**知识管理系统需求和设计笔记**

**企业的问题所在**

管理机制：对知识管理仍采用分散管理的形式，企业知识四处散落并且难于寻找，缺乏公司层面的管理机制。

知识流失：员工跳槽离开公司将企业知识带走，或员工升迁、换岗时知识流失。

知识复用：由于知识的不成体系，开展新项目或需要多个部门协同合作时，缺乏足够的知识获取途径，致使工作效率低下，或重复犯出现过的错误。

决策支持：管理层缺乏历史的资料做参考和支持，使得决策依据缺乏。

岗位知识：新员工上岗时缺乏指导，适应工作比较困难。

知识传递：公司的知识传递机制主要通过面授和培训，存在着范围狭窄和频率过低的缺陷。

**管理知识积累的几种途径**

页面: 企业内部的新闻, 行政规范, 帮助说明等网页页面.

文档：日常工作产生的每一份文档

任务：通过KM系统的任务系统发出的每项任务( 通常会有文档以附件形式关联, 以文档的形式汇报)

会议：每次会议的会议演讲稿、会议发言纪要、会议结论

消息: 发送给个人或组织的消息, 通告等信息, 此类信息可关联至其他类型的某项知识条目

业务流程：工作流中流转的文档，最终流向KM系统

**框架级需求点**

**知识实体**. 系统内所有类型的知识都可以被抽象为知识实体(KE, KnowledgeEntity). 新闻, 文档, 任务等知识项都是知识实体抽象类的一个具体子类. 知识实体所具备的基本属性包括:

系统全局范围内ID

知识类型标识符. 以字符串标识, 例如:"System.Doc.Office.Excel" 标识这是一份标准的MS office excel文档. 在客户端处理此KE的时候, 需要根据此信息找到可以处理此KE的模块. 例如, 用户在查看一份文档的时候, 可以看到有一个任务项和此文档相关, 用户点击此任务项想查看任务项的详细信息. 这个时候, 系统就会根据此KE的类型信息, 从配置信息中查询到类型标识符为"System.Task"的KE应该由A模块打开, 则系统会自动加载A模块, 传递这个KE作为参数, 让A模块打开.

系统全局定位符. 用于指明此KE的存放路径. 例如: <http://host/library1/doc1.doc> [\\fileserver\disk1\folder1\doc1.doc](file:///\\fileserver\disk1\folder1\doc1.doc) 等.

权限信息. 系统内每一个KE都应能附着一定的权限数据. 权限信息应具备如下特性:

权限信息的最小粒度应该到单个KE, 例如一份文档, 一个任务项等.

权限信息应细分为多个级别, 例如, 可读,可写,可删除, 可复制等.

权限信息应支持继承关系, 例如给某个文件夹赋予可写权限, 则缺省情况下, 文件夹中的文档也具备可写权限.

权限信息应支持最近原则. 例如给某个文件夹赋予可写权限, 但其中某个文件的权限是不可写, 则以最近的不可写权限为准.

元数据集. 每个KE都可以附着若干种元数据信息, 例如创建日期, 作者, 创建语言, 是否归档, 隶属部门等. 特别需要指出的是

1) 元数据应是网状结构.

2) 元数据应划分为两类, 系统级, 和用户自定义.

3) 元数据应可以继承. 例如某个任务附着了元数据 "大客户支持", 则

关联关系. 系统内任何一个KE都可以与其他KE之间建立关联关系. 这种关联关系体现了知识累积之后带来附加效应, 是知识管理的核心价值所在. 例如, 用户A给用户B创建了一个待办事项, 同时也可以为这个待办事项关联上一份操作手册文档. 用户B在打开这个待办事项时, 不需要再自己去找操作手册, 直接在待办事项的关联资源中就能找到这个文档.

