:
  - Дынамічных інтэрфейсаў
  - Неадпаведнасці асяроддзяў
  - Нестабільнай інтернет-сувязі

Інструменты ШІ ужываюць машыннае навучанне, каб:

- Выяўляць, калі змяняюцца лакатары элементаў
- Абнаўляць селектары, захоўваючы мэты тэставання
- Вучыцца на памылках і лепшыць устойлівасць тэстаў у будучыні

Гэта памяншае высілкі на абслугоўванне тэстаў і дазваляе ўстойліва аўтаматызаваць працу ў вялікіх маштабах.

- **Візуальная праверка з дапамогай камп'ютэрнага зроку:**
  - Традыцыйныя інструменты правяраюць структуру HTML

- Інструменты візуальнага ШІ ужываюць абортачныя нейрасеці для выяўлення візуальных перамен на ўзоруні пікселяў і ігнаравання неістотных змяненняў.
- **Аналіз уздзеяння тэстаў на базе машыннага навучання** —  
Інструменты МН могуць:
  - Ацаніць змяненні кода з дапамогай статычнага аналізу
  - Мадыфікаваць карту ужытых тэставых выпадкаў
  - Прыярытэзаваць выкананне тэстаў на аснове гістарычных даных аб дэфектах, паказыкаў складанасці кода і бізнес-крытычнасці
- **Аўтаматызацыя без ведання кода** зніжае бар'ер аўтаматызацыі наступнымі спосабамі:
  - Ужыванне апрацоўкі натуральнай мовы для перакладу простага маўлення ў тэставыя сцэнарыі
  - Прапанова візуальных інтэрфейсаў мадэлявання
  - Аўтаматычнае генерыраваць скрыпты падтрымкі

У выніку, аўтаматызацыя без кода:

- Дазваляе экспертам у предметнай вобласці ствараць тэсты
- Змяншае разрыў у навыках аўтаматызацыі
- Паскарае пашырэнне ахопу тэставання

## **Роля перыяду тэсціравання на аснове ШІ**

Сучасная эпоха змяняе не толькі тое, як мы тэстуем, але і тое, што значыць ствараць якаснае праграмнае дачыненне ў свеце, кіраваным штучным інтэлектам.

Арганізацыі, якія квітнеюць у гэтай новай парадыгме, разглядаюць якасць як стратэгічную перавагу, а не як сродак праверкі адпаведнасці.

У адрозненне ад традыцыйнай скрыптовай аўтаматызацыі, ШІ-тэсціраванне ўжывае алгарытмы ў наступных мэтах:

- Генерацыя тэстальных выпадкаў
- Самааднаўленне тэстальных скрыптоў
- Прагназаванне дэфектаў
- Аптымізацыя тэставага пакрыцця
- Аналіз вынікаў тэстаў

У выніку, найбольш відавочнымі перавагамі тэсціравання на аснове штучнага інтэлекту з'яўляюцца:

- **Хутчэйшае стварэнне тэстаў** — Штучны інтэлект генеруе тэсты на аснове патрабаванняў або паводзін спажыўцоў
- **Меншая патрабавальнасць** — Тэсты з самааднаўленнем мінімізуюць неабходнасць падтрымкі скрыптоў
- **Больш плённы запуск тэстаў** — Праз вызначэнне крытычных тэстальных выпадкаў
- **Палепшаная дакладнасць** — Зніжае колькасць ілжывага становішчаў і ілжывага адмоўных вынікаў
- **Пашыраны ахоп тэставання** — Штучны інтэлект даследуе памежныя выпадкі, якія людзі могуць прапусціць

Хоць штучны інтэлект і не з'яўляецца поўнай заменай тэсціроўшчыкам, ён значна скарачае ручную працу і павышае надзеінасць праграмнага дачынення.

ШІ не заменіць тэсціроўшчыкаў у бліжэйшай будучыні.

Але тэсціроўшчыкі, якія ўжываюць ШІ, безумоўна могуць замяніць тых, хто яго не ўжывае.

---

## **II. Якасць ПД**

## 2. Якасць ПД

### Якасць

**Якасць** — гэта набор уласцівасцей аб'екта, якія дазваляюць яму задавальняць відавочныя або няяўныя патрэбы.

### Якасць праграмнага дачынення

**Якасць праграмнага дачынення** — гэта ступень, у якой ПД надзейна выконвае запланаваныя аперацыі без памылак або адхіленняў ад заданых патрабаванняў.

Гэта павінна азначаць, што праграма не ўтрымлівае ўразлівасцей, якія былі альбо наўмысна ўбудаваны ў ПД, альбо выпадкова ўстаўлены ў любы момант яго жыццёвага цыклу.

### Спрыянне

**Спрыянне** — гэта вынік змен у прадукце або паслуже, які выклікае ўпэўненасць.

Яно гарантуе, што прадукт працуе належным чынам у адпаведнасці з чаканнямі або патрабаваннямі.

### Спрыянне якасці

**Спрыянне якасці** — СЯ — гэта спосаб прадухілення дэфектаў у створаных прадуктах і выключэння праблем пры пастаўцы прадуктаў або паслуг кліентам.

Спрыянне якасці сканцэнтравана на паляпшэнні працэсу распрацоўкі і павышэнні яго эфектыўнасці і дзеіснасці ў адпаведнасці са стандартамі якасці, вызначанымі для прадукцыі.

## **Спрыянне якасці праграмнага дачынення**

**Спрыянне якасці праграмнага дачынення** — СЯ ПД — гэта пастаянная каардынацыя інжынерных працэсаў, накіраваная на дасягненне належнай якасці праграмнага дачынення і яго адпаведнасці вызначаным стандартам.

## **Інжынер па спрыянні якасці ПД**

**Інжынер па спрыянні якасці ПД** — гэта інжынер, які спецыялізуецца на спрыянні якасці праграмнага дачынення ў адпаведнасці з патрабаваннямі і стандартамі на ўсіх этапах працэсу яго распрацоўкі.

---

## 2.1. Сферы якасці ПД

Якасць праграмнага дачынення ахоплівае тры крытычныя вобласці, кожная з якіх адыгрывае асобную ролю ў стварэнні надзейных прадуктаў:

- **Тэсціраванне якасці ПД** — гэта дзейнасць, накіраваная на выяўленне праблем у прадукце і правядзенне сістэматычных праверак для выяўлення дэфектаў, якія ахопліваюць праверкі функцыянальнасці, якаснасці і бяспекі
- **Кантроль якасці ПД** — гэта аканечны працэс выяўлення праблем у ПД перад яго дастаўкай канчатковым спажыўцам, які ўключае сканаванне дэфектаў шляхам праверак, аглядаў і маніторынгу ў рэжыме рэальнага часу
- **Спрыянне якасці ПД** — гэта працэс, які гарантуе, што ўсе працэсы, метады, дзеянні і рабочыя элементы распрацоўкі ПД кантралююцца, аптымізуюцца і адпавядаюць вызначаным стандартам



Software quality scopes

Разам гэтыя вобласці ўтвараюць цэласную структуру якасці, якая гарантуе, што праграмнае дачыненне эфектыўна адпавядае як тэхнічным патрабаванням, так і чаканням спажыўцу:

- **Тэсціраванне** — выяўляе дэфекты і правярае правільнасць
- **Кантроль** — кантралюе якасць і забяспечвае выкананне патрабаванняў
- **Спрыянне** — прадухіляе дэфекты і паляпшае працэсы

Інтэгруючы ўсе тры фактары, арганізацыі дасягаюць большай надзейнасці, зніжэння выдаткаў і ўстойлівага даверу кліентаў.

---

## 2.2. Тэставанне якасці ПД

**Тэставанне якасці ПД** — гэта арыентаваная на ацэнку дзейнасць, прызначаная для таго, каб упэўніць, што ПД адпавядае заданым патрабаванням, функцыянуе слушна і прапануе якасны спажывецкі досвед.

Гэта ўключае ў сябе выяўленне дэфектаў, праверку функцыянальнасці і ацэнку якаснасці, бяспекі, зручнасці ужытку і надзейнасці.

Тэсціраванне можа праводзіцца тэсціроўшчыкам або спецыяльнай камандай тэсціроўшчыкаў.

Гэты працэс можа таксама ўключаць этап планавання тэставання.

### Асноўныя мэты

Тэставанне якасці ПД кіруецца некалькімі ключавымі мэтамі, якія разам вядуць да стварэння надзейнага і трывалага прадукта.

Гэтыя мэты ўключаюць:

- **Выяўленне дэфектаў** — каб сістэматычна выяўляць і документаць памылкі, пахібы і неадпаведнасці да выпуску
- **Праверка патрабаванняў** — каб праверыць, ці адпавядае канчатковае ПД усім вызначаным бізнес-мэтам і спажывецкім патрабаванням
- **Ацэнка якаснасці** — каб ацаніць крытычныя нефункцыянальныя атрыбуты, такія як хуткасць, маштабаванасць і стабільнасць пры чаканых умовах нагрузкі
- **Спрыянне бяспечы** — каб праактыўна выяўляць уразлівасці бяспекі і забяспечваць абарону ПД ад патэнцыйных пагроз і парушэнняў

- **Праверка зручнасці ужытку** — каб ацаніць спажывецкі інтэрфейс і зручнасць спажывання, пераканаўшыся, што ПД інтуітыўна зразумелае, эфектыўнае і пасуе канчатковаму спажыўцу
- **Праверка адпаведнасці** — каб пацвердзіць, што ПД адпавядае ўсім дзеючым стандартам галіны, заканадаўчым нормам і ўнутраным палітыкам

## Роля тэсціравання ПД

Асноўная мэта тэсціравання — выяўленне збояў праграмнага дачынення так, каб можна было вызначыць і выправіць яго дэфекты.

Тэсціраванне якасці ПД:

- **Зніжае рызыкі** збояў ПД
- **Паляпшае вопыт** кліента
- **Зніжае выдаткі** доўгатэрміновага абслугоўвання
- **Забяспечвае адпаведнасць** з галіновымі правіламі

Роля тэсціравання праграмнага дачынення часта азначае праверку кода, а таксама яго выканання ў розных асяроддзях і ўмовах, кіруючыся наступнымі сцвярджэннямі:

- Тэсціраванне **не можа** вызначыць, што прадукт функцыянуе належным чынам ва ўсіх умовах
- Тэсціраванне **можа толькі** вызначыць, што яно не функцыянуе належным чынам у пэўных умовах.

Укараняючы строгія працэсы тэсціравання, арганізацыі могуць пастаўляць надзейныя, бяспечныя і высокаякасныя ПД.

---

## 2.3. Кантроль якасці ПД

**Кантроль якасці ПД** — арыентаваная на праверку дзейнасць, прызначаная для выяўлення і выпраўлення дэфектаў ПД шляхам праверкі адпаведнасці загадзя вызначаным стандартам якасці.

Кантроль якасці засяроджваецца на выяўленні адхіленняў ад патрабаванняў праз сістэматычнае тэсціраванне, агляды і праверкі перш, чым прадукт дасягне спажыўцу.

### Асноўныя мэты

Асноўныя мэты кантролю якасці наступныя:

- **Выяўленне дэфектаў** — Пошук памылак, пахібаў або неадпаведнасцей у кодзе, дызайне або дакументацыі
- **Адпаведнасць стандартам** — Дасягненне адпаведнасці функцыянальным патрабаванням, галіновым стандартам і арганізацыйным рэкамендацыям
- **Удасканаленне працэсу** — Праверка выканання працоўных працэсаў распрацоўкі, у тым ліку практик напісання кода і пратаколаў тэсціравання

### Асноўныя віды дзейнасці

Дзейнасць па кантролі якасці ПД звычайна ўключае ў сябе:

- **Тэсціраванне** — Выкананне тэст-кейсаў для выяўлення функцыянальных або нефункцыянальных дэфектаў
- **Рэцэнзіі калег** — Ручная праверка кода або дакументаў членамі каманды
- **Статычны аналіз** — Аўтаматызаваная праверка на наяўнасць проблем з якасцю кода без яго выканання
- **Дынамічны аналіз** — Маніторынг паводзін ПД падчас яго выканання

- **Аўдыты** — Фармальныя праверкі для праверкі адпаведнасці працэсам і стандартам

## Роля контролю якасці

Падчас контролю якасці праграмнага дачынення каманда тэсціравання правярае адпаведнасць прадукта функцыянальным патрабаванням і такім чынам:

- **Зніжае колькасць збояў пасля рэлізу** — падзенняў, парушэнняў бяспекі і г.д.
  - **Эканоміць выдаткі** — пазбягае карэкцыі дэфектаў пасля запуску, якая ў разы даражэйшая
  - **Забяспечвае давер спажыўцу** — шляхам дастаўкі надзейнага і якаснага ПД
-

## 2.4. Спрыянне якасці ПД

**Спрыянне якасці праграмнага дачынення** — СЯ ПД — гэта планавая і комплексная арыентаваная на валідацыю дзеянасць, скіраваная на выкананне патрабаванняў да якасці прадукта з мэтай далейшага ўдасканалення сістэмы якасці.

СЯ ПД гарантуе, што ўсе працэсы, метады, дзеянні і рабочыя элементы распрацоўкі праграмнага дачынення кантралююцца, аптымізуюцца і адпавядаюць вызначаным стандартам.

Спрыянне якасці праграмнага дачынення ахоплівае ўсе працэсы распрацоўкі ПД, пачынаючы ад вызначэння патрабаванняў да кадавання і выпуску ў свет.

### Ключавая роля

Спрыянне якасці праграмнага дачынення заснавана на інжынерных працэсах, якія гарантуюць якасць больш эфектыўным спосабам, чым кантроль якасці праграмнага дачынення.

Тэставанне якасці праграмнага дачынення і кантроль якасці праграмнага дачынення здольныя выявіць асноўную колькасць праблем у прадукце, але гэта не азначае, што гэтыя дэфекты не паўторацца ізноў.

Роля СЯ ПД заключаецца ў перапрацоўцы сістэмы, каб прадухіліць далейшае ўзнікненне гэтых дэфектаў; і на самой справе, яно можа наогул не ўключыць ніякага тэсціравання.

Галоўная мэта спрыяння якасці праграмнага дачынення — гарантаваць якасць праграмных прадуктаў або паслуг, якія арганізацыя прадастаўляе кліентам.

## **Кантроль якасці і спрыянне якасці**

Кантроль якасці праграмнага дачынення адрозніваецца ад спрыяння якасці праграмнага дачынення.

Кантроль якасці праграмнага дачынення — гэта праверка адпаведнасці якасці ўстаноўленым крытэрыям — іншымі словамі, **пошук дэфектаў**.

Спрыянне якасці праграмнага дачынення ахоплівае працэсы і стандарты для пастаяннага падтрымання высокай якасці прадукта, напрыклад, ПД, документацыі і працэсаў — іншымі словамі, **пазбяганне дэфектаў**.

Розніцу паміж кантролем якасці праграмнага дачынення і спрыяннем якасці ПД можна выказаць наступным чынам:

- SQC — гэта дзейнасць, арыентаваная на **прадукт**
- SQA — гэта дзейнасць, арыентаваная на **працэс**

Кантроль якасці праграмнага дачынення адсочвае адпаведнасць **вынікаў** распрацоўкі ПД яго першапачатковым патрабаванням.

Спрыянне якасці праграмнага дачынення адсочвае адпаведнасць усіх **працэсаў** распрацоўкі ПД запатрабаванаму належнаму чыну.

---

## 2.5. Характарыстыкі якасці ПД

Характарыстыкі якасці ПД — альбо так званыя атрыбуты якасці, ці нефункцыянальныя патрабаванні — гэта ўласцівасці ПД, якія вызначаюць яго мэтавую плённасць, зручнасць абслугоўвання, адпаведнасць патрабаванням і маюць мажлівасць вымярэння.



Ключавыя характеристыкі якасці праграмнага дачынення

Гэтыя харкторыстыкі служаць эталонамі для ацэнкі таго, наколькі добра праграмнае дачыненне працуе, паводзіць сябе і адпавядзе чаканням зацікаўленых бакоў.

## **Функцыянальнасць**

**Функцыянальнасць** — гэта здольнасць праграмнага дачынення несці пэўныя ўласцівасці.

Функцыянальнасць ахоплівае:

- Прыдатнасць
- Цэласнасць
- Бяспека
- Дакладнасць і іншыя ўласцівасці

## **Зручнасць спажывання**

**Зручнасць спажывання** — гэта здольнасць праграмнага дачынення быць лёгкім для разумення і ужытку.

Зручнасць ужытку пераважае:

- Працаздольнасць
- Навучальнасць
- Даступнасць
- Прывабнасць і гэтак далей

## **Эфектыўнасць**

**Эфектыўнасць** — гэта сувязь паміж узроўнем якаснасці праграмнага дачынення і колькасцю неабходных рэсурсаў.

Эфектыўнасць прадугледжвае:

- Ёмістасць
- Паводзіны ў часе

- Спажыванне рэурсаў
- Эфектыўнасць і гэтак далей

## **Надзейнасць**

**Надзейнасць** — гэта здольнасць праграмнага дачынення працягаць функцыянаваць у заданых умовах на працягу заданага перыяду часу.

Дыяпазоны надзейнасці:

- Аднаўляльнасць
- Даступнасць
- Стабільнасць
- Паслядоўнасць і г.д.

## **Падtrzymальнасць**

**Падtrzymальнасць** — гэта намаганні, неабходныя для ўнясення пэўных змяненняў.

Падtrzymальнасць улічвае:

- Маштаванасць
- Паўторнае ужыццё
- Модульнасць
- Тэставанасць і іншое

## **Партатыўнасць**

**Партатыўнасць** — гэта здольнасць праграмнага дачынення пераносіцца з аднаго асяроддзя ў іншае.

Партатыўнасць распаўсюджваецца на:

- Магчымасць усталёўкі
- Замяняльнасць

- Сумяшчальнасць
- Суіснаванне і г.д.

## Ключавая роля

Разуменне і прыярыйтэтызацыя харектарыстык якасці праграмнага дачынення з'яўляеца фундаментальнай дзейнасцю ў праграмнай інжынерыі.

---

## **2.6. Крытэрыі ўваходу і выхаду**

**Крытэрыі** — гэта прынцып або стандарт ацэнкі чаго-небудзь.

Кожны этап жыццёвага цыклу распрацоўкі і тэсціравання ПД мае свае ўласныя крытэрыі ўваходу і выхаду.

### **Крытэрыі ўваходу**

**Крытэрыі ўваходу** — гэта крытэрыі, якія павінны быць выкананы перад пачаткам этапу распрацоўкі або тэстування.

Гэта загадзя вызначаны набор умоў, якія павінны існаваць да пачатку распрацоўкі або тэсціравання.

Ён ужываецца ў якасці механізма кіравання працэсамі для вызначэння эканамічнай эфектыўнасці пачатку пэўнага этапу.

Каманда павінна перайсці ў наступны этап толькі пасля таго, як будуть выкананы крытэрыі выхаду з папярэдняга.

### **Крытэрыі выхаду**

**Крытэрыі выхаду** — гэта крытэрыі, якія павінны быць выкананы перад завяршэннем канкрэтнай задачы або працэсу.

Гэта загадзя вызначаны набор умоў, якія павінны выканацца, перш чым пэўны этап можна лічыць завершаным.

Ён ужываецца ў якасці механізма кантролю працэсаў для праверкі таго, што этап распрацоўкі або тэстування быў завершаны і што якасць прадукцыі прымальная.

Крытэрыі выхаду вызначаюць вынікі, якія павінны быць выкананы да таго, як этап будзе пакінуты.

## **III. ЖЫЩЁВЫ ЦЫКЛ распрацоўкі ПД**

### **3. Жыццёвы цыкл распрацоўкі**

#### **ПД**

**Жыццёвы цыкл распрацоўкі ПД** — ЖЦР ПД — гэта працэс пабудовы праграмнага дачынення з намерам спраектаваць, распрацаваць і пратэсціраваць ПД найвышэйшай якасці, з найменшымі выдаткамі і ў самыя кароткія тэрміны.

ЖЦР ПД — таксама называны **Працэс распрацоўкі ПД** — можна назваць фрэймворкам для вызначэння задач, якія выконваюцца на кожным этапе вытворчасці праграмнага дачынення.

#### **Этапы SDLC**

Распрацоўка праграмнага дачынення структуравана ў некалькі этапаў, і жыццёвы цыкл распрацоўкі праграмнага дачынення служыць для іх ахопу.

Цыкл не завяршаецца, пакуль не будуць выкананы ўсе патрабаванні, і працягваецца, пакуль усе патэнцыйныя патрэбы не будуць скарэкціраваны ў сістэме.

ЖЦР ПД забяспечвае добра структурованы паток этапаў, якія дапамагаюць арганізацыі хутка ствараць якаснае, надзейна пратэсціраванае і гатовае да ўжытку праграмнае дачыненне.

Жыццёвы цыкл распрацоўкі праграмнага дачынення складаецца з падрабязнай паслядоўнасці кроکаў, якія апісваюць, як распрацоўваць, падтрымліваць, змяняць і замяняць конкретнае праграмнае дачыненне, і ўключае наступныя этапы:

1. Планаванне ПД
2. Праектаванне ПД
3. Распрацоўка ПД
4. Тэсціраванне ПД

5. Разгортванне ПД
6. Абслугоўванне ПД



## Перавагі

ЖЦР ПД працуе за кошт зніжэння кошту распрацоўкі праграмнага дачынення, адначасова паляпшаючы якасць і скарачаючы час вытворчасці.

ЖЦР ПД дасягае гэтых, здавалася б, супярэчлівых мэтаў, прытрымліваючыся плана, які ліквідуе тыповыя недахопы праектаў распрацоўкі ПД.

ЖЦР ПД мае моцны акцэнт на этапе тэсціравання — з'яўляючыся паутаральнай методыкай, яна гарантуе якасць кода на кожным цыкле.

Найбольшая перавага жыццёвага цыклу распрацоўкі праграмнага дачынення заключаецца ў тым, што ён забяспечвае контроль над працэсам распрацоўкі ў пэўнай ступені і гарантуе, што сістэма адпавядае ўсім заданным патрабаванням.

## Недахопы

ЖЦР ПД працуе не так добра там, дзе ёсць узровень нявызначанасці або непатрэбныя накладныя выдаткі.

Ён накіроўвае намаганні па распрацоўцы з акцэнтам на планаванне, але сістэма не заахвочвае творчы ўнёсак або інавацыі на працягу ўсяго жыццёвага цыклу.

---

## **3.1. Этап планавання ПД**

**Планаванне ПД** — гэта пачатковы і фундаментальны этап ЖЦР ПД, асноўная мэта якога — пераўтварыць расплывістую ідэю або бізнес-неабходнасць ў добра акрэсленую, жыццяздольную і афіцыйна зацверджаную прапанову праграмнага праекта.

Этап адказвае на асноўныя пытанні:

- Што мы будуем?
- Чаму мы гэта будуем?
- Ці магчыма і ці варта гэта будаваць?

Планаванне праграмнага дачынення сканцэнтравана на даследаванні і прыняцці рашэнняў на высокім узроўні, вызначэнні стратэгічнага кірунку і межаў для ўсяго праекта да таго, як будуць выдзелены значныя рэсурсы.

Звычайна гэты этап прадугледжвае наступныя падрыхтоўчыя мерапрыемствы:

- Канцэпцыя праграмнага дачынення
- Аналіз патрабаванняў

### **Канцэпцыя праграмнага дачынення**

**Канцэпцыя праграмнага дачынення** — або **Распрацоўка ідэй праграмнага дачынення** — гэта творчы і даследчы працэс стварэння, вызначэння і ўдасканалення праграмнага рашэння для ўяўнай проблемы або бізнес-магчымасці.

Гэта падэтап, на якім праект запускаецца.

Распрацоўка канцэпцыі праграмнага дачынення ўключае ў сябе:

- **Генерацыя ідэй** — Мазгавы штурм патэнцыйных рашэнняў, функцый або цалкам новых прадуктаў

- **Вызначэнне праблемы** — Выразнае фармуляванне канкрэтнай праблемнай кропкі, прабелу на рынку або неэфектыўнасці, якую праграмнае дачыненне будзе вырашаць
- **Пошук рашэнняў** — Даследаванне існуючых рашэнняў, аналіз канкурэнтаў і разгляд розных тэхналагічных падыходаў
- **Пачатковае бачанне** — Фарміраванне агульнага бачання таго, што будзе рабіць праграмнае дачыненне і для каго, часта зафіксаванае ў кароткай заяве аб бачанні

## **Аналіз патрабаванняў**

**Аналіз патрабаванняў** — гэта працэс сістэматычнага выяўлення, аналізу, спецыфікацыі і праверкі патрэб і амежавань, якім павінна адпавядаць новая праграмная сістэма.

Гэты падэтап пераходзіць ад расплывістай ідэі да канкрэтнага набору спецыфікацый і ўключае ў сябе:

- **Выяўленне** — Збор інфармацыі ад зацікаўленых бакоў праз інтэрв'ю, семінары і апытани
- **Аналіз** — Удакладненне, прыярытэтызацыя і структураванне сабранай інфармацыі для вырашэння канфліктаў і няпэўнасцей
- **Спецыфікацыя** — Документаванне патрабаванняў у зразумелай, адназначнай і праверанай форме

Аналіз патрабаванняў з'яўляецца асновай паспяховай распрацоўкі праграмнага дачынення, бо ён гарантуе адпаведнасць канчатковага прадукту бізнес-мэтам, патрэбам кліентаў і тэхнічнай магчымасці.

## **Асноўныя мэты**

**Планаванне праграмнага дачынення** — гэта працэс вызначэння дарожнай карты праекта, уключаючы яго аб'ём, рэсурсы, графік, бюджет і рызыкі, для стварэння дзейснага плана выканання.

Гэты этап вызначае траекторыю ўсяго праекта, забяспечваючы ўзгадненне паміж зацікаўленымі бакамі, рэалістычныя чаканні і выразную дарожную карту выканання.

## **Асноўныя віды дзеянасці**

Планаванне праграмнага дачынення складаецца з наступных асноўных мерапрыемстваў:

- **Вызначэнне вобласці** — Фармальнае дакументаванне мэтаў, вынікаў, задач, выдаткаў і тэрмінаў праекта ў Статуце праекта:
  - У межах або па-за межамі праекта — выразнае вызначэнне межаў праекта
  - Структура дэкампазіцыі работ — іерархічная дэкампазіцыя паставак
  - Заява аб аб'ёме дзеяння — Фармальнае пагадненне аб прадухіленні "узняцця аб'ёму дзеянь"
- **Планаванне рэурсаў** — Вызначэнне неабходных чалавечых, тэхнолагічных і фінансавых рэурсаў:
  - Структура каманды — распрацоўшчыкі, тэсціроўшчыкі, менеджеры па праекце і іншыя пасады з іх абавязкамі
  - Тэхнолагічны стэк — мовы праграмавання, фрэймворкі, базы даных
  - Разлік бюджету — выдаткі на распрацоўку, тэставанне, інфраструктуру і абслугоўванне

- **Ацэнка рызык і іх змяншэнне** — Вызначэнне патэнцыйных тэхнічных, фінансавых і аперацыйных рызык з пачатковым вызначэннем стратэгіі іх змякчэння:
  - Вызначэнне рызык — вызначэнне патэнцыйных пагроз
  - Аналіз рызык — ацэнка ўздзеяння і верагоднасці
  - Стратэгія змякчэння рызык — Планаванне на выпадак надзвычайных сітуацый для высокапрыярытэтных рызык
- **Графік і планаванне этапаў** — Створэнне агульнага графіка з ключавымі этапамі:
  - Дыяграммы Ганта і дарожныя карты — візуальныя храналагічныя шкалы для кожнага этапу SDLC
  - Agile Sprints — вызначэнне працягласці спрынтаў і прыярытэтаў задач
  - Метод крытычнага шляху — Вызначэнне задач, якія могуць затрымаць проект
- **Тэхніка-эканамічнае аргументаванне** — Ацэнка жыццяздольнасці праекта па некалькіх аспектах — тэхнічным, эканамічным, аперацыйным і юрыдычным:
  - Тэхнічная магчымасць — ці можна пабудаваць з ужыццём сапраўднай тэхналогіі?
  - Эканамічная мэтазгоднасць — на сколькі дарагім будзе прадукт?
  - Эксплуатацыйная мэтазгоднасць — ці спадабаецца рашэнне канчатковым спажыўцам?
  - Юрыдычная мэтазгоднасць — ці адпавядае гэта рэгулюючым нормам?
- **Зацвярджэнне і пачатак** — Уключэнне жыццёвага цыклу распрацоўкі праграмнага дачынення:
  - Статут праекта — дазваляе рэалізаваць праект і размяркоўвае рэсурсы

- Узгадненне з зацікаўленымі бакамі — афіцыйнае пагадненне аб аб'ёме, бюджетце і тэрмінах
- Уступная сустрэча — узгадненне мэтаў і працэсаў усіх каманд

Разуменне патрабаванняў да спрыяння якасці і вызначэнне рызык, звязаных з проектам, з'яўляюцца найважнейшымі задачамі дасягнення якасці на этапе планавання праграмнага дачынення.

## Крытэрыі ўваходу і выхаду

Асноўныя падзеі, якія запускаюць этап планавання праграмнага дачынення, ўключаюць:

- Вызначэнне патрэб бізнесу — распазнанне магчымасці
- Стратэгічная ініцыятыва — адпаведнасць мэтам арганізацыі
- Запыт зацікаўленых бакоў — афіцыйны запыт ад бізнес-падраздзяленняў
- Тэхналагічныя магчымасці — ужыванне новых тэхналогій

Вынікі, якія сведчаць аб гатоўнасці ПД да наступнага этапу, залежаць ад падыходу да кіравання проектам і могуць уключаць:

- **Документ з патрабаваннямі да бізнесу** — спецыфікацыя бізнес-мэтаў высокага ўзроўню як набор функцыянальнасцей, якім ПД павінна адпавядаць для дасягнення поспеху
- **Спецыфікацыя патрабаванняў да праграмнага дачынення** — документ для выразнага вызначэння і запісу патрабаванняў да ПД і іх зацвярджэння заказчыкам або аналітыкамі.
- **Документ з функцыянальнымі патрабаваннямі** — спецыфікацыя, якая ўключае падрабязную інфармацыю пра паводзіны сістэмы
- **Гісторыі спажыўцу і выпадкі ўжытку** — документы, якія апісваюць функцыі з пункту гледжання канечнага спажыўца

- **Статут праекта** — сціслы дакумент, які афіцыйна пацвярджае існаванне праекта і дае яго кіраўніку паўнамоцтвы ужываць арганізацыйныя рэсурсы для дзейнасці па праекце.
- **План праекта** — агульная дарожная карта з тэрмінамі, этапамі і рэсурсамі
- **Справаздача аб тэхніка-эканамічным абгрунтаванні** — справаздача, якая абгрунтоўвае тэхнічную, фінансавую і юрыдычную жыццяздольнасць праекта
- **План кіравання рызыкамі** — дакумент, які вызначае рызыкі і стратэгіі іх змяншэння

## Роля этапу планавання

Паспяховае выкананне этапу планавання завяршаецца прыняццем афіцыйнага рашэння "завершана — не завершана", у якім зацікаўленыя бакі вырашаюць, ці варта ўхваляць Статут праекта і выдзяляць рэсурсы на этапы падрабязнага праектавання і распрацоўкі.

Этап планавання з'яўляецца краевугольным каменем паспяховай распрацоўкі праграмнага дачынення, які пераўтварае абстрактныя ідэі ў дзеісныя стратэгіі.

Укладваючы час у дбайнае абмеркаванне ідэй, аналіз патрабаванняў і планаванне, каманды могуць пазбегнуць распаўсядженых памылак, аптымізаваць рэсурсы і своечасова рэалізоўваць праекты ў рамках бюджету.

---

## 3.2. Этап практавання ПД

**Практаванне ПД** — гэта этап апісання пажаданых функцый і аперацый сістэмы.

Мэта этапу практавання — вызначыць тып кліентаў і сервераў, неабходных для тэхнічнай рэалізацыі сістэмы.

### Асноўныя віды дзейнасці

Этап складаецца з наступных асноўных мерапрыемстваў:

- Практаванне архітэктуры сістэмы
- Практаванне ІТ-інфраструктуры

Практаванне праграмнага дачынення можа ўключаць:

- Іерархічныя дыяграмы
- Макеты інтэрфейса экранаў
- Дыяграмы "сутнасць-сувязь"
- Дыяграмы працэсаў
- Слоўнікі даных
- Бізнес-правілы
- Псеўдакод

— і іншыя неабходныя звесткі, сабраныя ў адпаведнасці са спецыфікацыяй практавання праграмнага дачынення.

### Асноўныя мэты

На этапе практавання сістэма павінна быць апісана як сукупнасць модуляў — або падсістэм — у адпаведнасці з патрабаваннямі, вызначанымі ў зацверджанай спецыфікацыі патрабаванняў да праграмнага дачынення.

Практаванне праграмнага дачынення заключаецца ў выразным вызначэнні ўсіх архітэктурных модулей прадукта разам

з прадстаўленнем яго сувязі са знешнім і староннім модулямі, а таксама патокаў данных.

## Крытэрыі выхаду

Выкананая Спецыфікацыя праектавання праграмнага дачынення з'яўляецца асноўным крытэрыем выхаду на этапе праектавання праграмнага дачынення.

**Спецыфікацыя праектавання праграмнага дачынення** — гэта прадстаўленне праекта ПД, прызначанае для захоўвання праектнай інфармацыі, вырашэння розных проблем і перадачы сабранных данных зацікаўленым у праектаванні бакам.

Спецыфікацыя праектавання праграмнага дачынення часта называецца наступным чынам:

- Апісанне праектавання праграмнага дачынення
- Документ па праектаванні праграмнага дачынення
- Праектны документ

— і ўключае, як правіла, **Дыяграму архітэктуры**, якая кранаеца дробных элементаў праектавання.

---

## 3.3. Этап распрацоўкі ПД

**Распрацоўка праграмнага дачынення** — гэта этап, на якім абстрактны праект і патрабаванні увасабляюцца ў функцыянальны працоўны код.

Асноўная мэта этапу распрацоўкі — увасобіць падрабязныя спецыфікацыі праекта ў працоўны праграмны прадукт шляхам сістэматычнага праграмавання, забяспечваючы пры гэтым якасць кода, яго падтрымку і адпаведнасць патрабаванням.

Этап распрацоўкі ўключае ў сябе наступныя асноўныя кампаненты, якія належыць стварыць

- Код
- Базы даных
- Інфраструктура

### Асноўныя віды дзейнасці

Дзейнасць на першасным этапе распрацоўкі звычайна павінна ўключаць:

- **Кадаванне або праграмаванне** — Напісанне зыходнага кода на выбраных мовах праграмавання — Python, JavaScript і інш.
- **Модульнае тэсціраванне** — Проверка асобных кампанент, якія называюцца модулямі, каб асобна пераканацца ў іх належнай працы
- **Агляд кода** — Калегіяльная проверка кода для паляпшэння якасці, абмену ведамі і ранняга выяўлення дэфектаў
- **Кантроль версій** — Кіраванне зменамі кода з дапамогай такіх сістэм, як Git, SVN або Mercurial
- **Інтэграцыя** — Аб'яднанне асобных праграмных модуляў у цэласную сістэму

Дадатковыя мерапрыемствы на этапе распрацоўкі могуць наступнымі:

- **Рэалізацыя базы даных** — Стварэнне і запаўненне баз даных у адпаведнасці з праектнымі заданнямі
- **Распрацоўка API** — Стварэнне ўнутраных і зневідных інтэрфейсаў
- **Кіраванне канфігурацыяй** — Кіраванне наладамі і параметрамі, спецыфічнымі для асяроддзя
- **Адладка** — Выяўленне і выпраўленне дэфектаў на ўзоруны кода

Праграмнае і аппаратнае дачыненне таксама неабходна набыць і ўсталяваць на гэтым этапе.

## Уводы і вывады

Распрацоўшчыкі грунтуюць сваю працу на наступных даных:

- **Падрабязныя праектныя дакументы** — Тэхнічныя характеристыстыкі, схемы баз даных, контракты API
- **Дызайн UI/UX** — Схемы, макеты, правілы стылю
- **Дыяграмы архітэктуры** — Кампаненты сістэмы і іх взаемадзеянне
- **Налада асяроддзя распрацоўкі** — Серверы, IDE, інструменты

Асноўныя вынікі этапу распрацоўкі звычайна ўключаюць:

- **Зыходны код** — Код з контролем версій і дакументацыяй
- **Тэставыя выпадкі і вынікі** — Аўтаматызаваныя тэсты на справаздачамі аб поспехах/няўдачах
- **Тэхнічная документацыя** — Коментарыі да кода, дакументацыя API

- **Стварэнне артэфактаў** — Выканальныя файлы, пакеты, кантэйнеры

## Мовы праграмавання

На этапе распрацоўкі мова праграмавання — МП — выбіраецца ў залежнасці ад тыпу распрацоўваемага праграмнага дачынення.

Каб стварыць код, распрацоўшчыкі павінны таксама прытрымлівацца рэкамендацый па кадаванні, вызначаных іх арганізацыяй, і такіх інструментах праграмавання, як кампілятары, інтэрпрэтатары і адладчыкі.

## Крытэрыі выхаду

Этап распрацоўкі мае сваё выразнае вызначэнне гатоўнасці:

- Усе функцыі рэалізаваны ў адпаведнасці са спецыфікацыямі
  - Напісаныя і зданыя модульныя тэсты з дасягненнем мэставага пакрыцця
  - Праверка кода завершана для ўсіх змяненняў
  - Тэставанне інтэграцыі паспяховае
  - Тэхнічная документацыя абноўлена
  - Гатоўнасць да наступнага этапу тэсціравання
-

## 3.4. Этап тэсціравання ПД

**Тэсціраванне ПД** — гэта этап, прызначаны для сістэматычнай ацэнкі і праверкі таго, што ПД адпавядзе зададным патрабаванням, функцыянуе слушна і забяспечвае якасны спажывецкі досвед.

Дзейнасць па тэсціраванні, з большага, уключана ва ўсе этапы жыццёвага цыклу распрацоўкі праграмнага дачынення.

Аднак этап тэсціравання тычыцца толькі тэсціравання прадукту.

Этап тэсціравання ПД служыць асноўнай **Брамай якасці** перад выпускам у вытворчасць, паколькі дэфекты выяўляюцца і выпраўляюцца, покуль прадукт не дасягне ўмоў якасці, вызначаных у спецыфікацыі патрабаванняў да праграмнага дачынення.

### Асноўная дзейнасць

Падчас этапу тэставання ўсе фрагменты кода інтэгруюцца і разгортаюцца ў спецыяльным асяроддзі, каб інжынеры па спрыянні якасці маглі праверыць ПД на наяўнасць пахіб, недахопаў і дэфектаў, а таксама пераканацца, што яно функцыянуе належным чынам.

Этап тэставання ўключае наступныя асноўныя заходы:

- **Планаванне і праектаванне тэстаў:**
  - Распрацоўка стратэгіі тэсціравання — агульны падыход і вызначэнне мэтай
  - Стварэнне тэставых выпадкаў — падрабязныя тэставыя сцэнарыі з крокамі і праектаваннем чаканых вынікаў
  - Падрыхтоўка тэставых даных — стварэнне і кіраванне тэставымі наборамі даных

- Налада тэставага асяроддзя — канфігурацыя абсталявання, праграмнага дачынення і сетак
- **Выкананне і ацэнка тэстаў:**
  - Выкананне тэставых выпадкаў — Тэсты выконваюцца ў адпаведнасці са створанымі планамі тэставання
  - Справаздачы аб дэфектах — выяўленыя праблемы рэгіструюцца, адсочваюцца і кіруюцца
  - Аналіз вынікаў — ацэнка вынікаў у параўнанні з чаканымі вынікамі
  - Рэгрэсійнае тэставанне — праверка таго, што новыя змены не парушаюць існуючу функцыянальнасць
- **Справаздачнасць аб тэсціраванні і закрыццё:**
  - Зводныя справаздачы аб тэставанні — дакументацыя па выніках і выніках тэставання
  - Збор метрык — збор даных аб пакрыцці тэстамі, шчыльнасці дэфектаў і г.д.
  - Ацэнка крытэрыяў выхаду — ПД гатова да рэлізу

## **Тэсціраванне паказчыкаў і выміярэнняў**

**Паказчыкі тэсціравання ПД** — гэта колькасныя паказчыкі, ужывытыя для ацэнкі эфектыўнасці, прадукцыйнасці, прагрэсу і якасці працэсу тэсціравання.

Яны даюць аналітычныя даныя для прыніяцца аргументаў на рашэнняў, вызначэння абласцей для паляпшэння і ацэнкі эфектыўнасці тэсціравання ў адпаведнасці з пастаўленымі мэтамі:

- Пакрыццё тэстамі — працэнт патрабаванняў, пакрытых тэставымі выпадкамі

- Шчыльнасць дэфектаў — колькасць дэфектаў на размер або метрыку
- Эфектыўнасць тэставых выпадкаў — працэнт дэфектаў, выяўленых тэставымі выпадкамі
- Узечка дэфектаў — дэфекты, выяўленыя пасля выпуску ў параўнанні з дэфектамі падчас тэставання

## Крытэрыі ўваходу і выхаду

Крытэрыі ўваходу вызначаюць, калі можа пачацца этап тэсціравання:

- Патрабаванні стабільныя і зацверджаныя
- Распрацоўка практычна завершана
- Тэставае асяроддзе гатовае
- Тэставыя выпадкі рыхтуюцца і ўдасканальваюцца

Крытэрыі выхаду вызначаюць, калі этап тэсціравання павінен завяршыцца:

- Усе крытычныя і асноўныя дэфекты выпраўлены
- Мэты пакрыцця тэстамі дасягнуты
- Дасягнуты паказчыкі якаснасці і бяспекі
- Згода зацікаўленых бакоў атрымана

## Роля этапа тэсціравання

Тэставанне — гэта найважнейшая частка жыццёвага цыклу распрацоўкі ПД, якая можа зэканоміць шмат часу, грошай і сіл.

Каб забяспечыць якаснае праграмнае дачыненне, арганізацыя павінна сістэматычна праводзіць тэсціраванне.

## **3.5. ЭТАП РАЗГОРТВАННЯ ПД**

**Разгортванне ПД** — гэта этап, на якім правераная ПД выпускаецца і пачынае працаваць у асяроддзі, якое дазваляе спажыўцам атрымаць да яго доступ і спажывацца ім.

Этап разгортвання, які таксама называюць:

- Этап дастаўкі
- Этап прымянення
- Этап усталёўкі

— уяўляе сабой пераход ад распрацоўкі да эксплуатацыйнага ўжытку на адпаведным рынку.

### **Асноўныя віды дзеянасці**

Этап разгортвання звычайна ўключае наступныя ключавыя дзеянні:

- **Падрыхтоўка да разгортвання:**
  - Планаванне разгортвання — Стварэнне падрабязнай стратэгіі разгортвання, графіка і планаў адкату
  - Налада асяроддзя — Наладаванне сервера, базы даных, сеткі і параметраў бяспекі
  - Канчатковая праверка — выкананне дымавых тэстаў і праверак на працаздольнасць ва ўмовах, падобным да вытворчых
  - Стварэнне рэзервовай копіі — рэзервовае капіраванне існуючых сістэм і даных перад разгортваннем
  - Камунікацыя з зацікаўленымі бакамі — паведамленне спажыўцам, службам падтрымкі і зацікаўленым бакам аб будучых зменах

- **Выкананне разгортвання:**
  - Разгортванне пакетаў — Усталёўка бінарных файлаў, бібліятэк і залежнасцей
  - Міграцыя базы даных — выкананне SQL-скрыптоў, перадача даных і абнаўленне схемы
  - Канфігурацыя ПД — Усталёўка параметраў і налад, спецыфічных для асяроддзя
  - Актывацыя інтэграцыі — Падключэнне да зневніх сістэм і інтэрфейсаў прыкладных праграм, API
  - Ініцыялізацыя службы — Запуск спадарожных служб і фонавых працэсаў
- **Праверка пасля разгортвання:**
  - Праверка спраўнасці — пераканацца, што ўсе службы працуюць правільна
  - Функцыянальнае тэставанне — пацвердзіць працу крытычна важных бізнес-функцый у вытворчым рэжыме
  - Праверка якаснасці — забяспечыць адпаведнасць часу рэагавання патрабаванням уздоўнія аблугоўвання
  - Тэставанне доступу спажыўцу — праверыць механізмы аутэнтыфікацыі і аутарызацыі
  - Налада маніторынгу — наладзіць паведамленні, журналы і маніторынг працы

## Стратэгіі разгортвання

Стратэгіі разгортвання ўключаюць:

- **Традыцыйныя падыходы:**

- Разгортванне Big-Bang — поўная замена сістэмы адразу
  - Паралельнае разгортванне — старыя і новыя сістэмы працуюць адначасова
  - Паступовае разгортванне — разгортванне па модулях, рэгіёнах або групах спажыўцуў
- **Сучасныя падыходы — бесперапыннае разгортванне:**
    - Сіне-зялёнае разгортванне — два аднолькавыя асяроддзі для пераключэння трафіку паміж імі
    - Рэліз канарэйкі — паступовае адкрыццё новай версіі для невялікай колькасці спажыўцуў
    - Сцяжкі функцый — разгортванне кода з выключанымі функцыямі, каб паступова ўключачыць іх
    - Паступовае разгортванне — паступовае абнаўленне экзэмпляраў пры захаванні службы

## Крытэрый ўваходу і выхаду

Перадумовы для пачатку разгортвання наступныя:

- Усе этапы тэсціравання паспяхова завершаны
- Атрымана адабрэнне зацікаўленых бізнес-бакоў
- Падрыхтавана і праверана працоўнае асяроддзе
- План адкату задокументаваны і пратэставаны
- Документацыя і навучанне спажыўцуў завершаны

Патрабаванні да завяршэння разгортвання наступныя:

- Дацыненне паспяхова запушчана ў працоўным рэжыме
- Усе кропкі інтэграцыі працуюць правільна
- Выкананы контрольныя паказчыкі якаснасці
- Маніторынг і паведамленні ў аператыўным рэжыме
- Каманды падтрымкі падрыхтаваныя і готовыя
- Праведзены агляд пасля разгортвання

## **3.6. ЭТАП АБСЛУГОЎВАННЯ ПД**

**Абслугоўванне ПД** — гэта этап карэктіровак, паправак і ўдасканаленя, прызначаных для яго абнаўлення, падтрымання працаздольнасці і якаснасці пасля першапачатковага разгортвання.

### **Асноўныя мэты**

Асноўная мэта этапу абслугоўвання — рэгулярнае абнаўленне і мадэрнізацыя ПД для яго адаптацыі да будучых праблем.

У цэлым, абслугоўванне накіравана на наступныя мэты:

- Выпраўленне праблем — выяўленне і выпраўленне памылак, выяўленых у працоўнай версіі
- Адаптацыя да змен — мадыфікацыя праграмнага дачынення для працы ў зменлівых умовах
- Паширэнне функцыянальнасці — даданне новых функцый і паляпшэнне існуючых магчымасцей
- Прадухіленне пагаршэння — аптымізацыя якаснасці і вырашэнне праблемы тэхнічнага доўгу
- Забеспячэнне бесперапыннасці — падтрымка спалучэння з платформамі і стандартамі, якія пастаянна развіваюцца

### **Асноўныя віды дзейнасці**

Дзейнасць на этапе абслугоўвання можна падзяліць на наступныя катэгорыі:

- **Карэктіручае абслугоўванне:**
  - Вырашэнне праблем — выпраўленне памылак і памылак, пра якія паведамляюць спажыўцы
  - Экстраныя выпраўленні — вырашэнне крытычных праблем, якія ўплываюць на даступнасць сістэмы

- Кіраванне патчамі — разгортванне невялікіх, мэтанакіраваных абнаўленняў

- **Адаптыўнае аслугоўванне:**

- Абнаўленні платформы — адаптацыя да аперацыйных сістэм, аbstалявання або воблачных платформ
- Інтэграцыя са староннімі сістэмамі — падтрыманне спалучэння са зневшнімі сістэмамі
- Адпаведнасць рэгулятарным патрабаванням — выкананне заканадаўчых і нарматыўных правіл

- **Прагрэсіўнае аслугоўванне:**

- Аптымізацыя якаснасці — паляпшэнне хуткасці, эфектыўнасці і спажывання рэурсаў
- Паляпшэнні зручнасці ўжытку — паляпшэнне спажывецкага інтэрфейсу і досведу
- Дапаўненні функцый — рэалізацыя новых магчымасцей на аснове водгукаў спажыўцу

- **Прафілактычнае аслугоўванне:**

- Рэфактарынг кода — змена кода без змен функцыянала
- Скарачэнне тэхнічнай запазычанаасці — ліквідацыя недахопаў, дапушчаных падчас распрацоўкі
- Абнаўленні дакументацыі — падтрыманне дакументацыі ў актуальным стане з улікам змен у сістэме

## **Мадэлі аслугоўвання**

Мадэлі аслугоўвання ПД можна падзяліць на наступныя групы:

- **Традыцыйныя мадэлі:**

- Мадэль хуткага выпраўлення — імгненнае выпраўленне без падрабязнага аналізу
  - Ітэратыўнае паляпшэнне — цыклічнае паляпшэнне праз аналіз і перапраектаванне
  - Арыентацыя на паўторнае ўжыццё — спажыванне наісных кампанентаў для абслугоўвання
- Сучасныя падыходы:
    - Agile-абслугоўванне — рэгулярныя спрынты па аблугоўванні ПД са зваротнай сувяззю спажыўцу
    - Аблугоўванне DevOps — падтрыманне выконваюць інтэграваныя каманды распрацоўкі і аперацый
    - Аблугоўванне на аснове ШІ — прагнастычная аналітыка для прадухілення проблем

## Роля этапу аблугоўвання

Аблугоўванне ПД — гэта не проста выпраўленне памылак, але і стратэгічны, пастаянны працэс падтрымання каштоўнасці, бяспекі і адпаведнасці ПД патрэбам бізнесу.

Найбольш паспяховыя кампаніі разглядаюць аблугоўванне як стратэгічную інвестыцыю, а не як неабходныя выдаткі.

Эфектыўнае аблугоўванне патрабуе балансу рэактыўнага падтрымання з праактыўным удасканаленнем, гарантуючы, што ПД працягвае прыносіць каштоўнасць на працягу ўсяго свайго жыццёвага цыклу.

Тэхнічнае аблугоўванне — гэта не заключны этап, а мяккае вяртанне да этапу планавання з мэтай далейшых паляпшэнняў на наступным узроўні ЖЦР ПД.

## **3.7. Мадэлі ЖЦР ПД**

Мадэлі жыццёвага цыклу распрацоўкі ПД коратка называюць Мадэлямі працэсаў распрацоўкі ПД.

**Мадэль працэсу распрацоўкі ПД** — МППД — гэта структура, якая вызначае паслядоўнасць дзеянь, задач і працоўных працэсаў для распрацоўкі праграмных прадуктаў.

МППД забяспечвае сістэматычны падыход да ўвасаблення спажывецкіх патрабаванняў у функцыянальнае ПД, адначасова кіруючы такім абмежаваннямі, як час, кошт і якасць.

Кожны МППД выконвае шэраг кроکаў, унікальных для свайго тыпу, каб забяспечыць поспех працэсу распрацоўкі ПД.

### **Падыходы МППД**

Усе МППД можна падзяліць на дзве вялікія катэгорыі ў залежнасці ад падыходаў, ужытых да распрацоўкі:

- Прагнозныя
- Адаптыўныя

### **Прагнозныя МППД**

**Прагнозныя МППД** — або **Планавыя** — гэта мадэлі, якія мяркуюць, што патрабаванні могуць быць цалкам вызначаны ў пачатку праекта і ахінуцца адносна стабільнымі падчас распрацоўкі.

Яны засяроджваюцца на падрабязным аналізе і планаванні будучыні і ўліку вядомых рызык. У крайнім выпадку, прагназісты могуць дакладна паведаміць, якія функцыі і задачы запланаваны на працягу ўсяго працэсу распрацоўкі.

Прагнозныя мадэлі абапіраюцца на эфектыўнасць аналізу на ранній стадыі, і калі яна нізкая, праект можа напаткаць цяжкасці падчас змены свайго кірунку.

## **Адаптыўныя МППД**

**Адаптыўныя МППД** — або **Каштоўнасныя** — гэта мадэлі, якія ўлічваюць, што патрабаванні змяняюцца падчас распрацоўкі, і падкрэсліваюць патрэбу ў гнуткасці, супрацоўніцтве і несупынным ўдасканаленні.

Яны сканцэнтраваны на хуткай адаптацыі да зменлівых рэалій — калі патрэбы праекта змяняюцца, змяняеца і гнуткая каманда.

Адаптыўнай камандзе цяжка дакладна апісаць, што адбудзецца ў будучыні — чым далей дата, tym больш расплывістай паказвая сябе адаптыўная мадэль адносна таго, што адбудзецца ў гэтую дату.

---

## **IV. Прагнозные МПД**

## 4. Прагнозныя МППД

Як вынікае з назвы, прагнозная МППД мяркуе, што можна прадказаць увесь працоўны працэс.

Гэта прадугледжвае поўнае разуменне канчатковага прадукту і вызначэнне працэсу яго дастаўкі.

У гэтай форме жыццёвага цыклу праекта кошт, аб'ём і тэрміны выканання вызначаюцца на ранніх этапах праекта.

### Перавагі

Асноўныя перавагі прагнозных МППД наступныя:

- Іх лёгка зразумець і прасачыць, бо кожны этап пачынаецца пасля завяршэння папярэдняга
- Зразумелыя інструкцыі і лаканічны працоўны працэс дазваляюць распрацоўшчыкам лягчэй працеваць у межах пэўнага бюджету і тэрмінаў
- Калі ўсё ідзе паводле плану, гэта дазваляе арганізацыям прадказаць чаканы бюджету праекта і яго тэрміны
- Кожны этап мае пэўныя тэрміны і вынікі, што спрашчае камандам працу і контроль за ўсім праектам
- Галоўны клопат прагнастычнага падыходу — распрацоўка і падтрыманне спецыфікацый канчатковага прадукту

Гэта робіць яго ідэальным для праектаў, дзе ўсе патрабаванні вызначаны і добра зразумелыя, а таксама ёсць выразнае бачанне канчатковага прадукту.

У рамках прагнастычнага падыходу чакаюцца мінімальныя змены, бо праца ўжо прадказаная і добра вядомая.

Каманда мае выразнае ўяўленне аб тым, куды рухаецца праект і як прытрымлівацца паслядоўнасці.

## Недахопы

Асноўныя недахопы прагнастычнага падыходу наступныя:

- Працоўнае праграмнае дачыненне ствараецца на познім этапе, што прыводзіць да затрымкі выяўлення памылак і ўразлівасця ў дадатку
- Арганізацыям часта даводзіцца несці дадатковыя выдаткі на затрымку заявак, калі пахібы выяўляюцца на этапе тэставання
- Складаныя праекты дрэнна кіруюцца: яны не падыходзяць для дынамічных праектаў, якія прадугледжваюць гнуткія патрабаванні або навызначанасць у канчатковым прадукце

Карацей кажучы, прагнозны падыход можа быць надзвычай жорсткім, патрабуючы ад распрацоўшчыкаў падтрымання строгіх і нязменных стандартоў на працягу ўсяго жыццёвага цыклу.

Паколькі паслядоўнасць работ ужо вызначана загадзя, любыя наступныя змены могуць быць вельмі дарагімі і працаёмкімі.

## Прагнозныя мадэлі

Ніжэй прыведзены найбольш важныя прагнозныя мадэлі працэсу распрацоўкі ПД:

- Мадэль Вялікага выбуху
- Мадэль Вадаспаду
- Інкрементальная мадэль
- Ітэрацыйная мадэль
- Спіральная мадэль
- V-Мадэль

## **4.1. Мадэль Вялікага выбуху**

**Мадэль Вялікага выбуху** — гэта працэс распрацоўкі ПД з увагай на ўсіх тыпах рэурсаў у распрацоўцы і кадаванні без або з нязначным іх планаваннем.

Патрабаванні разглядаюцца і ўвасабляюцца падчас іх паступлення.

### **Ключавыя характеристыкі**

Мадэль Вялікага выбуху — гэта МППД, якая не прытрымліваецца ніякага канкрэтнага працэсу або працэдуры і патрабуе вельмі мала планавання.

Распрацоўка пачынаецца са сродкаў і высілкаў, неабходных у якасці ўваходных рэурсаў, а распрацаванае ПД служыць вынікам.

Нават кліент можа быць не ўпэўнены ў сваіх патрабаваннях, і яны рэалізуюцца на хаду без асаблівага аналізу.

Звычайна мадэль Вялікага выбуху ўжываецца для невялікіх праектаў, дзе каманды распрацоўшчыкаў вельмі малыя.

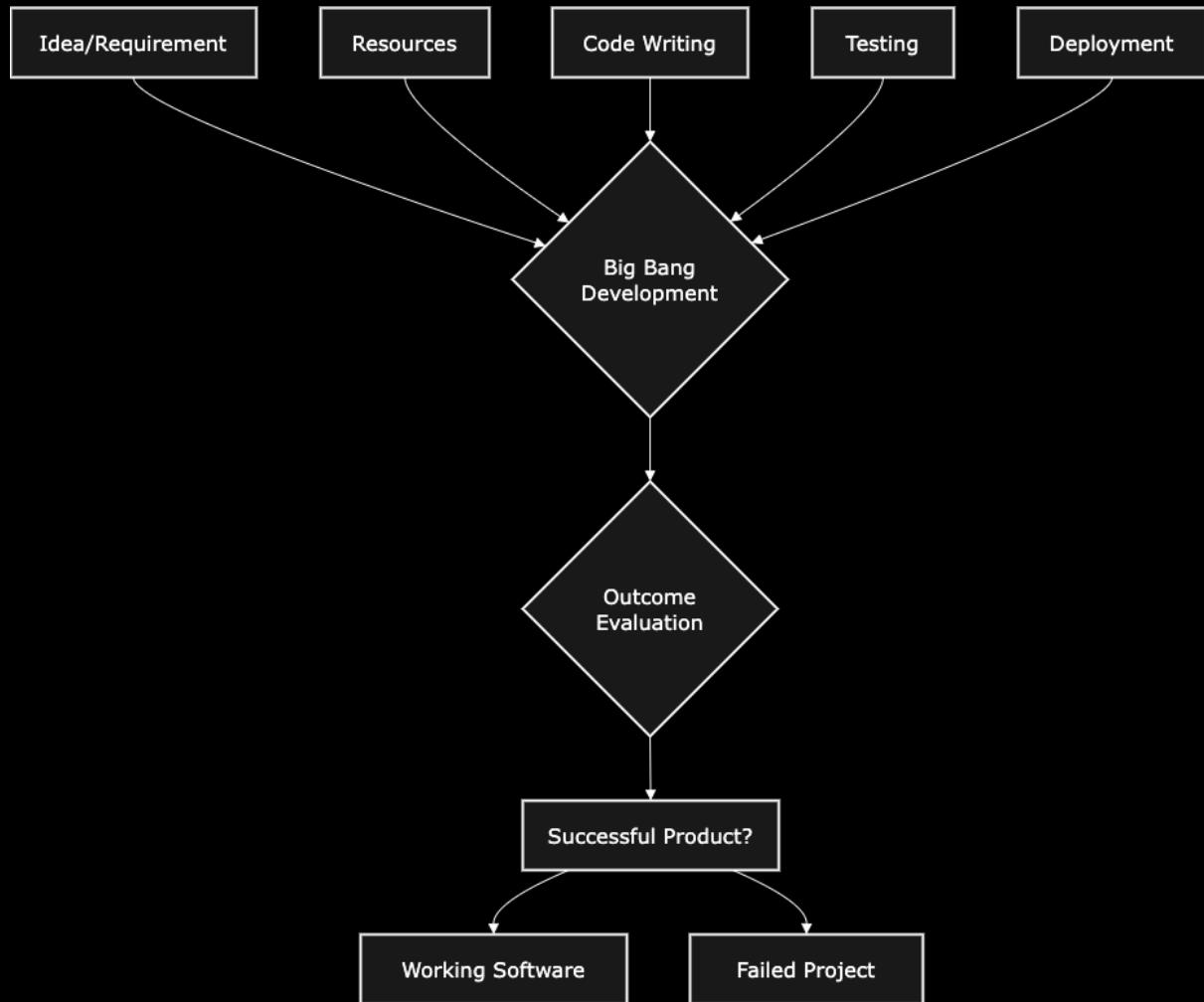
### **Асноўныя віды дзейнасці**

Мадэль Вялікага выбуху факусуе ўсе магчымыя рэсурсы на распрацоўцы праграмнага дачынення, практычна без планавання.

Патрабаванні разумеюцца і рэалізуюцца па меры іх з'яўлення; любыя неабходныя змены могуць запатрабаваць — або не — поўнага перагляду праграмнага дачынення.

Гэта мадэль ідэальна падыходзіць для невялікіх праектаў з адным або двумя распрацоўшчыкамі, якія працуюць супольна, а таксама плённая для акадэмічных праектаў і практикі.

Гэта таксама ідэальная мадэль для прадукту, патрабаванні да якога не вельмі добра зразумелыя, а канчатковая дата выпуску не пазначана.



*Bing Bang Model*

## Перавагі

Мадэль Вялікага выбуху — гэта метад распрацоўкі:

- Просты — патрабуе вельмі мала або зусім не патрабуе планавання
- Кіруемы — не патрабуе фармальных працэдур
- Даступны — патрабуе вельмі мала рэурсаў
- Гнуткі — патрабуе меншай кваліфікацыі

У заключэнні, мадэль Вялікага выбуху ідэальна падыходзіць для паўтаральных або невялікіх праектаў з мінімальнымі ризыкамі.

## Недахопы

Асноўныя недахопы мадэлі Вялікага выбуху наступныя:

- Вельмі высокая ризыка і навызначанасць
- Выключает складаныя і аб'ектна-арыентаваныя праекты
- Адхінае працяглыя і бягучыя праекты
- Можа абысціся вельмі дорага, калі патрабаванні няправільна зразумець

Такім чынам, мадэль Вялікага выбуху мае вельмі высокую ризыку, бо няправільна зразумелыя або змененныя патрабаванні могуць прывесці да поўнай адмены або зрыву праекта.

---

## 4.2. Мадэль Вадаспаду

**Мадэль Вадаспаду** — гэта працэс распрацоўкі ПД, якая падзяляе ўесь жыццёвы цыкл распрацоўкі ПД на розныя этапы.

Вадаспадная МППД была першай мадэллю, прадстаўленай у кампаніяй Winston Royce 1970 годзе, і мае таксама іншую назву — Лінейна-паслядоўнай МППД.

### Асноўныя віды дзейнасці

Вадаспадная мадэль — гэта паслядоўная МППД, якая падзяляе распрацоўку ПД на загадзя вызначаныя этапы.

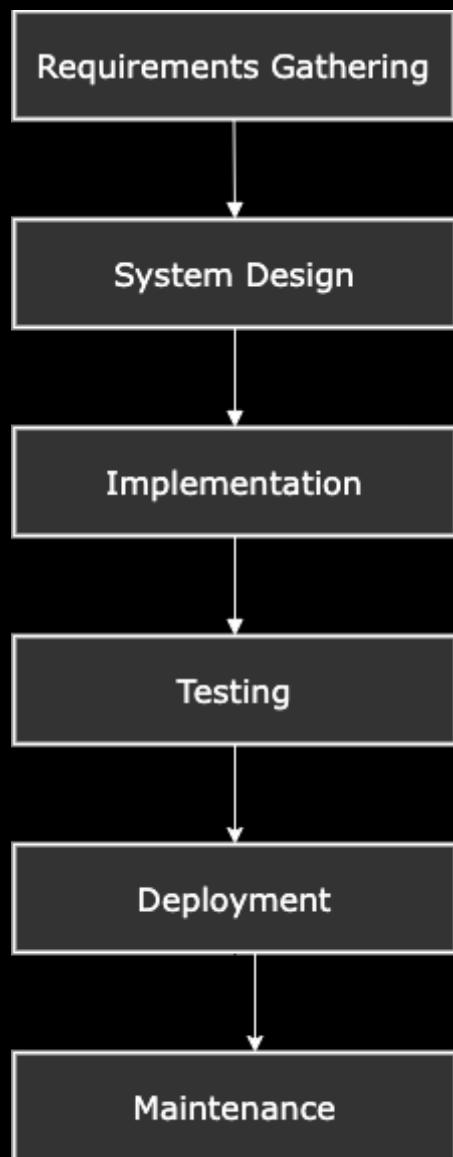
Кожны этап прызначаны для пэўнай дзейнасці і павінен быць завершаны да пачатку наступнага этапу без іх нахлёсту.

### Прымяненне

Кожнае праграмнае дачыненне адрозніваецца і патрабуе адпаведнага МППД, якога трэба прытрымлівацца ў залежнасці ад унутраных і знежніх фактараў.

Некаторыя сітуацыі, калі ўжыванне мадэлі Вадаспаду найбольш мэтазгодна:

- Патрабаванні рупліва задокументаваныя, ясныя і фіксаваныя
- Тэхналогіі і інструменты, якія ўжываюцца, знаёмыя і паслядоўныя
- Даступны дасведчаныя рэсурсы, і іх дастаткова
- Дачыненне простае і кампактнае
- Праграмнае асяроддзе стабільнае
- Тэрмін праекта кароткі



*Waterfall Model*

## Перавагі

Мадэль Вадаспаду — гэта самая ранняя МППД, якая ўжывалася для распрацоўкі ПД; яе вельмі проста зразумець і прымяніць.

