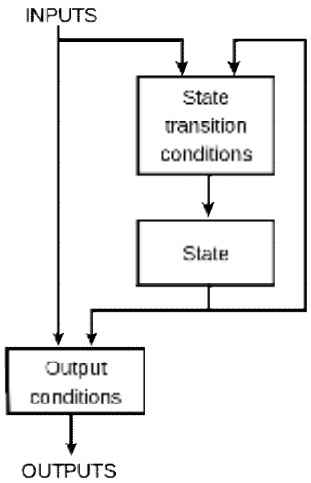
BPM Quick notes

= Basic Concepts

Concept	Related information	Reference
FSM Finite State Machine 有穷状态机	FSM logic: The next state and output of an FSM is a function of the input and of the current state. 	https://en.wikipedia.org/wiki/Finite-state_machine
BPM Business Process Management Biz Proc Mgmt		https://en.wikipedia.org/wiki/Business_process_management

业务流程管理		
BPME BPM Engine BPM 引擎	Business rules engine In any IT application, business rules change more frequently than other parts of the application code. Rules engines or inference engines serve as pluggable software components which execute business rules that a business rules approach has externalized or separated from application code. This externalization or separation allows business users to modify the rules without the need for IT intervention. The system as a whole becomes more easily adaptable with such external business rules, but this does not preclude the usual requirements of QA and other testing. 在信息科技应用系统中，商业逻辑比应用代码变化得更为频繁。规则引擎或者推理引擎作为插件式软件能执行商业逻辑的方法需要脱离于代码独立实现。这个独立实现能让用户在不需要信息部门的介入下直接修改商业逻辑。	https://en.wikipedia.org/wiki/Business_rules_engine BPMN Designer: Camunda: https://camunda.org/ ProcessOn: https://www.processon.com/
BPMN « BPM & Notation	Business Process Model and Notation (BPMN) is a graphical representation for specifying business processes in a business process model. **Element** BPMN models consist of simple diagrams constructed from a limited set of graphical elements. For both business users and developers, they simplify understanding business activities' flow and process. BPMN's four basic element categories are: **Flow objects** Events, activities, gateways **Connecting objects** Sequence flow, message flow, association **Swim lanes** Pool, lane **Artifacts** Data object, group, annotation These four categories enable creation of simple business process diagrams (BPDs). BPDs also permit making new types of flow object or artifact, to make the diagram more understandable.	https://en.wikipedia.org/wiki/Business_Process_Model_and_Notation

= BPMN Tools

http://bpmn.io from camunda => .bpmn file
http://processon.com/ => .pos file

= Our Design

- Design BPME with BPMN

https://www.processon.com/embed/570759a0e4b0dcddf98190f1

- Our BPMN Elements vs Standard BPMN Elements

Seq	Our BPMN Elements definitions		Refer to BPMN Elements	
1	FO (**Flow Objects/Element)**		Flow objects	
		Event(Start/Intermediate/End/etc),		Events
		Program (Task/Call Activity/etc),		Activities
		Gateway (Exclusive/Inclusive/Complex/Parallel)		Gateways
2	LINK (Connection objects)		Connection objects	
		Sequence flow, Message flow, Association		Sequence flow, message flow, association
3	Swim lanes		Swim lanes	
		Pool, Lane		Pool, Lane
4	Artifacts		Artifacts	
		Data Object, Group, Annotation, Data Store, etc		Data Object, Group, Annotation, Data Store, etc

= Our Target

- BPMN Simple (but practical) Engine

Compile bpmn file(.pos, generated by processon) into structure below.

Structure & Properties			Description
{			
all:	{		//hashtable
	$id:	{	//the id of element auto generated by design tools, but will be overwrite by **$_m** in properties //if overridden by _m, it is **activity_name**
		id:	Ditto.
		name:	bpmn activity name
		type:	i.e. tagname e.g. Linker/SequenceFlow/TextAnnotation/ExclusiveGateway/etc
		title:	title
		}	
	}		
properties:	{		//Properties of the BPMN
	start		Value of default start _m
	BpmMode		Default, Func, Orm
	}		
idx_src:	{ $src_id: [{link_id:$link_id, src: $tgt_id}] }		Src-Tgt links
idx_tgt:	{ $tgt_id: [{link_id:$link_id, src: $src_id}] }		Tgt-Src links
}			

- BPME-PHP

1. Default mode
   DefaultMode 的耦合是用 同进程间 用 类.方法来耦合
   类 的空间，执行完后 fo_id/bp_id 不须保存

2. ORM mode
   OrmMode 是跨进程的，用数据库来做耦合

3. Func mode

将来有可能要应用这个数据结构到一些高密集运算, 例如： fibolacci，代码的耦合是 函数级的
就是说代码是用 碎片方式 加载的，比如：
include ("bpm.$_c.$_m.php");
这样在 _m 之间其实是 函数体内耦合，而不是 new $_c;
Func mode 非常接近 FSM 状态机计算；

- BPME-NODEJS

Similar as the concept above. (But Func mode seems not easy to implement)

Data Structure of FO

For the Engine to process the FOs, logically we need the data structure below. (Implementation should be different for different BpmMode)

Field	Mandatory(M) /Optional(O)	Type	Description
fo_id:	M	STRING	ID of the Flow Object
bp_id:	M	STRING	ID of the Biz Proc Instance
_c:	M	STRING	
_m:	M	STRING	
_p	O	JSON	NOTES: could have _c/_m/lang/_s/etc. inside _p.
env	O	JSON	The env when call. (Sometime the FO itself may need to use $thisBP->getStartContextEnv() to get the initial env of whole bpm.)

result		JSON
		||
		PAGESTRING
			//process result
			JSON:
			{
			STS: $STS, //MADATORY
			$other_data_name: $other_data,
			}
			
			PAGESTRING:
			STRING of UI

result_headers	O	JSON	some extra headers maybe need to send client
prev_fo_id	O	STRING	ID of previous FO
prev_fo_STS_a		STRING	Previous FO result status array
next_fo_id	O	STRING	ID of next FO
status	M	STRING	NEW | LOCK | DONE | JUMP NEW : means the FO is just created and enqueued LOCK: means BPME locked an going to process this FO DONE: the BPME have done the processing and no JUMP JUMP: processing done and need JUMP. Use with the next_id
engine	O	STRING	ID of bpme if needed