**知识元数据**. 这也是知识管理中的一个核心概念. 知识元数据是指附着在KE上的附加信息. 这些信息可以是系统在管理KE时自动附件的, 例如, 创建者, 版本号, 文件大小等, 也可以是从KE中抽取出来的, 例如针对图片文件, 元数据可以是图片的尺寸等信息. 知识管理系统中的知识元数据应具有如下特性:

层次/网状结构: 元数据项之间允许存在关联关系, 这种关联关系应可描述为层次网状结构. 例如针对某份销售方案文档, 其"针对客户"元数据的值为"上海移动", 而"上海移动" 又隶属于"上海大客户", "上海大客户"隶属于 "华东区客户". 那么如果用户指定搜索"华东区客户" 相关的销售方案文档, 这份文档应该显示出来. 需要说明的是, 元数据的内部存储和表达形式可以设计的很简单, 也可以设计的很复杂, 我们可以根据资源和时间的情况选择一种适合的设计方案. 另一方面, 不管设计的复杂程度如何, 系统对外暴露出的接口基本是一致的, 基本搜索和高级搜索. 我们在设计的时候, 需要考虑的一个问题是, 在保持对外接口不变的情况下, 如何能够以渐进累加的方式增强元数据的功能.

可扩展性: 系统会预定义一套元数据体系, 例如文档大小, 创建日期, 作者等. 同时应允许用户根据实际应用场景, 扩展出新的元数据结构. 并且扩展出的新的元数据结构在使用范围, 使用方式上应该与系统预定义的完全保持一致.

**知识订阅**. 知识订阅是指用户针对系统中的某个KE, 要求系统自动推送其状态变更的信息, 这是知识管理中的一项重要功能, 将原本需要由使用者手工进行的知识搜集工作转变为系统自动完成, 极大提高了知识处理的效率. 例如, 用户可以订阅某份文档的更新状态, 如果有其他用户对此文档进行更新, 系统会自动推送信息给用户, 或者以某种方式展现在用户界面上. 再例如, 用户可以订阅一个任务项的完成情况(有权限的情况下), 则系统会自动推送此任务的状态更新信息给该用户.

订阅的粒度应该可以是任何KE或者KE的容器. 每一个展现在UI上的KE及KE的容器都应该附着一个"订阅"/"取消订阅"的操作. 订阅后进入到个人的"订阅库"中.

系统应提供一个UI容器(Grid/tree view/List) 能够根据指定的过滤条件呈现订阅信息. 过滤条件例如: "本部门, 未关闭的, 和某次会议相关的任务".

客户端应能定时向服务器请求刷新用户的订阅信息. 当订阅信息有更新时, 需要以某种直观方便的方式提示给用户.

**知识搜索.** 知识检索可以划分为简单搜索和复杂搜索两种模式.

简单检索模式类似于google搜索界面. 用户输入任意关键字, 系统返回匹配到的查询结果.

复杂检索模式类似与google的高级检索. 用户可以在输入关键字的同时, 指定若干查询条件, 例如文档创建时间, 所属部门等元数据过滤条件. 系统返回匹配到的查询结果.

简单搜索和复杂搜索都应支持在搜索集合中进一步过滤的操作. 例如, 用户使用简单搜索, 得到搜索结果为100篇文档, 分别属于5个部门, 则用户可以继续勾选部门, 去掉他认为不需要的部门的搜索结果, 只留下感兴趣的部门搜索结果. 具体操作模式可参考主流电子商务网站的产品搜索操作.

知识搜索的一个高级功能是搜索订阅, 即用户可以定义一个知识搜索条件组合, 提交给系统作为知识订阅源, 系统如果检测到匹配知识搜索条件组合的知识集合有更新, 则会主动通知用户. 知识搜索的概念可以在KM系统很多地方使用到. 例如, PMO部门为了跟踪各个工作项的进展. 可以订阅一个搜索条件"类型=工作项, 状态=未完成, 相关部门=xx部门". 那么当符合条件的任务项的状态或元数据发生改变后, 订阅用户就会收到通知.

**分知识类别的需求点**

**页面:**

用户应可以创建自定义内容的页面. 例如某次活动的新闻稿, 针对某个业务流程的帮助说明等. 创建页面的界面可参照SharePoint的界面或者其他常规的CMS系统界面, 不需要特别复杂.

页面的发布需要有一个简单的审批机制, 这个机制不需要很复杂, 但是很多场景下都会使用到.

页面应允许挂载自定义操作, 例如评论, 评级等. 保持足够的可扩展性.

用户浏览页面信息时, 在界面上应该有某个位置让用户可以很方便的进行导航操作, 例如上一篇, 下一篇, 相关文章等.