Некаторыя з асноўных пераваг мадэлі Вадаспаду наступныя:

- Выразна акрэсленыя этапы
- Добра зразумелыя вехі

- Лёгка арганізаваныя задачы
- Добра дакументаваныя працэсы
- Гатовыя для кіравання рэсурсы
- Вынікі на кожным этапе
- Этапы выконваюцца адзін за адным
- Зручныя для агляду вынікі

## Недахопы

Асноўныя недахопы мадэлі Вадаспаду наступныя:

- Разгалінаваная нявывераная дакументацыя
  - Працоўнае ПД з'яўляецца позна ў ЖЦР ПД, таму этап тэставання пачынаецца занадта позна
  - Змены ў праекце непрадказальныя
  - Рызыкі і нявызначанасць высокія
  - Складаныя або аб'ектна-арыентаваныя праекты выключаюцца
  - Доўгія і працяглыя праекты немагчымыя
  - Прэкты са зменай патрабаванняў дрэнна кіруюцца
  - Для выканання праекта патрабуеца падтрымка стабільнасці асяроддзя
  - Інтэграцыя складаная і клопатная
  - Прагрэс на стадыях распрацоўкі цяжка вымераць
  - Тэхналагічныя вузкія месцы дрэнна выяўляюцца
  - Бізнес-проблемы цяжка пераадолець
-

## 4.3. Інкрементальная мадэль

**Інкрементальная мадэль** — гэта працэс распрацоўкі праграмнага дачынення, у якім патрабаванні разбіваюцца на некалькі асобных модуляў жыццёвага цыклу распрацоўкі праграмнага дачынення.

### Асноўныя віды дзейнасці

Інкрементальную МППД можна разглядаць як серию вадаспадных цыклаў.

Патрабаванні падзяляюцца на групы на початку праекта; для кожнай групы распрацоўваецца ПД па вадаспаднай мадэлі.

#### Працэс

Кожны рэліз дадае больш функцыянальнасці, пакуль не будуць выкананы ўсе патрабаванні; кожны цыкл служыць этапам абслугоўвання папярэдняга рэлізу.

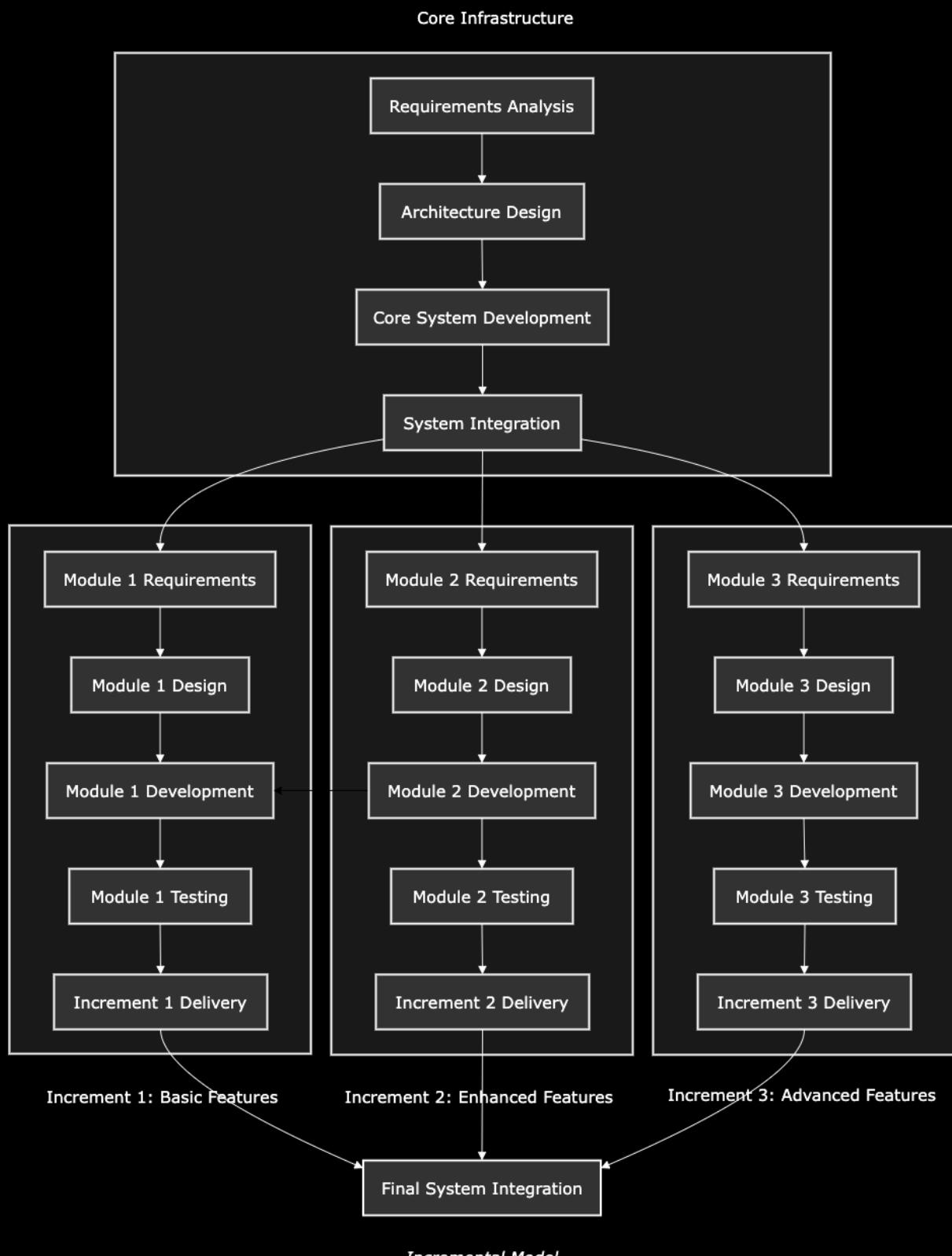
Паступовыя мадыфікацыі мадэлі дазваляюць цыклам распрацоўкі перакрывацца, так што наступны цыкл можа пачацца да завяршэння папярэдняга.

#### Ітэрацыі

Кожная ітэрацыя праходзіць праз наступныя чатыры этапы:

- Аналіз патрабаванняў
- Праектаванне
- Кадаванне
- Тэсціраванне

і кожны наступны выпуск сістэмы дадае функцыі да папярэдняга выпуску, пакуль не будуць рэалізаваны ўсе функцыі паводле намераў.



## **Працэдура**

ПД запускаецца ў вытворчасць, калі будзе выпушчаны першы інкрэмэнт.

Першы інкрэмэнт часта з'яўляеца стартавым прадуктам, у якім выконваюцца асноўныя патрабаванні, а дадатковыя функцыі дадаюцца ў наступных прыростах.

Пасля таго, як кліент прааналізуе стартавы прадукт, адбываеца распрацоўка наступнага інкрэмэнту.

## **Характарыстыкі мадэлі**

Ключавыя характарыстыкі Інкрэментальнай мадэлі ўключаюць:

- Распрацоўка ПД падзяляеца на мноства міні-праектаў распрацоўкі
- Частковыя сістэмы паслядоўна будуюцца для стварэння канчатковай поўнай сістэмы
- Патрабаванне з найвышэйшым прыярытэтам вырашаеца ў першую чаргу
- Пасля распрацоўкі патрабаванняў, усе патрабаванні для гэтага інкрэмэнту замарожваюцца

## **Прымяненне**

Інкрэментальная мадэль можа ўжывацца, калі:

- Патрабаванні да ПД выразна зразумелыя
- Ранні выпуск прадукта мае вялікае значэнне
- Існуюць асаблівасці і мэты, якія маюць значную рызыку
- Распрацоўшчыкі не маюць высокай кваліфікацыі або досведу
- ПД распрацоўваецца прадуктовай кампаніяй
- Дацыненне працуе ў вэб-рэжыме і можа лёгка абнаўляцца

## **Перавагі**

Асноўныя перавагі Інкрементальнай мадэлі наступныя:

- Высокая хуткасць распрацоўкі
- Высокая гнуткасць
- Меншыя выдаткі
- Магчымасць змен
- Кліент здолен ацаніць кожную зборку
- Памылкі лёгка выявіць

## **Недахопы**

Ніжэй прыведзены асноўныя недахопы Інкрементальнай мадэлі:

- патрабуеца дальнабачнае планаванне
  - кожны ітэрацыйны цыкл дастаткова жорсткі
  - выпраўленне памылкі ў любой з адзінак упłyвае на ўсё ПД
  - выпраўлення ў шмат, і яны займаюць шмат часу
-

## 4.4. Ітэрацыйная мадэль

**Ітэрацыйная мадэль** — або **Мадэль эвалюцыйных змен** — гэта працэс распрацоўкі ПД, які пачынаецца з рэалізацыі невялікага простага падмнства патрабаванняў і ітэратыўна ўдасканальвае версіі, якія развіваюцца, паکуль поўнае дачыненне не будзе рэалізавана і гатова да разгортвання.

### Асноўныя віды дзейнасці

Ітэрацыйная МППД не спрабуе пачаць з поўнай спецыфікацыі патрабаванняў.

Замест гэтага распрацоўка пачынаецца з вызначэння і рэалізацыі толькі часткі праграмнага дачынення, якая затым правяраецца для вызначэння далейшых патрабаванняў.

Затым гэты працэс паўтараецца, ствараючы новую версію праграмнага дачынення ў канцы кожнай ітэрацыі мадэлі.

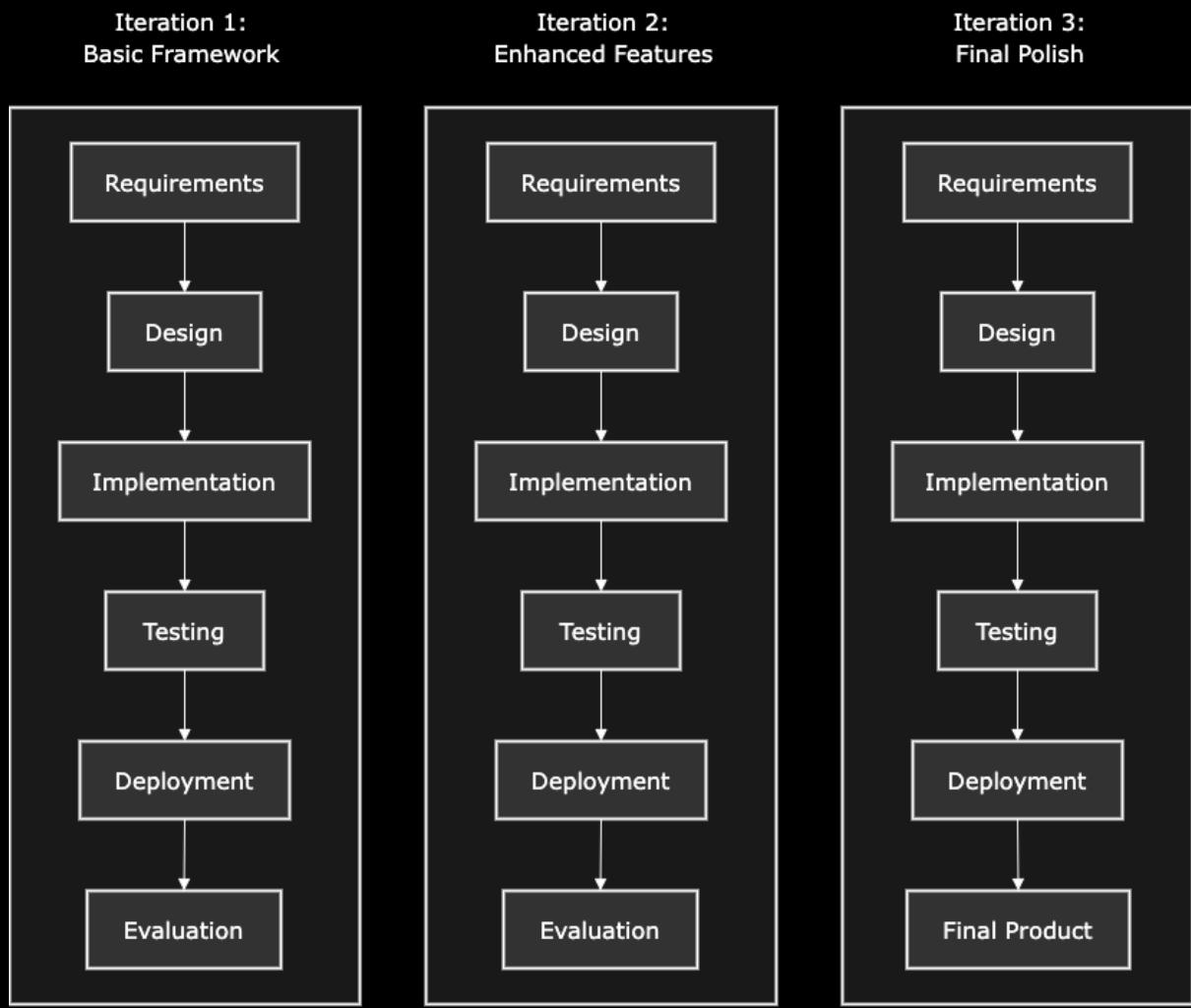
На кожнай ітэрацыі ўносяцца змены ў дызайн і дадаюцца новыя функцыянальныя магчымасці.

Асноўная ідэя гэтага метаду заключаецца ў распрацоўцы сістэмы праз паўтаральныя, ітэрацыйныя цыклы і меншымі, паступовымя часткамі за раз.

### Працоўны працэс

Ітэрацыйная мадэль — гэта спалучэнне ітэрацыйнага праектавання і інкрементальнай распрацоўкі.

Падчас распрацоўкі адначасова можа выконвацца і больш за адну ітэрацыю цыклу распрацоўкі праграмнага дачынення.



*Iterative Model*

## Зборкі

Ітэрацыйная мадэль мае на ўвазе, што ўсе патрабаванні падзелены на розныя зборкі.

Падчас кожнай ітэрацыі модуль распрацоўкі праходзіць этапы аналіза патрабаванняў, практавання, реалізацыі і тэставання.

Кожны наступны рэліз модуля дадае новую функцыянальнасць да папярэдняй.

Працэс працягваецца да таго часу, пакуль сістэма не будзе гатовая да поўнага выканання патрабаванняў.

## Роля ТЭСЦІРОЎШЧЫКА

Ключавымі асаблівасцямі Ітэрацыйнай мадэлі з'яўляюцца строгая праверка патрабаванняў, а таксама праверка кожнай версii ПД на адпаведнасць патрабаванням у рамках кожнай ітэрацыі.

Па меры развіцця ПД на працягу паслядоўных цыклаў тэсты паўтараюцца і пашыраюцца для праверкі кожнай версii ПД.

## Прымяненне

Як і іншыя МППД, ітэрацыйная распрацоўка мае некаторыя спецыфічныя прымяненні ў індустрыйнай праграмнага дачынення.

Мадэль найчасцей ужываецца ў наступных сітуацыях:

- Патрабаванні да прадукту выразна вызначаны і зразумелыя
- Каманда распрацоўшчыкаў павінна ўжываць новыя тэхналогіі
- Пэўныя функцыі ці патрабаванні могуць мяніцца ў час працы
- У наступным могуць з'явіцца новыя функцыі і мажлівасці ПД, якія нясуць высокі уровень рызыкі
- Мэты пэўных ітэрацый могуць запатрабаваць аутсорсінг

## Перавагі

Асноўная перавага Ітэрацыйнай мадэлі — гэта ПД, якое распрацоўваецца на самых ранніх этапах распрацоўкі.

У прыватнасці, працоўнае праграмнае дачыненне дазваляе лягчэй знайсці функцыянальныя або канструктыўныя недахопы.

Недахопы, выяўленыя на ранніх этапах жыццёвага цыклу, дазваляюць распрацоўшчыкам эканоміць бюджет падчас уніяснення карэктніровак у праграмнае дачыненне.

Іншыя перавагі ітэрацыйнай мадэлі наступныя:

- Вялікія і крытычна важныя праекты выканальныя
- Працоўнае ПД з'яўляецца на ранніх этапах жыццёвага цыклу
- Кліент хутка атрымлівае ПД і ацэньвае яго ў зваротнай сувязі
- Кожны інкремент забяспечвае новую функцыянальнасць ПД
- Атрыманыя вынікі з'яўляюцца неадкладнымі і перыядычнымі
- Кожная ітэрацыя служыць кіраванай вехай
- Прагрэс у распрацоўцы праграмнага дачынення вымяральны
- Паралельная распрацоўка мажліва
- Меншыя ітэрацыі спрашчаюць тэсціраванне і адладку
- Змены патрабавань лёгка рэгулююцца з меншымі выдаткамі
- Праблемы, цяжкасці і рызыкі могуць разглядацца паступова
- Паўторны пошук, аналіз і змякчэнне рызык больш стабільныя
- Высокія рызыкі можна змякчыць адразу

## Недахопы

Асноўны недахоп ітэрацыйнай мадэлі заключаецца ў tym, што яна падыходзіць толькі для значных праектаў распрацоўкі ПД: разбіць невялікую сістэму на далейшыя невялікія працаздольныя інкременты або модулі складана.

Іншыя недахопы ітэрацыйнай мадэлі наступныя:

- Для прымянення мадэлі можа спатрэбіцца больш рэурсаў
- Частыя змены патрабаванняў трываюць кошт ПД высокім
- Мажлівы значныя праблемы на узроўні архітэктуры
- Увага кірауніцтва мае большую патрэбу і складанасць
- Аналіз рызык патрабуе высокакваліфікованых рэурсаў
- Прагрэс моцна залежыць ад этапу аналіза рызык
- Перавызначэнні інкрементаў перавызначаюць усю сістэму
- Рызыкоўны ўзровень навызначанасці напрыканцы праекта
- Мадэль дрэнна падыходзіць для невялікіх праектаў

## 4.5. Спіральная мадэль

**Спіральная мадэль** — гэта спалучэнне ітэрацыйнай мадэлі з сістэматычнымі, кантраляванымі аспектамі паслядоўнага лінейнага развіцця, уласцівых мадэлі Вадаспаду.

Іншымі словамі, мадэль уяўляе сабой сувязь ітэрацыйных і каскадных мадэляў з выразным акцэнтам на аналізе рызык.

Гэта дазваляе выпускаць рэлізы інкрементальна — або весці **Паступовае ўдасканаленне** — праз кожную ітэрацыю па спіралі.

### Спіралі

Спіральная МППД уключае чатыры этапы, апісаныя ніжэй.

Праграмны праект неаднаразова праходзіць гэтыя этапы ў ітэрацыях, якія называюцца Спіралямі.

На падставе ацэнкі кліента працэс распрацоўкі ПД уваходзіць у наступную ітэрацыю і пасля гэтага вядзецца паводле лінейнага падыходу для апрацоўкі водгукаваў, прапанаваных кліентамі.

Працэс ітэрацыі па спіралі працягваецца на працягу ўсяго тэрміну службы праграмнага дачынення.

### 1. Ідэнтыфікацыя

Гэты этап пачынаецца са збору бізнес-патрабаванняў у базавую спіраль.

На наступных спіралах, па меры разбудовы ПД, на гэтым этапе вызначаюцца патрабаванні да сістэмы, падсістэмы і модуляў.

Гэты этап таксама ўключае разуменне сістэмных патрабаванняў шляхам пастаяннага меркавання паміж заказчыкам і сістэмным аналітыкам.

У канцы спіралі прадукт разгортваецца на вызначаным рынку.

## **2. Праектаванне**

Этап праектавання пачынаецца з праектавання кацэпцыі базавай спіралі і ўлучае архітэктурнае праектаванне, праектаванне лагічных модуляў, праектаванне фізічнага прадукту і канчатковае праектаванне ў наступных спіралях.

## **3. Зборка альбо мантаж**

Этап зборкі адносіцца да вытворчасці рэальнага праграмнага прадукту на кожнай спіралі.

На базавай спіралі, калі прадукт толькі задумваецца і распрацоўваецца дызайн, на гэтым этапе распрацоўваецца Доказ Канцэпцыі, каб атрымаць водгукі кліентаў.

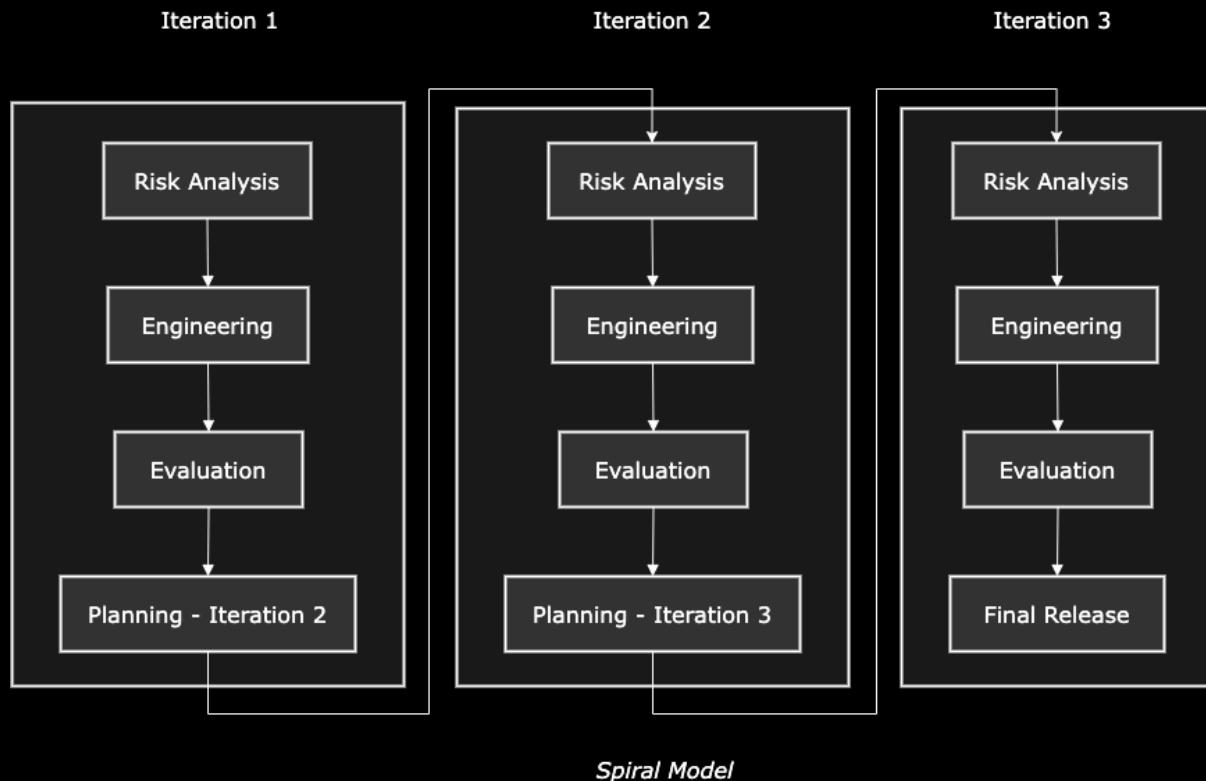
У наступных спіралях, з большай яснасцю патрабаванняў і дэталяў праектавання, ствараецца рабочая мадэль праграмнага дачынення, якая называецца зборкай, з нумарам версіі.

Гэтыя зборкі адпраўляюцца кліенту для адпаведнай сувязі.

## **4. Ацэнка і аналіз рызык**

Аналіз рызык уключае вызначэнне, ацэнку і маніторынг тэхнічных магчымасцей і рызыкі кіравання — такіх як парушэнне графіка і перавышэнне выдаткаў.

Пасля тэставання зборкі, у канцы першай ітэрацыі, кліент ацэньвае праграмнае дачыненне і дае водгук.



*Spiral Model*

## Асноўныя віды дзейнасці

Кожны этап спіральнай мадэлі ў праграмнай інжынерыі пачынаецца з мэты праектавання і спыняеца тым, што кліент аглядае прагрэс.

Працэс распрацоўкі пачынаецца з невялікага набору патрабаванняў і праходзіць праз кожны этап мадэлі Вадаспаду.

Каманда распрацоўшчыкаў дадае функцыянал дадатковых патрабаванняў ўсё больш высокіх віткоў спіралі, пакуль праграмнае дачыненне не будзе гатова да этапу вытворчасці.

## Перавагі

Перавага Спіральнай мадэлі заключаецца ў тым, што яна дазваляе дадаваць элементы прадукту, калі яны становяцца даступнымі або вядомымі.

Гэта гарантую адсутнасць супярэчнасцей з папярэднімі патрабаваннямі і дызайном.

Гэты метад адпавядзе падыходам, якія маюць некалькі зборак і рэлізаў ПД, што дазваляе ажыццяўляць упарадкаваны пераход да дзейнасці па тэхнічным абслугоўванні.

Яшчэ адзін станоўчы аспект гэтага метаду заключаецца ў тым, што спіральная мадэль прымушае спажыўцу да ўдзелу ў распрацоўцы сістэмы на ранніх этапах.

Іншыя перавагі спіральнай мадэлі наступныя:

- Мадэль дазваляе шырока спажываць прататыпы
- Стварэнне прататыпа спрашчае ацэнку выдаткаў
- Ітаратыўнае развіццё спрашчае кіраванне рызыкамі
- Больш хуткая распрацоўка і больш рэгулярныя паляпшэнні
- Кліенты хутка атрымліваюць ПД і даюць адпаведную сувязь
- Змены патрабаванняў лёгка ўлічваюцца
- Патрабаванні сабраны больш дакладна
- Дадатковыя функцыі могуць быць перанесены на больш позні этап
- Распрацоўку можна падзяліць на меншыя часткі
- Рызыкоўныя часткі ПД могуць быць распрацаваны раней

## Недахопы

Асноўным недахопам Спіральнай мадэлі з'яўляецца тое, што для выканання такіх праектаў патрабуеца вельмі строгае кіраванне, і існуе рызыка запуску спіралі ў бясконцым цыклі.

Дысцыпліна і аб'ём запытаў на змены вельмі важныя для паспяховай распрацоўкі і разгортвання прадукту.

Іншыя недахопы Спіральнай мадэлі наступныя:

- Вельмі імаверны разыходжанні ў графіку або бюджетэце
- Шмат памежных этапаў патрабуюць шмат дакументацыі

- Для беспербойнай працы неабходна строга выконваць мадэльны пратакол
- Працэс і яго кіраванне больш складаныя
- Досвед ацэнкі рызык абавязковы для Спіральнай мадэлі
- Спіральная распрацоўка падыходзіць адно буйным праектам
- Непрыстасаванасць для малых праектаў з-за вялікіх выдаткаў
- Для маларызыкоўных праектаў мадэль празмерна складаная
- Спіраль можа працягвацца бясконца

## Прымяненне

Спіральная мадэль шырока ўжываецца ў IT-індустрый, бо яна сінхранізавана з натуральным працэсам распрацоўкі любога прадукту — навучанне адбываецца падчас сталення, што нясе мінімум рызыкі як для кліента, так і для IT-кампаніі.

Наступныя асаблівасці служаць тыповымі выпадкамі ўжывання Спіральнай мадэлі:

- Праект вялікі
- Рэлізы павінны быць частымі
- Водгукі кліентаў неабходныя
- Стварэнне прататыпа магчыма
- Ацэнка рызыкі важная
- Бюджэтныя абмежаванні сур'ёзныя
- Змены бізнес-прыярыйтэтаў адбываюцца часта
- Праект мае сярэднюю або высокую рызыку
- Патрабаванні невыразныя, зменлівыя або складаныя
- Перагляд праекта можа адбыцца ў любы час
- Чакаюцца істотныя змены

## 4.6. V-Model

**V-мадэль** з'яўляеца пашырэннем Вадаспаднай МППД і заснавана на асацыяцыі этапу тэставання з кожным адпаведным этапам распрацоўкі.

Мадэль складаецца з этапаў верыфікацыі і валідацыі і таксама вядомая як **Мадэль верыфікацыі і валідацыі**.

### Асноўныя віды дзейнасці

V-мадэль — гэта МППД, дзе выкананне працэсу адбываецца ў форме літары V.

Гэта азначае, што з кожным крокам распрацоўкі непасрэдна звязаны этап тэставання, які выконваецца паралельна-паслядоўна.

Гэта мадэль высокай дысцыпліны, дзе наступны этап пачынаеца толькі пасля завяршэння папярэдняга.

### Этапы верыфікацыі

У V-мадэлі ёсьць некалькі этапаў праверкі, кожны з якіх падрабязна тлумачыцца ніжэй.

#### 1. Аналіз бізнес-патрабавань

Гэта першы этап цыклу распрацоўкі, на якім патрабаванні да прадукту разумеюцца з пункту гледжання кліента.

Гэты этап прадугледжвае дэталёвую камунікацыю з кліентам, каб зразумець яго чаканні і дакладныя патрабаванні.

Гэта вельмі важная дзейнасць, і ёй трэба добра кіраваць, бо большасць кліентаў не ўпэўненыя, што менавіта ім трэба.

Планаванне праекта прыёмачных выпрабаванняў ажыццяўляеца на гэтым этапе, паколькі бізнес-патрабаванні могуць быць ужыты ў якасці ўваходных даных падчас прыёмачных выпрабаванняў.

## **2. Праектаванне сістэм**

Пасля таго, як патрабаванні да прадукту сталі зразумелымі і падрабязнымі, надыходзіць час распрацоўваць сістэму цалкам.

Праектаванне сістэмы ўключае усведамленне і падрабязнае апісанне ўсяго абсталявання і наладу сувязі для ПД у распрацоўцы.

На аснове праекта ПД распрацоўваеца План тэставання.

Калі зрабіць гэта на больш раннім этапе, пазней застасеца больш часу на фактычнае выкананне тэстаў.

## **3. Распрацоўка архітэктуры**

На гэтым этапе вызначаюцца і распрацоўваюцца архітэктурныя спецыфікацыі.

Звычайна прапануеца больш за адзін тэхнічны падыход, і канчатковае рашэнне прымаецца на падставе тэхнічнай і фінансавай мэтазгоднасці.

Праектаванне сістэмы далей дзеліцца на модулі, якія выконваюць розныя функцыі.

Інакш гэта называеца Высоказроўневым праектаваннем.

На гэтым этапе выразна разумеюцца і вызначаюцца сродкі перадачы даных і сувязь паміж унутранымі модулямі і знешнім светам — іншымі сістэмамі.

З дапамогай інфармацыі вышэй на гэтым этапе можна распрацаваць і задокументаваць інтэграцыйныя тэсты.

## **4. Праектаванне модулей**

Гэты этап вызначае падрабязны ўнутраны змест усіх модуляў сістэмы і называецца Нізкаўзроўневым праектаваннем.

Важна, каб дызайн быў сумяшчальны з іншымі модулямі ў архітэктуры сістэмы і іншымі знешнімі сістэмамі.

Юніт-тэсты з'яўляюцца неад'емнай часткай любога працэсу распрацоўкі і дапамагаюць ліквідаваць максімальнью колькасць памылак і недахопаў на самай ранняй стадыі.

Модульныя тэсты могуць быць распрацаваны на гэтым этапе на аснове даных аб унутранай пабудове модулей.

## **5. Кадаванне**

Фактычнае кадаванне праграмных модулей, распрацаваных на этапе праектавання, выконваецца на этапе кадавання.

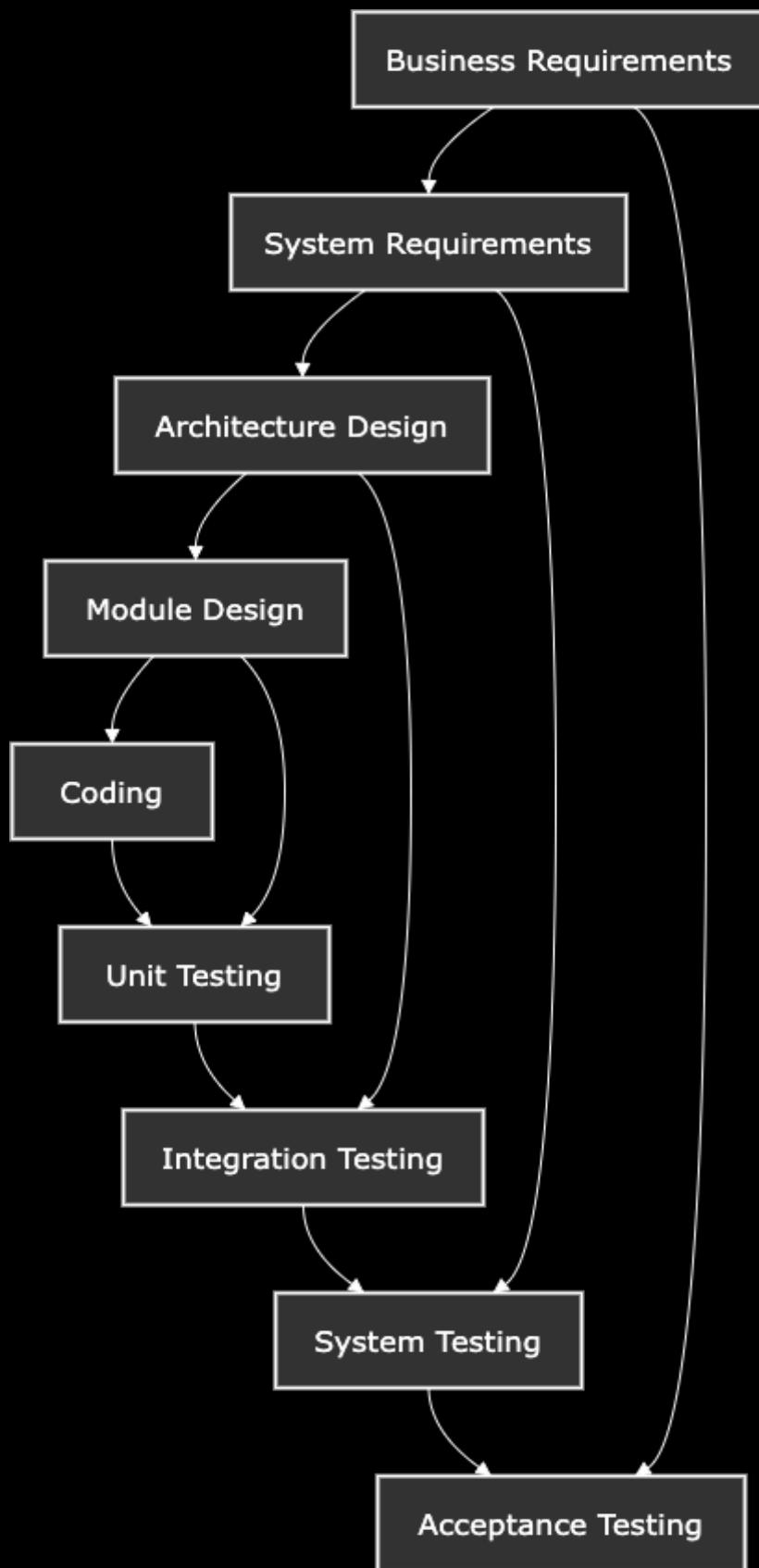
Найлепшая мова праграмавання вызначаецца ў залежнасці ад сістэмных і архітэктурных патрабаванняў.

Кадаванне выконваецца ў адпаведнасці з рэкамендацыямі і стандартамі кадавання.

Код праходзіць шматлікія праверкі і аптымізуецца дзеля лепшай якаснасці, перш чым канчатковая зборка будзе дабаўлена ў рэпозіторый.

## **Этапы валідацыі**

Розныя этапы валідацыі ў V-мадэлі падрабязна тлумачацца ніжэй.



*V-Model*

## **1. Юніт-тэсціраванне**

Юніт-тэсты, распрацаваныя на этапе праектавання модуляў, выконваюцца над кодам падчас гэтага этапу праверкі.

Юніт-тэставанне — гэта тэсціраванне на ўзоруі кода, якое дапамагае выявіць памылкі на ранній стадыі, хоць усе дэфекты нельга выявіць з дапамогай юніт-тэставання.

## **2. Інтэграцыйнае тэсціраванне**

Інтэграцыйнае тэсціраванне звязана з этапам архітэктурнага праектавання.

Інтэграцыйныя тэсты праводзяцца для праверкі суіснавання і сувязі ўнутраных модулей у сістэме.

## **3. Тэставанне сістэмы**

Тэставанне сістэмы непасрэдна звязана з этапам праектавання сістэмы.

Сістэмныя тэсты правяраюць усю функцыянальнасць сістэмы і сувязь распрацоўваемай сістэмы са зневшнімі сістэмамі.

Большасць проблем сумяшчальнасці праграмнага і апаратнага дачынення можна выявіць падчас выканання гэтага сістэмнага тэсту.

## **4. Прыёмачнае тэсціраванне**

Прыёмачнае тэсціраванне звязана з этапам аналіза бізнес-патрабаванняў і прадугледжвае тэставанне прадукту ў спажывецкім асяроддзі.

Прыёмачныя тэсты выяўляюць проблемы сумяшчальнасці з іншымі сістэмамі, даступнымі ў спажывецкім асяроддзі.

Ён таксама выяўляе нефункцыянальныя праблемы, такія як дэфекты нагрузкі і якаснасці ў рэальнym спажывецкім асяроддзі.

## Перавагі

Галоўная перавага V-мадэлі заключаецца ў тым, што яе вельмі лёгка зразумець і прымяніць.

Прастата гэтай мадэлі таксама спрашчае кіраванне ёю.

Усе перавагі V-мадэлі наступныя:

- Простая і лёгкая для разумення і ўжывання
- Лёгка кіраваць дзякуючы трываласці мадэлі
- Кожны этап мае пэўныя вынікі і працэс праверкі
- Дазваляе працеваць з невялікімі праектамі, патрабаванні якіх добра зразумелыя
- Этапы выконваюцца па адным

## Недахопы

V-мадэль не з'яўляецца гнуткай да змен, якія з'яўляюцца звычайнімі ў дынамічным свеце, і гэта найбольшы недахоп.

Патрабаванні да ПД мяняюцца, і іх рэалізацыя абыходзіцца вельмі дорага.

Іншыя недахопы V-мадэлі наступныя:

- рызыка і нявызначанасць знаходзяцца на высокім узроўні
- складаныя і аб'ектна-арыентаваныя праекты адхіляюцца
- працяглыя і бягучыя праекты адхінаюцца
- праекты, патрабаванні якіх маюць сярэдні або высокі рызыка змены, адхіляюцца
- на этапе тэставання да праграмы цяжка вярнуцца і змяніць функцыянальнасць
- працоўнае ПД адсутнічае да завяршэння жыццёвага цыклу

## Прымяненне

Паводле прымянення V-мадэль вельмі падобная да мадэлі Вадаспаду, паколькі абедзве мадэлі з'яўляюцца паслядоўнымі.

Патрабаванні павінны быць вельмі выразна сформуляваны да пачатку праекта, бо вяртацца і ўносіць змены звычайна дорага.

Гэтая мадэль часта ўжываецца ў галіне медыцынскіх распрацовак, якая з'яўляецца строга дысцыплінаванай вобласцю.

Ніжэй прыведзены некаторыя з найбольш прыдатных сцэнарыяў ужытку V-мадэлі:

- патрабаванні добра акрэслены, выразна задокументаваны і зафіксаваны
  - вызначэнне прадукту стабільнае
  - тэхнологія не з'яўляецца новай і добра зразумела камандзе праекта
  - няма двухсэнсоўных або неакрэсленых патрабаванняў
  - праект кароткі паводле часу
-

## **V. Адаптыўныя МПД**

## 5. Адаптыўныя МППД

**Адаптыўныя МППД** — гэта мадэлі, якія ўлічваюць, што патрабаванні змяняюцца падчас распрацоўкі, і падкрэсліваюць гнуткасць, супрацоўніцтва і паставанне ўдасканаленне.

Адаптыўная МППД спалучае ў сабе інкрементальную і ітэрацыйную распрацоўку.

Гэта прадугледжвае паступовае дабаўленне функцый і ўнясенне змененняў і ўдасканаленняў у адпаведнасці з водгукамі.

Іншымі словамі, праца можа лёгка адаптавацца да зменлівых патрабаванняў на аснове новых водгукаў, атрыманых ад кліента.

### Ключавы элемент

Ключавым элементам адаптыўнай МППД з'яўляецца тое, што, хоць яна і вызначае пэўныя этапы на працягу ўсяго ЖЦР ПД, але ў той час і надае ім пэўнай гнуткасці.

Адаптыўная МППД сканцэнтравана на дасягненні жаданай канчатковай мэты шляхам хуткай адаптацыі да дынамічных бізнес-патрабаванняў.

Гэта і надае больш увагі бягучым патрабаванням, і пакідае простору для змен маштаба праекта ў будучыні.

### Гнуткасць

Супольна усе адаптыўныя МППД называюцца Гнуткімі (*Agile*) метадамі, пасля таго, як **Agile-маніфест** быў апублікаваны ў 2001 годзе.

*Agile*-падыход узнік на ранніх этапах распрацоўкі ПД і стаў папулярным дзякуючы сваёй гнуткасці і адаптыўнасці.

## Гнуткія метады

Ніжэй прыведзены найбольш папулярныя Agile-метады:

- Хуткая распрацоўка ПД — RAD
- Хуткае прататыпаванне
- Распрацоўка дынамічных сістэм
- Рацыянальны ўніфікаваны працэс
- Скрам (Scrum — схватка)
- Крышталёва чысты
- Экстрэмальнае праграмаванне
- Распрацоўка, арыентаваная на функцыі

Скрам, безумоўна, з'яўляецца самым папулярным і фактычна стандартным метадам Agile-распрацоўкі, хутчэй за ўсё, таму, што яго лёгка і рэалізаваць, і падтрымліваць.

---

## 5.1. Гнуткая метадалогія

**Гнуткая метадалогія** — гэта спалучэнне ітэрацыйных і інкрементальных МППД з акцэнтам на адаптыўнасць працэсаў і задавальненне кліентаў шляхам хуткай паставкі працоўнага ПД.

### Дзейнасці

Гнуткія метады разбіваюць ПД на невялікія інкрементальныя зборкі.

Гэтыя зборкі прапануюцца ў ітэрацыях.

Звычайна кожная ітэрацыя доўжыцца ад аднаго да трох тыдняў.

Кожная ітэрацыя ўключае ў сябе міжфункцыянальныя каманды, якія адначасова працуюць над рознымі галінамі, такімі як:

- Планаванне
- Аналіз патрабаванняў
- Праектаванне
- Кадаванне
- Модульнае тэсціраванне
- Прыёмачнае тэставанне

У канцы ітэрацыі працоўны прадукт дэманструеца кліенту і важным зацікаўленым бакам.

### Скрынкі часу

Agile-метадалогія лічыць, што кожны праект павінен разглядацца па-рознаму, і існуючыя метады павінны быць адаптаваныя да патрабаванняў праекта.

У Agile задачы падзяляюцца на **Скрынкі часу** — невялікія тэрміны — для дабаўлення ў рэліз пэўных функцый.

Далей ужываецца ітэрацыйны падыход, і пасля кожнай ітэрацыі паставляеца працоўная зборка ПД.

Кожная зборка паступова пашыраеца з пункту гледжання функцый.

Канчатковая зборка мае ўсе функцыі, неабходныя кліенту.

---

## 5.2. Гнуткія метады

Гнуткія метады шырока ўжываюцца ў свеце ПД. Аднак гэтыя метады не заўсёды падыходзяць для ўсіх дачыненій.

Ніжэй — некаторыя перавагі і недахопы методыкі Agile.

### Перавагі

Асноўныя перавагі методыкі Agile наступныя:

- Зручная камандная праца і крос-трэнінгі
- Хуткая распрацоўка і дэманстрацыя рашэнняў
- Мінімальная патрабаванні да рэурсаў
- Гатоўнасць да фіксаваных або зменлівых патрабаванняў
- Пастаянная адаптацыя да змен у асяроддзі
- Мажлівасць адначасовай распрацоўкі і паставкі
- Мінімум планавання, правілаў і дакументацыі
- Сапраўдная гнуткасць для распрацоўшчыкаў і простае кіраванне

### Недахопы

Асноўныя недахопы методыкі Agile наступныя:

- Дрэнная апрацоўка складаных залежнасцей
- Устойлівасць, магчымасць абслугоўвання і пашыральнасць знаходзяцца пад высокай рызыкай
- Абавязкова неабходны агульны план, лідар і кіраунік праекта
- Строгія тэрміны паставкі паправак і новай функцыянальнасці
- Моцная залежнасць ад узаемадзеяння з кліентамі
- Складанасці з перадачай тэхналогій і высокая індывідуальная залежнасць з-за недахопу дакументацыі

## 5.3. Маніфест Agile

### Краевугольныя камяні Agile

Ніжэй прыведзены асноўныя прынцыпы Agile Manifesto:

- Асобы і іх узаемаадносіны важней Працэсаў і інструментаў
- Працоўнае ПД важней Падрабязнай документацыі
- Супрацоўніцтва з кліентам важней Пунктаў контракта
- Рэагаванне на змены важней Выканання плана

### Прынцыпы Agile

Прынцыпы, на якіх грунтуецца Agile-маніфест:

- Наш галоўны прыярытэт — здавальненне патрэб кліента праз своечасовую і бесперапынную паставку плённага ПД
- Змены патрабавань вітаюцца нават напрыканцы распрацоўкі
- Гнуткія працэсы спажываюць змены на плён канкурэнтнай перавагі кліента
- Паставляйце працоўнае ПД часта, кожныя некалькіх тыдняў ці месяцаў, надаючы перавагу больш сціслым тэрмінам
- Бізнесмены і распрацоўшчыкі павінны штодня працеваць разам на працягу ўсяго праекта
- Будуйце праекты вакол матываваных людзей. Забяспечце ім неабходнае асяроддзе і падтрымку і даверце ім выкананне працы
- Найбольш эфектыўны і дзейсны метад перадачы інфармацыі камандзе распрацоўшчыкаў і ўнутры яе — гэта асабістая размова
- Працоўнае ПД — асноўны паказчык прагрэсу
- Гнуткія працэсы спрыяюць устойліваму развіццю. Спонсары, распрацоўшчыкі і спажыўцы павінны мець мажлівасць ахінаць тэмп нязменным бясконца доўга

- Пастаянная ўвага да тэхнічнай дасканаласці і пільнасці праектавання павышае гнуткасць
  - Простасць — мастацтва максімізацыі колькасці амінутай працы — мае важнае значэнне
  - Найлепшыя архітэктура, патрабаванні і праектаванне з'яўляюцца вынікам самаарганізаваных каманд
  - Рэгулярна камандзе варта паразважаць, як стаць больш эфектыўнай, а потым адпаведна выправіць і перапрацаўваць свае паводзіны
-

## 5.4. Хуткая распрацоўка ПД

**Хуткая распрацоўка дачыненняў** — Rapid Application Development, RAD — гэта метад распрацоўкі ПД, заснаваны на прататыпаванні і ітэрратыўнай распрацоўцы без планавання наогул.

### Ключавыя віды дзейнасці

Хуткая распрацоўка ПД мае наступныя правілы:

- Збор патрабаванняў кліента праз семінары або фокус-групы
- Раннєе тэставанне прататыпаў заказчыкам, ужываючы карцэпцыю ітэрацый
- Пераўживанне існуючых прататыпаў — кампанентаў
- Бесперапынная інтэграцыя і хуткая дастаўка
- Аўтаматызацыя руцінных аперацый
- Візуальнае праграмаванне прататыпаў

### Ключавыя мэты

Хуткая распрацоўка дачыненняў — гэта метадалогія распрацоўкі ПД, якая ўжывае мінімальнае планаванне на плён хуткаму прататыпаванию.

**Прататып** — гэта працоўная мадэль, якая функцыянальна эквівалентна кампаненту прадукта.

У мадэлі RAD функцыянальныя модулі распрацоўваюцца паралельна ў якасці прататыпаў і інтэгруюцца для стварэння цэльнага прадукта для больш хуткай пастаўкі.

Паколькі няма падрабязнага папярэдняга планавання, гэта спрашчае ўключэнне змяненняў у працэс распрацоўкі.

## **Працоўная плынь**

Праекты RAD трывамаюцца ітэрацыйнай і інкрементальнай мадэлі, у якой невялікія каманды распрацоўшчыкаў, экспертаў у предметнай вобласці, прадстаўнікоў кліентаў і іншых IT-асоб паступова працуюць над сваім кампанентам або прататыпам.

Найважнейшым аспектам поспеху гэтай мадэлі з'яўляецца магчымасць паўторнага ўжытку распрацаваных прататыпаў.

## **Перавагі**

Асноўныя перавагі мадэлі RAD наступныя:

- Змены патрабаванняў улічваюцца
- Прагрэс можна вымераць
- Час ітэрацыі можа быць кароткім пры спажыванні магутных інструментаў RAD
- Якаснасць з меншай колькасцю людзей за кароткі час
- Скарочаны час распрацоўкі
- Павялічаная магчымасць паўторнага ўжытку кампанентаў
- Хуткія першыя водгукі кліентаў даюць хуткія плён
- Інтэграцыя з самага пачатку вырашае мноства дашейных проблем

## **Недахопы**

Асноўныя недахопы мадэлі RAD наступныя:

- Залежнасць ад тэхнічна моцных членаў каманды для вызначэння бізнес-патрабаванняў
- Можна пабудаваць толькі модульную сістэму.
- Патрабуюцца дасведчаныя распрацоўшчыкі і дызайнеры
- Высокая залежнасць ад навыкаў мадэлявання
- Непрыдатна для танных праектаў, бо кошт мадэлявання і аўтаматызаванай генерацыі кода вельмі высокі

- Складанасць кіравання больш пасуе для сістэм, якія заснаваныя на кампанентах і маштабуюцца
- Патрабуецца ўдзел спажыўца на працягу ўсяго ЖЦР ПД
- Пасуе для праектаў з карацейшымі тэрмінамі распрацоўкі

## Прымяненне

Мадэль RAD можа быць паспяхова прыменена да праектаў, у якіх магчымая выразная мадулярызацыя.

Калі праект нельга разбіць на модулі, RAD можа пацярпець няўдачу.

Наступныя парады апісваюць тыповыя сцэнарыі, у якіх можна ўжыць RAD:

- Сістэму можна мадуляваць для паступовай пастаўкі
- Існуе высокая даступнасць дызайнераў для мадэлявання
- Бюджэт дазваляе спажыць аўтаматызаваныя інструменты генерацыі кода
- Эксперты ў даменнай вобласці валодаюць адпаведнымі бізнес-ведамі
- Змены патрабаванняў і працоўныя прататыпы павінны быць прадстаўлены кліенту невялікімі ітэрацыямі па 2-3 месяцы.

Мадэль RAD забяспечвае хуткую дастаўку, бо скарачае агульны час распрацоўкі дзякуючы магчымасці паўторнага ўжытку кампанентаў і паралельнай распрацоўцы.

RAD працуе добра толькі ў тым выпадку, калі ёсць высокакваліфікованыя інжынеры, а кліент таксама імкненца стварыць мэтавы прататып у заданыя тэрміны.

Калі хоць адзін са складнікаў бракуе, мадэль можа пацярпець няўдачу.

## **5.5. Хуткае прататыпаванне**

**Хуткае прататыпаванне** адносіцца да стварэння прататыпаў, здольных засведчыць функцыянальнасць ПД у распрацоўцы.

Аднак прататып можа не цалкам адпавядыць логіцы арыгінальнага ПД.

Хуткае прататыпаванне становіцца вельмі папулярным, бо дазваляе камандзе зразумець патрабаванні кліентаў на раннай стадыі распрацоўкі.

Яно таксама дапамагае распрацоўшчыкам ПД атрымліваць водгукі ад кліентаў і адчуць, што менавіта чакаеца ад створанага прадукту.

### **Асноўныя віды дзейнасці**

**Прататып** — гэта рабочая мадэль ПД з абмежаванай функцыянальнасцю.

Прататып не заўсёды дакладна адпавядае логіцы, ужытай у рэальным дачыненні, і патрабуе дадатковых намаганняў, якія трэба ўлічваць пры ацэнцы намаганняў.

Прататыпаванне ўжываецца для таго, каб спажыўцы маглі ацаніць прапановы распрацоўшчыкаў і паспрабаваць іх перад рэалізацыяй.

Гэта таксама дапамагае зразумець патрабаванні, якія з'яўляюцца спецыфічнымі для спажыўца і, магчыма, не былі ўлічаны распрацоўшчыкам падчас распрацоўкі ПД.

## Ключавыя мэты

Прататыпаванне ПД варта ўжываць толькі тады, калі намаганні, затрачаныя на стварэнне прататыпа, дадаюць значную каштоўнасць канчаткова распрацаванаму дачыненню.

### Перавагі

Асноўныя перавагі хуткага прататыпавання наступныя:

- Павышаная зацікаўленасць спажыўцу ў ПД яшчэ да яго стварэння
- Паколькі адлюстроўваецца рабочая мадэль сістэмы, спажыўцы лепш разумеюць ПД у распрацоўцы
- Скарачае час і выдаткі, бо дэфекты можна выявіць значна раней
- Даступная хутчэйшая зворотная сувязь са спажыўцамі, што прыводзіць да лепшых рашэнняў
- Адсутныя, заблытаныя або складаныя функцыі можна лёгка вызначыць

### Недахопы

Асноўныя недахопы хуткага прататыпавання наступныя:

- Рызыка недастатковага аналізу патрабаванняў праз занадта вялікую залежнасць ад прататыпа
- Спажыўцы могуць блытацца ў прататыпах і рэальных сістэмах
- На практицы гэтая методыка можа павялічыць складанасць сістэмы, бо аб'ём сістэмы можа паширыцца за межы першапачатковых планаў
- Распрацоўшчыкі могуць паспрабаваць паўторна ўжываць існыя прататыпы для стварэння ПД, нават калі гэта тэхнічна немагчыма
- Намаганні, укладзеныя ў стварэнне прататыпаў, могуць быць занадта вялікімі, калі яны не цэнтралююцца належным чынам

## Прымяненне

Хуткае прататыпаванне мае найбольшую каштоўнасць для распрацоўкі ПД з высокім узроўнем узаемадзеяння са спажыўцом, напрыклад, для онлайн-дачыненя.

Сістэмы, якія патрабуюць ад спажыўцу запаўнення форм альбо пераход на розныя экраны перад апрацоўкай даных, могуць вельмі трапна ўжываць прататыпаванне, каб надаць ПД дакладны выгляд і ўражанне яшчэ да таго, як яно будзе створана фактычна.

ПД, якое патрабуе занадта шмат апрацоўкі даных і ахінае большую частку функцыянальнасці ўнутры, з вельмі абмежаваным спажывецкім інтэрфейсам, звычайна не атрымлівае выгады ад прататыпавання.

Распрацоўка прататыпа можа прынесці ў такіх праектах новыя выдаткі і запатрабаваць шмат дадатковых намаганняў.

---

## 5.6. Скрам

**Скрам** — гэта лёгкі, ітэратыўны і інкрементальны Agile фрэймворк распрацоўкі, пастаўкі і падтрымкі складаных прадуктаў.

Ён прызначаны для каманд з дзесяці ці менш чалавек, якія разбіваюць сваю працу на ітэрацыі, якія называюцца спрынты, працягласцю не больш за адзін месяц, а часцей за ўсё — два тыдні.

Каманда Скрам адсочвае прагрэс на штодзённых сустрэчах працягласцю 15 хвілін, якія называюцца **Штодзённымі скрамамі**.

Напрыканцы Спрынта каманда праводзіць агляд Спрынта, каб прадэманстраваць выкананую працу, і рэтраспектыву Спрынта для пастаяннага ўдасканалення працэсаў.

### Прынцыпы Скрам

Скрам дазваляе камандам самаарганізоўвацца, заахвочваючы сумеснае спажыванне рэурсаў або цесную супрацу онлайн ўсіх членоў каманды, а таксама штодзённую асабістую камунікацыю паміж усімі членамі каманды і зацікаўленымі дысцыплінамі.

Ключавы прынцыпам Скрам — узаемнае ўсведамленне таго, што кліенты могуць змяніць сваё меркаванне адносна таго, чаго яны хочуць або патрабуюць, што вядзе да хісткасці патрабаванняў, а таксама што будуць непрадказальныя праблемы, для якіх прагнастычны падыход не падыходзіць.

Такім чынам, Скрам ужывае эмпірычны падыход, заснаваны на доказах, — прызнаючы, што праблему нельга цалкам зразумець або вызначыць загадзя, і замест гэтага засяроджваючыся на tym, як максымізаваць здольнасць каманды хутка выконваць працу, рэагаваць на новыя патрабаванні і адаптавацца да новых тэхналогій і змен рынковых умоў.

## **VI. Фрэймворк Скрам**

## 6. Фрэймворк Скрам

**Скрам** — гэта эмпірычны падыход, заснаваны на зваротнай сувязі, які падмацоўваецца празрыстасцю, аглядам і адаптацыяй.

**Празрыстасць** азначае, што кожныя плынъ, працоўны працэс або прагрэс у рамках Скрам павінны быць бачныя тым, хто адказвае за вынік.

**Часты агляд** распрацоўвае мага працу тут і таго, на сколькі добра кожны з іх працуе, — гэта найлепшы спосаб зрабіць працэсы бачнымі для ўсёй каманды.

**Адаптацыя**, заснаваная на частым аглядзе, — гэта здольнасць каманды выяўляць адхіленні ад пачатковых мэтаў і выпраўляць працэс распрацоўкі.

### Каштоўнасці Скрам

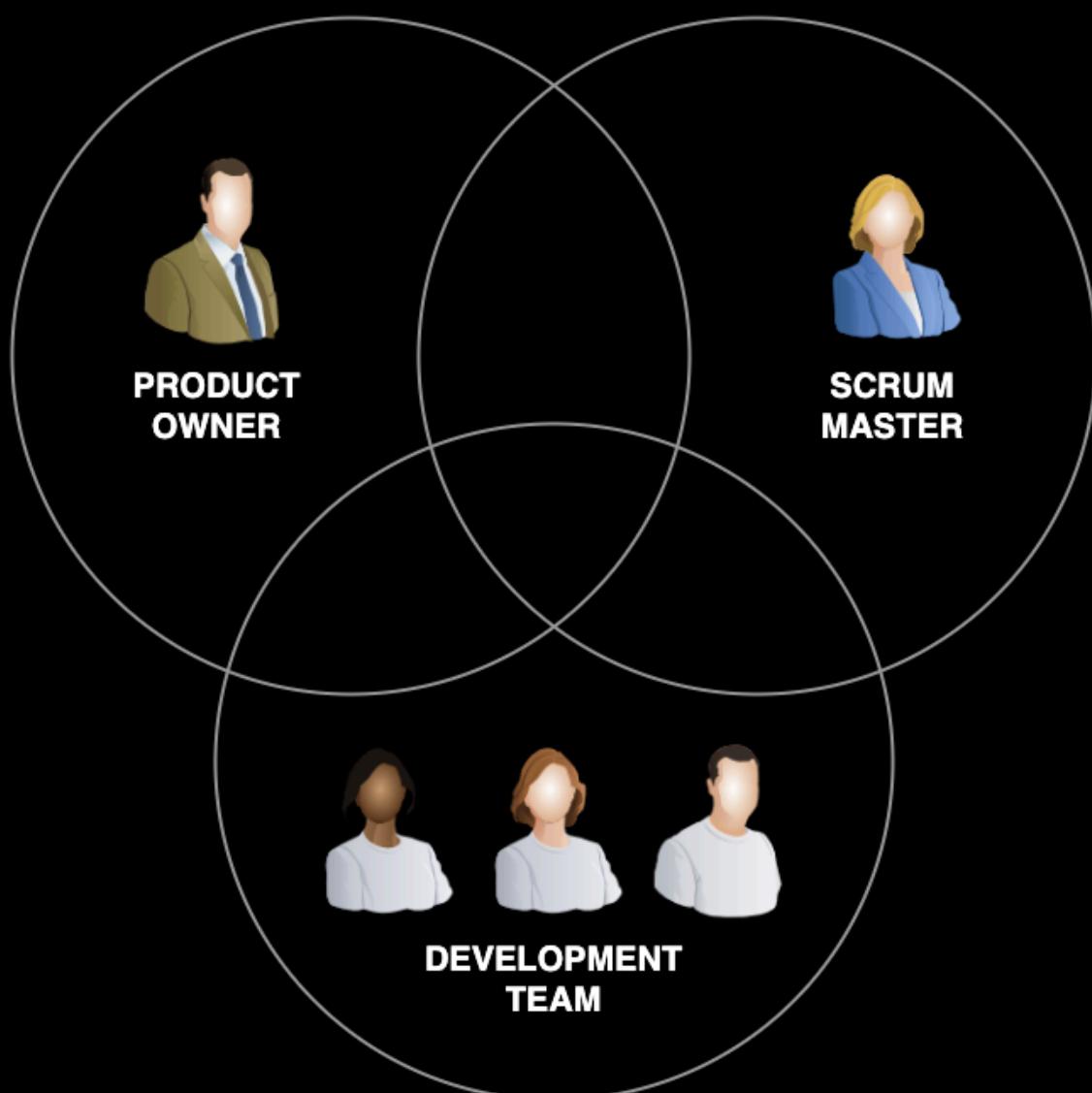
Фрэймворк Скрам падкрэслівае пяць асноўных каштоўнасцей, якімі кіруюцца праца, дзеянні і паводзіны членаў Скрам-каманды:

- Канцэнтрацыя — дазваляе камандзе засяродзіцца на сваёй мэце, пазбягаючы іншай працы
- Павага — прымушае ўдзельнікаў паважаць адзін аднаго як кваліфікованых спецыялістаў
- Адкрытасць — спрыяе згодзе паміж камандай і зацікаўленымі бакамі, каб адкрыта абмяркоўваць сваю працу
- Прыхільнасць — дапамагае кожнаму ўдзельніку прысвяціць сябе дасягненню агульнай мэты
- Смеласць — дазваляе камандзе рабіць правільныя рэчы, сутыкаючыся са складанымі проблемамі

## 6.1. Каманда Скрам

**Каманда Скрам** з'яўляецца фундаментальнай адзінкай фрэймворка Скрам — згуртаванай групай прафесіяналаў, якія працуюць разам для дасягнення каштоўных, якасных вынікаў праз супрацоўніцтва і ітэратыўны прагрэс.

### Структура Скрам-каманды



Скрам-каманда складаецца з трох ключавых роляў, кожная з якіх мае асобныя абавязкі:

- **Уладальнік прадукту** — Прадстаўляе зацікаўленыя бакі, вызначае прыярытэты і забяспечвае максімальную каштоўнасць для бізнесу ад каманды
- **Скрам-майстар** — Выступае ў якасці пасярэдніка, ліквідуючы перашкоды і ствараючы атмасферу, у якой каманда можа эфектыўна працаваць
- **Каманда распрацоўшчыкаў** — Самаарганізаваная міжфункцыянальная група прафесіяналаў, якія праектуюць, ствараюць і тэстуюць прырашчэнне прадукту

## Ключавыя прынцыпы

Скрам прыносіць плён дзякуючы празрыстасці, супрацоўніцтву і пастаяннаму ўдасканаленню.

Каб дасягнуць поспеху, каманда павінна:

- Аб'яднацца вакол агульнай мэты — кожны ўдзельнік уносіць свой уклад у дасягненне адзінай мэты
- Падтрымліваць каштоўнасці Скрам — адданасць справе, мужнасць, мэтанакіраванасць, адкрытасць і павагу
- Часта меркавацца — штодзённыя ўзаемадзеянні (выступы, агляды спрынтаў) забяспечваюць згоду і адаптыўнасць

Развіваючы культуру даверу, падсправаздачнасці і адаптыўнасці, каманда Скрам максімізуе эфектыўнасць і дасягае значных вынікаў у кожным спрынце.

---

## **6.2. Уладальнік прадукта**

**Уладальнік прадукта**, УП — альбо пастваўшчык ПД на рынак — гэта прадстаўнік зацікаўленых у ПД бакоў і кліентаў, адказны за станоўчыя бізнес-вынікі.

Уладальнік прадукта вызначае функцыянальнасць ПД у арыентаваных на кліента тэрмінах — звычайна гэта Гісторыі спажыўцу — і прыярытэзуе яе ў залежнасці ад важнасці для ПД.

У Скрам-каманды павінен быць толькі адзін УП, хоць адзін УП можа падтрымліваць больш чым адну каманду.

### **Ключавая дзейнасць**

Уладальнік прадукту сканцэнтраваны на бізнес-баку распрацоўкі ПД, аб'ядноўваючы намаганні зацікаўленых бакоў і каманды Скрам.

Уладальнік прадукту не паказвае, як каманда Скрам дасягае тэхнічнай мэты, а хутчэй імкненцца да згоды сярод членаў каманды Скрам.

Гэтая роля вельмі важная і патрабуе глыбокага разумення абодвух бакоў, бізнесу і распрацоўшчыкаў, у камандзе Скрам.

### **Ключавыя мэты**

Добры ўладальнік прадукту павінен умець паведаміць патрэбы бізнесу, выбраць найлепшыя спосабы дасягнення сваіх мэтаў і данесці сваё паведамленне да ўсіх зацікаўленых бакоў, у тым ліку да распрацоўшчыкаў, ужываючы тэхнічную мову, па меры неабходнасці.

Уладальнік прадукту ўжывае эмпірычныя інструменты Скрам для кіравання вельмі складанай працай, цэнтралюючы рызыкі і дасягаючы каштоўнасці.

## **Ключавыя абавязкі**

Камунікацыя — гэта асноўная адказнасць уладальніка прадукта.

Здольнасць перадаваць прыярытэты і падтрымліваць членаў каманды і зацікаўленыя бакі мае жыццёва важнае значэнне, каб накіроўваць распрацоўку ПД ў правільнае рэчышча.

Роля ўладальніка прадукта ліквідуе разрыў у камунікацыі паміж камандай і яе зацікаўленымі бакамі, выступаючы ў якасці пасярэдніка зацікаўленых бакоў перад камандай і прадстаўніка каманды перад усёй супольнасцю зацікаўленых бакоў.

## **Задачы**

Як твар каманды перад зацікаўленымі бакамі, уладальнік прадукта павінен выканаць наступныя камунікацыйныя задачы:

- Вызначаць і паведамляць пра рэліз
  - Паведамляць пра стан каманды і ПД
  - Абмяркоўваць прагрэс падчас кіроўных сустрэч
  - Дзяліцца з зацікаўленымі бакамі інфармацыяй пра значныя рызыкі, перашкоды, залежнасці і здагадкі
  - Узгадняць прыярытэты, аб'ём, фінансаванне і графік
  - Забяспечваць бачны, празрысты і зразумелы вобраз ПД
-

## 6.3. Скрам-майстар

**Скрам-майстар** — гэта асоба, адказная за ліквідацыю перашкод на шляху каманды да дасягнення мэтаў развіцця і паставленых задач.

Скрам-майстар гарантуе, што каманда прытрымліваецца прынцыпаў фрэймворка Скрам, вядзе ключавыя сесіі і заахвочвае каманду да ўдасканалення.

### Ключавыя абавязкі

Асноўныя абавязкі Скрам-майстра:

- Дапамога ўладальніку прадукта
- Падтрымка каманды і пашырэнне магчымасцей
- Коўчынг Скрам-каманды
- Стварэнне высокакаштоўных інкрэментаў
- Правядзенне Скрам-мерапрыемстваў

Скрам-майстру не дазволена мець абавязкаў па кіраванні персаналам, таму роля Уладальніка Прадукта ніколі не павінна сумяшчацца з ролем Скрам-майстра.

---

## 6.4. Каманда распрацоўшчыкаў

**Каманда распрацоўшчыкаў** — гэта адзінка, якая выконвае ўсе задачы, неабходныя для стварэння прырашчэння каштоўнай прадукцыі.

Членаў каманды называюць **Распрацоўшчыкамі**.

Тэрмін **распрацоўшчык** адносіцца да любога чалавека, які ўдзельнічае ў распрацоўцы і падтрымцы Пд, таму сюды ўваходзяць:

- Аналітыкі
- Архітэктары
- Спецыялісты па датых
- Дызайнеры
- Распрацоўшчыкі
- Даследчыкі
- Тэсціроўшчыкі

— і іншыя.

### Арганізацыя каманды

Каманда распрацоўшчыкаў самаарганізуецца.

Нягледзячы на тое, што ніякая праца не павінна паступаць у каманду, акрамя як праз Уладальніка Прадукту, а Скрам-майстар павінен абараняць каманду ад празмернай развеі ўвагі, каманду ўсё ж неабходна заахвочваць да непасрэднага ўзаемадзеяння з зацікаўленымі бакамі і кліентамі, каб атрымаць максімальнае разуменне і неадкладнасць зваротнай сувязі.

---

## 6.5. Артэфакты Скрам

**Артэфакты Скрам** — гэта інфармацыя, якую Скрам-каманда і зацікаўленыя бакі ўжываюць для падрабязнага апісання ПД, дзей па яго стварэнні і дзеянняў, якія выконваюцца падчас праекта.

У рамках Скрам ёсць трох асноўных артэфакты, якія адыгрываюць вырашальную ролю ў дасягненні празрыстасці і садзейнічанні эфектыўнаму супрацоўніцтву:

- Бэклог прадукту
- Бэклог спрынту
- Павялічэнне

Гэтыя артэфакты забяспечваюць празрыстасць, спрыяюць прыняцю рашэнняў і служаць асновай для адаптацыі ўнутры каманды Скрам і яе зацікаўленых бакоў.

### Бэклог прадукта

**Бэклог прадукта** — гэта дынамічны спіс элементаў, якія прадстаўляюць працу, неабходную для стварэння прадукта.

Ён уключае новыя функцыі, паляпшэнні, выпраўленні памылак, задачы і іншыя патрабаванні да працы.

УП адказвае за падтрыманне і прыярытэтызацыю элементаў у бэклогу.

Намер, звязаны з бэклогам прадукту, — гэта мэта прадукту.

### Бэклог спрынта

**Бэклог спрынта** — гэта падмножства элементаў Бэклога прадукта, адабраных для распрацоўкі на працягу пэўнага перыяду часу, які называецца Спрынт.

Каманда распрацоўшчыкаў сумесна вырашае, якія элементы ўключыць у БЭКЛОГ спрынта.

Намер, звязаны з БС, — гэта мэта спрынта.

## Інкрэмэнт

**Інкрэмэнт** — прырашчэнне — гэта сума працы, выкананай падчас спрынту, якая дадае каштоўнасць прадукту.

Ён уключае ў сябе распрацаваныя функцыі, паляпшэнні і выпраўленні памылак, якія адпавядаюць крытэрам гатоўнасці.

Намер, звязаны з Інкрэмэнтам, — гэта вызначэнне крытэров гатоўнасці.

---

## 6.6. Бэклог прадукта

**Бэклог прадукта** — БП — гэта падрабязны пералік прац для выкананння, які ўтрымлівае ўпараткованы спіс патрабаванняў да прадукта, падтрыманы Скрам-камандай.

Патрабаванні вызначаюць функцыі, выпраўленні памылак, нефункцыянальныя патрабаванні і г.д. — усё, што трэба зрабіць для стварэння жыццяздольнага ПД.

Адказнасць за бэклог прадукта і бізнес-каштоўнасць кожнага элемента бэклога прадукта нясе Уладальнік прадукта.

Уладальнік прадукта прыярытызуе элементы БП, зыходзячы з наступных меркаванняў:

- Рызыка
- Бізнес-каштоўнасць
- Залежнасці элемента
- Памер предмета
- Патрэбная дата

## Ключавая дзеяйнасць

Звычайна Уладальнік Прадукту і Скрам-каманда сумесна распрацоўваюць план выканання работ. Бэклог Прадукта:

- Фіксуе запыты на змяненне прадукта, утым ліку на даданне, замену і выдаленне пэўных функцый і выпраўленне проблем
- Забяспечвае, каб распрацоўшчыкі мелі працу, якая максімізуе бізнес-выгаду ад прадукту

БП развіваецца па меры з'яўлення новай інфармацыі пра прадукт і яго кліентаў, таму пазнейшыя спрынты могуць быць накіраваны на новую працу.

## 6.7. Бэклог спрынта

**Бэклог спрынта** — БС — гэта спіс задач, якія каманда павінна выкананаць падчас наступнага спрынта.

Спіс складаецца Скрам-камандай, якая паступова выбірае элементы Бэклога Прадукта паводле прыярытэту, пачынаючы з верху Бэклога Прадукта.

### Задачы

Распрацоўшчыкі могуць разбіць элементы БП на задачы.

Задачы ў БС ніколі не прызначаюцца — і не перадаюцца — членам каманды кімсьці іншым.

Хутчэй за ўсё, члены каманды падпісваюцца на задачы — або выбіраюць іх — па меры неабходнасці ў залежнасці ад прыярытэту бэклога, а таксама ад уласных навыкаў і магчымасцей.

Гэта спрыяе самаарганізацыі распрацоўшчыкаў.

### Дошка задач

БС з'яўляецца ўласнасцю распрацоўшчыкаў, і ўсе яго ацэнкі прапануюцца распрацоўшчыкамі.

Часта для прагляду і змены стану задач бягучага спрынту, такіх як "Зрабіць", "У працэсе" і "Гатава", ужываецца Дошка задач.

### Паўторная прыярытэзацыя

Пасля таго, як БС вызначаны, ніякая дадатковая праца не можа быць дадана ў БС нікім, апроч Каманды.

Пасля завяршэння Спрынта аналізуецца БП, пры неабходнасці пераглядаюцца яго прыярытэты, а ў наступны Спрынт выбіраецца наступны набор функцыянальнасцей.

---

## **6.8. Інкремент**

**Інкремент** — гэта патэнцыйна рэлізуемы вынік Спрынта, які адпавядаетя яго мэце.

### **Ключавая дзейнасць**

Інкремент фарміруеца з усіх рэлізаваных элементаў Бэклога Спрынту, інтэграваных з працай усіх папярэдніх Спрынтаў.

Інкремент павінен быць завершаны адпаведна крытэрыю гатоўнасці каманды Скрам.

### **Ключавая мэта**

Інкремент павінен быць цалкам функцыянальным і ў прыдатным для ўжытку стане незалежна ад таго, ці вырашыць Уладальнік Прадукту яго разгарнуць і спажыць.

---

## 6.9. Падзеі Скрам

**Падзеі Скрам** — інакш называныя **Цырымоніямі** — гэта асноўныя дзеянні, якія адбываюцца ў кожнай ітэрацыі Спрынта для падтрымкі структуры, стымулявання супрацоўніцтва і пастаяннага ўдасканалення ў працэсе Скрам.

У рамках Скрам існуе пяць асноўных падзей, якія адыгрываюць вырашальную ролю ў дасягненні празрыстасці, адаптацыі і эфектыўнага супрацоўніцтва:

- Спрынт
- Планаванне спрынта
- Штодзённая скрам
- Агляд спрынта
- Рэтраспектыва спрынта

### Спрынт

**Спрынт** — гэта фундаментальны абмежаваны па часе перыяд, калі Скрам-каманда сумесна працуе над дасягненнем пэўнай мэты.

Спрынт службыць сэрцам Скрам, дзе ідэі ператвараюцца ў матэрыяльную каштоўнасць.

Гэта забяспечвае паслядоўную і кароткую ітэрацыю для звартнай сувязі, што дазваляе камандзе праверыць і адаптаваць як свае працоўныя працэсы, так і элементы, над якімі яны працуяць.

Спрынты маюць фіксаваную працягласць, якая доўжыцца адзін месяц або менш.

Карацейшыя спрынты генеруюць больш цыклаў навучання і абмяжоўваюць рызыку меншымі часавымі рамкамі.

## **Планаванне спрынту**

**Планаванне спрынту** — гэта найважнейшая падзея ў структуры Скрам, якая рыхтуе глебу для прадуктыўнага Спрынту, вызначаючы мэту Спрынту і спосаб яе выканання.

Планаванне Спрынту ініцыюе яго, вызначаючы працу, якую трэба выкананаць падчас гэтага Спрынту, і гарантуе, што найбольш важныя элементы з Бэклога Прадукту абмеркаваныя і ўзгодненыя з Мэтай прадукту.

## **Штодзённы Скрам**

**Штодзённы Скрам** — таксама вядомы як **Штодзённы стэндап** — гэта кароткая штодзённая сустрэча распрацоўшчыкаў для праверкі прагрэсу ў дасягненні мэты Спрынту і адаптацыі адставання ад запланаваных задач па меры неабходнасці.

Мэта штодзённага Скрам — стварыць дзейсны план працы на наступны дзень, які спрашчае прыняцце рашэнняў і выяўляе перашкоды для іх пераадолення.

## **Агляд спрынту**

**Агляд спрынту** — гэта мерапрыемства па завяршэнні Спрынту, падчас якога каманда Скрам абмяркоўвае Інкрэмент з зацікаўленымі бакамі і вызначае будучыя папраўкі.

Агляд спрынту — гэта сустрэча, на якой дэманструюцца элементы бэклога прадукту, якія яшчэ чакаюць выканання.

Скрам-каманда сумесна абмяркоўвае, што рабіць далей, забяспечваючы каштоўны ўнёсак у наступнае планаванне спрынту.

## **Рэтраспектыва спрынту**

**Рэтраспектыва спрынту** — гэта сустрэча Скрам-каманды, на якой ацэньваецца выкананы Спрынт з пункту гледжання асоб, узаемадзеяння, працэсаў, інструментаў і іх азначэння гатоўнасці.

Распрацоўшчыкі дзеляцца тым, што прайшло добра падчас спрынту, проблемамі, якія ўзніклі, і тым, як гэтыя проблемы былі вырашаны ці не былі вырашаны.

Скрам-каманда вызначае вобласці для паляпшэння, каб прыстасаваць іх і павысіць эфектыўнасць наступнага спрынту.

---

## 6.10. Спрынт

**Спрынт** — таксама вядомы як **Ітэрацыя** або **Скрынка Часу** — гэта асноўная адзінка працягласці распрацоўкі ў Скрам.

Працягласць Спрынту — гэта працягласць, якая ўзгоднена і фіксуеца загадзя для кожнага Спрынту і звычайна складае ад аднаго тыдня да аднаго месяца, прычым часцей за ўсё два тыдні.

### Працоўны працэс

Кожны Спрынт пачынаецца з падзеі планавання Спрынту, падчас якога вызначаюцца яго мэта і запатрабаваныя элементы Бэклога прадукту.

Каманда прымае тое, што, на яе думку, готова, і пераўтварае гэта ў бэклог Спрынту з падрабязнай інфармацыяй аб неабходнай працы і прагнозам дасягнення мэты Спрынту.

Кожны Спрынт завяршаецца Аглядам Спрынту і Рэтраспектывай Спрынту, у якіх разглядаецца прагрэс, каб паказаць яго зацікаўленым бакам, а таксама вызначыць урокі і паляпшэнні для наступных Спрынтаў.

### Увага

Скрам падкрэслівае каштоўныя, плённыя вынікі ў канцы спрынту, якія сапраўды готовыя — праграмнае дачыненне, якое было цалкам інтэгравана, пратэставана, задокументавана і патэнцыйна готова да выпуску.

### Планаванне спрынту

У пачатку спрынту каманда Скрам праводзіць мерапрыемства па планаванні спрынту, каб:

- Узаемна абмеркаваць і ўзгадніць аб'ём працы, якую плануецаць выкананаць падчас Спрынту
- Выбраць элементы бэклога прадукту, якія можна выкананаць за адзін спрынт
- Падрыхтаваць бэклог спрынту, які ўключае працу, неабходную для выканання выбранных элементаў бэклога прадукту
- Узгадніць мэту спрынту і кароткае апісанне таго, што прагназуецаць да канца спрынту.

Па меры працаўкі дэталей некаторыя элементы бэклогу прадукту могуць быць падзелены або вернуты ў бэклог прадукту, калі каманда больш не лічыць, што можа выкананаць неабходную працу за адзін спрынт.

Максімальная працягласць планавання спрынту не павінна перавышаць 8 гадзін для 4-тыднёвага спрынту.

---

## 6.11. Штодзённы Скрам

**Штодзённы Скрам** — гэта штодзённая падзея абмежаванай працягласці, падчас якой Каманда Распрацоўкі сінхранізуе дзеянні, правярае прагрэс у дасягненні Мэты спрынту і адаптуе план на наступныя 24 гадзіны.

Простымі словамі, гэта штодзённая сустрэча каманды, каб дасягнуць узаемаразумення, але не падрабязная справаздача кіраўніку аб стане спраў.

Асноўныя рэкамендацыі для штодзённага Скрам наступныя:

- Усім распрацоўшчыкам варта падрыхтавацца
- Запрашаюцца усе, але прамаўляюць толькі распрацоўшчыкі
- Толькі каманда вырашае, як праводзіць свой штодзённы Скрам.

### Ключавыя характеристыкі

Штодзённы Скрам:

- Павінен адбывацца ў адзін і той жа час і ў адным і тым жа месцы штодня
- Абмежаваны — па часе — пятнаццаццю хвілінамі

Падчас штодзённага Скрам не павінна адбывацца падрабязных абмеркаванняў.

Падчас сустрэчы кожны член каманды звычайна адказвае на трывалыя пытанні, каб прайсці праверку і сінхранізацыю:

- Што я зрабіў учора для каманды, каб дасягнуць мэты спрынту?
- Што я планую зрабіць сёння, каб каманда дасягнула мэты спрынту?

- Ці бачу я якія-небудзь перашкоды на шляху каманды да дасягнення мэты спрынту?

## Ключавая мэта

Галоўная мэта штодзённага Скрам — забяспечыць, каб уся каманда мела выразнае, агульнае разуменне працы і магла хутка выявіць любыя праблемы, якія могуць перашкодзіць ім дасягнуць мэты спрынту.

Гэта гуртуе намеры, спрыяе самаарганізацыі і мінімізуе неабходнасць у іншых сустрэчах.

## Пасля сустрэчы

Пасля завяршэння сустрэчы асобныя ўдзельнікі могуць сабрацца разам, каб падрабязна абмеркаваць пытанні.

Такая сустрэча часам называецца групавой або асабістай сустрэчай.

---

## **6.12. Агляд спрынту**

Пасля завяршэння спрынту каманда праводзіць два мерапрыемствы:

- Агляд спрынту
- Рэтраспектыва спрынту

**Агляд спрынту** — гэта мерапрыемства для проверкі вынікаў Спрынту і вызначэння будучых адаптацый.

Каманда Скрам прадстаўляе вынікі сваёй працы ключавым зацікаўленым бакам, і абмяркоўвае прагрэс у дасягненні мэты ПД.

### **Ключавыя характеристыкі**

Падчас агляду спрынту каманда:

- Аглядзе выкананую працу і запланаваную працу, якая не была выканана
- Прапануе гатовыя вынікі зацікаўленым бакам і абмяркоўвае з імі далейшыя напрамкі працы

Агляд спрынту — гэта рабочая сесія, і каманда Скрам павінна пазбягаць яе абмежавання презентацыяй.

### **Рэкамендацыі**

Рэкамендацыі па аглядах спрынту:

- Незавершаную працу немагчыма прадэманстраваць
- Рэкамендаваная працягласць — дзве гадзіны для двухтыднёвага спрынту.

## **6.13. Рэтраспектыва спрынту**

**Рэтраспектыва спрынту** — гэта сустрэча для планавання шляхоў павышэння яго якасці і эфектуўнасці.

Скрам-каманда ацэньвае, як прайшоў апошні Спрынт з пункту гледжання ўдзельнікаў, іх узаемадзеяння, працэсаў, інструментаў і крытэрыя гатоўнасці.

На рэтраспектыве спрынту каманда:

- Разважае пра мінулы спрынты
- Вызначае і ўзгадняе дзеянні па пастаянным удасканаленні працэсаў

### **Рэкамендацыі**

Рэкамендацыі па падрыхтоўцы рэтраспектыў спрынту:

- Рэкамендаваная працягласць — паўтары гадзіны для двухтыднёвага спрынту.
- Арганізатор мерапрыемства — Скрам-майстар

### **Ключавыя характеристыстыкі**

Падчас рэтраспектывы спрынту неабходна адказаць на тры асноўныя пытанні:

- Што атрымалася добра падчас спрынту?
- Што пайшло не так падчас спрынту?
- Што палепшиць у наступным спрынце?

### **Абмежаванні**

Рэтраспектыва Спрынту — гэта завяршэнне Спрынту.

Такім чынам, яго звычайны максімум складае трох гадзіны на працягу месячнага спрынту.

Чым карацейшы спрынт, тым карацей павінна доўжыцца Рэтраспектыва.

---

## 6.14. Удасканаленне бэклога

**Удасканаленне бэклога** — альбо яго **Перагляд** — гэта пастаянны працэс ацэнкі элементаў Бэклога прадукту і праверкі іх належнай падрыхтоўкі такім чынам, каб яны былі зразумелымі і выканальнымі для каманд пасля таго, як трапляюць у Спрынты падчас іх планавання.

Падчас перагляду:

- Элементы бэклога прадукту могуць быць разбітыя на некалькі меншых
- Крытэрыі прыёмкі могуць быць удакладнены
- Залежнасці могуць быць выяўлены і даследаваны

Нягледзячы на тое, што першапачаткова ўдасканаленне бэклога не было асноўнай практыкай Скрам, яно было дадана ў працэс Скрам і прынята як спосаб кіравання якасцю элементаў БП, якія ўваходзяць у спрынт, з рэкамендаванымі інвестыцыямі да 10% ад магутнасці спрынту каманды.

## Тэхнічны доўг

**Тэхнічны доўг** — альбо **Запазычаны код** — гэта меркаваны кошт будучай пераробкі, выкліканы выбарам хуткага і простага коду зараз замест лепшых падыходаў, якія занялі б больш часу.

Тэхнічная запазычанасць таксама можа абмяркоўвацца падчас удасканалення бэклога.

## Адмена спрынту

Уладальнік прадукту можа адмяніць спрынт пры неабходнасці.

Уладальнік прадукту можа зрабіць гэта з улікам меркаванняў каманды, Скрам-майстра або кірауніцтва.

Напрыклад, кірауніцтва можа пажадаць, каб Уладальнік Прадукту адмяніў Спрынт, калі знешнія абставіны зводзяць на нішто каштоўнасць мэты Спрынту.

Калі Спрынт завяршаецца пазапланава, наступным крокам з'яўляецца правядзенне новага Планавання Спрынту, падчас якога разглядаецца прычына мінулага завяршэння.

---

## 6.15. Працоўны працэс Скрам

**Скрам** з'яўляеца гнуткім фрэймворкам, прызначаным для ітэратыўнай і паступовай распрацоўкі высакаякаснага ПД.

Яго структураваны, але адаптыўны працоўны працэс падкрэслівае:

- Гнуткасць
- Супрацоўнасць
- Пастаяннае ўдасканаленне

— і можа быць прадстаўлены з дапамогай схемы ніжэй.

### Ключавыя этапы працэсу Скрам

#### 1. Стварэнне і ўдасканаленне бэклога прадукту

- Уладальнік прадукту падтрымлівае прыярытэтны спіс функцый, паляпшэнняў і выпраўленняў, які называецца Бэклог прадукту
- Элементы БП — гісторыі спажыўцу, памылкі, задачы — вызначаюцца з дапамогай ацэнак і крытэрыяў гатоўнасці

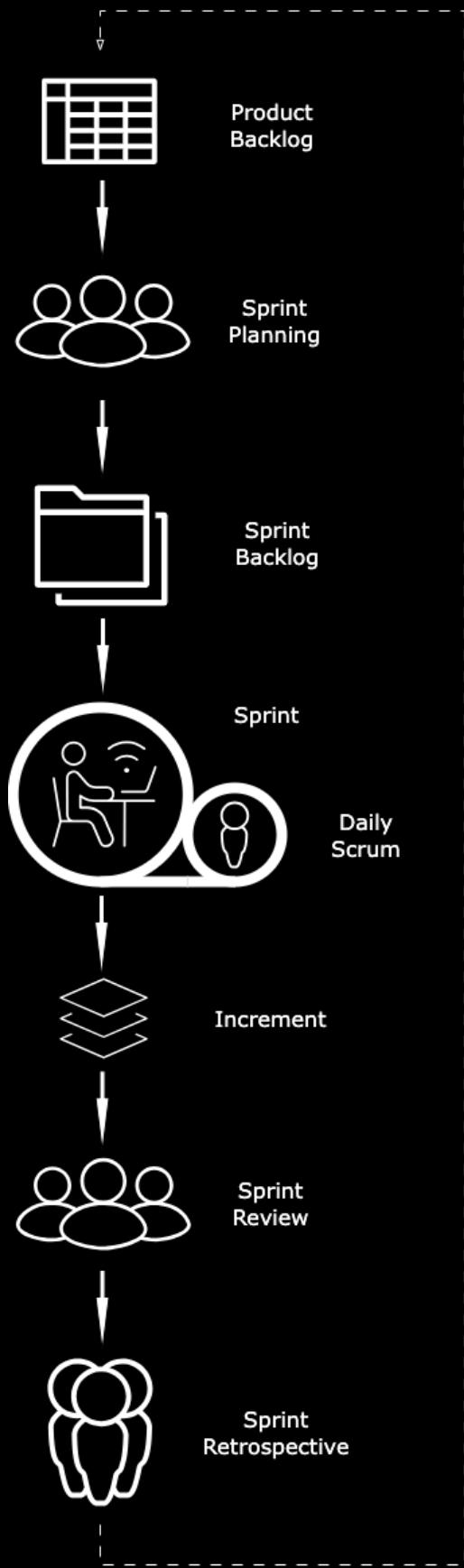
#### 2. Планаванне спрынту

- Каманда выбірае элементы БП на наступны Спрынт
- Далей — вызначае Мэту спрынту і стварае Бэклог спрынту

#### 3. Штодзённы Скрам

Гэта 15-хвілінная штодзённая сустрэча каманды, каб:

- Сінхранізаваць рабочую дзейнасць членаў каманды
- Забяспечце празрыстасць і хуткае вырашэнне проблем



*Scrum Process*

## **4. Выкананне спрынту**

- Каманда распрацоўшчыкаў збірае, тэстуе і інтэгруе функцыі за кароткія цыклы
- Праца адсочваецца праз Спрынт-табліцу з колонкамі "План", "У працы" і "Гатова"

## **5. Агляд спрынту**

- У канцы спрынту каманда дэманструе інкрэмэнт — прырашчэнне — працоўнага ПД зацікаўленым бакам
- Адбываецца збор адпаведнай сувязі для праўкі прыярытэтаў у бэклогу прадукту

## **6. Рэтраспектыва спрынту**

- Каманда разважае пра тое, што прайшло добра, што не, і як палепшыць сітуацыю ў наступным спрынце
- Увага — на паляпшэнні працэсаў, інструментаў, камунікацыі і Г.Д.

## **Асноўныя артэфакты Скрам**

Асноўныя артэфакты Скрам:

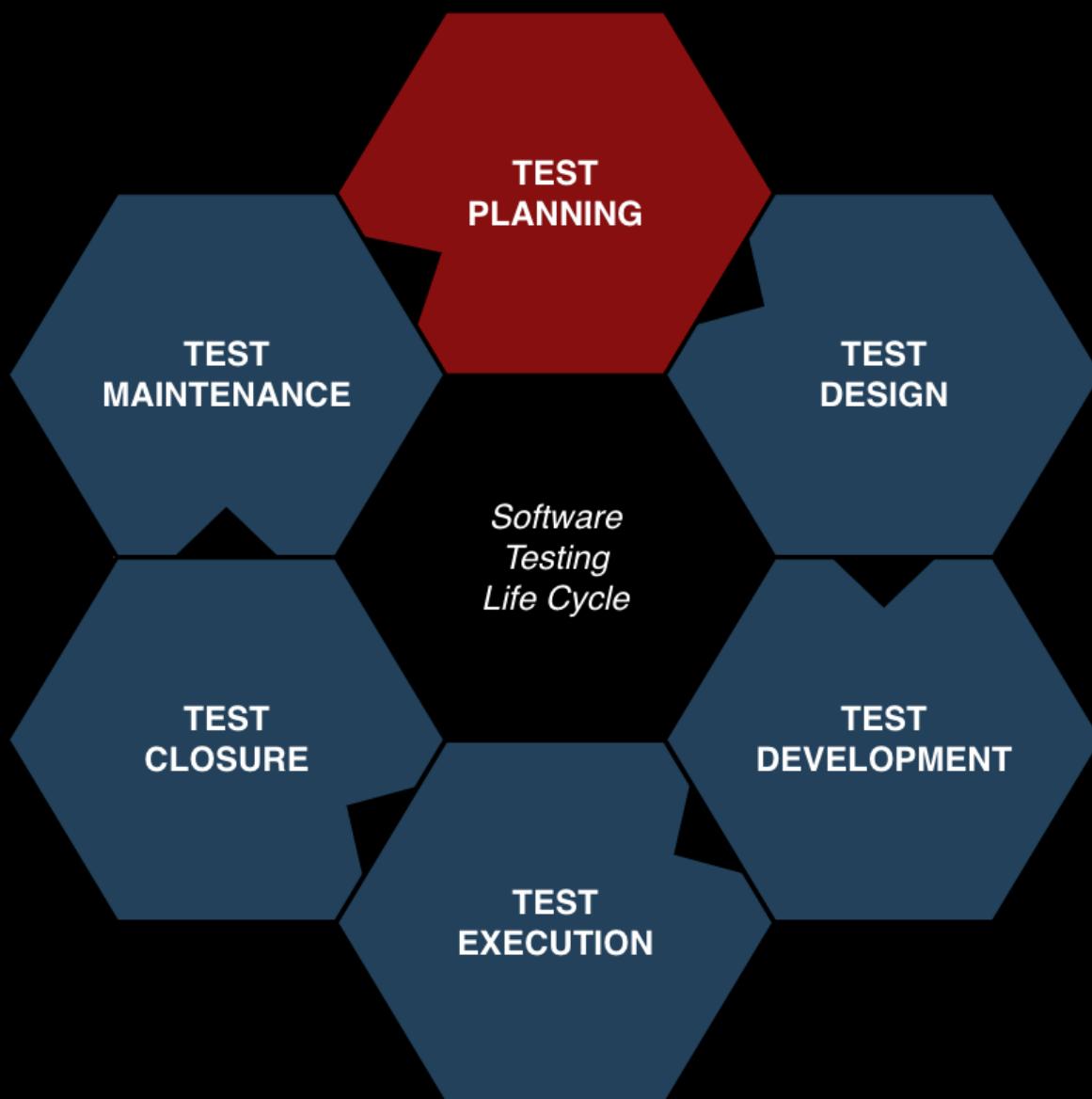
- Спіс пажаданняў да ПД — дынамічны спіс пажаданняў з усімі патрэбнымі функцыямі
- Бэклог спрынту — падмножства элементаў бэклога, выкананых у спрынце
- Інкрэмэнт — версія прадукту, даступная для усталёўкі пасля кожнага спрынту

## **VII. ЖЫЦЦЁВЫ ЦЫКЛ ТЭСЦІРАВАННЯ ПД**

## 7. Жыццёвы цыкл

### тэсціравання ПД

**Жыццёвы цыкл тэсціравання ПД** — ЖЦТ ПД — гэта паслядоўнасць дзеянняў па праверцы і валідацыі, якія праводзяцца падчас ЖЦР ПД дзеля дасягнення мэтаў якасці дачынення.



## **Этапы ЖЦТ**

Жыццёвы цыкл тэсціравання праграмнага дачынення складаецца з наступных метадалагічных этапаў сертыфікацыі праграмнага прадукту:

1. Планаванне тэставання
2. Праектаванне тэстаў
3. Распрацоўка тэстаў
4. Выкананне тэстаў
5. Закрыццё тэставання
6. Падтрымка тэстаў

Кожны з гэтых этапаў мае пэўныя крытэрыі ўваходу і выхаду, мерапрыемствы і звязаныя з ім вынікі.

---

## 7.1. Этап планавання тэставання

**Планаванне тэставання** — гэта этап вызначэння мерапрыемстваў і рэсурсаў, неабходных для дасягнення мэтаў тэсціравання.

Планаванне тэставання звычайна выконваецца ў адпаведнасці са спецыфікацыяй патрабаванняў да праграмнага дачынення, стратэгіяй тэставання і аналізам рызык.

### Ключавая дзейнасць

Дзейнасць па планаванні тэставання звычайна мае намерам

- Вызначыць тыпы тэстаў, якія трэба стварыць
- Сабраць падрабязную інфармацыю аб прыярытэтах тэсціравання
- Праверыць меркаванае тэставае асяроддзе
- Падрыхтаваць матрыцу адсочвання патрабаванняў
- Вызначыць паказчыкі тэставання і спосабы назірання за імі
- Ацаніць магчымасць аўтаматызацыі тэсціравання

### План тэставання

**План тэставання** — гэта афіцыйны документ, які апісвае аб'ём, падыход, рэсурсы і графік запланаваных мерапрыемстваў па тэсціраванні.

Ён служыць планам працэсу тэсціравання, вызначаючы што, як, хто і калі будзе тэсціраваць ПД.

Асноўныя кампаненты плана тэсціравання:

- Мэты тэсціравання

- Аб'ём тэставання
- Рызыкі тэсціравання
- Тэставае пакрыццё
- Неабходныя рэсурсы
- Ролі ў камандзе
- Інструменты тэсціравання
- Графік тэсціравання
- Вынікі тэставання

## Графік тэсціравання

**Графік тэсціравання** — гэта падрабязны графік, які паказвае, калі будзе праводзіцца кожная тэставая дзейнасць, колькі часу яна зойме, якія рэсурсы патрэбныя і паслядоўнасць тэставых задач.

## Ключавыя вынікі

Асноўнымі вынікамі этапу планавання тэставання з'яўляюцца:

- План тэставання
  - Графік тэсціравання
  - Матрыца адсочвання патрабаванняў
  - Тэхніка-эканамічнае аргументаванне аўтаматызацыі
-

## 7.2. Этап практавання тэстаў

**Практаванне тэстаў** — гэта этап стварэння, выпраўлення і абнаўлення спісаў праверкі і тэставых выпадкаў у адпаведнасці з планам тэставання.

Этап практавання тэстаў можа і павінен пачынацца раней, чым сам працэс распрацоўкі праграмнага дачынення.

Гэты этап забяспечвае сістэматычнае і эфектыўнае тэсціраванне, вызначаючы што тэставаць, як тэставаць і якія даныя выкарыстоўваць.

### Ключавая дзейнасць

Этап практавання тэстаў прызначаны стварыць:

- Спісы праверкі (чэк-лісты), заснаваныя на адвольным тэставанні
- Тэставыя выпадкі (тэст-кейсы) на аснове пашыраных спісаў праверкі
- Тэставыя скрыпты — для аўтаматызацыі тэст-кейсаў
- Тэставыя даныя, неабходныя для вышэйзгаданых артэфактаў

### Ключавыя вынікі

Асноўныя вынікі этапу практавання тэстаў:

- Чэк-лісты
- Тэст-кейсы
- Тэст-скрыпты
- Тэставыя даныя

## 7.3. Этап распрацоўкі тэстаў

**Распрацоўка тэстаў** — этап, на якім каманда распрацоўвае, стварае і рыхтуе ўсе неабходныя артэфакты для выканання тэстаў.

Артэфакты — тэст-кейсы, тэст-сценарыі, тэставыя даныя — адсочваюцца на этапе выканання тэсту, каб гарантаваць, што ПД паводзіць сябе належным чынам.

### Асноўныя мэты

Асноўныя мэты этапу распрацоўкі тэстаў:

- Пакрыццё тэстамі — упэўніцца ў праверцы ўсіх патрабаванняў
- Шматразовыя тэст-кейсы — для бягучага і рэгрес-тэставання
- Падрыхтоўка тэст-даных — уваходных даных, баз даных, API
- Аўтаматызацыя тэставых сценарыяў — там, дзе гэта магчыма
- Адсочвальнасць — сувязь тэст-кейсаў з патрабаваннямі

Этап аб'ядноўвае этапы праектавання і выканання тэстаў, забяспечваючы структураваную, паўтаральную і эфектыўную праверку праграмнага дачынення.

Інвестуючы ў добра распрацаваныя тэст-кейсы, тэставыя даныя і аўтаматызацыю тэсціравання, каманды могуць:

- Знізіць колькасць дэфектаў у вытворчасці
- Паскорыць рэгресійнае тэсціраванне
- Палепшыць адпаведнасць аўдыту

### Ключавая дзейнасць

На гэтым этапе каманда тэсціравання засяроджваецца на наступных асноўных відах дзейнасці:

- Распрацоўка тэст-кейсаў — стварэнне падрабязных крокавых працэдур для праверкі канкрэтных патрабаванняў

- Падрыхтоўка тэставых даных — стварэнне і кіраванне данымі, неабходнымі для выканання тэст-кейсаў
- Распрацоўка тэставых сцэнарыяў — аўтаматызаваных тэстаў для рэгрэсійнага тэсціравання або тэсціравання якаснасці
- Наладаванне тэставага наваколля — падрыхтоўка і налада апаратнага і праграмнага асяроддзя для тэсціравання

## Ключавыя вынікі

Асноўныя вынікі этапу распрацоўкі тэстаў:

- Тэст-кейсы — пакрокавыя працэдуры праверкі
- Тэставыя даныя — уваходныя даныя, базы даных і налады асяроддзя
- Тэст-скрыпты — код для аўтаматызаванага выканання тэстаў
- Матрыца адсочвання — сродак праверкі ўсіх патрабаванняў
- Кароткі змест тэставага набору — агляд пакрыцця тэстамі

The screenshot shows a Google Sheets document titled "Requirements Traceability Matrix". The spreadsheet has a green header row labeled "REQUIREMENTS TRACEABILITY MATRIX". Column A contains project metadata: Project Name, Project Type, Project Start Date, Project End Date, Project Sponsor, and Project Manager/Department. Columns B through E represent the traceability matrix itself. Row 8 defines the columns: ID, Technical Assumption(s) and/or Customer Need(s), Functional Requirement, Status, and Architectural/Design Document. Rows 9 through 15 list specific requirements, their descriptions, and their status relative to the defined columns.

REQUIREMENTS TRACEABILITY MATRIX				
Project Name				
Project Type				
Project Start Date				
Project End Date				
Project Sponsor				
Project Manager/Department				
ID	Technical Assumption(s) and/or Customer Need(s)	Functional Requirement	Status	Architectural/Design Document
001	The system must support multiple user logins simultaneously	The system shall allow users to log in using unique credentials	Approved	User Authentication Design Doc
002	Customer needs the application to be mobile-friendly	The system shall be responsive and support mobile devices	In Progress	UI/UX Design Guidelines
003	Data integrity is crucial for transaction processes	The system shall ensure all transactions are atomic	Approved	Transaction Processing Design
004				
005				
006				
007				

Матрыца адсочвання патрабаванняў

## 7.4. ЭТАП ВЫКАНАННЯ ТЭСТАЎ

**Выкананне тэстаў** — гэта этап, на якім каманда кантролю якасці запускае распрацаваныя тэст-кейсы і тэставыя сцэнарыі на рэальных зборках ПД у адпаведнасці з планам тэставання.

### Ключавая дзеянасць

Выкананне тэстаў — гэта "практычны" этап, на якім праграмнае дачыненне актыўна правяраецца на адпаведнасць патрабаванням і звычайна ўключае ў сябе:

- Разгортванне і налада асяроддзя
- Выкананне тэст-кейсаў і тэставых сцэнарыяў
- Карэктроўка тэст-кейсаў і тэставых сцэнарыяў
- Паведамленне аб дэфектах
- Паўторнае тэставанне дэфектаў

Пасля выпраўлення памылак каманда распрацоўшчыкаў стварае новую зборку для каманды па спрыянні якасці для паўторнага тэставання праграмнага дачынення.

### Ключавыя вынікі

Асноўныя вынікі выканання тэстаў:

- Матрыца адсочвання патрабаванняў, дзе адлюстроўваюцца:
  - Памылкі
  - Тэст-кейсы
  - Тэст-скрыпты
- Тэст-кейсы, адаптаваныя да бягучага стану дачынення
- Тэст-скрыпты, абноўленыя да апошняй версіі ПД
- Справаздачы аб дэфектах

## 7.5. ЭТАП ЗАКРЫЦЦЯ ТЭСТАВАННЯ

**Закрыцце тэставання** — гэта этап, на якім выкананне тэстаў бягучага цыклу распрацоўкі фармальна завяршаецца.

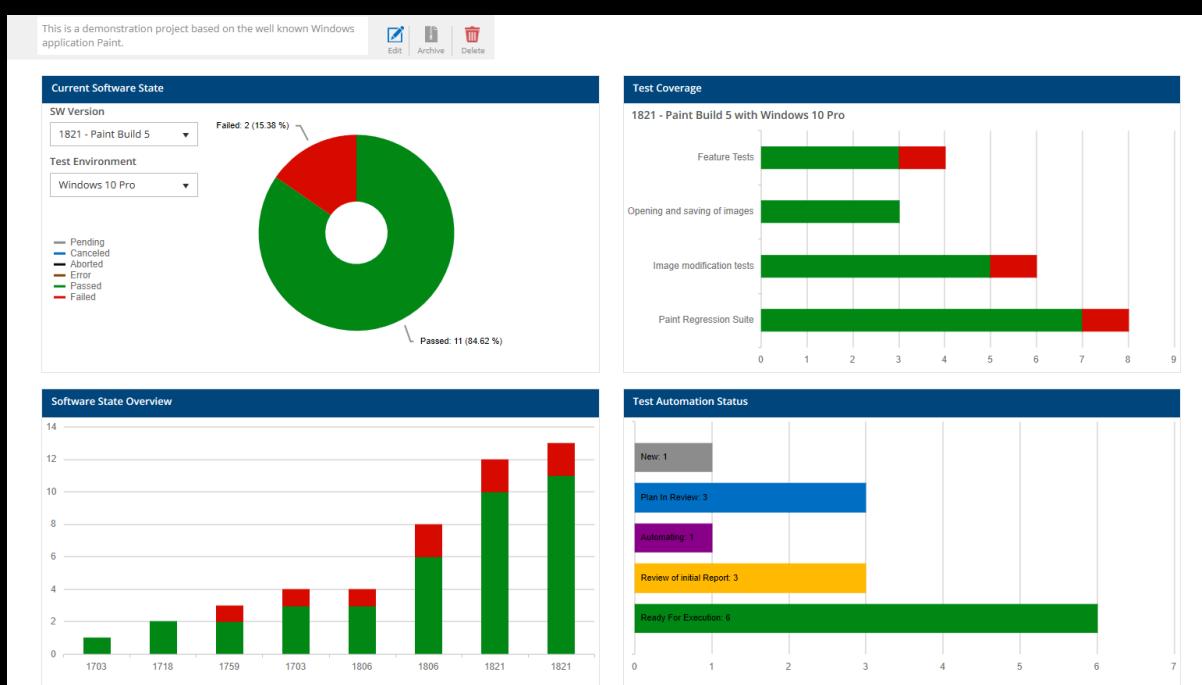
### Ключавыя мэты

Асноўная мэта завяршэння тэставання — уважыць і ацаніць агульную эфектыўнасць і якаснасць этапу выканання тэстаў.

Закрыцце тэставання дазваляе таксама задокументаваць вынікі выкананых тэстаў, збіраючы іх у справаздачы аб выніках тэставання і ахінуць тэст-артэфакты для далейшага ўжыцця.

### Справаздача аб выніках тэставання

**Справаздача аб выніках тэставання** — або **Справаздача аб закрыцце тэставання** — гэта асноўны документ этапу, які змяшчае інфармацыю аб выяўленых і развязаных проблемах.



Справаздача аб выніках тэставання

Справаздача аб выніках тэставання звычайна ўключае:

- Зводныя вынікі тэставання
- Падрабязны аналіз памылак
- Прэзентацыю метрык

Справаздача аб выніках тэставання сігналізуе аб завяршенні тэставай дзейнасці і інфармуе зацікаўленых бакоў аб заканчэнні этапу тэставання.

## Ключавыя вынікі

Вынікі закрыцца тэставання звычайна ўключаюць:

- Справаздачу аб стане тэставання
- Справаздачу аб выніках тэставання
- Тэставыя метрыкі

Карацей кажучы, закрыцце тэставання гарантуе, што мэты тэставання дасягнуты, памылкі задокументаваны, а этап тэставання паспяхова завершаны.

---

## 7.6. Падтрымка тэстаў

**Падтрымка тэстаў** — гэта няспынны этап, на якім тэставыя рэсурсы — тэст-кейсы, сцэнарыі, даныя і асяроддзі — абнаўляюцца, аптымізуюцца і выдаляюцца, каб ісці ўсцяж з развіццём ПД.

У адрозненне ад аднаразовага выканання тэстаў, падтрымка тэстаў забяспечвае доўгатэрміновую актуальнасць, эфектыўнасць і дакладнасць працэсаў тэставання па меры змянення ПД.

### Ключавыя мэты

Асноўныя мэты этапу тэхнічнай падтрымкі тэстаў ўключаюць:

- Падтрыманне слушнасці тэстаў — гарантуе, што як ручныя тэст-кейсы, так і аўтаматызаваныя тэст-сцэнарыі застаюцца актуальнымі і эфектыўнымі па меры развіцця праграмы.
- Бесперапыннасць сістэмы аўтаматызацыі — кампаненты сістэмы аўтаматызацыі павінны быць узгоднены са зменамі ў інструментах або бібліятэках іншых вытворцаў.
- Рэгрэсійнае тэставанне — гарантуе, што змены ў кодзе не парушаюць існуючу функцыянальнасць праз набор аўта-тэстаў, які звычайна змяшчае іх вялікую колькасць, што патрабуе пастаяннае абслугоўванне для адпаведнай праверкі ПД.
- Бесперапынная інтэграцыя — аўта-тэсты выконваюцца праз канвееры бесперапыннай інтэграцыі для хуткага выяўлення і вырашэння праблем; спрыянне абнаўленню тэстаў мае вырашальнае значэнне ў гэтым кантэксце, бо зборкі не ўважаюцца завершанымі, покуль не прайдуць тэсты.
- Справаздачнасць — рэгулярная справаздачнасць дапамагае інжынерам па спрыянні якасці выяўляць пашкоджаныя тэсты, якія патрабуюцца абнаўлення.

## Ключавая дзейнасць

- Агляд і абнаўленні тэстовых артэфактаў:
  - Тэст-кейсы — дабаўленне або выдаленне кроکаў у залежнасці ад змен у функцыях і прыярытэтах
  - Тэстывыя даныя — падтрымка адпаведнасці са зменамі ў вытворчасці і аナンімізацыя канфідэнцыйных даных
- Аўтаматызаванае аблугоўванне тэстовых сценарыяў:
  - Выпраўленне няспраўных лакатараў пасля правак
  - Рэфактарынг скрыптоў
  - Абнаўленне бібліятэк, фрэймворкаў і залежнасцей
- Аптымізацыя тэставага набору:
  - Выдаленне лішніх — Аб'яднанне дублікатаў тэстаў
  - Паширэнне пакрыцця — дабаўленне новых тэстаў
  - Класіфікацыя тэстаў — на дымавыя, рэгрэсійныя і г.д. — для лепшага кіравання іх выкананнем
- Абнаўленні матрыцы адсочвання:
  - Актуалізацыя тэст-кейсаў да сапраўдных патрабаванняў
  - Вылучыць прabelы, дзе патрэбныя новыя тэсты

## Ключавыя вынікі

Асноўныя вынікі этапу тэсціравання — гэта абноўленыя:

- Тэст-кейсы — узгодненыя з новымі функцыямі ПД
- Тэстывыя скрыпты — з палепшанай стабільнасцю і якаснасцю
- Аўдыт тэстаў — з выяўленнем даданых або выдаленых тэстаў
- Лог хісткіх тэстаў — з данымі па хісткіх тэстах і іх выпраўленні

# **VIII. Тэстовая документация**

## 8. Тэставая дакументацыя

**Тэставая дакumentацыя** — гэта набор тэставых дакументаў, якія ўжываюцца для спрыяння якасці ПД.

### Ахопы тэставай дакumentацыі

Ахопы тэставай дакumentацыі можна прадставіць наступным чынам.



## **Катэгорыі дакументацыі**

Тэставую дакumentaцыю можна падзяліць на троі асноўныя катэгорыі:

- Дакumentaцыя па выкананні тэстаў
- Скрыпты і даныя для аўтаматызацыі тэсціравання
- Дакumentaцыя па планаванні тэставання

### **Дакumentaцыя па выкананні тэстаў**

Асноўныя дакumentы па выкананні тэстаў:

- Чэк-ліст
- Тэст-кейс — або тэставы сценар
- Набор тэстаў

Тэст-кейс з'яўляецца найважнейшым тэставым дакumentам.

### **Дакumentы па аўтаматызацыі тэсціравання**

Асноўныя дакumentы па аўтаматызацыі тэсціравання ўключаюць:

- Тэставы скрыпт
- Тэставыя даныя

У аўтаматызацыі тэставыя даныя почасту і ствараюцца аўтаматычна.

Нягледзячы на тое, што тэставыя даныя ўжываюцца і у ручным тэсціраванні, у аўтаматызацыі яны адыгрываюць асаблівую ролю і значэнне.

## **Документацыя па палітыцы тэсціравання**

Асноўная документацыя па планаванні тэсціравання  
наступная:

- План тэставання
- Тэставая стратэгія
- Палітыка тэставання

План тэставання і стратэгія тэставання могуць быць асобнымі  
документамі або адным; выбар залежыць ад распрацоўшчыка  
документацыі і значнасці праграмнага дачынення.

Калі праект большы па памеры, мае сэнс мець розныя  
документы — інакш іх можна аб'яднаць у адзін.

---

## **8.1. Чэк-ліст**

**Чэк-ліст** — гэта кароткі спіс праверак, якія неабходна выкананць падчас этапу тэсціравання.

Чэк-ліст дапамагае падзяліць функцыянальнасць ПД на асобныя блокі тэставання, а таксама дазваляе ўбачыць аб'ёмы тэставання і колькасць праверак з адмоўным вынікам.

Парадак тэстаў чэк-ліста можа быць выпадковым, бо ён не мае значэння.

Чэк-лісты звычайна ўжываюцца на пачатковых этапах праекта, калі тэст-кейсы яшчэ толькі плануюцца для распрацоўкі.

## Переваги

Асноўныя перавагі чэк-лістоў наступныя:

- Гнуткасць — можна прымяняць ва ўсіх тыпах тэсціравання
  - Простасць — лёгка ствараць і падтрымліваць

- Хуткасць — хутка распрацаваць і зразумець
- Сцісласць — трэба запоўніць толькі некалькі палёў
- Аналізуемасць вынікаў — лёгка выкананаць і даследваць
- Адкрытасць — спрашчаюць далучэнне новых членаў каманды
- Кантроль тэрмінаў — дазваляюць адсачыць выкананне тэстаў

## Недахопы

Асноўныя недахопы ўжыцця чэк-лістоў у тэсціраванні:

- Розная інтэрпрэтацыя — інжынеры могуць выконваць аднолькавыя праверкі, ужываючы розныя падыходы
  - Прабелы ў ахопе — цяжка ахапіць усе функцыянальныя або структурныя кампаненты, асабліва — вышэйшага ўзроўню
  - Паўтор задач — спроба ахапіць вялікі аб'ём правераў можа прывесці да лішняга і, як вынік, да празмернага тэставання
  - Складанасць апісання — чэк-лістамі цяжка апісаць складаныя кампаненты сістэмы, функцыі і іх взаємадзеянне
-

## 8.2. Тэст-кейс або тэставы

### Сцэнар

**Тэст-кейс** — ці інакш **Тэставы сцэнар** — гэта паслядоўнасць кроکаў, прызначаных для праверкі чаканых паводзін ПД.

Кардынальнае адрозненне паміж чэк-лістом і тэст-кейсам заключаецца ў полі "Чаканы вынік", якое уласціва для тэст-кейсаў.

Test Case Template					
Test Case Header					
1. Test Case Name					
2. Module Name					
3. Requirement No.					
4. Test Data					
5. Severity					
6. Precondition					
7. Test Case Type					
8. Brief Description					
Test Case Body					
Step No.	Action	Input	Expected Result	Status	Comment
Test Case Footer					
1. Author Name					
2. Received By					
3. Approved By					
4. Approved Date					

Тэст-кейс — гэта асноўны тэставы документ, і ён часта з'яўляеца сінонімам самога тэрміна "тэст".

### Тыпы тэст-кейсаў

Існуе два тыпы тэст-кейсаў:

- Нефармальны
- Фармальны

**Нефармальны тэст-кейс** — гэта тэст-кейс, які ўжваеца для праверкі ПД, якое не мае фармальных патрабаванняў і заснавана

на прынятым звычайным рэжыме працы праграмнага дачынення падобнага класа.

**Фармальны тэст-кейс** — гэта тэст-кейс, які характерызуецца вядомымі уваходнымі і чаканымі выходнымі данымі, вызначанымі перад выкананнем тэсту.

## Абавязковыя палі

Фармат фармальнага тэст-кейсу можа ўключачь розныя параметры, але абавязковыя палі наступныя:

- Ідэнтыфікатар
- Апісанне
- Крокі
- Чаканы вынік
- Статус
- Прыярытэт

У самым кароткім фармаце тэст-кейсу поле "Прыярытэт" можа адсутнічаць.

## Матрыца адсочвання патрабаванняў

Каб цалкам пераканацца ў тым, што ўсе патрабаванні да праграмнага дачынення выкананы, для кожнага патрабавання павінна быць як мінімум два тэст-кейсы:

- Стanoўчи
- Адмоўны

Калі патрабаванне мае пад-патрабаванні, то і кожнае пад-патрабаванне павінна мець як мінімум два тэст-кейсы.

Адсочванне сувязі паміж патрабаваннем і тэстам часта ажыццяўляецца з дапамогай матрыцы адсочвання патрабаванняў.

## **Фармальная структура тэст-кейсу**

Пісьмовыя тэст-кейсы павінны ўключачь апісанне тэставанай функцыянальнасці і крокі падрыхтоўкі, неабходнай для дасягнення магчымасці правядзення тэсту.

Тыповы фармат пісьмовага тэст-кейсу можа ўключачь:

- ID — Унікальны ідэнтыфікатор
- Аўтар — Імя распрацоўшчыка
- Катэгорыя тэсту — група, да якой належыць тэст-кейс
- Апісанне — або рэзюмэ — Мэта тэст-кейсу
- Папярэдня ўмовы — або папярэдня патрабаванні — умовы, якія павінны быць выкананы перад выкананнем тэст-кейсу
- Тэставыя даныя — зменныя і іх значэнні ў тэст-кейсе
- Крокі — Дзеянні, якія неабходна выкананы
- Чаканы вынік — чаканы вынік выкананага кроку
- Пасляумовы — вынік выканання кроку
- Фактычны вынік — вынік, атрыманы пасля выканання кроку
- Статус — вынік параўнання чаканых і фактычных вынікаў — пройдзена/не пройдзена
- Прыярытэт — важнасць тэст-кейсу для рэгрэсіі
- Аўтаматызацыя — адзнака таго, ці належыць тэст-кейс аўтаматызацыі
- Аўтаматызаваны — адзнака таго, ці тэст-кейс аўтаматызаваны
- Заўвагі — набор заўваг, звязаных з этапам тэставання

Большасць вышэйпералічаных палёў неабавязковыя і могуць быць прапушчаны ў шаблоне тэст-кейсу.

## **Тэставы сцэнар**

**Тэставы сцэнар** — гэта пакрокавыя дзеі канечнага спажыўца, задокументаваныя для праверкі пэўнай функцыянальнасці.

Калі тэст-кейс лепш падыходзіць для модульнага тэставання, то тэставы сцэнар лепш пасуе функцыянальнаму тэставанню "ад краю да краю".

Тэст-кейсы звычайна з'яўляюцца кароткімі праверкамі, у той час як тэставыя сцэнарыі ахопліваюць значную колькасць кроаку праверкі чаканага выніку і лепш падыходзяць для аўтаматызацыі.

TESTING SCENARIO		
Test Scenario	Test Case 1	
	Test the login functionality of the e-commerce site to make sure that registered user is allowed to login into site using valid credentials	
Test Case 1	Make sure that site under test is available and testable	
Test Step 1	Launch the ecommerce application with the given URL	<input checked="" type="checkbox"/> <input type="checkbox"/>
Test Step 2	Navigate to Login page	<input checked="" type="checkbox"/> <input type="checkbox"/>
Test Step 3	Enter valid Username and password in required field	<input checked="" type="checkbox"/> <input type="checkbox"/>
Test Step 4	Click on login button	<input checked="" type="checkbox"/> <input type="checkbox"/>
Test Case 2	Make sure that site under test is available and testable	
Test Case 2	Launch the ecommerce application with the given URL	
Test Step 1	Launch the ecommerce application with the given URL	<input checked="" type="checkbox"/> <input type="checkbox"/>
Test Step 2	Navigate to Login page	<input type="checkbox"/> <input checked="" type="checkbox"/>
Test Step 3	Enter valid Username and password in required field	<input type="checkbox"/> <input checked="" type="checkbox"/>
Test Step 4	Click on login button	<input type="checkbox"/> <input checked="" type="checkbox"/>

Мэта тэставага сцэнарыя — прасачыць усе вынікі кроаку для праверкі канкрэтнай складанай проблемы або проблемнага выпадку ўжывання.

## 8.3. Тэст-скрыпт і тэставыя даныя

**Тэст-скрыпт** — гэта праграма, прызначаная для аўтаматызацыі этапаў тэставання і праверак.

Кожны тэставы скрыпт звычайна асацыюеца з тэст-кейсам.

```
normalizeHtml.spec.js
1 import normalizeHtml, {normalizeHtmlAttribute} from "./normalizeHtml"
2
3 describe("normalizeHtml", function(){
4     it("Converts & to &amp'", function(){
5         expect(normalizeHtml("&")).toBe("&amp;")
6     })
7     it("Converts &quo; to »", function(){
8         expect(normalizeHtml("&quo;")).toBe("»")
9     })
10 })
11
12 describe("normalizeHtmlAttribute", function(){
13     it("Converts & to &amp'", function(){
14         expect(normalizeHtmlAttribute("&")).toBe("&amp;")
15     })
16     it("Converts &quo; to &amp;quo;", function(){
17         expect(normalizeHtmlAttribute("&quo;")).toBe("&amp;quo;")
18     })
19     it("Converts quote signs to &quot", function(){
20         expect(normalizeHtmlAttribute('\"')).toBe("&quot;")
21     })
22 })
23 }
```

Пасля падрыхтоўкі тэст-скрыпту звычайна замяняе звязаны з ім тэст-кейс падчас ЖЦТ ПД.

Тэст-скрыпт звычайна ствараецца для праверкі часткі функцыянальнасці праграмнай сістэмы.

Тэст-скрыпты можна ствараць на мовах праграмавання агульнага прызначэння, спецыяльных мовах для тэст-скрыптоў або інструментамі запісу тэстаў з графічным інтэрфейсам спажыўца.

Тэст-скрыпты незаменныя ва ўмовах, немажлівых для паўтарэння чалавекам, напрыклад, пры нагрузкочным тэставанні.

## Кампаненты тэст-скрыпта

Асноўныя кампаненты тэст-скрыпта наступныя:

- Этапы тэставання — інструкцыі для кожнага тэставага руху
- Чаканыя вынікі — паводзіны, якія чакаюцца ад ПД
- Тэставыя даныя — уваходныя значэнні, неабходныя для тэсту
- Праверкі — умовы, якія павінны выкананцацца ў выніку скрыпта

## Распрацоўка тэставых сцэнарыяў

Працэс распрацоўкі тэст-сцэнарыя звычайна ўключае ў сябе:

- Аналіз тэставых выпадкаў — разуменне патрабаванняў да ПД
- Распрацоўка тэст-кейсаў — падрабязная пабудова тэст-кейсаў
- Пабудова тэст-скрыпта — пераклад тэст-кейсу ў тэст-скрыпт
- Падрыхтоўка тэставых даных — генерацыя належных даных
- Выкананне тэставага скрыпта — Запуск тэставага скрыпта
- Захоўванне тэст-скрыптоў — Захоўванне кода ў рэпозіторыі

## Перавагі

Аўта-тэсты маюць перевагі па некалькіх прычынах:

- Тэставыя скрыпты могуць выконвацца бесперапынна
- Тэставым скрыптам не патрэбны чалавек
- Тэставыя скрыпты лёгка паўтараць
- Тэставыя скрыпты значна хутчэйшыя

## Недахопы

Тэставы скрыпт як ПД для тэставання ПД:

- Можа ўтрымліваць свае ўласныя недахопы
- Патрабуе больш высокі ўзровень экспертызы
- Можа даследаваць толькі тое, на што запраграмаваны

## Тэставыя даныя

**Тэставыя даныя** — даныя, прызначаныя для ўжытку ў тэстах.

Тэставыя даныя могуць быць атрыманы самім тэсціроўшчыкам або праграмай, якая дапамагае яму.

title	release_year	length	replacement_cost
West Lion	2006	159	29.99
Virgin Daisy	2006	179	29.99
Uncut Suicides	2006	172	29.99
Tracy Cider	2006	142	29.99
Song Hedwig	2006	165	29.99
Slacker Liaisons	2006	179	29.99
Sassy Packer	2006	154	29.99
River Outlaw	2006	149	29.99
Right Cranes	2006	153	29.99
Quest Mussolini	2006	177	29.99
Poseidon Forever	2006	159	29.99
Loathing Legally	2006	140	29.99
Lawless Vision	2006	181	29.99
Jingle Sagebrush	2006	124	29.99
Jericho Mulan	2006	171	29.99
Japanese Run	2006	135	29.99
Gilmore Boiled	2006	163	29.99
Floats Garden	2006	145	29.99
Fantasia Park	2006	131	29.99
Extraordinary Conquerer	2006	122	29.99
Everyone Craft	2006	163	29.99
Dirty Ace	2006	147	29.99
Clyde Theory	2006	139	29.99
Clockwork Paradise	2006	143	29.99
Ballroom Mockingbird	2006	173	29.99

Тэставыя даныя могуць быць запісаны для паўторнага ўжыцця або спажыты адзін раз, а потым забытыя.

Тэставыя даныя можна падзяліць на два наступныя тыпы:

- Сінтэтычныя даныя — інакш вядомыя як фэйкавыя даныя
- Рэальныя даныя — інакш вядомыя як сапраўдныя даныя

**Сінтэтычныя тэставыя даныя** ствараюцца альбо ўручную, альбо з дапамогай інструментаў генерацыі даных.

**Рэальныя тэставыя даныя** звычайна бяруцца з кліентскага асяроддзя, а затым ананімізуюцца.

---

## 8.4. Тэставы набор

**Набор тэстаў** — гэта група тэст-кейсаў (тэставых сцэнарыяў) або тэст-скрыптоў для выканання ў пэўным тэставым запуску.

Набор тэстаў часта змяшчае падрабязныя інструкцыі або мэты для кожнага набору тэставых элементаў і інфармацыю аб канфігурацыі сістэмы, якая будзе ўжывацца падчас запуску.

The screenshot shows the Aqua Test Management software interface. The top navigation bar includes 'New', 'Open', 'Refresh', 'Collapse all projects', 'Show views', 'Details' (selected), 'Dependency', 'History', 'Filter items', 'Show filter row', 'Select columns', 'Reset to default', '(Un)Archive items', and a search bar. The left sidebar features 'Favorites', 'Projects' (ShopDemo selected), and sections for 'Ordering', 'Products', and 'Sign-In'. The main grid displays a list of requirements with columns for ID, Name, Requirement coverage (e.g., TS1118219, TS1118219), Covered by Test Cases (e.g., Not covered, Covered), Status (Ready, QA, In Progress), Creation date, and Last modified. Below the grid is a detailed view of requirement TC4544343, showing its relationships to other requirements and test cases, and its status (Passed, Not run, Not complete, Not applicable).

ID	Name	Requirement coverage	Covered by Test Cases	Status	Creation date	Last modified
RQ074435	Create new account	TS1118219, TS1118219	Not covered	Ready	12.04.2022	26.07.2022
RQ054634	Sign up via Google	TS1118219	Covered	QA	12.04.2022	26.07.2022
RQ052439	Update eCoffe profile	TS1118219	Covered	Ready	12.04.2022	22.07.2022
RQ054030	Sign up with credentials	TS1118219, TS1118219	Not covered	In Progress	12.04.2022	22.07.2022
RQ052431	Change the username	TS1118219, TS1118219	Not covered	Ready	12.04.2022	22.07.2022
RQ051430	News block on the website	TS1118219	Covered	Ready	12.04.2022	17.07.2022
RQ053433	Create new requirements	TS1118219	Covered	QA	12.04.2022	16.07.2022
RQ051432	Option to mark user profile	TS1118219, TS1118219	Covered	Ready	12.04.2022	14.07.2022

Ён таксама можа ўтрымліваць папярэдне неабходныя стан або крокі, а таксама апісанні наступных тэстаў.

### Ключавыя мэты

Набор тэстаў — гэта кантэйнер з наборам тэстаў, які памагае тэсціроўшчыкам выкананць іх і паведаміць статус выканання.

У наборы тэстаў тэст-кейсы або тэст-скрыпты арганізованы ў лагічным парадку — напрыклад, тэст-кейс для рэгістрацыі будзе папярэднічаць тэст-кейсу для ўваходу ў сістэму.

Тэставы набор можа прымачь адзін з наступных станаў:

- Актыўны
- Выконваецца
- Завершаны

## Тыпы набору тэстаў

Набор тэстаў служыць для групавання падобных тэст-кейсаў.

Наборы тэстаў часта дзеляцца на наступныя тыпы:

- **Набор для дымавога тэставання** — Калекцыя тэст-кейсаў для базавай праверкі большасці функцыянальных абласцей і запускаецца пасля кожнай зборкі ПД перад тым, як зборка будзе прапанавана для ўжытку больш шырокай аудыторыяй
- **Набор для тэставання цэласнасці ПД** — Калекцыя тэст-кейсаў для валідацыі базавага функцыяналу ПД і першы ўзровень праверкі ПД пасля ўнясення ў яго змен
- **Набор тэстаў крытычнага шляху** — Калекцыя тэст-кейсаў, якія выконваюцца паміж кампанентамі ПД і правяраюць надзейнасць кропак інтэграцыі паміж модулямі
- **Набор тэстаў для праверкі функцыяналу** — Калекцыя тэст-кейсаў для праверкі асобнай функцыі ПД, каб запэуніць, што асноўныя аспекты яе працы пратэставаны
- **Набор рэгресійных тэстаў** — Калекцыя тэст-кейсаў, якая служыць для рэгресійнага аналізу функцыянальных абласцей ПД і часта дадаецца ў некалькі набораў тэстаў і планаў тэставання

## **8.5. План тэставання**

**План тэставання** — гэта документ на ўзроўні ПД, які апісвае мэты, графік, ацэнку, вынікі і рэсурсы, неабходныя для тэставання дачынення.

### **Ключавыя мэты**

План тэставання служыць шаблонам правядзення праверкі ПД і вызначае працэс, які назіраецца і кантралюецца менеджарам па тэсціраванні.

План тэсціравання павінен адпавядаць документам кампаніі па палітыцы тэсціравання і стратэгіі тэсціравання.

### **Ключавыя кампаненты**

План тэставання звычайна павінен уключаць:

- Ідэнтыфікатар плана тэсціравання
- Спасылкі
- Уводзіны
- Тэставыя элементы
- Рызыкі развіцця ПД
- Функцыі, якія будуць пратэставаны
- Функцыі, якія не падлягаюць тэставанню
- Падыход да тэсціравання
- Крытэрыі паспяховасці/непаспяховасці тэсту
- Крытэрыі прыпынення і патрабаванні да аднаўлення
- Вынікі тэставання
- Налада тэставага наваколля
- Патрэбы ў персанале і яго навучанні
- Абавязкі членаў каманды
- Графік тэсціравання
- Планаванне рызык і непрадбачаных абставін

- Зацвярджэнні
- Гласарый

— і іншыя раздзелы.

Вышэйзгаданыя кампаненты плана тэсціравання не з'яўляюцца абавязковымі і могуць быць прапушчаны пры неабходнасці.

ProjectManager		Test Plan Template								
Test Plan Identifier:		123								
Introduction:		To define testing strategy for widget								
Objectives:		Ensure that all functionalities meet specified requirements								
Test Scope:		This test covers functionalities, but not system security								
Date: 10/12/24		Updated on: 10/12/24								
Test Items	Test Environment & Tools	Test Types	Test Strategy and Approach	Entry Criteria	Exit Criteria	Deliverables	Start Date	End Date	Assignee	KPIs
Started	Server	functional	unit testing	all functional requirements must be finalized and approved	at least 95% of the planned test cases must be executed	document listing all test cases	10.12.24	10.14.24	J. Smith	test case execution rate: executed test cases / total test cases x 100
In Progress	Client machines	functional	integration testing	test cases must be written, reviewed and approved	a minimum of 90% of executed test cases must pass	report summarizing outcomes of test case executions	10.12.24	10.15.24	J. Smith	pass rate: passed cases / executed test cases x 100
Test Plan Team										
Name	Role	Responsibilities								
J. Smith	QA	test functionality								

Часта кампаніі ствараюць свае ўласныя фарматы, наогул не заснаваныя на якім-небудзь стандарты.

## Перавагі

Стварэнне плана тэсціравання мае некалькі пераваг, бо ён:

- Дапамагае людзям па-за тэставай камандай — кліентам, распрацоўшчыкам і г.д. — зразумець дэталі тэсціравання
- Кіруе мысленне тэставай каманды на завершанасць працэсаў
- Документуе ключавыя аспекты — ацэнку, аб'ём і стратэгію тэставання — для перагляду і ўжывання ў іншых праектах

## **8.6. Стратэгія тэставання**

**Стратэгія тэставання** — гэта дакумент ўзроўню арганізацыі, які акрэслівае падыход, мэты і план выканання тэставання ПД.

Ён служыць планам, які гарантую адпаведнасць тэсціравання бізнес-мэтам, тэхнічным патрабаванням і стандартам якасці.

### **Ключавыя мэты**

Мэта стратэгіі тэставання:

- Вyzначыць аб'ём, фокус і метадалогіі тэсціравання
- Узгадніць намаганні па тэсціраванні з тэрмінамі праекта і бізнес-мэтамі
- Забяспечыць паслядоўнасць паміж камандамі
- Аптымізаваць размеркаванне рэурсаў — інструменты, бюджет, персанал
- Змякчыць рызыкі праз структураванне планавання

### **Ключавыя кампаненты**

Асноўныя раздзелы стратэгіі тэставання наступныя:

- Аб'ём і мэты — што будзе тэставацца і чаму
- Тыпы тэставання — функцыянальнае, нефункцыянальнае, рэгрэсійнае і г.д.
- Узроўні тэставання — модуль, інтэграцыя, сістэма і г.д.
- Тэставае асяроддзе — неабходнае абсталяванне, праграмнае дачыненне і інструменты
- Кіраванне тэставымі данымі — як будуць стварацца, захоўвацца і анатомізавацца тэставыя даныя
- Ролі і абавязкі — хто распрацоўвае, выконвае і зацвярджае тэсты
- Аналіз рызык — Вызначэнне зон высокай рызыкі і планаў па змякчэнні наступстваў

- Крытэрыі ўваходу і выхаду — умовы для пачатку і спынення тэсціравання
- Падыход да аўтаматызацыі — інструменты, пакрыццё і інтэграцыя CI/CD
- Кіраванне дэфектамі — як памылкі рэгіструюцца, прыярытызуюцца і паўторна правяраюцца
- Метрыкі і справаздачныя — ключавыя паказчыкі эфектыўнасці і частата справаздачныя
- Вынікі — планы тэставання, тэст-кейсы, справаздачы аб выніках тэставання і журналы аўдыту

## Роля стратэгіі тэставання

Добра акрэсленая тэставая стратэгія гарантуе, што тэставанне будзе структурованым, эфектыўным і мэтанакіраваным.

Яна ліквідуе разрыў паміж бізнес-мэтамі і тэхнічным выкананнем, зніжаючы выдаткі і максімізуючы якасць.

---

## **8.7. Палітыка тэставання**

**Палітыка тэставання** — гэта дакумент на ўзроўні кампаніі, які вызначае прынцыпы тэсціравання, прынятые арганізацыяй.

Палітыка тэставання вызначаецца генеральнымі дырэктарамі кампаніі, якія забяспечваюць арганізацыінае ўяўленне аб тэставай дзеянасці.

### **Ключавыя мэты**

Палітыка тэставання звычайна апісвае:

- Месца правядзення тэсціравання ў кампаніі
  - Мэты тэставання ў арганізацыі
  - Вызначэнне працэсу тэсціравання
  - Вымярэнне эфектыўнасці тэставання
  - Падыход да паляпшэння працэса тэсціравання
-

## **8.8. Сістэмы кіравання**

### **тэставаннем**

**Сістэма кіравання тэставаннем** — СКТ — гэта ПД, прызначанае для таго, каб дапамагчы камандам эфектыўна кіраваць працэсамі тэсціравання праграмнага дачынення.

СКТ можа ахопліваць усе ўзроўні тэставай документацыі і забяспечваць цэнтралізаванае месца для захоўвання і кіравання тэставымі выпадкамі, сцэнарыямі, патрабаваннямі і дэфектамі.

### **Ключавыя мэты**

Сістэмы кіравання тэставаннем адыгryваюць вырашальную ролю ў спрыянні якасці ПД, аўтаматызуючы наступныя працэсы:

- Адсочванне патрабаванняў
- Адсочванне задач праекта
- Кіраванне тэставымі выпадкамі
- Аўтаматызаванае выкананне скрыптоў
- Адсочванне дэфектаў
- Прадастаўленне справаздачнасці і паказчыкаў

### **Перавагі**

Асноўныя перавагі сістэм кіравання тэставаннем наступныя:

- Кансалідацыя працэсу тэсціравання
- Эфектыўны доступ да даных і іх аналіз
- Эфектыўная камунікацыя паміж камандамі

## **ІХ. Дэфекты якасці ПД**

## 9. Дэфекты якасці ПД

**Дэфект якасці ПД** — альбо **Памылка** ці проста **Баг** — гэта недахоп, пахібка або недасканаласць ПД, якая прыводзіць да яго нечаканых паводзін або неадпаведнасці пэўным патрабаванням.

Дэфекты ўяўляюць сабой адхіленні паміж фактычнымі і чаканымі патрабаваннямі, спецыфікацыямі або чаканнямі кліента, што прыводзіць ці да функцыянальных праблем — збояў, пахібных вынікаў і г.д. — ці да нефункцыянальных праблем — нізкай якаснасці, уразлівасцей бяспекі і г.д.

### Ключавыя характеристыстыкі

Дэфекты можна класіфікаваць, ужываючы наступныя ключавыя характеристыстыкі:

#### 1. Першапрычыны:

- Памылкі кадавання
- Недахопы праектавання або архітэктуры
- Няправільна зразумелыя патрабаванні
- Несумяшчальнасць з наваколлем

#### 2. Уздзеянні:

- Крытычныя — збоі сістэмы, страта даных
- Важныя — неналежнасць асноўных функцый
- Нязначныя — касметычныя праблемы

#### 3. Методы выяўлення:

- Тэсціраванне
- Агляды кода
- Водгукі спажыўцу

## Роля кіравання дэфектамі

Кіраванне дэфектамі адыгryвае вырашальную ролю па наступных асноўных прычынах:

- **Кошт** — выпраўленне дэфектаў пасля рэлізу ў шмат разоў даражэйшае, чым падчас распрацоўкі
- **Рэпутацыя** — звычайна кожны чацвёрты кліент адмаўляеца ад ПД пасля яго збою
- **Адпаведнасць** — дэфекты ПД у сферах аховы здароўя або фінансаў могуць весці да парушэння законаў і прыводзіць да штрафаў

## Найлепшыя практыкі

Найлепшыя практыкі памяншэння дэфектаў ПД павінны ўключачыць:

- **Тэставанне са зрухам ўлева** — раннєе выяўленне памылак праз як мага ранейшае ўкараненне модульных або інтэграцыйных тэстаў
- **Статычны аналіз** — рэалізацыя спецыяльных інструментаў праверкі якасці кода
- **Рэцэнзіі калег** — 60–90% прадухілення дэфектаў адбываецца дзякуючы рэгулярным праверкам кода
- **Аўтаматызаванае рэгрэсійнае тэставанне** — прадухіленне паўторнага ўзнікнення старых памылак шляхам аўтаматызацыі тэстовых выпадкаў для вядомых проблем

## 9.1. Класіфікацыя дэфектаў

Дэфекты якасці ПД звычайна класіфікуюцца па наступных крытэрыях.

### Класіфікацыя дэфектаў па ступені сур'ёзнасці

Улічаючы ўплыў на сістэму, дэфекты праграмнага дачынення можна разглядаць як:

- **Крытычныя** — Прычыны збою, страты даных або падзення
  - Паходжанне базы даных
  - Збой праграмы пры запуску і г.д.
- **Важныя** — Асноўныя функцыі парушаныя, але сістэма працуе:
  - Памылкі ўваходу
  - Памылкі апрацоўкі плацяжу і гэтак далей
- **Памяркоўныя** — Частковыя пахібы з абходнымі шляхамі:
  - Пошук вяртае няпоўныя вынікі
  - Проблемы з фарматаваннем спажывецкага інтэрфейсу
- **Нязначныя** — Касметычныя проблемы без уплыву на працу:
  - Памылкі ў правапісе
  - Невялікія проблемы з выраўноўваннем тэкста

### Класіфікацыя дэфектаў па прыярытэце

Па тэрміновасці выпраўлення праграмныя дэфекты можна падзяліць на наступныя групы:

- **Неадкладныя** — Павінны быць неадкладна выпраўлены, бо блакуюць далейшую працу і павінна быць выпраўлены ў наступнай зборцы

- **Тэрміновыя** — Важна выпраўліць у хутка, бо ўплываюць на ключавыя функцыі і павінны быць выпраўлены ў існай версіі
- **Памяркоўныя** — Маюць абыходныя шляхі і павінны быць выпраўлена ў наступным рэлізе
- **Нетэрміновыя** — Нязначныя праблемы, якія могуць быць выпраўлены пры неабходнасці, як правіла, у будучым рэлізе

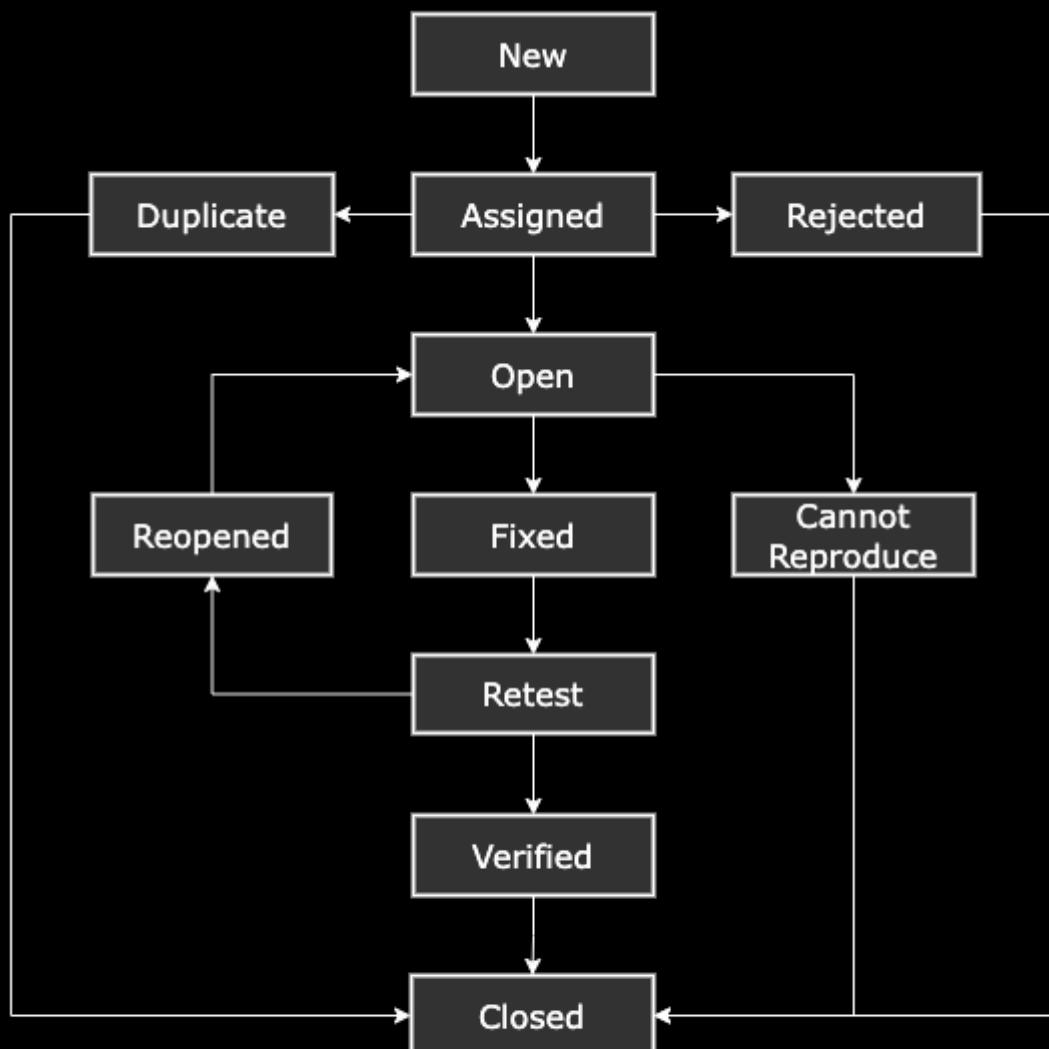
## Класіфікацыя дэфектаў па паходжанні

Улічваючы крыніцу дэфекту, іх можна падзяліць на наступныя катэгорыі:

- **Патрабаванні** — Няслушныя або адсутныя патрабаванні:
  - Незразумелыя правілы вядзення бізнесу
  - Супярэчлівыя патрабаванні і г.д.
- **Праектаванне** — Недахопы архітэктуры ці дызайну:
  - Дрэнная структура базы даных
  - Небяспечная архітэктура і г.д.
- **Кадаванне** — Памылкі рэалізацыі:
  - Сінтаксічныя памылкі
  - Лагічныя памылкі
  - Праблемы з узаемадзеяннем кампанентаў і г.д.
- **Тэсціраванне** — Памылкі ў тэстовых выпадках або асяроддзі:
  - Няправільныя тэставыя даныя
  - Праблемы з канфігурацыяй асяроддзя і г.д.
- **Дэфекты інтэграцыі** — Праблемы ўзаемадзеяння модулей:
  - Праблемы сумяшчальнасці API
  - Неадпаведнасці фармату даных і інш.

## 9.2. Жыццёвы цыкл дэфекту

Жыццёвы цыкл дэфекту можна прадстаўіць з дапамогай наступнай схемы.



*Defect Life Cycle*

## **Падрабязныя пераходы станаў**

Разуменне дэфектаў ПД і іх жыццёвага цыклу мае фундаментальнае значэнне для эфектыўнага дасягнення якасці.

Добра кіраваны працэс ліквідацыі дэфектаў, як правіла, уключае наступныя падрабязныя пераходы станаў:

- Новы — Дэфект выяўлены і зарэгістраваны ўпершыню
  - Прызначаны — Дэфект прызначаны распрацоўшчыку
  - Дублікат — дэфект ужо быў заяўлены кімсьці іншым
  - Адхілены — Дэфект несапраўдны або не з'яўляеца гэтакім
  - Адкрыты — Распрацоўшчык прымае дэфект і пачынае яго выпраўляць
  - Выпраўлены — распрацоўшчык рэалізуваў выпраўленне
  - Паўторна тэставаны — тэсціроўшчык правярае выпраўленне
  - Паўторна адкрыты — выпраўленне няпоўнае або дэфект выяўляеца зноў
  - Правераны — правільнасць выпраўлення пацверджана
  - Закрыты — Дэфект фармальна закрыты ў сістэме
-

## **9.3. Справаздача аб дэфекце**

**Справаздача аб дэфекце** — гэта афіцыйны документ, які апісвае недахоп, што прыводзіць да нечаканых паводзін ПД або няправільных вынікаў яго працы.

Ён служыць асноўным інструментам маркавання паміж тэсціроўшчыкамі, распрацоўшчыкамі і зацікаўленымі бакамі для адсочвання і вырашэння праблем з ПД.

### **Ключавыя кампаненты**

Асноўныя кампаненты справаздачы аб дэфекце:

- Ідэнтыфікатар дэфекту — унікальнае абазначэнне
- Рэзюмэ — кароткі, апісальны загаловак
- Паведаміў — хто выявіў дэфект
- Дата паведамлення — калі быў выяўлены дэфект
- Кампанент — закрануты раздел праграмы
- Сур'ёзнасць — уплыў на функцыянальнасць сістэмы
- Крокі ўзнаўлення — крокі для ўзнаўлення дэфекту
- Чаканы вынік — што павінна адбыцца ў адпаведнасці з патрабаваннямі
- Фактычны вынік — што адбываецца насамрэч
- Асяроддзе — наваколле выяўлення дэфекту
- Доказы — скрыншоты, запісы, журналы, здымкі БД

### **Ключавыя характеристыкі**

Эфектыўная справаздача аб дэфекце:

- Зразумелая — не мае двухсэнсоўнасці
- Лаканічная — без лішняй інфармацыі
- Поўная — змяшчае ўсю неабходную інфармацыю
- Паслядоўная — адпавядае арганізацыйным стандартам
- Узнаўляльная — іншыя могуць узнавіць праблему

Напісанне эфектыўных справаздач аб дэфектах патрабуе:

- Аб'ектыўнай мовы —*Пасля націскання кнопкі "Адправіць" з'яўляецца паведамленне пра памылку: "Памылка далучэння да БД", а не "Сістэма рушыцца пры спробе захаваць даныя ў базе"*
- Канкрэтныі і дэталяў —*У выпадаючым спісе адлюстроўваеца 5 элементаў замест чаканых 7: адсутны ролі "Адміністратор" і "Менеджар", а не "Выпадаючыя параметры няслушныя"*
- Аднаго дэфекту на справаздачу —*Дэфект A: "Кнопка ўваходу не працуе", Дэфект B: "Поле пароля дазваляе ўжываць забароненыя сімвалы", а не "Проблемы з логінам і паролем"*
- Далучэння візуальных доказаў:
  - Стрэлак ці кружкоў для вылучэння бага на скрыншотах
  - Запісу відэа для складаных шматэтапных дэфектаў
  - Копій адпаведных запісаў журнала з часовымі меткамі

Гожая справаздача аб дэфекце — гэта не проста апісанне, але важны інструмент яго абмеркавання і спрыяння павышэнню якасці.

## Ключавыя мэты

Эфектыўнае паведамленне аб дэфектах:

- Паскарае вырашэнне праблем дзякуючы паразуменню
- Надае даныя для паказчыкаў якасці і удасканалення працэсаў
- Забяспечвае падсправаздачнасць праз належнае адсочванне
- Спрыяе аргументаваному рашэнню аб мажлівасці рэлізу

Якасць справаздач аб дэфектах непасрэдна ўплывае на эфектыўнасць працэсу распрацоўкі і якасць канчатковага прадукту.

Інвеставанне часу ў напісанне зразумелых, поўных і прафесійных справаздач аб дэфектах прыносіць значныя дывідэнды ва ўсім ЖЦР ПД.

# **Х. Класіфікацыя відаў тэсціравання ПД**

## **10. Класіфікацыя тыпаў тэсціравання ПД**

**Класіфікацыя відаў тэсціравання ПД** — гэта сістэматычная класіфікацыя відаў тэсціравання ПД ў адпаведнасці з крытэрыямі іх прызначэння.

Успрыманне класіфікацыі дазваляе прымаць аргументаваныя рашэнні і спрыяць распрацоўцы надзейных праграмных прадуктаў.

### **Асноўныя мэты**

Мэты класіфікацыі відаў тэсціравання ПД:

- Комплексная сегментация
- Адаптация стратэгіі тэсціравання
- Эфектыўнае размеркаванне рэсурсаў
- Прымусовае змяншэнне рызык
- Адаптыўнае спрыянне якасці

Выбар тыпу тэставання залежыць ад такіх фактараў, як:

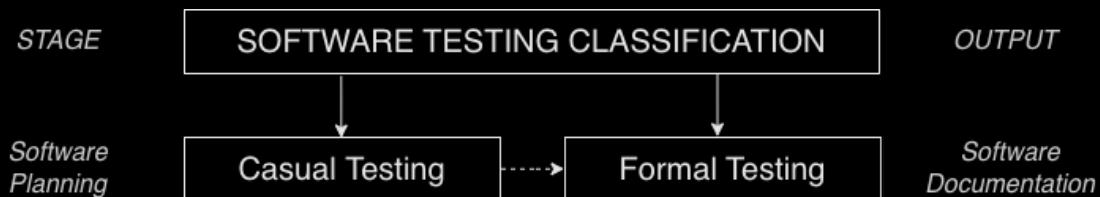
- Аб'ём сістэмы
  - Этап ЖЦР ПД
  - Бюджэт праекта
  - Значнасць ПД
-

## 10.1. Віды тэставання на этапе планавання ПД

**Планаванне ПД** — гэта этап вызначэння падыходу да стратэгіі документавання будучай распрацоўкі, тэсціравання і звязаных з імі працэсаў.

На этапе планавання праграмнага дачынення ёсць два асноўныя тыпы тэсціравання:

- Выпадковае тэсціраванне
- Фармальнае тэсціраванне



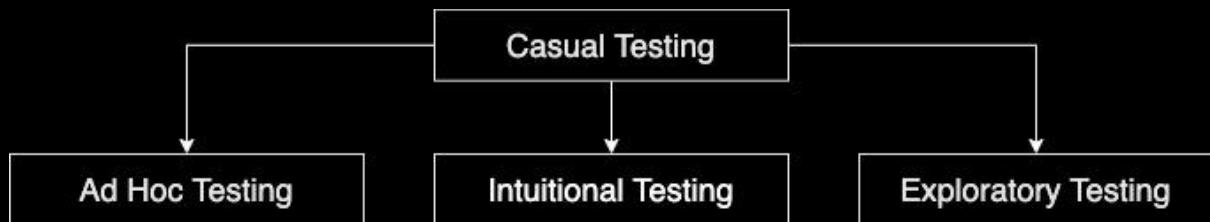
Пункцірная лінія ілюструе тэндэнцыю пераходу ад выпадковага да фармальнага тэсціравання, паколькі фармальнае тэставанне забяспечвае такі ўзровень надзейнасці, паўтаральнасці і падсправаздачнасці, якіх выпадковае тэсціраванне па сваёй сутнасці мець не можа.

### Выпадковае тэсціраванне

**Выпадковае тэсціраванне** — гэта тып тэсціравання праграмнага дачынення, які праводзіцца без якога-небудзъ спецыяльнага планавання, дакументацыі і працэdur.

Існуе трох асноўных стылі нефармальнага тэсціравання:

- Адвольнае тэсціраванне
- Інтуітыўнае тэставанне
- Даследчыцкае тэставанне



## Адвольнае тэсціраванне

**Адвольнае тэсціраванне** — гэта найменш фармальны стыль Выпадковага тэсціравання, які таксама называецца.

Адвольныя тэсты выконваюцца толькі адзін раз, калі не выяўлена памылка, і паколькі само тэставанне не дакументавана, выяўленыя дэфекты цяжка ўзнавіць.

Моц адвольнага тэставання заключаецца ў яго прымяненні на самых ранніх этапах жыццёвага цыклу распрацоўкі, дзякуючы чаму відавочныя памылкі выяўляюцца даволі хутка.

	Test Planning	Documentation	Prior Experience
Ad Hoc Testing	No	No	No
Intuitive Testing	No	No	Yes
Exploratory Testing	No	Yes	Yes

## Інтуітыўнае тэставанне

**Інтуітыўнае тэставанне** — часта называнае **Пошукам памылак** — гэта стыль выпадковага тэсціравання, заснаваны на папярэднім вопыце і ведамі ў галіне тэсціравання.

Для пошуку памылак няма відавочных правілаў тэсціравання і яно цалкам залежыць ад сітуацыі.

Гэта вызначаецца выключна мінульм вопытам і інтуіцыяй інжынера па спрыянні якасці, які, магчыма, ведае праблемныя вобласці, якія звычайна выклікаюць збоі праграмнага дачынення.

## Даследчыцкае тэставанне

**Даследчыцкае тэставанне** — гэта стыль выпадковага тэсціравання, які выконваецца як працэс адначасовага навучання, распрацоўкі тэстаў і выканання тэстаў.

Джэм Канер, які ўвёў гэты тэрмін у 1984 годзе, вызначае даследчыцкае тэставанне наступным чынам:

*"Даследчыцкае тэставанне" — гэта стыль тэсціравання ПД, які падкрэслівае асабістую свободу і адказнасць асобнага інжынера за пастаянную аптымізацыю якасці сваёй працы, разглядаючы навучанне, звязанае з тэставаннем, распрацоўку, выкананне і інтэрпрэтацыю вынікаў тэстаў як узаемна дапаўняльныя дзеянні, якія выконваюцца паралельна на працягу ўсяго праекта.*

Падчас тэставання ПД тэсціроўшчык вывучае рэчы, якія разам з вопытам і творчасцю ствараюць новыя тэсты для запуску.

Кантрольны спіс — гэта адзіны артэфакт, які ўжываецца для структуравання даследчыцкага тэставання, каб не марнаваць час на паўтарэнне адных і тых жа тэстаў.

## Фармальнае тэсціраванне

**Фармальнае тэсціраванне** — гэта тып тэставання ПД, які выконваецца з належным планаваннем, дакументацыяй і працэдурамі.

Ён выконваецца з паўнавартаснай дакumentацыяй тэст-кейсаў і рупліва прытрымліваецца жыццёвага цыклу тэсціравання ПД.

Фармальнае тэсціраванне каштуе даражэй з-за выдаткаў на падрыхтоўку тэста-кейсаў, навучанне персаналу і напісанне дакументацыі.

## Крыніцы дакументацыі

Фармальнае тэсціраванне мае тры асноўныя крыніцы дакumentaцыі:

- Гісторыі спажыўцуў
- Бізнес-сцэнарыі
- Спецыфікацыя патрабаванняў да ПД

### Гісторыі спажыўцуў

Гісторыі спажыўцуў — гэта кароткія сцвярджэнні, якія ўжываюцца ў распрацоўцы ПД і асвятляюць адну функцыю, задачу або мэту з пункту гледжання спажыўца.

Тыповы фармат гісторыі спажыўца выглядае наступным чынам: "Як спажывец, я хачу мець нейкую мэту, каб мець нейкую прычыну", напрыклад:

*Як пакупнік у інтэрнэце, я хачу мець магчымасць фільтраваць тавары па катэгорыях, каб лёгка знаходзіць патрэбныя мне рэчы.*

### Бізнес-сцэнарыі

Бізнес-сцэнарыі — гэта нястрога аkrэсленныя апісанні канкрэтных сітуацый, якія тлумачаць погляд спажыўца на дасягненне мэты або выкананне задачы.

Яны раскрываюць погляд кліента на функцыю ПД, амінаючы працэс яе рэалізацыі, і могуць быць прадстаўлены ў выглядзе серыі кроکаў.

Бізнес-сцэнарыі даюць контэкст і разуменне таго, як пэўны тып чалавека можа ўзаемадзейнічаць з прадуктам або паслугай.

## **Спецыфікацыя патрабаванняў да ПД**

**Спецыфікацыя патрабаванняў да ПД** — гэта документаваны набор чаканняў, які ахінае ўсе функцыянальныя аспекты і фарміруе аснову для выканання тэстаў для розных функцый, абмежаванняў і магчымасцей.

Ён служыць адзінай крыніцай праўды як для праграмнага кода, так і для распрацоўкі тэставых выпадкаў на наступных этапах жыццёвага цыклу распрацоўкі праграмнага дачынення.

## **Тыпы фармальнага тэсціравання**

Адпаведна, існуе тры асноўныя тыпы фармальнага тэсціравання:

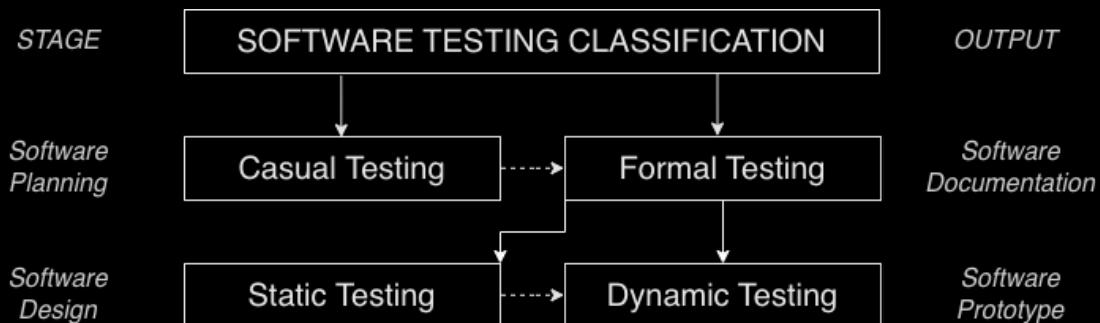
- Тэсціраванне на аснове гісторый
  - Тэсціраванне на аснове сцэнарыяў
  - Тэсціраванне на аснове спецыфікаций
-

## 10.2. Віды тэстування на этапе праектавання ПД

**Праектаванне ПД** — гэта этап, які дазваляе атрымаць першыя працоўныя прататып або, інакш кажучы, мінімальна жыццяздольны прадукт.

На этапе проектавання ПД ёсць два асноўныя тыпы тэсціравання:

- Статычнае тэстуванне
- Дынамічнае тэстуванне



Пункцірная лінія паказвае перавагу дынамічнага тэсціравання перад статычным, паколькі першае выяўляе больш шырокі шэраг конкретных дэфектаў, а другое падыходзіць толькі для ранняга прадухілення дэфектаў.

### Статычнае тэстуванне

**Статычнае тэстуванне** — гэта тып фармальнага тэстування, які выконваецца без выканання праграмнага кода.

Па гэтай прычыне статычнае тэстуванне таксама называюць:

- Тэсціраванне дакументацыі
- Тэставанне без выканання
- Праверачнае тэставанне
- Тэставы агляд

## Документы для праверкі

Найважнейшыя документы, якія належаць статычнаму тэставанню, ўключаюць:

- Документы аналізу патрабавання:
  - Гісторыі спажыўцу
  - Бізнес-сценарыі
  - Спецыфікацыі патрабавання да ПД
- Спецыфікацыі праектавання ПД
- Документацыя па распрацоўцы тэстаў:
  - Планы тэставання
  - Тэставыя выпадкі
- Документацыя па распрацоўцы тэстаў:
  - Зыходны код
  - Тэставыя скрыпты
- Документацыя па выпуску:
  - Інструкцыі спажыўца
  - Дапаможныя звесткі

## Мэта статычнага тэсціравання

Статычнае тэставанне, якое не патрабуе запуску праграмнага дачынення, дазваляе выяўляць важныя памылкі і неадназначнасці на самых ранніх этапах распрацоўкі.

Чым раней будуць выяўлены лагічныя памылкі і супярэчнасці патрабаванням, тым меншы кошт іх выпраўлення і тым прасцейшым будзе працэс распрацоўкі праграмнага дачынення.

## **Дынамічнае тэставанне**

**Дынамічнае тэставанне** — гэта тып фармальнаага тэсціравання, які выконваецца з выкананнем праграмнага кода.

Яно дазваляе праверыць функцыянальныя паводзіны ПД, памяць, працэсы і агульную якаснасць у дынаміцы.

Вось чаму дынамічнае тэставанне таксама называецца **Тэставаннем выканання** або **Валідацыйным тэставаннем**.

### **Мэта дынамічнага тэставання**

Дынамічнае тэставанне факусуеца на ацэнцы паводзін праграмнага кода падчас выканання і назіранні за яго паводзінамі ў розных умовах.

Яно накіравана на забеспячэнне правільнай працы ПД падчас і пасля ўстаноўкі, і мае на ўвазе парашуннне фактычнага выводу з чаканым.

Дынамічнае тэставанне можа пачацца да таго, як праграмнае дачыненне будзе гатова; некаторыя блокі або кампаненты могуць быць пратэсціраваны дынамічна з ужываннем заглушак, драйвераў або адладчыкаў.

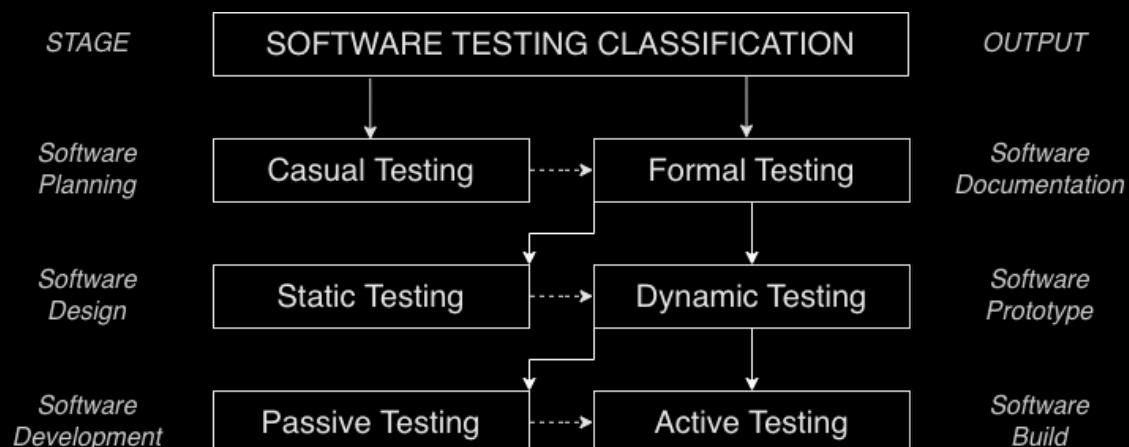
---

## 10.3. Віды тэстування на этапе распрацоўкі ПД

**Распрацоўка ПД** — гэта этап стварэння адначасова і працоўнай, і тэстуемай зборкі мэтавага праграмнага дачынення.

На этапе распрацоўкі праграмнага дачынення існуе два асноўныя тыпы тэсціравання:

- Пасіўнае тэстуванне
- Активнае тэстуванне



Пункцірная лінія адлюстроўвае тэндэнцыю пераходу ад пасіўнага да активнага тэстування, абумоўленую патрэбай у хуткасці, надзейнасці і пастаяннай зваротнай сувязі ў сучаснай распрацоўцы праграмнага дачынення.

