ICS 35.240.01
CCS M30 49

团 体 标 准

T/SZAS 77—2024

企业级低代码开发平台技术要求

Technical requirements for enterprise-level low-code development platform

2024-02-20 发布　　2024-03-20 实施

深圳市标准化协会　　发 布

目　　次

前　言

本文件按照GB/T 1.1—2020《标准化工作导则 第1部分：标准化文件的结构和起草规则》的规定起草。

本文件由腾讯云计算（北京）有限公司提出。

本文件由粤港澳大湾区标准创新联盟工业互联网委员会归口。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件起草单位：深圳市腾讯计算机系统有限公司、腾讯云计算（北京）有限公司、深圳市标准化协会、中国电子技术标准化研究院、浪潮通用软件有限公司、北京百度网讯科技有限公司、南方电网数字集团有限公司、云南电网有限责任公司信息中心、卡奥斯工业智能研究院（青岛）有限公司、海信视像科技股份有限公司、Thoughtworks Limited、香港数字制造研究院有限公司、欣旺达电子股份有限公司、佛山市灵泽万川人工智能科技有限公司、中山大学、华南理工大学、同济大学、广州农村商业银行股份有限公司、深圳市六度人和科技有限公司、云安全联盟大中华区、澳科质量（珠海横琴）科技有限公司、道尔(中国)有限公司、万高信息科技有限公司。

本文件主要起草人：丁涛、揭光发、王永霞、代威、骆勤、宁鹏伟、吕洋、林楠、但丹、苏伟、国建勋、郑伟波、魏代森、杨楠楠、黄翔、潘征、李玲璠、李申章、孙浩、郑文霄、张宏伟、王之奎、杨璐、李蓬勃、赵海涛、王旭、李涛、冯昕、邓瑶、何双峰、冯颖、吴新勇、许木娣、李鹏、全晶丽、王金。

本文件为首次发布。

企业级低代码开发平台技术要求

1 范围

本文件规定了企业级低代码开发平台系统架构和通用技术要求，包括领域专用语言（DSL）层、核心能力层、集成开发、运行方式、运营运维服务层等能力要求。

本文件适用于企业级低代码开发平台的设计、开发和实施。

2 规范性引用文件

本文件没有规范性引用文件。

3 术语和定义

下列术语和定义适用于本文件。

3.1

企业级低代码开发平台 enterprise-level low-code development platform

为组织数字化业务提供降低编码量甚至无需编码而达到研制开发服务/产品能力的平台。

3.2

领域专用语言 domain specified language

针对某一特定领域,具有受限表达性的一种计算机程序设计语言,为低代码平台提供领域开发语言。

注：DSL具有以下三个特点：

a）语言性（language nature）：DSL是一种程序设计语言，具备连贯的表达能力，不管是一个表达式还是多个表达式组合在一起；

b）受限的表达性（limited expressiveness）：通用程序设计语言提供广泛的能力：支持各种数据、控制，以及抽象结构。这些能力很有用，但也会让语言难于学习和使用。DSL只支持特定领域所需要特性的最小集。使用DSL，无法构建一个完整的系统，相反，却可以解决系统某一方面的问题；