同其他KE, 用户可以针对页面容器的更新进行订阅操作.

**文档:**

能够查看某个文档/文件夹的访问信息: 某个范围内的用户在某个时间段内进行的某种操作的统计信息

针对某份文档的自定义操作.

添加注释,

讨论

评级

发起任务

发起流程

为文档添加元数据. 例如: 保密等级、所属产品、有效日期、权限继承、共享范围、文档协作者、审批流程选择等

所有文档类知识都应支持批量操作, 例如批量删除, copy等

在文档容器上(主要是文件夹), 可设定允许容纳的文件类型

可指定允许放入系统的文档类知识的大小

系统应提供一种直观方便的树形浏览方式, 让用户可以很方便的定位到某个文档类知识

**任务:**

发起任务应该是一种受控活动. 例如, 用户A只能发起1,2,3种任务, 其中任务1的受理人不能是部门经理等.

系统应允许转发和代理任务. 支持自动和手动两种

支持任务分解. 例如用户A给用户B设定了一项任务, 用户B可以分解这个任务, 将其拆分为3个子任务, 分别设定为C, D和自己. 任务发起者应该能指定任务是否可分解. [任务分解时需要指定权重]

分解后的子任务的进度信息应能自动汇总到上级.

任务的相关人员及具有权限的人员可方便的查阅任务的上下级汇总信息.

任务应有提醒功能 (超时, 重要级别等), 提醒方式包括系统界面提示, 邮件, 短信等.

任务的状态迁移方式应该能灵活定义. 例如, 一般来说, 任务的状态可以是, 已发起, 执行中, 已关闭. 某些业务场景下, 任务不允许由执行人自行关闭, 需要添加一个状态, 已完成. 执行人可以设置任务的状态为已完成, 然后发起人审核后再调整为已关闭. 具体参考VSTS中的任务状态机.

系统应该提供一个界面, 用户能够以直观的查阅和自己相关的当前任务信息, 具体包括:

我自己的任务 [未完成]

我发起的任务 [未完成]

我关心的任务

系统应该提供一个界面, 用户能够以直观的查阅和自己相关的任务历史信息, 类别与当前任务相同.

系统应该提供多种任务查询和展现方式, 例如根据任务类别展现, 根据任务时间展现等.

[高级] 新创建的任务可以保存为任务模板, 供以后创建相似任务时重复使用.

**会议:**

会议可以理解为一中自定义的知识集合模板. 里面包含有若干种按照时间顺序组织起来的知识类型, 例如: 文档, 任务项, 人员等. 因为在企业中会议是一种很常见的知识聚合, 所以单独列出来作为一种知识类型来表达. 需要注意的是, 系统应该支持其他类型的知识扩展.

会议类型知识至少包含这些类型的信息: 会议时间, 会议地点, 与会人, 主持人, 会议标题, 会议议程, 会议纪要, 会议决议, 会议相关资料, 待办事项等. 在实际应用场景中应允许自定义和扩展.

一般情况下, 会议将由一个会议流程驱动, 例如会议发起, 会议准备, 会议召开, 会议反馈, 会议结束等. 这个流程应可以灵活配置和调整. 例如, 在某些应用场景下, 必须等提交了会议纪要之后就允许关闭会议流程, 而有些应用场景下, 需要等待会议待办事项全部完成之后才能关闭会议流程.

**消息:**

消息可划分为通告和个人消息, 两者的区别主要在于UI呈现的方式不同, 公告可能会是在UI上一块专门的区域以列表或网格形式呈现. 例如 "部门公告", "公司通告"等. 而个人消息则是以类似于常用的即时消息工具来呈现.

通告.

通告的接受对象可以为个人, 或若干人员的集合, 例如所有部门经理, 部门全体成员等.

通告分为实时通告和非实时通告.

通告有优先级：紧急，高，普通等.

通告有有效期

通告可要求用户点击回应.

个人消息

个人消息的接收对象为个人.

个人消息只可以为实时消息.

个人消息的其他需求和设计点参见常用的即时消息工具.

**业务流程**

涉及到业务流程的知识很有价值, 但也更难于整合到KM系统中, 这一块建议放在后续阶段考虑.