### Пасіўнае тэстуванне

**Пасіўнае тэстуванне** — гэта тып дынамічнага тэстування, які выконваецца праз выкананне кода без узаемадзеяння спажыўца з праграмным дачыненнем.

Пасіўнае тэставанне азначае праверку паводзін сістэмы праз аўтаномную праверку выканання, праверку лога і аналіз трасіроўкі стэка.

## **Актыўнае тэсціраванне**

**Актыўнае тэсціраванне** — гэта тып дынамічнага тэсціравання, якое выконваецца праз выкананне кода разам з узаемадзеяннем спажыўца з праграмным дачыненнем.

Актыўнае тэставанне прызначана для ацэнкі надзейнасці, устойлівасці і аптымальнай якаснасці праграмнага дачынення ў дынамічных асяроддзях.

---

## 10.4. Віды тэставання на этапе тэсціравання ПД

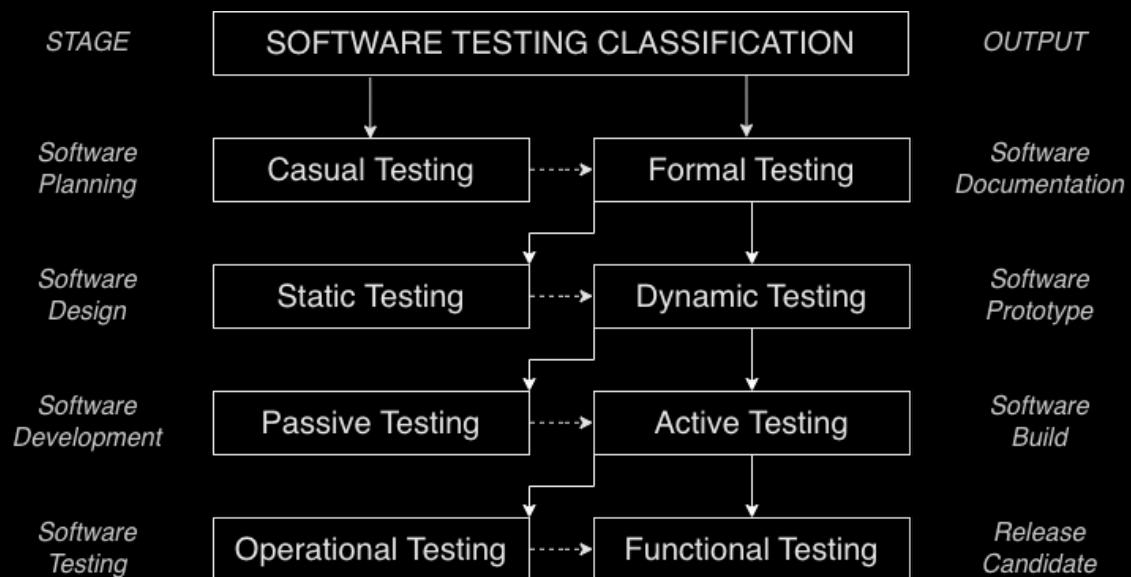
**Тэсціраванне ПД** — гэта этап паўторных праверак зборкі ПД для давядзення яго да стану рэліз-кандыдата.

**Рэліз-кандыдат** — гэта патэнцыйна доступная версія праграмнага дачынення, якая мае поўны набор функцый і лічыцца гатовай да фінальнага тэставання перад афіцыйным выпускам.

Гэта ўяўляе сабой заключны этап цыклу распрацоўкі, на якім ПД, верагодна, стане разгорнутай спажывецкай версіяй, калі не будуць выяўлены крытычныя праблемы.

На этапе тэсціравання ПД актуальны два іх тыпы:

- Аперацыйнае тэставанне
- Функцыянальнае тэставанне



Пункцір паказвае большую важнасць функцыянальнага тэсціравання ў параўнанні з аперацыйным, падкрэсліваючы, што працоўны праграмны прадукт з'яўляецца неабходнай умовай для ацэнкі яго хуткасці, бяспекі або надзейнасці.

## **Аперацыйнае тэставанне**

**Аперацыйнае тэставанне** — гэта тып актыўнага тэставання для праверкі аперацыйных — нефункцыянальных — аспектаў ПД.

Нефункцыянальныя аспекты — гэта аспекты, якія выяўляюць якасць прадукту, асабліва ў кантэксце прыдатнасці для спажыўцу, і ніколі не звязаны з канкрэтнай функцыяй ПД:

- Якаснасць
- Зручнасць спажывання
- Надзейнасць
- Маштабаванасць
- Бяспека і г.д.

Аперацыйнае тэсціраванне прызначана для праверкі нефункцыянальных аспектаў, якія ніколі не разглядаюцца пры функцыянальным тэсціраванні ПД — тых, якія не могуць быць звязаныя з пэўнай функцыяй або дзеяннем спажыўца.

Яно правярае, якім чынам працуе ПД, у адрозненне ад праверкі функцыянальных паводзін.

Аперацыйныя патрабаванні адлюстроўваюць якасць ПД у цэлым, у прыватнасці, у кантэксце яго прыдатнасці для спажыўцу.

## **Функцыянальнае тэставанне**

**Функцыянальнае тэставанне** — гэта тып актыўнага тэставання, скіраванага на праверку адпаведнасці ПД яго функцыянальным патрабаванням і спецыфікацыям.

Функцыянальнае тэсціраванне служыць для праверкі дзеяння ПД шляхам прадастаўлення ўваходных даных і ацэнкі вынікаў у адпаведнасці з дакументаванымі ўмовамі.

Праграмная функцыя мае дзве адметныя неаперацыйныя рысы, якія маюць вырашальнае значэнне для функцыянальнага тэсціравання:

- Ацэньваемасць — наяўнасць чаканых і рэальных вынікаў
- Двухбаковасць — мажлівасць станоўчага і адмоўнага тэставання

**Ацэньваемасць** — гэта ўзровень прастаты і эфектыўнасці, з якімі функцыю можна ацаніць, вымераць і вызначыць адпаведна пэўных крытэрыяў.

**Двухбаковасць** з'яўляецца адметнай уласцівасцю праграмнай функцыі, якая дасягаецца дзякуючы яе ўзаемнасці для мэт тэсціравання.

## Тыпы функцыянальнага тэставання

Двухбаковасць функцыі дазваляе падзяліць функцыянальнае тэсціраванне на два асноўныя тыпы:

- Станоўчае — Пазітыўнае — тэставанне
- Адмоўнае — Негатыўнае — тэставанне

### Станоўчае тэставанне

**Станоўчае тэставанне** — часам называнае **Тэставаннем шчаслівага шляху** — гэта тып функцыянальнага тэсціравання ПД, якое праводзіцца шляхам прадастаўлення сапраўдных, правільных і карэктных звестак у якасці ўваходных даных.

Пазітыўнае тэставанне правярае, ці паводзіць сябе ПД належным чынам з пазітыўнымі і чаканымі ўводамі спажыўца, каб пацвердзіць саму жыццяздольнасць ПД.

Станоўчы тэст павінен папярэднічаць адмоўнаму, бо дэфекты, выявленыя ў выніку станоўчых тэстаў, прадухіляюць неабходнасць далейшага тэсціравання.

## **Адмоўнае тэставанне**

**Адмоўнае тэставанне** — гэта тып функцыянальнага тэсціравання, які праводзіцца над ПД шляхам прадастаўлення няправільных, памылковых або пашкоджаных звестак у якасці ўваходных даных.

Негатыўнае тэставанне правярае, ці паводзіць сябе ПД належным чынам пры адмоўных або непажаданых дзеяннях спажыўца, каб выявіць прабелы ў прадухіленні памылак.

Гэта азначае, што ПД можа і павінна працаваць правільна ў няправільных умовах, гэта значыць пры адмоўным тэсціраванні.

## **Падыходы да функцыянальнага тэсціравання**

Для функцыянальнага тэсціравання характэрныя трывалыя падыходы:

- Чорная скрыня
- Шэрай скрыня
- Белая скрыня

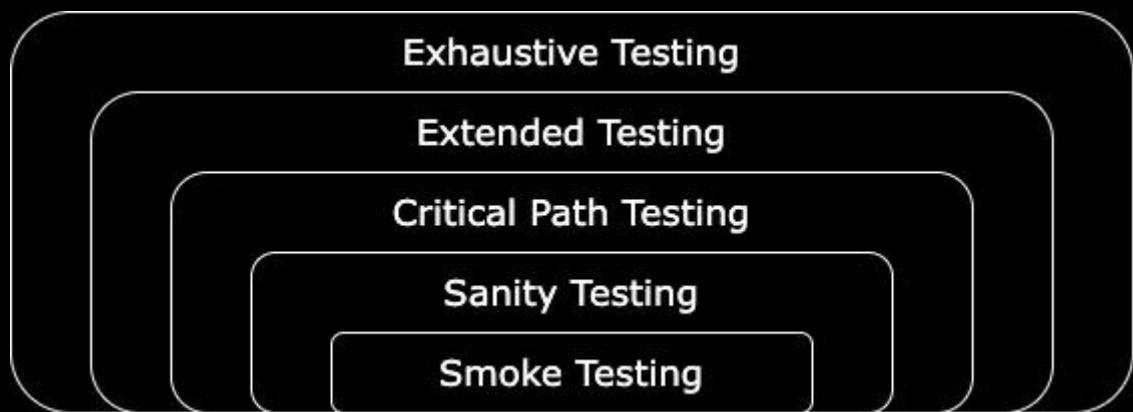
### **Чорная скрыня**

**Чорная скрыня** — гэта падыход да функцыянальнага тэсціравання без ведання праграмнага кода, унутранай структуры і дэталяў рэалізацыі.

Гэты тэрмін сімвалізуе немагчымасць убачыць унутраную рэалізацыю праграмнага дачынення, што робіць кране важным досвед тэсціроўшчыкаў.

Існуе пяць узроўняў тэставання чорнай скрыні ў залежнасці ад аб'ёму тэставання:

- Дымавое тэставанне
- Тэставанне цэласнасці ПД
- Тэставанне крытычнага шляху
- Паширанае тэсціраванне
- Вычарпальнае тэставанне



**Дымавое тэставанне** — гэта базавы ўзровень тэсціравання чорнай скрыні, які праводзіцца для выяўлення відавочных пахіб, дастаткова сур'ёзных для адхілення зборкі ПД, прапанаванай для далейшай праверкі.

**Тэставанне цэласнасці ПД** — гэта тып тэсціравання чорнай скрыні, які праводзіцца для хуткай ацэнкі працаздольнасці ПД у адпаведнасці з базавым наборам праверак.

**Тэставанне крытычнага шляху** — ці **Скразное тэставанне** — гэта тып тэставання чорнай скрыні, які праводзіцца для аналізу той функцыянальнасці, якая найчасцей ужываецца большасцю спажыўцу ПД.

**Пашыранае тэсціраванне** — гэта тып тэсціравання чорнай скрыні, якое праводзіцца для вывучэння ўсіх функцыянальных магчымасцей, заяўленых у спецыфікацыі патрабаванняў да ПД.

**Вычарпальнае тэставанне** — або **Поўнае тэсціраванне** — гэта тып тэставання чорнай скрыні, які праводзіцца для проверкі таго, што ПД працуе правільна ў любой магчымай сітуацыі.

Хоць вычарпальнае тэставанне насамрэч недасягальнае, стараннае тэставанне дапамагае ствараць надзейныя праграмы з мінімальнымі дэфектамі.

## **Шэрай скрыня**

**Шэрай скрыня** — гэта падыход да функцыянальнага тэсціравання з частковым веданнем праграмнага кода, унутранай структуры і дэталяў рэалізацыі.

Тэрмін азначае, што гэты падыход з'яўляецца спалученнем тэсціравання чорнай і белай скрыні.

## **Белая скрыня**

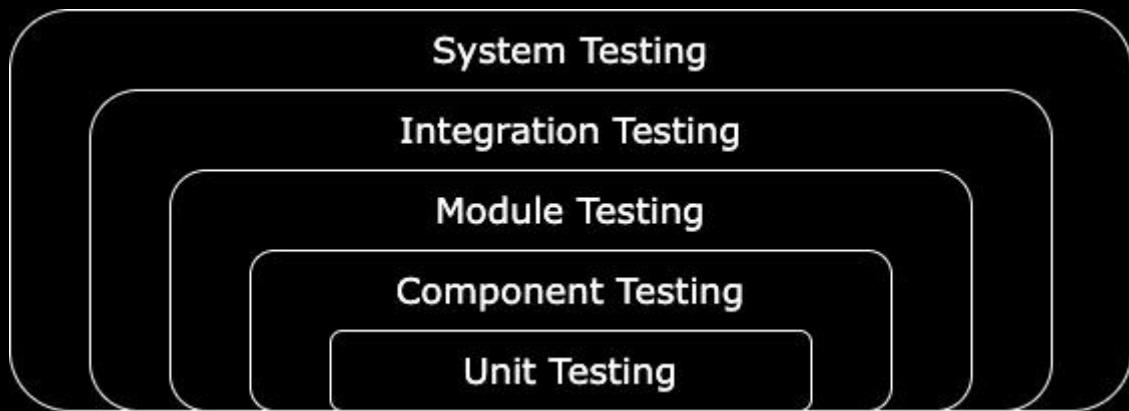
**Белая скрыня** — гэта падыход да функцыянальнага тэсціравання з веданнем праграмнага кода, унутранай структуры і дэталяў рэалізацыі.

Гэты тэрмін адносіцца да канцэпцыі празрыстай шыбы, якая сімвалізуе здольнасць бачыць скрозь вонкавую абалонку ПД яго ўнутраны стан.

Існуе пяць узроўняў тэсціравання белай скрыні ў залежнасці ад аб'ёму тэставання:

- Юніт-тэсціраванне
- Тэсціраванне кампанентаў
- Модульнае тэсціраванне
- Інтэграцыйнае тэсціраванне

- Тэсціраванне сістэмы



**Юніт-тэсціраванне** — базавы ўзровень тэсціравання белай скрыні, які звычайна выконваюць распрацоўшчыкі і які сканцэнтраваны на тэсціраванні асобных блокаў ізалявана.

**Тэсціраванне кампанентаў** — гэта тып тэсціравання белай скрыні, які выконваецца на кампаненце ПД, атрыманым праз спалучэнне незалежных адзінак, асобна.

**Модульнае тэсціраванне** — гэта тып тэсціравання белай скрыні, якое факусуецца на праверцы незалежных модуляў, зробленых са згрупаваных кампанентаў, асобна.

**Інтэграцыйнае тэсціраванне** — гэта тып тэсціравання белай скрыні, які праводзіцца на наборы модуляў, аб'яднаных у падсістэмную сутнасць, для праверкі іх сумеснай працаздольнасці.

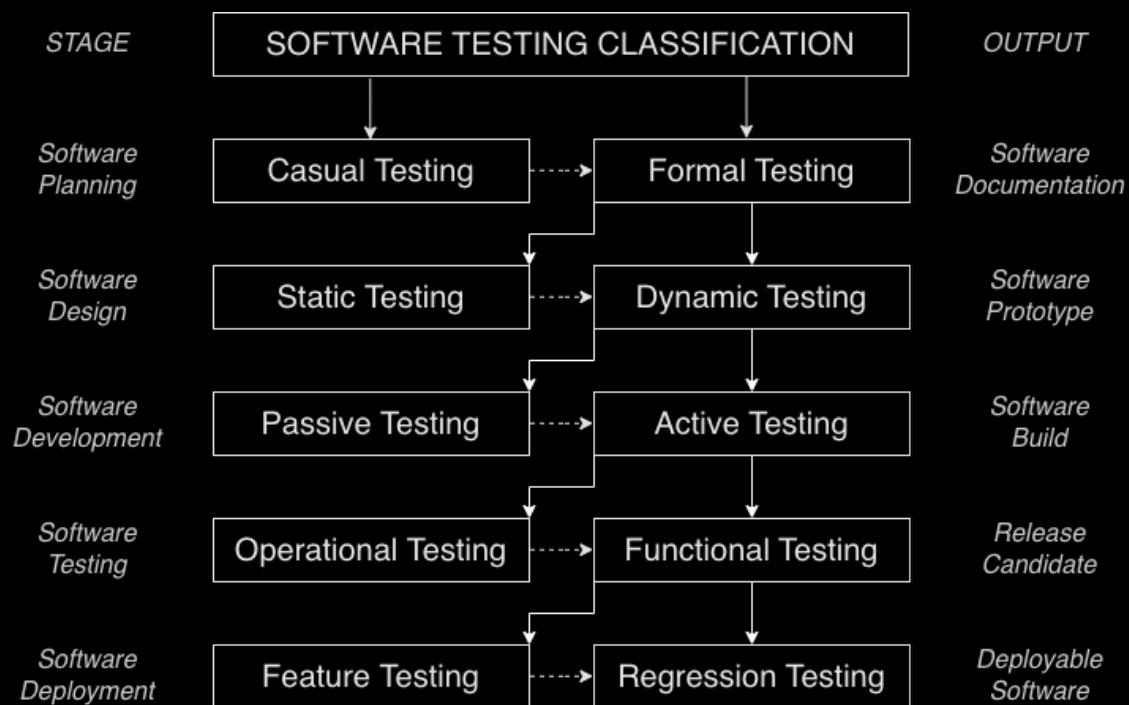
**Тэсціраванне сістэмы** — гэта тып тэсціравання белай скрыні, якое праводзіцца над інтэграваным праграмным дачыненнем для ацэнкі яго адпаведнасці заданым патрабаванням.

## 10.5. Віды тэставання на этапе разгортання ПД

**Разгортанне праграмнага дачынення** — гэта этап, на якім рэліз-кандыдат тэстуецца, зацвярджаецца і перадаецца кліенту.

На этапе разгортання праводзяцца два тыпы тэсціравання ПД:

- Тэсціраванне новых функцый — Прагрэсіўнае тэсціраванне
- Рэгрэсійнае тэставанне



Пункцір адлюстроўвае распаўсюджанае і апраўданае меркаванне ў прафесійнай індустрыві распрацоўкі ПД, што абарона наяўнай функцыянальнасці з'яўляецца больш высокім прыярытэтам, чым далучэнне новай.

## **Тэсціраванне новых функцый**

**Тэсціраванне новых функцый** — гэта тып функцыянальнага тэсціравання, які праводзіцца для праверкі адпаведнасці новага кода або змененняў у асяроддзі патрабаванням да ПД.

Мэта тэсціравання новых функцый — гарантаваць, што новыя функцыі працуюць належным чынам, не ўносяць дэфектаў і падтрымліваюць агульную якасць праграмнага дачынення.

## **Рэгрэсійнае тэставанне**

**Рэгрэсійнае тэставанне** — гэта тып функцыянальнага тэсціравання, які праводзіцца для праверкі таго, ці захоўваюць новыя змены ў кодзе або асяроддзі раней дасягнутую якасць.

Рэгрэсійнае тэставанне азначае, што ўжо распрацаваныя тэсты выконваюцца паўторна, каб праверыць уплыў змененняў на функцыянальнасць, якая існавала раней.

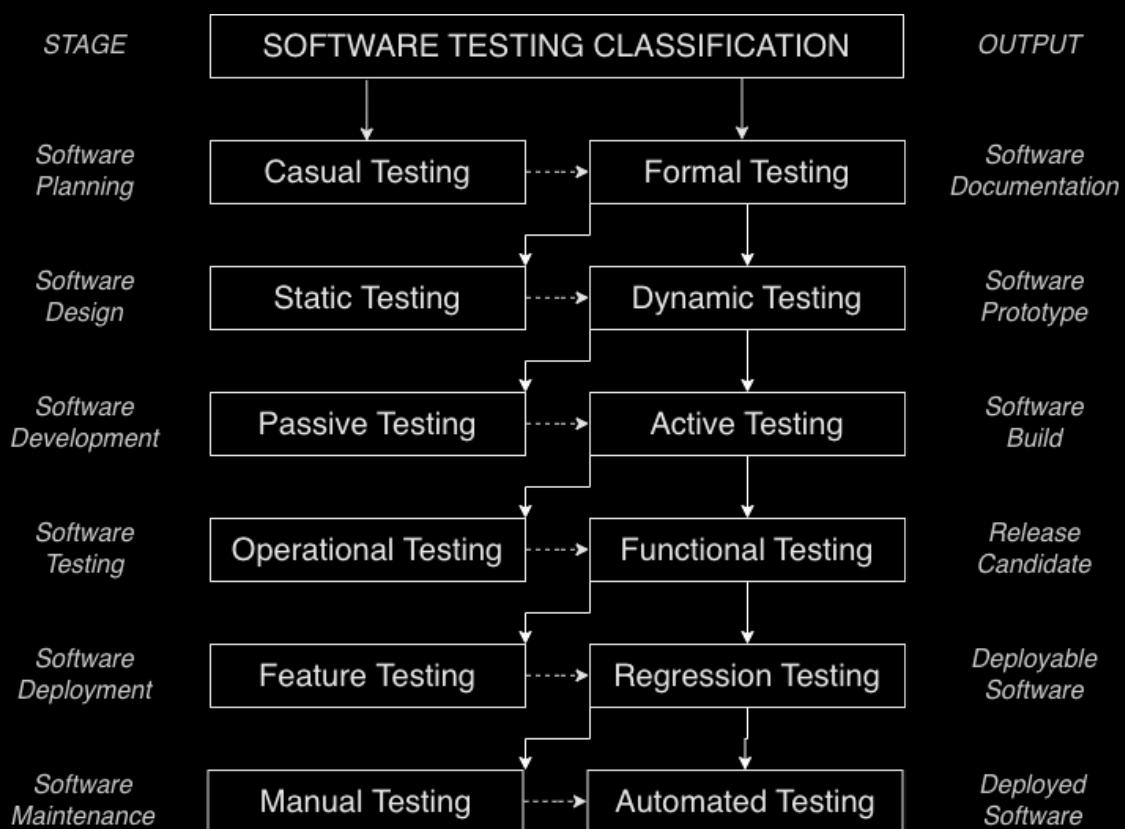
---

## 10.6. Віды тэстування на этапе аблугоўвання ПД

**Аблугоўванне ПД** — гэта этап пашырэння ахопу тэстування ПД і, такім чынам, дасягнення яго больш высокай якасці падчас наступнага цыклу распрацоўкі.

На этапе аблугоўвання выконваюцца два тыпы рэгрэсійнага тэсціравання:

- Ручное тэсціраванне
- Аўтаматызаванае тэсціраванне



Пункцір ад ручнога тэсціравання да аўтаматызаванага падкрэслівае большую важнасць апошняга, бо яно забяспечвае хуткасць, маштабаванне і надзейнасць такім чынам, якога ручное тэсціраванне не можа дасягнуць, што робіць яго асновай любой каманды, якой неабходна часта і ўпэўнена выпускаць ПД.

## **Ручное тэсціраванне**

**Ручное тэсціраванне** — гэта тып рэгрэсійнага тэсціравання, якое выконваецца ўручную.

Ручное тэсціраванне азначае, што ўсе важныя праверкі функцый праграмнага дачынення і стварэнне тэставых справаздач выконваюцца без аўтаматызаваных інструментаў тэсціравання.

## **Аўтаматызаванае тэсціраванне**

**Аўтаматызаванае тэсціраванне** — гэта тып рэгрэсійнага тэсціравання, якое выконваецца аўтаматычна праз запуск тэставых сцэнарыяў з ужыццем аўтаматызаваных інструментаў тэсціравання.

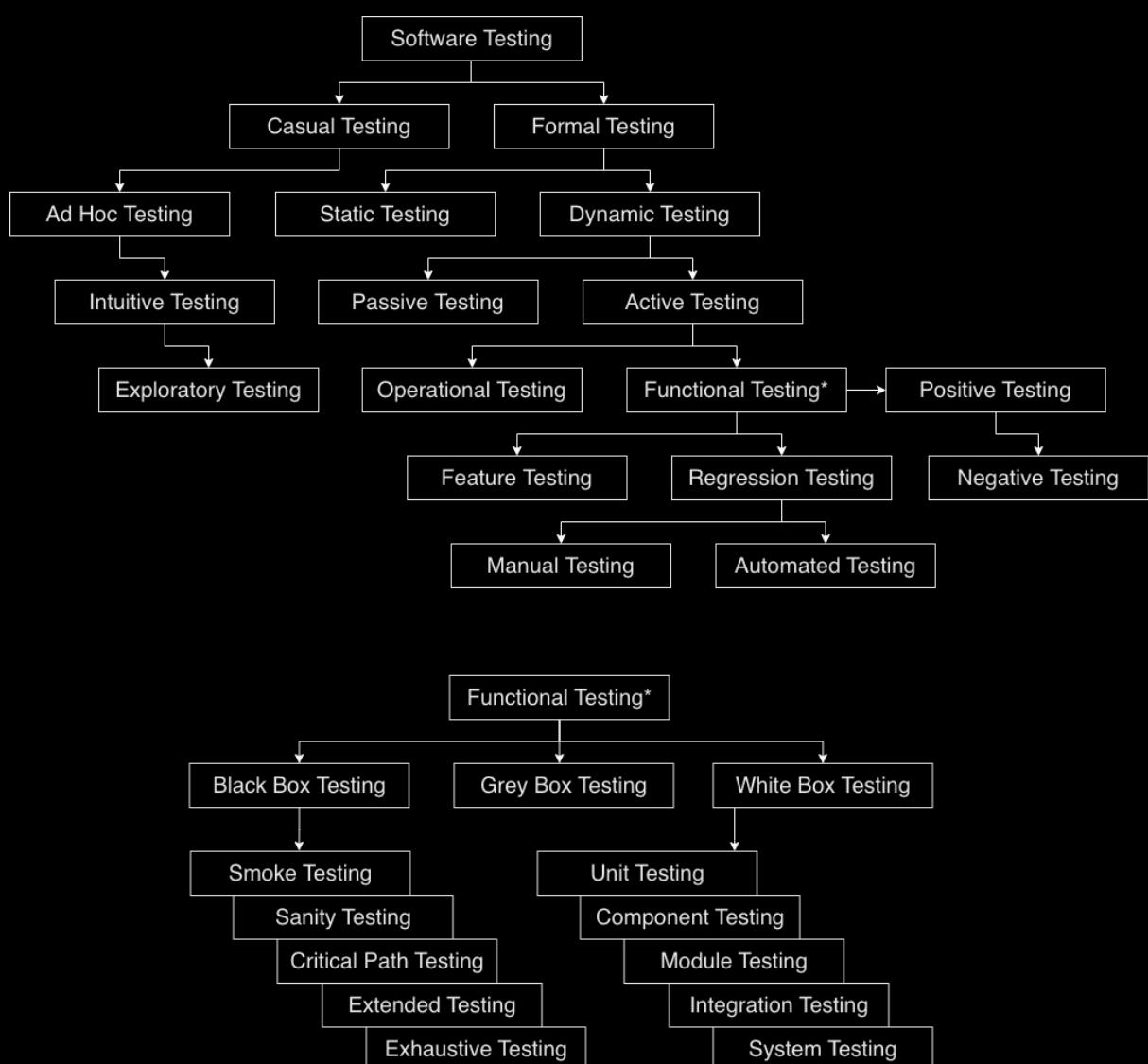
Аўтаматызаванае тэсціраванне прадугледжвае ўжытак адпаведных інструментаў аўтаматызацыі для распрацоўкі, удасканалення і выканання тэставых сцэнарыяў для праверкі ПД.

---

## 10.7. Схема класіфікацыі

### тэсціравання ПД

Класіфікацыю тэсціравання праграмнага дачынення можна канчаткова прадстаўіць наступнай схемай.



Згодна са схемай, функцыянальнае тэставанне — гэта самы важны, працаёмкі, рэурсаёмкі і адказны этап з улікам мэтаў ЖЦТ ПД.

Функцыянальнае тэставанне з'яўляецца найважнейшым этапам распрацоўкі, паколькі яно спалучае распрацоўку і рэальнае спажыванне ПД, гарантуючы яго надзейнасць, бяспеку і гатоўнасць да спажывання.

Без яго нават ідэальна распрацаваныя сістэмы рызыкуюць пацярпець катастрофічныя збоі.

---

# XI. Практика

## 11. Практыка

Вывучэнне асноў спрыяння якасці ПД, а затым пераход да рэальнага праекта — адзін з найбольш эфектыўных спосабаў пабудаваць плённую і перспектыўную кар'еру.

Высветлім, чаму — і распрацуем практычны шлях наперад.

### Чаму

Індустрывя ПД зразумела, што якасць нельга праверыць напрыканцы — яна павінна быць убудавана ў сам працэс.

Гэта стварыла значны попыт на кваліфікаваных спецыялістах па спрыянні якасці, якія разумеюць як тэорыю, так і практыку — кожныя ПД, вэб-сайт і лічбавая сістэма павінны быць пратэставаны.

Незалежна ад того, ці застанецеся вы ў сферы контролю якасці, ці пяройдзеце ў распрацоўку, кіраванне ці праектаванне, разуменне асноў контролю якасці зробіць вас больш грунтоўнымі, арыентаванымі на спажыўца і каштоўнымі — вы навучыцесь думаць пра памежныя выпадкі, рызыкі і спажывецкі досвед такім чынам, якім звычайнія распрацоўшчыкі часта ігнаруюць.

Тэорыя контролю якасці дае вам уяўленне пра тое, што тэставаць, чаму тэставаць, а практыка паказвае вам рэальнасць таго, як рэчы на самой справе ламаюцца, як мець зносіны з распрацоўшчыкамі, як расстаўляць прыярытэты — і гэта спалучэнне вельмі магутнае!

Выяўленне крытычнай памылкі да таго, як яна дасягне кліента, неверагодна задавальняе — вы непасрэдна прадухіляеце расчараўванне, абараняеце прыбытак кампаніі і абараняеце спажывецкі досвед.

Вы пераходзіце ад пасіўнага навучання да актыўнага стварэння каштоўнасці.

## Як

Не бярыцеся адразу за велькі праект — прытрымлівайцеся наступных крохаў, каб паступова развіць свае ўпэўненасць і навыкі.

- Крок 1. Замацуйце асновы — пераканайцеся, што вы разумееце асноўныя канцэпцыі:
  - ЖЦР і ЖЦТ ПД
  - Тыпы тэстаў
  - Асноўная тэставая документацыя
  - Іншыя асноўныя паняцці
- Крок 2: Патрэніруйцеся на кантраліваным рэальным праекце — пратэсціруйце тое, што існуе, але там, дзе стаўкі нізкія:
  - Абярыце папулярны вэб-сайт або праграму з адкрытым зыходным кодам — любы, які вы любіце найбольш
  - Стварыце просты чэк-ліст — паспрабуйце дадаць туды самыя каштоўныя праверкі, не ганіцеся за колькасцю
  - Напішыце фармальныя тэст-кейсы — не проста клікайце па кнопках, пішыце падрабязна адпаведна фармату
  - Выканайце тэсты і паведамце пра памылкі — знайдзіце рэальный збоі або проблемы з ПД і паведамляйце пра іх
- Крок 3: Далучыцеся да рэальнага праекта:
  - ПД з адкрытымі крыніцамі
  - Валанцёрскія праекты
  - Стажыроўкі або пасады малодшага тэсціроўчыка

Гэта дасць вам магчымасць стварыць партфоліё і атрымаць практычныя вопыт працы.

Такая рэальнасць — месца для сапраўднага навучання.

А мэта — гэта вы, гатовыя да каманднай працы.

## Калі

Тэарэтычныя веды — гэта карта, а практика над проектам — падарожжа, якое навучыць арыентавацца на рэальнай мясцовасці.

Вы будзеце збянтэжаныя, вы будзеце разгубленыя, але вы навучыцесь значна хутчэй, чым толькі дзякуючы тэорыі.

Варта развіць такія мяkkія навыкі, як размова, перакананне і цярпенне, якія гэтак жа важныя, як і вашы тэхнічныя навыкі.

Перастаньце вагацца, пачынайце здзейсняць:

- далучайцесь да Групы SQAF у Telegram, да іншых чытачоў гэтай кнігі — аб'яднаўшы высілкі, вы будзеце рухацца хутчэй
- сачыце за мной у LinkedIn, каб атрымліваць апошнія абнаўленні і навіны, звязаныя з кнігай
- пішыце мне ў асабістую або на al.vorvul@gmail.com, калі вам патрэбна дапамога — і я паспрабую дапамагчы

Я заўсёды адкрыты вашых пытанняў, крытыкі і прапаноў.

Пераход ад вывучэння тэставання да практикі — не скачок.

Гэта шэраг невялікіх, памяркоўных кроکаў.

Найлепшы час для пачатку — быў учора.

Другі найлепшы час — зараз.

Дзейнічайце.

Тэхналагічны індустрый патрэбныя больш кваліфікованыя, захопленыя сваёй справай спецыялісты па контролі якасці.

**Аляксандр Ворвуль**

**АСНОВЫ ТЭСТАВАННЯ  
ПРАГРАМНАГА ДАЧЫНЕННЯ**

Электронная версія

Першае выданне

Кніга абаронена аўтарскім правам.

Капіраванне для мэтаў, акрамя асабістага спажывання,  
дазволена толькі са згоды ўладальніка аўтарскіх праў.

[al.vorvul@gmail.com](mailto:al.vorvul@gmail.com)



©Аляксандр Ворвуль

Мінск 2025