c）针对领域（domain focus）：只有在一个明确的小领域下，这种能力有限的语言才会有用。

3.3

逻辑可视化 logic visualization

通过图形化界面，将业务逻辑~~代码~~表达为逻辑节点与线条，对业务代码逻辑、业务流程和数据处理逻辑等进行编排的一种实现方式或工具。

3.4

UI可视化 UI visualization

通过包括但不限于组件拖拽、表单填写等交互方式，完成UI中页面、布局、组件、容器、插件、事件和模板的编辑和开发。

注1：组件是页面编排的最小粒度，容器中可以嵌套组件和容器。

注2：事件是页面和组件交互逻辑的基本单元。

注3：模板是基于不同业务逻辑和技术栈固定格式，包含组件和容器。

注4：插件指的是无UI的组件。

3.5

组件 component

UI可视化和逻辑可视化编排所需的最小单元统称为组件,通过对组件的编排可以搭建出完整的业务逻辑。

注：常见的组件类型有UI组件、事件组件、逻辑组件、模板组件等。通常组件可以被业务方自定义扩展。

3.6

数据模式 data schema

用于描述数据模型的元数据集合。

注：后端数据库连接的相关连接参数(主机、端口等)。

3.7

数据模型 data model

系统借助数据库与元数据配置信息通过自动生成常用数据操作接口并用数据模式来记录这些配置的一种模型。

3.8

一码多端 write once run multiterminal

通过一份低代码设计和开发数据(包括逻辑、页面等)，生成多个场景/平台下运行的应用，且具备一致的表现内容和交互逻辑。

3.9

解析引擎 parsing engine

将UI模式（UI Schema）的数据和逻辑模式（Logic Schema）的数据作为输入，并解析运行的程序框架。

3.10

编译引擎 compiling engine

通过对UI模式（UI Schema）和逻辑模式（Logic Schema）进行一定规则的处理，翻译为指定高级语言的程序框架。

3.11

灰度发布 grayscale publishing

一种支持平滑过渡的发布方式，按照一定策略选取部分用户，让其先行访问体验产品新版本功能，通过收集这部分用户对新版本功能的反馈以及监控服务运行状态，以及对新版本功能、性能、稳定性等指标进行分析，进而决定继续放大新版本投放范围直至全量升级或回滚至老版本。

3.12

热更新 hot fix

通过动态下发和加载代码，使App或SDK在不重新下载和安装的情况下，改变其原有代码逻辑或资源文件。

注：低代码平台的热更新，本质上是对Schema信息的更新，在动态解析(解析执行)模式下，开发者可以使用配置等方式将Schema信息进行配置，管理和动态下发，完成热更新的能力。

3.13

低代码开发平台产物 low-code development platform product

开发者通过低代码开发平台生产出来的应用配置与代码数据，包括但不限于配置文件、数据、代码、二进制文件。

4 缩略语

下列缩略语适用于本文件。

AI：人工智能（Artificial Intelligence）

API：应用程序编程接口（Application Programming Interface）

APP：应用程序（Application）

CPU：中央处理器（central processing unit）

DSL：领域专用语言（Domain Specified Language）
IP：网际互连协议（Internet Protocol）
JSON：轻量级数据交换格式（JavaScript Object Notation）
OS：操作系统（Operating System）
PC：个人电脑（Personal Computer）
PV：页面浏览量（Page View）
SaaS：软件即服务（Software as a Service）
SDK：软件开发工具包（Software Development Kit）
SLA：服务水平协议（Service Level Agreement）
TCC：类型正确性条件（Type Correctness Condition）
UI：用户界面（User Interface）
XML：可扩展标记语言（eXtensible Markup Language）
YAML：数据序列化表示格式（YAML Aint a Markup Language）

5 系统架构

企业级低代码开发平台，包括领域专用语言（DSL）、核心能力、集成开发、运行方式、运营与运维服务等模块，具体为：

a) 领域专用语言模块，提供低代码开发平台的开发语言能力；
b) 核心能力模块，提供低代码开发平台的核心能力引擎，包括前后端逻辑编排、前端页面编排与多端适配，后端数据模型与数据接口编排等前端与后端能力；
c) 集成开发，提供低代码开发平台所需的代码编辑、可视化设计、AI开发等能力；
d) 运行方式定义了低代码开发平台主要的运行方式；
e) 运营与运维服务，提供低代码开发平台开发产物相关的多种能力要求。

企业级低代码开发平台系统架构如图1。

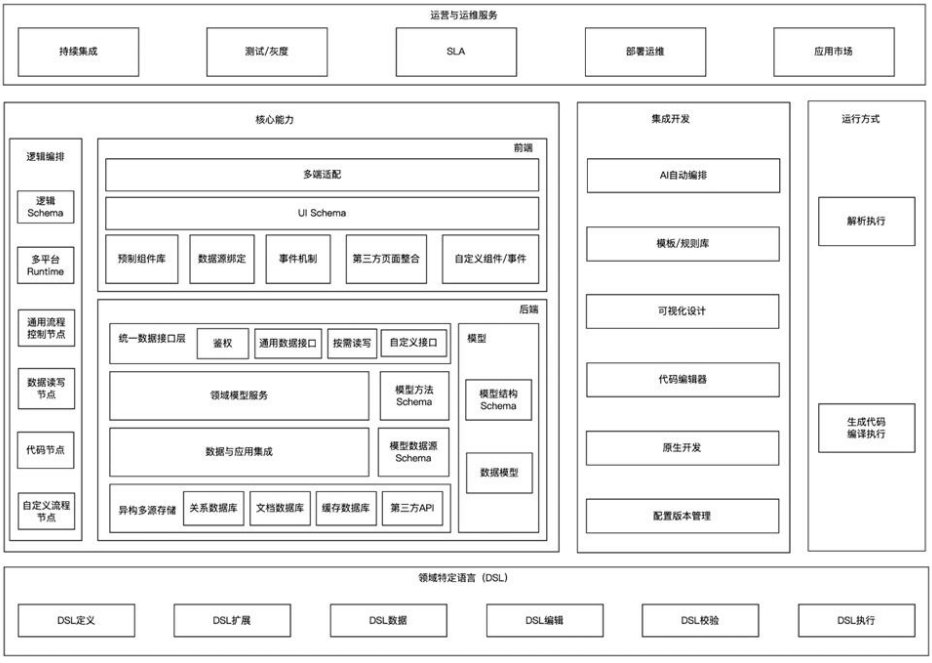

图 1 企业级低代码开发平台系统架构

6 领域专用语言（DSL）

低代码开发平台至少支持领域专用语言（DSL），或至少一种高代码语言，以满足无法通过平台编排手段支撑的业务个性化需求。DSL提供文本编辑的配置方式，并满足以下技术要求：

a) DSL定义：提供DSL所需的配置文件的结构，包括格式、语法、参数、语义、领域模型等；
b) DSL扩展：应提供DSL动态扩展的能力，应对业务的个性化开发需求；
c) DSL数据：应提供用于构建和运行低代码应用的配置数据；
d) DSL编辑：指配置数据的编辑方式，应支持手动编写、可视化编辑或AI自动生成三种方式；
e) DSL校验：应具有DSL编辑期间对所生产的DSL格式及合理性进行检测并提示的能力；
f) DSL执行：应具有解释DSL并实时运行的解析执行，以及编译DSL为原生代码再运行的编译执行两种能力。

7 核心能力

7.1 逻辑编排

低代码开发平台应当提供逻辑编排的能力，解决平台内置组件及功能模块未实现的开发能力，包括但不限于以下能力：

a） 逻辑模式（Logic Schema）：低代码平台应能够通过逻辑编排，生成描述逻辑执行流程的逻辑模式（Logic Schema），包括条件、循环、赋值、计算、调用关系等内容，描述文件可为JSON，YAML，XML等主流格式数据文件；
b） 多平台Runtime：应支持多种运行环境，具备在不同的平台上运行同一套应用程序，实现跨平台应用开发和部署；
c） 通用流程控制节点：应提供用于控制业务流程的节点能力，实现流程控制、条件判断、循环等功能，构建各类业务流程；
d） 数据读写节点：应提供业务流程中读取和写入数据的能力，实现数据的持久化和共享；
e） 自定义流程节点：应提供扩展业务流程的节点，自定义业务流程，实现个性化业务需求；
f） 代码节点：应提供编写自定义代码节点的能力，实现特定的业务逻辑，扩展低代码平台的功能和灵活性。

7.2 前端

低代码开发平台应当提供用户界面的低代码开发能力，包括但不限于以下要求：

a) 用户模式（UI Schema）：应提供UI Schema用于描述用户界面，通过容器、布局、组件、样式、事件、路由等内容描述用户界面的元数据集合，包括但不限于页面布局、样式、尺寸、颜色、响应用户的行为、与后端数据交互等能力；
b) 多端适配：应具备多终端适配的能力，包括但不限于Web、小程序、Android、iOS、Windows、Mac OS等终端类型；
c) 预制组件库:应提供业务领域所需的基础组件和原子组件，并可基于生成器提供的基础能力扩展出定制的组件或者能力，具备灵活性；
d) 数据源绑定：应支持对低代码开发过程中所需要的数据源进行绑定的功能，包括但不限于绑定局部变量，全局变量到指定的数据库表、接口返回或者其它自定义的数据源的能力；
e) 事件机制：应具备对事件响应的实现，包括但不限于事件触发器、生命周期事件管理以及响应事件的存储、网络、UI行为；
f) 第三方页面整合：应具备对第三方页面整合的能力，包括但不限于使用页面跳转，微前端技术；
g) 自定义组件/事件应当满足以下能力要求：
   1) 应支持自定义组件能力，自定义组件应按照约定规范进行设计和开发，并接入低代码开发平台，进行上架和使用；
   2) 应支持自定义事件能力，自定义事件允许组件自定义事件触发器和事件响应，通过不同组件之间自定义事件触发器和事件响应的编排，达到组件联动的效果。

7.3 后端

低代码开发平台应提供模型驱动的后端能力，包括但不限于以下要求：

a）模型：

低代码开发平台的模型应满足以下要求：

1) 模型结构：提供定义数据模型结构的规范，用于描述包括但不限于数据类型、关系、验证规则和其他数据模型规范，描述语言包括但不限于JSON、YAML、XML；
2) 数据模型：为业务系统设计对应的使用模型结构的数据模型；

b) 领域与模型服务：提供一组与业务领域相关的服务，通过这些服务可以创建、修改和查询领域内的数据；
c) 模型方法：提供模型交互的一组操作，包括创建、读取、更新、删除（CRUD）操作，并具有可扩展性，允许自定义逻辑；
d) 数据与应用集成：应支持整合存量的数据与应用，以实现传统应用系统现代化、连通企业数据、避免数据孤岛和烟囱应用；
e) 模型数据源：定义数据模型如何连接到其数据源。包括关系型数据库、NoSQL数据库、文件系统或云服务。应提供一种配置方法，使其可以轻松地切换或集成多个数据源；
f) 数据接口应满足以下要求：
   1) 鉴权：确保数据交互过程都要经过安全的鉴权步骤，以验证和授权数据访问；
   2) 通用数据接口：前端应用通过统一的方式请求数据；
   3) 按需读写：支持灵活的数据访问，允许根据需要读取或更新数据的特定部分；
   4) 自定义接口：允许开发者创建特定于应用的接口，以支持复杂的或非标准的数据交互需求；
g) 异构多源存储应具备对存储资源的访问能力，包括但不限于：
   1) 存储资源应包括但不限于关系数据库、内存数据库、对象存储、文件存储等，Redis和大数据平台等存储资源和中间件的访问能力；
   2) 应支持关系数据库，用于处理具有固定模式的数据；
   3) 宜支持半结构化的数据存储，如JSON或XML文档，允许灵活的数据模型和快速的数据访问；
   4) 宜支持缓存数据库，提供快速的数据读取和写入，通常用于提高应用性能；
   5) 应能够整合和管理来自第三方API的数据，以扩展系统的功能和访问外部数据。

8 集成开发

8.1 AI 自动编排能力

低代码开发平台可提供基于AI的智能编程能力，包括智能生成、设计稿转页面、智能搭建辅助等方面自动生成代码的能力，具体为：

a) 智能生成：基于AI将用户的自然语言描述转换成产品页面、产品逻辑、数据模型、业务流程等；
b) 智能搭建: 基于AI提供智能化辅助，自动推荐相关的行业属性、场景组件、逻辑节点、接口等，提高低代码编程效率和低代码开发质量；
c) 设计稿转页面：支持开发者通过上传设计稿自动生成页面，解析设计稿中的组件，自动转换为低代码平台中的组件并组装成页面。

8.2 模板/规则库

应为开发者提供模板、规则库，提高组件/功能的复用率，加速应用搭建过程。

8.3 可视化设计

应为开发者提供可视化设计器，实现包括但不限于数据模型建模、用户界面搭建、逻辑可视化编排功能。

8.4 代码编辑器

应提供源代码管理能力，包括但不限于：

a） 源码导出相关能力，编辑器应具备导出源码文件到本地的能力；
b） 源码编辑相关能力，包括源码插入，在线编辑以及在线实时预览等能力。

8.5 原生开发

应提供原生开发管理和运行能力，包括但不限于：

a) 支持开发者通过传统开发模式来开发应用的功能，并整合至低代码平台的产物中；
b) 提供调用Open API接口和注册回调，根据业务形态以及业务流程，把可视化能力接入到业务流转的环节中，最大程度的保留业务系统原有的逻辑。

8.6 配置版本管理

应提供配置管理的能力，包括但不限于：

a) 版本管理的能力，维护每一个版本的信息，开发者可以进行版本回退操作、查看每一个版本信息的操作；
b) 冲突处理的能力，多人协同开发的过程中，应具备版本冲突处理的能力；
c) 接入第三方配置版本管理工具。

9 运行方式

低代码开发平台的运行方式，包含以下两类：

a) 解析执行：遵循低代码组件描述协议、应用描述协议、低代码应用开发框架协议和逻辑协议等，通过读取UI可视化工具和逻辑可视化工具生成的对应模式（Schema），完成前端展示和后台服务的解析和运行；
b) 生成代码编译执行：根据模式（Schema）包含的元数据、数据接口信息自动生成，提供将平台中的多种模式（Schema）翻译为高级语言的能力，具备至少一种编程语言能力，包括但不限于Java、Nodejs、Golang编程语言。

10 运营与运维服务

10.1 持续集成

应具备持续集成的能力，包括但不限于：

a) 运行平台提供微服务治理框架和多语言运行环境，并支持异构语言运行；
b) 对整个系统提供负载保护功能，如入口流量控制，单位软硬件资源保护(CPU/内存/磁盘/网卡/队列等)，对关键路径进行熔断保护等；
c) 提供对系统关键指标支持实时或者离线对账的能力；
d) 支持业务动态扩缩容，在线回滚，功能降级等。

10.2 调试与测试

应具备对开发过程的调试与测试的能力，包括但不限于：

a) 支持单元测试、接口测试、压力测试以及安全测试，可接入混沌工程等能力；
b) 支持灰度发布中观察以及染色跟踪，链路跟踪，以及系统拓扑展示等能力。

10.3 灰度发布

应具备灰度发布的能力，包括但不限于：

a) 支持以可控制的比例选择特定用户群体使用新版本功能，剩余用户使用旧版本功能；
b) 对于移动端 H5页面、PC页面访问，支持以用户唯一标识或访问源IP决定访问前端资源的新旧版本；
c) 对于小程序、PC客户端、Mac客户端、iOS客户端、Android客户端，支持以用户唯一标识决定推送新版本更新；
d) 对于后端服务访问，支持以用户唯一标识或访问源IP决定访问后端服务的新旧版本；
e) 对于后端微服务，支持全链路灰度发布。

10.4 部署

应具备对开发应用的部署能力，支持以下相关功能：

a) 可视化部署前端UI资源、后端服务资源的能力；
b) 对产物导出支持私有化部署的能力；
c) 分布式一致性的能力，可采用分布式事务、TCC最终一致性保证等方式；
d) 多维度日常查询系统，如用户维度/请求维度等待；
e) 宜提供自动化部署能力；
f) 宜支持部署参数设置的能力；
g) 宜支持以容器化方式部署前端UI资源；
h) 宜支持以容器化方式部署后端微服务，支持平行扩容部署；
i) 可支持以无服务器方式部署逻辑编辑后端原子逻辑节点，支持水平扩容部署；
j) 可支持查看部署详情，可视化显示部署进度、部署日志；
k) 可预估系统容量，针对业务容量波动提前进行部署调整。

10.5 热更新

应具备热更新(发布)相关能力的要求，包括但不限于：

a) 后端热更新能力：在保证接口正常服务的情况下，完成后端业务逻辑调整之后平滑更新的能力；
b) 前端热更新能力应满足以下要求：
   1） H5页面的界面、样式、组件功能可在应用线上正常运行期间完成自动更新；
   2） 对小程序可完成对小程序的部分逻辑、UI、样式的热更新；
   3） 对Android应用，可完成对Android应用的UI、样式、逻辑的热更新；
   4） 对iOS客户端，可按iOS系统要求完成有限的热更新。

10.6 应用监控

低代码开发平台应具备应用监控的能力要求，包括但不限于：

a) 对计算资源以及相关基础软硬件资源进行监控，如CPU/内存/网卡/硬盘/网络以及对应云上设备；
b) 根据系统上报数据进行监控，基于阈值配置，可支持基于监控数据的智能化检测和告警；
c) 对于存储或资源等关键服务进行逻辑验证；
d) 对系统关键指标如流量，产值等进行异常检测以及对账等；
e) 与用户反馈、其他监控系统进行对接，及时发现各种隐患；
f) 告警产生时，给出异常原因，影响范围，相关人员，以及处理措施；
g) 支持工具链定位问题，通过数据分析工具定位问题，通过日志系统验证；
h) 支持多维度监控，包括不限于业务资源维度、系统维度、业务维度、安全维度、用户反馈维度等。

10.7 安全

10.7.1 鉴权能力

低代码平台开发的产品应具备分配和管理账号权限的能力，包括但不限于：

a) 账号体系：低代码平台应具备账号体系的管理能力，包括账号体系类型和账号体系权限；
b) 用户组：一个用户组应由多个账号组成，系统可同时维护多个用户组；

注：用户是指低代码开发的产品所面向的终端用户。

c) 鉴权：

用户访问低代码产品时，应具备以下鉴权能力：

   1） 鉴权粒度：系统按 "资源"+"参数" 粒度对 "用户/用户组" 进行权限授权；
   2） 鉴权方法：提供鉴权接口，用户在低代码平台使用资源时，可选择是否启用系统鉴权。

10.7.2 审计能力

应具备对用户操作的记录和审计功能：

a) 记录和存储用户对元数据、业务数据、平台数据等操作的能力；
b) 审计精度应支持租户级、应用级、对象级、字段级等。

10.8 SLA

10.8.1 平台 SLA

低代码开发平台应提供本身的SLA，满足以下要求：

a) 服务等级指标包括但不限于以下指标：
   1）低代码开发平台界面访问可用性，包括界面交互的可用性、显示的正确性、完整性以及响应时间等；
   2）低代码开发平台相关服务接口可用性，包括整个开发平台所有涉及到数据存储、数据处理、引擎转换等所有相关接口的服务可用性；

b) 服务等级目标包括但不限于以下指标：
   1）页面访问可用性要求；
   注：计算方式为：（一定周期内总时长-页面不可正常访问的时长)/一定周期内总时长。
   2）平台相关服务接口可用性要求。
   注：计算方式为：（一定周期内总时长-接口异常不可用的时长)/一定周期内总时长。

10.8.2 产物 SLA

低代码开发平台应提供开发的产物的SLA，满足以下要求：

a) 服务等级指标包括但不限于以下指标：
   1）逻辑兼容性：前端逻辑在各个终端和宿主（浏览器）环境中运行的兼容性；
   2）UI适配性：前端UI界面和交互效果在各个终端和宿主（浏览器）环境中的一致性，主要包括各尺寸终端的显示完整性和交互一致性；
   3）接口访问稳定性：后端服务接口访问的服务可用性；

b) 服务等级目标包括但不限于以下指标：
   1）逻辑兼容性要求；
   注：计算方式为：(产物在一定周期内总PV-因为兼容性异常的PV)/产物在一定周期内总PV。
   2）UI适配性要求；
   注：计算方式为：(产物在一定周期内总PV-因为UI适配性异常的PV)/产物在一定周期内总PV。
   3）接口访问稳定性要求。
   注：计算方式为：(产物在一定周期内总接口请求数-异常接口请求数)/产物在一定周期内总接口请求数。

10.9 应用市场

低代码开发平台宜支持发布到应用市场的能力，包括但不限于:
a) 为开发者提供成熟的应用市场模块，让开发者快速使用；
b) 开发者向应用市场提交应用，丰富市场的应用类型；
c) 应用市场可支持应用审核功能，避免质量不高的应用流入市场，影响用户体验；
d) 应用市场可具备统一的订阅管理，如应用购买支付、价格方案设置，应用的过期管理，支持多租户SaaS场景。

10.10 国际化

低代码平台应用宜具备多语言、时区配置的能力。

参 考 文 献

[1] GB/T 18978.151—2014 人-系统交互工效学 第151部分：互联网用户界面指南
[2] GB/T 24629—2009 元数据的XML Schema置标规则
[3] GB/T 30522—2014 科技平台　元数据标准化基本原则与方法
[4] GB/T 37729—2019 信息技术 智能移动终端应用软件（APP）技术要求