Моделювання верстатного обладнання з використанням високорівневих мов програмування для техніко-економічного аналізу

На сьогодні рівень розвитку обчислювальної техніки дає можливість широко використовувати її досягнення у всіх областях науки та техніки. Сучасні програмні засоби дозволяють автоматизувати великий об’єм математичних обчислень та моделювати процеси та системи будь-якої складності. Саме моделювання роботи машинобудівного підприємства з використанням ЕОМ та сучасних програмних продуктів дає можливість проектувальникам аналізувати та оцінити доцільність та можливість втілення того чи іншого рішення, провести порівняння або дослідження роботи у критичних умовах. З економічної точки зору, моделювання з використанням ЕОМ відкриває можливість планування на великі терміни з урахуванням великого числа факторів мінімізуючи вплив людського фактору. Також, системи моделювання та проектування підприємств відкривають великий потенціал накопичення організаційно-проектних рішень та централізацію доступу до останніх.

Для наглядності та демонстрації переваг такого підходу до проектування реалізуємо моделювання верстату мовою програмування Python з використанням об’єктно-оріентованої методології програмування. Згідно з данною методологією кожен верстат має бути представлений екземпляром окремого классу котрий буде інкапсулювати у собі атрибути та методи, що відповідають його характеристикам, діям, котрі можна виконувати з ним та способам взаємодії з ним. Для оптимізації процессу моделювання має бути підготовлена спеціальна база данних(бібліотека), котра буде зберігати класи основних та найпоширеніших моделей верстатів, а також абстрактні класи для того щоб максимально швидко моделювати відсутні верстати використовуючи механізм, що у термінах об’єктно-оріентованого програмування називається наслідування. Прикладом буде токарно-гвинторізний верстат 16К20.

Спочатку реалізується класс верстату та метод \_\_init\_\_():

class Lathe16K20:

def \_\_init\_\_(self, height, width, long, power, spindle\_position):

self.height = height

self.width = width

self.long = long

self.power = power

self.spindle\_position = spindle\_position

Вище реалізовано клас верстату 16К20 з основними атрибутами(для наочності), тепер для ініціалізації екземпляру верстату потрібно вказати назву екземпляру, назву класу та усі аргументи функції \_\_init\_\_() у порядку вказаному у функції, так як це порядкові аргументи.

Ініціалізація екземпляру:

lathe1=Lathe16K20(1500, 800, 5000, 2.2, ‘horizontal’)

Також для спеціалізації та уточнення характеристик верстату є можливість додавання окремих атрибутів конкретному екземпляру без внесення змін у клас верстату на випадок якщо верстат має якісь локальні модифікації чи удосконалення:

lathe1.rotation\_frequency = 8000

Для підвищення зручності моделювання та використання моделей атрибутам екземпляра можна присвоювати масиви даних або іменовані масиви, наприклад, для відображення позиції супорта верстата можна присвоїти атрибуту сaliper\_position іменований масив з вертикальною та горизонтальною координатами:

lathe1.clipper\_position = {‘y\_coordinate’:200, ‘z\_coordinate’:80}

Для моделювання методів, тобто дій які можна виконувати з верстатом потрібно реалізувати окрему функцію, наприклад переміщення супорта вздовж горизонтальної осі у відносних координатах:

class Lathe16K20:

. . .

def relative\_displacement(direction, value):

if direction == ‘+’:

self. clipper\_position[y\_coordinate]=+value

return self

if direction == ‘-’:

self. clipper\_position[y\_coordinate]=-value

return self

Методу передаються два аргументи, напрям та значення переміщення і в залежності від цього метод relative\_displacement змінює атрибут екземпляру.

ВИСТНОВКИ: